Universidade do Minho
Escola de Engenharia
Departamento de Informática

Artur Quintas

Creating Intelligible Metrics Road Traffic Analysis

April 2016

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Artur Quintas

Creating Intelligible Metrics Road Traffic Analysis

Master dissertation
Master Degree in Computer Science

Dissertation supervised by
César Analide
Fábio Silva

April 2016

ACKNOWLEDGEMENTS

First and foremost, I'd like to acknowledge the work of other authors, the myriad of books, articles, opinions, and other publications which allowed me to acquire the knowledge necessary to do this work. And equally important those who created the tools and the results upon which the developed solution and document were made.

Without the orientation of Cesar Analide and the continued supervision of Fábio Silva it would have not been possible to accomplish this work, for which I am thankful.

To my family, namely my mother Isabel and sister Joana, my sincerest thanks for your constant support not only during the making of this work but also throughout my years of education.

Lastly I would like to thank all the friends who have shown to be continuously available to both provide their opinions and companionship.

ABSTRACT

The increasing pervasiveness and lower cost of electronic devices equipped with sensors is leading to a greater and cheaper availability of localized information. The advent of the internet has brought phenomena such as crowd-sourced maps and related data. The combination of the availability of mobile information, community built maps, with the added convenience of retrieving information over the internet creates the opportunity to contextualize data in new ways.

This work takes that opportunity and attempts to generalize the detection of driving events which are deemed problematic as a function of contextual factors, such as neighbouring buildings, areas, amenities, the weather, and the time of day, week or month.

In order to research the problem at hand, the issue is first contextualized properly, providing an overview of important factors, namely Smart Cities, Data Fusion, and Machine Learning.

That is followed by a chapter concerning the state of the art, that showcases related projects and how the various facets of road traffic expression are being approached.

The focus is then turned to creating a solution. At first this consists in aggregating data so as to create a richer context than would be present otherwise, this includes the retrieval from different services, as well as the composition of a unique view of the same driving situation with new dimensions added to it. And then Models were created using different Machine Learning methods, and a comparison of results according to selected and justified evaluation metrics was made. The compared Methods are Decision Tree, Naive Bayes, and Support Vector Machine.

The different types of information were evaluated on their own as potential classifiers and then were evaluated together, leading to the conclusion that the various types combined allow for the creation of better models capable of finding problems with more confidence in such results.

According to the tests performed the chosen approach can improve the performance over a baseline approach and point out problematic situations with a precision of over 90%. As expected by not using factors concerning the driver state or acceleration the scope of problems which are detected is limited in domain.

RESUMO

A expansão e menor custo de dispositivos eletrónicos equipados com sensores está a levar a uma maior e mais barata disponibilidade de informação localizada. O advento da internet criou fenómenos como a criação de mapas e dados relacionados gerados por comunidades. A combinação da disponibilidade de informação móvel e mapas construídos pela comunidade, em conjunto com uma obtenção de informação através da internet mais conveniente, criou a oportunidade de contextualizar os dados de novas maneiras.

Este trabalho faz uso dessa oportunidade e tenta generalizar eventos de condução que são considerados problemáticos em função de factores contextuais, tais como a presença de edifícios, áreas, e comodidades na vizinhança, o clima, e a hora do dia, a semana, ou o mês.

De modo a investigar esta questão, o problema é contextualizado como emergente no tópico de Cidades Inteligentes, e explorado com recurso a Fusão de Dados e a Aprendizagem Máquina.

O estado da arte é exposto, através de projectos relacionados à expressão do tráfego rodoviário, dando relevo às várias facetas até então investigadas por outros autores de modo a enquadrar o trabalho presente.

Dado o enquadramento e concretização do problema, é proposta uma solução. Esta solução passa por inicialmente agregar dados de modo a enriquecer o contexto, incluindo a recolha destes de vários serviços, e uma composição dos dados recolhidos numa perspectiva única referente a uma situação de condução. Após este enriquecimento dos dados, são criados modelos com base em diferentes técnicas de Aprendizagem Máquina. Os métodos utilizados são Decision Tree, Naive Bayes, e Support Vector Machine.

Os resultados conseguidos com estes modelos são depois comparados de acordo com as métricas de avaliação seleccionadas.

Uma comparação foi feita também com diferentes tipos de informação separadamente e também em conjunto, levando à conclusão de que os vários tipos combinados permitem a criação de melhores modelos capazes de encontrar problemas com mais confiança nos resultados produzidos.

De acordo com os testes executados a abordagem escolhida consegue melhorar resultados de um modelo base e descobrir situações problemáticas de condução com uma precisão acima dos 90%. No entanto, como seria de esperar, o âmbito dos problemas detectados tem um domínio limitado aos aspectos seleccionados.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

1 INTRODUCTION

This document was produced in the context of a Master's Thesis in Computer Engineering at Universidade do Minho in Portugal.

The theme is that of Creating Intelligible Metrics when performing Road Traffic Analysis. For the purpose of this thesis metrics are measurements of some kind, namely related to the context of Road Traffic. These measurements do not need to be numeric in nature and can in fact be obtained simply by measuring the presence or absence of a categorical feature.

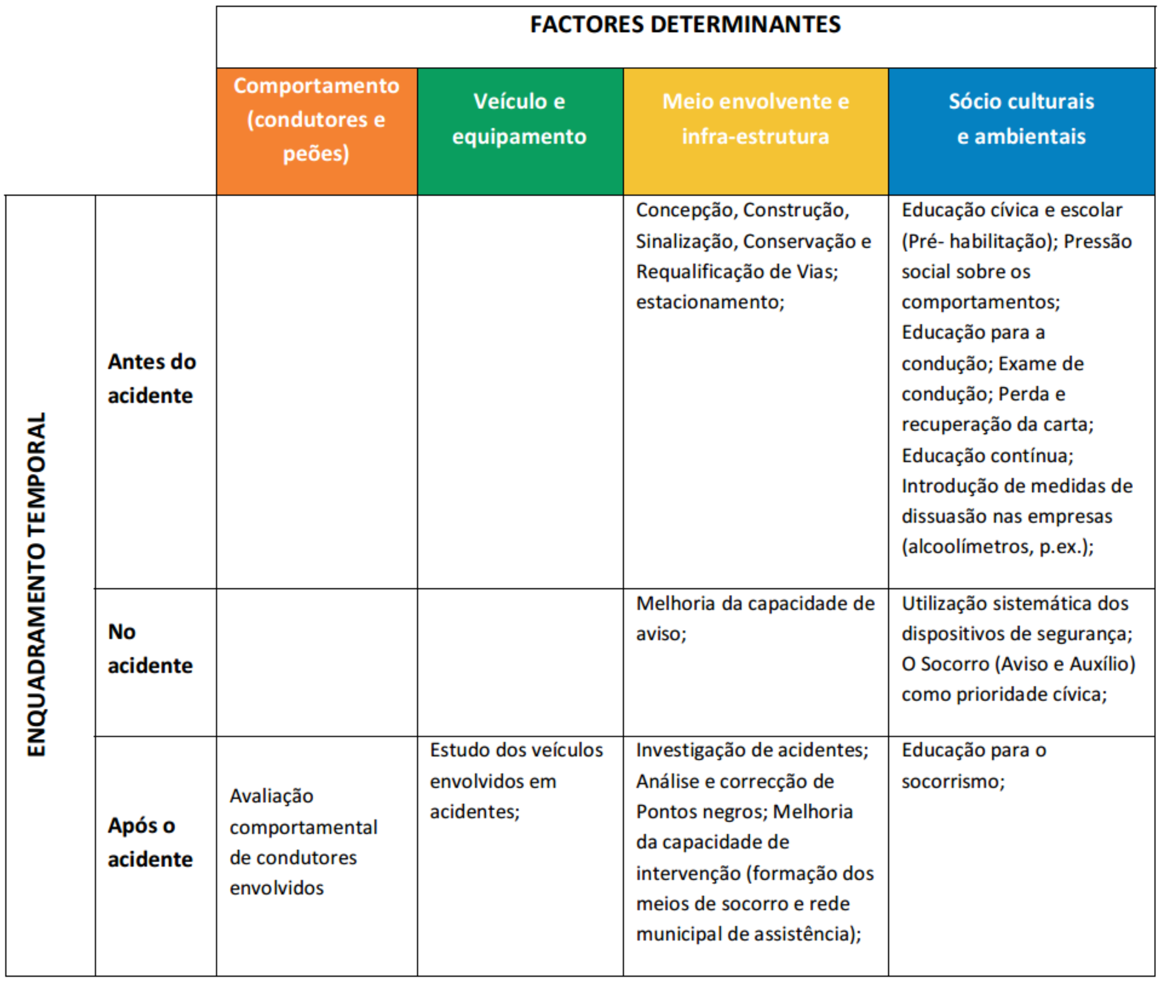
Metrics will be deemed intelligible when their appearance and comprehensibility are reasonably accessible to an observer, giving as such, importance to visualization and readability of the methods chosen. While human understandability is important, allowing an artificial agent to reason upon those metrics is also part of the vision behind this work.

The analysis performed concerns Road Traffic. This context is aggregated from several sources into a unique view for analysis. These sources add to the description of each object, thus a more complete view is created than it would be otherwise by providing a perspective over different aspects, adding up to a significant improvement in how each moment and place during a trip on the road can be described.

1.1 MOTIVATION

According to the Portuguese Ministry of Internal Administration, in a report published by the ANSR, which stands for Autoridade Nacional Segurança Rodoviária, concerning the period from the 1st of January to the 31st of December in 2015, there were 122,800 accidents on the road in Portugal. While the majority of accidents resulted in property damage alone, there were 37,958 minor injuries, 2,206 serious injuries, and 478 deaths.(ANSR, 2016) Given that these numbers have all increased, with the exception of deaths, in comparison to the same period in 2014, it seems clear that looking for solutions to improve road traffic is still a pressing matter.

In what concerns local planning, the Portuguese guide to the Planos Municipais de Segurança Rodoviária(PMSR) which refers to municipal planning, was released in 2009 by the ANSR(ANSR, 2009). This guide exposes a Haddon Matrix about when the municipal-



		FACTORES DETERMINANTES			
		Comportamento (condutores e peões)	Veículo e equipamento	Meio envolvente e infra-estrutura	Sócio culturais e ambientais
ENQUADRAMENTO TEMPORAL	Antes do acidente			Concepção, Construção, Sinalização, Conservação e Requalificação de Vias; estacionamento;	Educação cívica e escolar (Pré- habilitação); Pressão social sobre os comportamentos; Educação para a condução; Exame de condução; Perda e recuperação da carta; Educação contínua; Introdução de medidas de dissuasão nas empresas (alcoolímetros, p.ex.);
	No acidente			Melhoria da capacidade de aviso;	Utilização sistemática dos dispositivos de segurança; O Socorro (Aviso e Auxílio) como prioridade cívica;
	Após o acidente	Avaliação comportamental de condutores envolvidos	Estudo dos veículos envolvidos em acidentes;	Investigação de acidentes; Análise e correcção de Pontos negros; Melhoria da capacidade de intervenção (formação dos meios de socorro e rede municipal de assistência);	Educação para o socorrismo;

Figure 1.: Haddon Matrix presented in PMSR.

ities should intervene, in this matrix there's only advice regarding driver behaviour in the post-accident stage by advising the evaluation of the behaviour of the drivers involved. The work being presented relates to this very concept, it takes into account drivers' behaviour throughout whole trips, and since it does not do so live, it receives information post-event. Coupling such information with environmental factors, it then attempts to judge whether it is likely for there to be an issue of behaviour based on context. This may allow some insight into what leads to dangerous behaviour in the environment surrounding said behaviour.

1.2 OBJECTIVES

Transportation of goods and people using vehicles on roads is prevalent in today's society, it provides convenience and independence, but this is not without trade-offs. Some of those trade-offs that are of special interest to this dissertation, namely, the safety and comfort of people. This dissertation aims to study fusion of the smartphone gathered data together with outside sources such as geo referenced information like points of interest on the map, weather informations, and determining factors of sustainability. It is expected that the added structured information to the context will allow for new ways to automate the assessment of road traffic expression. The goal of this dissertation is therefore to experiment with aggregating external data with trip data gathered on a smartphone in order to generate meaningful and intelligible information. The hope is that such information can then be used to create mappings with metrics that can be justified. There will be an attempt to explore factors like neighbouring locations such as proximity to a bank or school. As well as when the driver is less prone to aggressive driver behaviour, depending on weather conditions and time of the day, week, or year.

The main goal of this work is to create richer information about road traffic, namely concerning locality and what sort of features can be learned and extracted regarding a position. This consists of obtaining intelligible metrics and fusing data from different sources into a single representation, as well as the use of proper machine learning techniques. To achieve this goal the following sub-objectives are of importance:

1. Study and application of Data Fusion.
2. Finding the appropriate Machine Learning Techniques.
3. Designing and studying a system capable of providing enriched information.
4. Creation of the designed system.

Data Fusion will be a crucial part of enriching information by fusing data and information from different sources and creating a more complete picture.

Machine Learning plays a very important role in automatically generating an analysis of the enriched data, finding the appropriate technique and properly delineating the problem are as such something this work will look into. This will be done taking into account the requirements that it needs to model the data with satisfiable accuracy, as well as, the need to provide intelligible information as a result.

A prototype system will be developed taking data from PHESS and demonstrating the added benefit of the system. The system will be created with the goal of making the automated analysis and enrichment of data, providing intelligible feedback as a service. An analysis of the system's capability will be included through testing for both performance and accuracy.

1.3 STRUCTURE

Structurally this document consists of seven main chapters each with several sections.

On the introductory chapter the topic of this dissertation is introduced, the document structure is described, followed by a section expanding on the motivation behind the theme of Road Traffic expression. This chapter is concluded with a section where objectives are detailed and the document structure.

The second chapter introduces the context where this work can be applied, namely smart cities, and then provides an overview of important items to the system that was developed. This consists of exploring Machine Learning methods and evaluation metrics.

The third chapter aims to explore the state of the art going over different studies that approach the subject of this dissertation from different perspectives both in terms of focus, as well as, the data types and methods used.

The fourth chapter explains the problems this works attempts to address and, then, explores some of its challenges and proposes a solution in the form of a system.

The fifth chapter documents a system created, what was created and how it was created. This chapter is divided in a few relevant sections, one that introduces decisions, these regard the data its sources and particularities as well as technologies used and some exploration that helped decisions, and then two more sections exists explaining the implementation of the two parts of software created, detailing structure, and the outcome of the implementation namely the endpoints and functionality that resulted.

The sixth chapter is dedicated to the experiments made. These experiments include a comparison between data features and the models that describe the data, and a simple performance experiment. The experimental setup is exposed in the due section, namely the working environment, physical hardware, and considerations as well as explanations of what was being tested. It also has a results and a discussion section where the results are exposed and discussed respectively.

There is one last chapter, the conclusion, with two sections, in the first section conclusions are made based on results obtained with the created system, as well as a summary of what objectives were fulfilled along with a general overview of what was achieved and what value was added. And the last section will discuss future work, namely optimizations and features that could be made, and that for one reason or another were not practical to make happen.

2

CONTEXT

This chapter will cover the context in which this work is framed, that of a Smart Cities, data fusion, and machine learning. The various aspects in providing support for urban life and development through technological means.

2.1 SMART CITIES

The Smart City is a concept that is defined differently throughout the literature. There are mainly two trends in how it is defined, a focus on a single urban aspect such as technology or ecology and definitions relating to the integration of the various urban aspects. Adopting the second trend, Smart City can therefore be described as a concept that refers to a technology based integration of both social and economic aspects of a city so as to maintain a sustainable and resilient development. (Monzon, 2015)

In Spain there are interesting Smart City projects occurring, an initiative headed by Endesa (Endesa.com, 2015a) a subsidiary of Enel showcases one of the two trends in Smart Cities. Enel has several Smart City projects with the goal of energy savings, this is a case of a one dimensional approach to Smart Cities, in Spain there are two testbed cities namely Málaga(Enel, 2013) and Barcelona(Endesa.com, 2015b). Málaga was able to reach its targets of 20% energy savings, and a reduction of CO2 emissions of 6,000-tonne per year (Endesa, 2014).

In the European Santander and Genova cities along with the Japanese cities of Mitaka and Fujisawa, the ClouT (*Cloud of Things for empowering the citizen clout in smart cities*) (Galache et al., 2014; Tei and Gurgen, 2014) project is an example of a citizen-centric multi aspect approach to a Smart City. It attempts to classify use cases of one of the three types they've found to be relevant after discussing with stakeholders, those types being: Smart city resource management, Safety and emergency management, and Citizen health and pleasant management (Yonezawa et al., 2015); an illustration of such can be found in Figure 2.

A concept related to the Smart City is the Internet of Things (IoT), the International Telecommunications Union (ITU) defines the Internet of Things as the infrastructure that

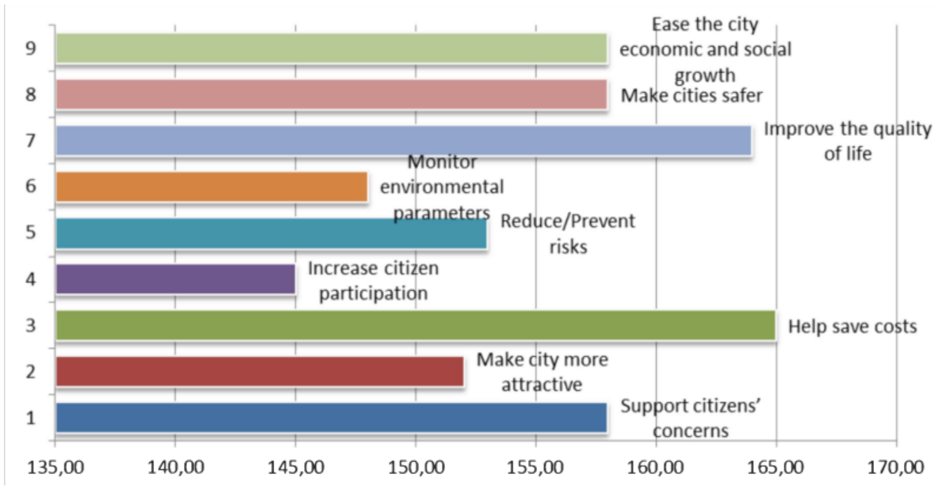


Figure 2.: Smart City needs as identified in a survey presented in Yonezawa et al. (2015).

enables advanced services by interconnecting physical and virtual things based on existing and evolving interoperable information and communication technologies. (itu, 2012)

The IoT plays an important role in shaping Smart Cities, it serves as a means to collect data to help integrate physical aspects of the world to information technologies that can leverage this into analysis, as well as, communication services.

As a consequence of the pervasiveness of networked devices and sensors, a lot of data may be collected making it an issue of dealing with Big Data. Big Data refers to data sets that are so large or complex that they demand an ability of data processing and analysis that goes beyond the typical database tools (Manyika et al., 2011). These tools are usually parallelizable and distributable over a myriad of machines to deal with the sheer volume of entities or their complexity. The theme of Big Data, while not the central issue of this thesis will accompany the work done throughout some of its stages where data volume and variety must be considerations.

A relevant case study of a Smart City, is one of the largest smart city experimental testbeds in the city of Santander in Spain. The project, named SmartSantander (SmartSantander.eu), deployed over 15000 sensors around the area of Santander so as to monitor the real-time state of the city. These sensors measure many environmental parameters such as light, temperature, and noise, as well as parameters like the occupancy of some parking slots.

Using data from sensor values can demand analytics to be performed over big data sets, to perform this, a platform named City Data and Analytics Platform(CiDAP) (Cheng et al., 2015) was created. This platform stores, processes, and analyzes the data generated by the SmartSantander project, focusing on dealing both with historical data and real time data. The choice of the NoSQL database was made due to the volume of Big Data that was needed

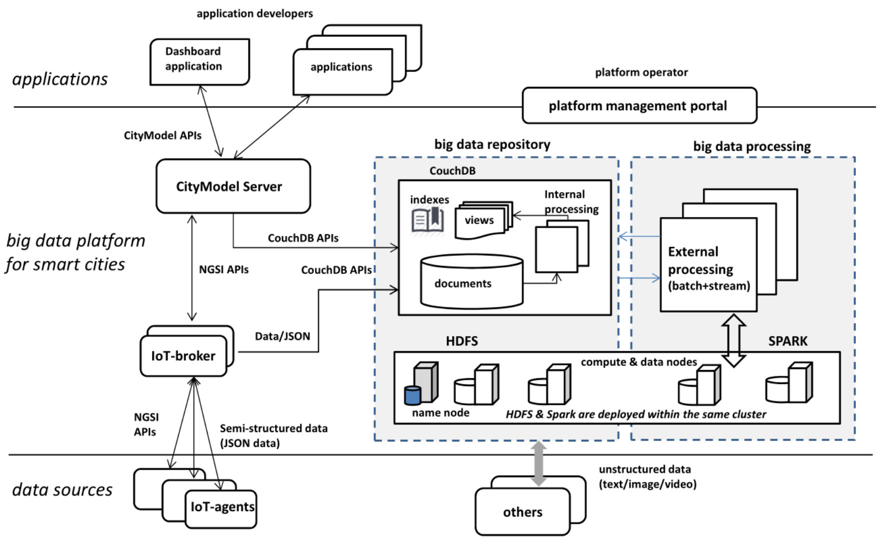


Figure 3.: System architecture of the CiDAP platform.

to store, as well as the incremental update views supported by CouchDB (CouchDB). Data analysis of a more complex nature, such as using machine learning methods, is made using Apache Spark (Spark). The system architecture of this platform can be seen in Figure 3.

Another interesting project that makes use of the Santander testbed is a project (Treboux et al., 2015) that attempts to answer the question: "Are the data from the Smart Santander consistent enough to predict a traffic jam and build a project to improve traffic management in the city?" the KNIME (Berthold et al., 2007) data-mining platform is used to explore this issue. The authors are able to point out traffic jams with a staggering 99.95% accuracy using multiple algorithms combining 3 prediction methods, namely Tree Ensemble, Fuzzy Rule, and Probabilistic Neural Network. In that solution the most important historical feature of data necessary to predict the traffic status is the rain.

Smart city plans are becoming more and more pervasive, in Portugal according to Vodafone (2016), using technological solutions the Sabugueiro village has managed energy savings of close to 20% in the domestic environment and 9% in the public network. Other relevant aspects mentioned are water savings of 12% and remote monitoring of life signs.

It is visible that several cities in different countries are starting to implement or design their own smart city plans and initiatives, that fact coupled with the results of such approaches show the increasing importance of this concept. Smart Cities were, therefore, considered to present an interesting context in which to fit this work, and some aspects of the created system were inspired by some of the aforementioned Smart City projects.

2.2 DATA FUSION

According to Hall and Llinas (1997), it can be said that "*Data fusion techniques combine data from multiple sensors, and related information from associated databases, to achieve improved accuracies and more specific inferences than could be achieved by the use of a single sensor alone.*". While Lahat et al. (2015) frames data fusion as "*the analysis of several datasets such that different datasets can interact and inform each other*".

Data fusion, therefore, plays an important role in this work, that of enriching the data. The foundational model of data fusion seems to originate from the JDL (Joint Directors of Laboratories). Their work divided the task of Data Fusion into five numbered levels and another pre-processing one as is visible in 4 as depicted in Hall and McMullen (2004). The authors also review the concepts and levels taken into account.

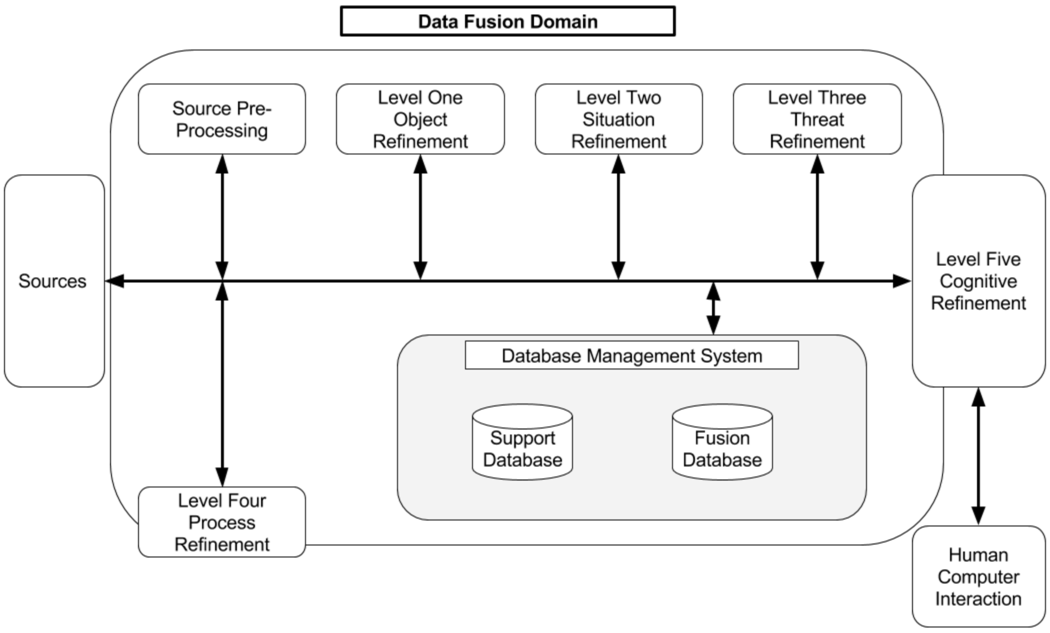

Data Fusion Domain
Source Pre-Processing
Level One Object Refinement
Level Two Situation Refinement
Level Three Threat Refinement
Sources
Level Five Cognitive Refinement
Database Management System
Support Database
Fusion Database
Level Four Process Refinement
Human Computer Interaction

Figure 4.: JDL Data Fusion Model

The DFIG (Data Fusion Information Group) later proposed an updated model as can be seen in figure 5.

For this dissertation the most relevant levels are zero through three, from Blasch et al. (2012) they can be summarized as levels:

LEVEL ZERO - Data assessment, estimation and prediction of signal/object observable states on the basis of pixel/signal level data association.

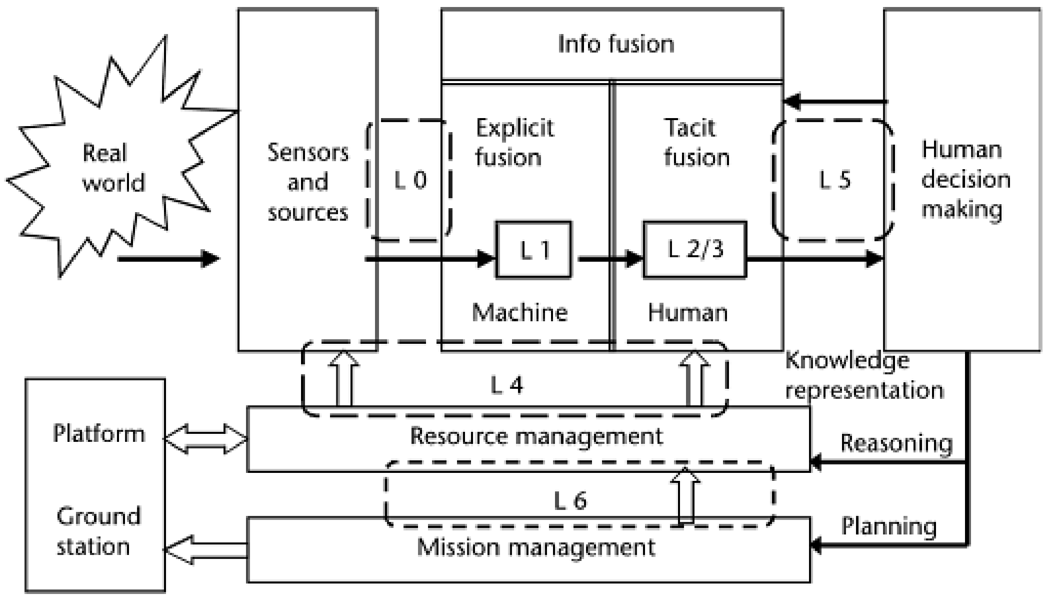


Figure 5.: DFIG Data Fusion Model

LEVEL ONE - Object assessment, estimation and prediction of entity states on the basis of data association, continuous state estimation, and discrete state estimation.

LEVEL TWO - Situation assessment, estimation and prediction of relations among entities, to include force structure and force relations, communications, and so forth.

LEVEL THREE - Impact assessment, estimation and prediction of effects on situations planned or estimated actions by the participants; to include interactions between action plans of multiple players.

In this thesis both low level and high level fusion will take place, the first will be present in the initial stage of aggregating all data inputs consistently. It corresponds to the first two levels of the DFIG model presented in figure 5. While high level fusion concerns levels two and three of the DFIG model, and it describes the later stages of this project.

In Blasch et al. (2012) an intuitive way of looking at this division is presented, by making a comparison between human situation awareness (SAW) and machine information fusion (MIF). This can be seen in figure 6.

Data fusion is used in several fields and should become increasingly apparent there are several different ways of understanding it,Castanedo (2013) the Input/Output (I/O) based characterization presented in Dasarathy (1997) is a popular means of understanding it. This classification takes into account five categories, these are: data in-data out (DAI-DAO), data in-feature out (DAI-FEO), feature in-feature out (FEI-FEO), feature in-decision out

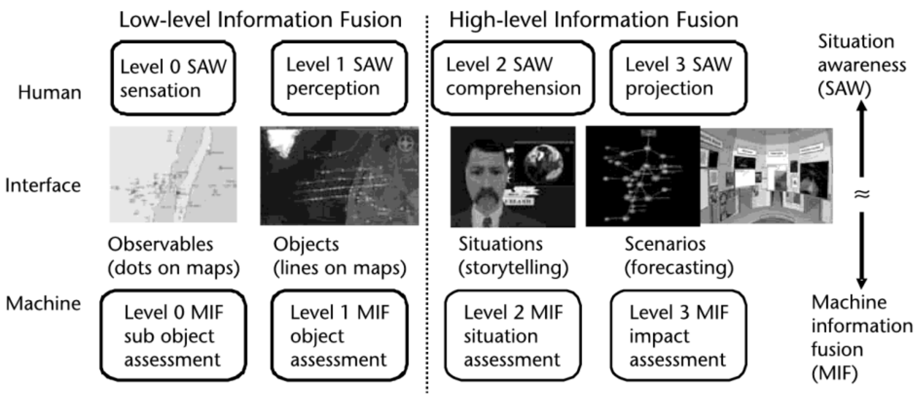

Low-level Information Fusion
High-level Information Fusion
Human
Level 0 SAW sensation
Level 1 SAW perception
Level 2 SAW comprehension
Level 3 SAW projection
Situation awareness (SAW)
Interface
≈
Observables (dots on maps)
Objects (lines on maps)
Situations (storytelling)
Scenarios (forecasting)
Machine
Level 0 MIF sub object assessment
Level 1 MIF object assessment
Level 2 MIF situation assessment
Level 3 MIF impact assessment
Machine information fusion (MIF)

Figure 6.: Comparison between SAW and MIF.

(FEI-DEO), and decision in-decision out (DEI-DEO). Where data refers to raw data from sensors and other sources, feature refers to characteristics of the entity being observed, and decision refers to results such as a class in classification or prediction in a prediction.

This last understanding was found to be intuitive and making for an easy way to interpret the created solution. Where GPS data leads to features such as date time from the system time of a record, and roads on a road network from the GPS points are extracted. It's also easy to understand that the analysis server is capable of making the decision whether a situation is expected to be problematic or otherwise. The various categories of I/O are as such intuitively discernible throughout the work done, and which will be presented in Chapters 4 and 5.

2.3 MACHINE LEARNING

Machine learning is a vast field within computer science, it is deeply connected with statistics, and dependent on data to serve as experience. Its usage ranges from prediction of future values, to classification of entries, to clustering of common features, or even the discovery of the underlying relations in the data.

> "The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience." - Mitchell (1997)

Norvig and Russell say that learning occurs if the performance of future tasks are improved after making observations about the world (Russell and Norvig, 2010). In their book they go on to distinguish four major factors on which a component of an agent depends on to improve. These factors are: the component, prior knowledge, representation of data, and available feedback.

Representation and prior knowledge comes down to a factored representation, that is a vector with attribute values both numerical and discrete, so as to enable the use of methods implemented in external libraries. Prior knowledge will be applied directly as filters over the dataset, but it is not otherwise represented in some form of logic within the system.

The goal of our system is to be able to learn how in road traffic a driver's behaviour varies with external factors, and present such in a human comprehensible form. This can be summarized as an instance of inductive learning since there is no prior knowledge that allows for the system to deduce the rules governing this behaviour.

We are therefore left with one last factor, one which may be considered the main factor used to distinguish machine learning methods, this is the matter of what feedback is available to learn from.

In accordance to this factor Machine Learning is commonly divided into three categories, Supervised Learning, Reinforcement Learning, and Unsupervised Learning.(Duda et al., 2001; Russell and Norvig, 2010) The term Semi-Supervised Learning is also in use and it characterizes a middle ground between supervised and unsupervised learning(Zhu and Goldberg, 2009; Russell and Norvig, 2010; Søgaard, 2013).

UNSUPERVISED LEARNING In Unsupervised Learning there is no explicit feedback or teacher, the goal is to learn patterns on the input data without such supervision. According to Zhu and Goldberg (2009) common Unsupervised Learning tasks include:

- **Clustering**, the task of dividing data into groups.
- **Novelty detection**, the task of identifying instances that are very different from most.
- **Dimensionality Reduction**, the task of representing each entry in a feature vector with less dimensions while maintaining the key characteristics.

REINFORCEMENT LEARNING Reinforcement Learning happens when rewards or punishments are the available feedback. This feedback only informs as to whether a result is right or wrong, it is then up to the learner to discern how or what actions prior to the feedback are right or wrong.

SUPERVISED LEARNING In Supervised Learning each data entry is composed of two aspects, the inputs and the outputs(Friedman et al., 2001). The inputs concern the information that is measured and fed to the machine learning method, and the outputs are what one is attempting to predict or classify based on the inputs. In the literature inputs can also have other names such as predictors, independent variables, and features. Analogously the outputs can be called responses, dependent variables, and labels.

Supervised Learning is a form of Inductive Learning, where the goal is to pick up an example data set, sometimes referred to as the training data, and create a model that is capable of making a generalization of the function that maps features into labels. In this manner new data can be labelled according to the same patterns that were present in the example data labelling.

This is the type employed in the execution of this work, where a vector of features is created from the aggregated context and the values of driving behaviour score retrieved are used as classifications.

There are many methods and tasks concerning machine learning, in this section we'll talk about some of the explored ones so as to create a context of what can be done and why some methods were more or less proper to this work's use case, and lastly justifying why Decision Trees were the model of choice to the developed system.

In order to properly contextualize this choice, it is relevant to showcase some of the decisions that had been made up to that point regarding the requirements of the learning method:

1. A binary classification will be used to discern between problematic or normal entries.
2. What contributes to a classification must be human comprehensible.
3. Features are bound to include attributes both of a quantitative nature as well as qualitative nature.

2.3.1 *Linear and Logistic Regression Analysis*

According to Seber and Lee (2012) the aim of regression analysis is to construct mathematical models which describe or explain relationships that may exist between variables.

The simple Linear Regression model is composed by the mean and variance functions(Weisberg, 2005):

$$E(Y|X = x) = \beta_0 + \beta_1 x$$
$$Var(Y|X = x) = \sigma^2$$

Constructing the model therefore consists of finding out the β_0, β_1, and σ parameters. The σ is often called the standard error of regression(Weisberg, 2005), and $\beta_0 + \beta_1 x$ is a line of mean values(Montgomery et al., 2012). A representation of a Linear Regression Model can be seen in Figure 7.

Still concerning regression analysis there's another method that was looked into, Logistic Regression.

This is a mathematical modelling approach that can describe the relationship between several inputs and a dichotomous output. This seems to fit properly with the requirements

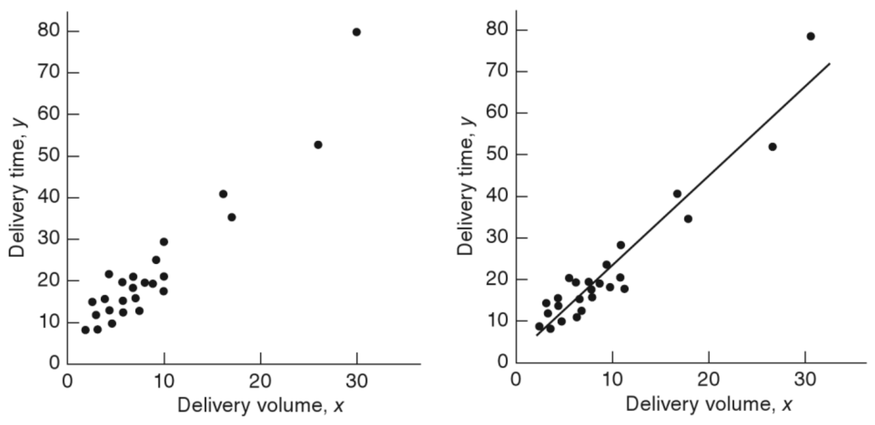


Figure 7.: Scatter plot and a line representing a linear regression model without the error from Montgomery et al. (2012).

at 1. The model is designed to describe a probability, i.e. a number between 0 and 1(Kleinbaum and Klein, 2010), and can be represented as:

$$P(D = 1|X_1, X_2, ..., X_k) = \frac{1}{1 + e^{-(\alpha + \sum \beta_i X_i)}}$$

Where α and β_i are the unknown parameters to be estimated in this model.

And while both Linear and Logistic Regression can handle the necessary forms of inputs and outputs, even if done through the use of some coding system, like dummy coding, it does not however provide a good way for a human to comprehend the causes of such classification since the classification is reached as a result of the respective functions alone, thus failing to meet the criteria in 2.

2.3.2 *Artificial Neural Networks (ANN)*

Artificial Neural Networks(ANN) are biologically inspired(Yegnanarayana, 2009), specifically by neuroscience(Hassoun, 1995), they are an attempt to model the information processing capabilities of the nervous system(Rojas and Feldman, 2013).

This popular machine learning approach is viable for a number of problems like pattern classification, speech synthesis and recognition, adaptive interfaces between humans and complex physical systems, function approximation, image compression, associative memory, clustering, forecasting and prediction, combinatorial optimization, nonlinear system modelling, and control(Hassoun, 1995).

The basic building block unit of an artificial neural network is the "artificial neuron" (Hassoun, 1995), this is named differently throughout the literature, because neurons are

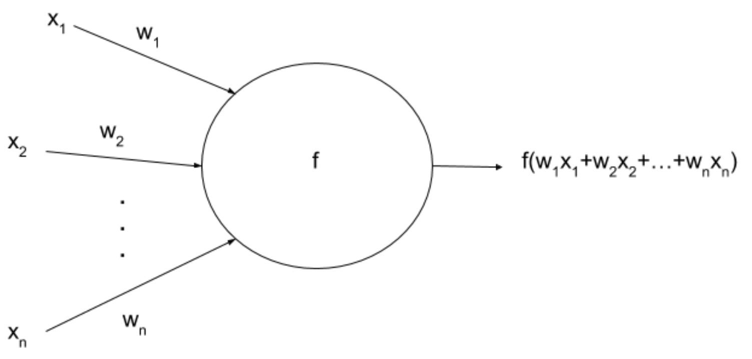


Figure 8.: Abstract representation of a neuron.

not fully understood and because each biological neuron is much more complex than their artificial counter parts, these building blocks are referred to as the "computing units" of artificial neural networks by some (Rojas and Feldman, 2013), while others name it Processing Units (Yegnanarayana, 2009) or Processing Elements (Rabuñal, 2005).

According to Rojas and Feldman (2013) in abstract terms, a neuron is structured as shown in Figure 8. The figure shows an abstract neuron with n inputs, where each i input channel transmits a real value x_i. The input channels usually have a weight associated, illustrated by w_i for each i input channel, these weights correspond to a multiplier applied to the x_i input. The information transmitted to the neuron through the input channels is then integrated, usually by summation, and the function f is evaluated accordingly.

The usefulness of an artificial neural network depends on the processing units being organised in a suitable manner to accomplish a given pattern recognition task, this organization is known in the literature as topology.(Yegnanarayana, 2009)

According to Costa and Simões (2008) in an artificial neural network, when a set of neurons is unconnected between its members, and has both a set of input and a set of output such nodes, this is known as a layer. There are three common types of layers, input layers which receive information from outside the network, output layers which transmit information to outside the network, and hidden layers which are not directly connected to the outside of the network.

Some authors like Yegnanarayana (2009) define layers as a set of processing units which have the same activation dynamics and output function, but that there can be interlayer and intralayer connections, i.e. connections between processing units in the same layer or different layers.

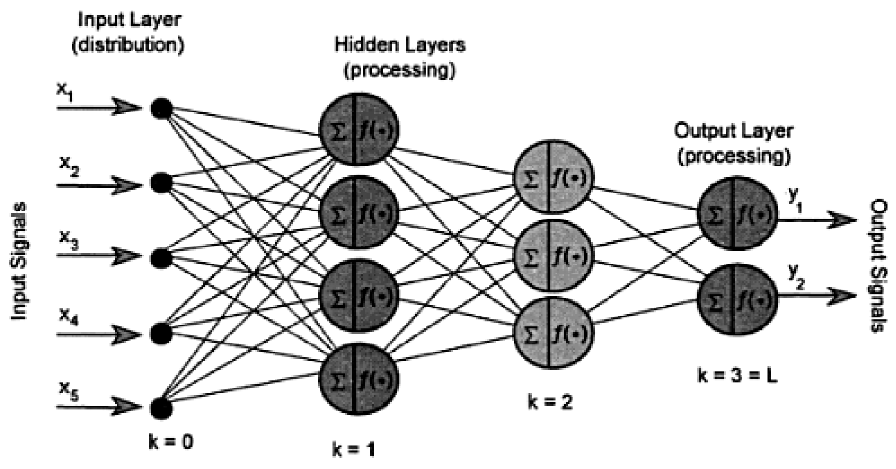


Figure 9.: Example of an ANN with a feedforward topology according to Priddy and Keller (2005).

In a neural networks adjusting the weights of connections between processing units, and in some cases the topology, can be considered the machine learning part of the process.

While technically Neural Networks can compute discrete inputs, this requires processes like dummy coding (Pham, 2006), fortunately the shape of our data makes it so that categorical features are by and large binary flags representing the presence of certain locations in the surrounding areas, dummy coding requires an amount of numerical features that is exactly the number of categories minus one in order to represent the original categorical feature (Hardy and Bryman, 2009) which ends up being the exact same number of features since they are composed of only two categories. The why and how of these binary flags are discussed ahead in the Chapter 4.3.

Normalizing data values beforehand is also an important step when dealing with differently ranged attributes and neural networks, so as to properly convey the contribution of an input in relation to others (Priddy and Keller, 2005). Furthermore normalizing data in a neural network tends to improve performance (Yu et al., 2010). Normalization was not deemed prohibitive, since it is an operation that can easily take advantage of the distributed environment provided by the Apache Spark engine. Similarly scaling may be necessary so that the range of the data is neither too small nor too large which may lead to issues with the precision limits (Yu et al., 2010).

Ultimately the issue with Neural Networks in the context of this work is that they work in a black box manner and, just like with linear and logistic regression, makes it hard for one to comprehend the specific why of a result other than understanding how the training or model itself works. This ends up being in conflict with 2, and was therefore not the method of choice.

Nevertheless Artificial Neural Networks are worth of consideration for future work, while tree-based methods are readily interpretable, ANNs commonly outperform them (Pham, 2006). In this instance this was a necessary trade-off for intelligibility.

2.3.3 *Support Vector Machine (SVM)*

The Support Vector Machine method was introduced by Vapnik and co-workers (Vapnik et al., 1992; Cortes and Vapnik, 1995; Vapnik, 1995), according to Vapnik (2013), the 2013 edition of Vapnik (1995), the support vector machine maps the input vectors to a high-dimensional feature space through some non-linear mapping chosen a priori. In this space an optimal separating hyperplane is constructed. A simple example of this is represented in Figure 10.

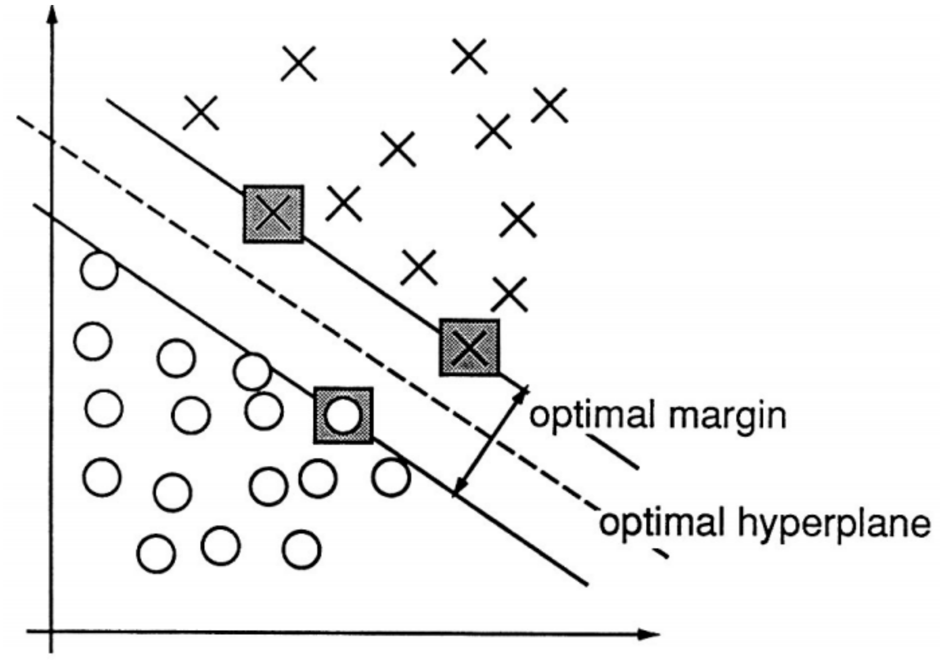


Figure 10.: An example of a separable problem in a 2 dimensional space. The grey squares are the support vectors that define the margin of largest separation between the two classes.(Cortes and Vapnik, 1995)

According to Suykens and Vandewalle (1999), given a training set of N data points $\{y_k, x_k\}_{k=1}^{N}$ with $x_k \in \mathbb{R}^n$ as the k-th input pattern and $y_k \in \mathbb{R}$ as the k-th output pattern, then the support vector method approach aims at constructing a classifier of the form given in Equation 1:

$$y(x) = sign\left[\sum_{k=1}^{N} \alpha_k * y_k * \psi(x, x_k) + b\right] \tag{1}$$

Where α_k are positive constants in $\mathbb{R}$ and b is a real constant. The ψ is a formula that typically characterizes either a linear SVM as described in Equation 2, a polinomial SVM of degree d as found in Equation 3, a RBF SVM shown in Equation 4 where σ is a constant, or a two layer neural SVM described in Equation 5 where κ and θ are constants.(Suykens and Vandewalle, 1999)

$$\psi(x, x_k) = x_k^T * x \tag{2}$$

$$\psi(x, x_k) = (x_k^T * x + 1)^d \tag{3}$$

$$\psi(x, x_k) = \exp\left(\frac{-\|x - x_k\|_2^2}{\sigma^2}\right) \tag{4}$$

$$\psi(x, x_k) = \tanh[\kappa * x_k^T * x + \theta] \tag{5}$$

Similar to the aforementioned methods, this too doesn't appear to have a very intuitive way of ascertaining what are the issues causing a problematic classification, so it ended up not being the method of choice. Nevertheless its performance was tested for comparison reasons. The results are available in Chapter 5.

2.3.4 *Decision Trees (DT)*

According to Barros et al. (2015) Decision Trees have been studied in the context of many disciplines, from engineering, statistics, and decision theory to machine learning more recently. The latter is the context for the application of Decision Tree Models in this work. Grabczewski (2013) states that DTs are often regarded as attractive approaches. One of the most important causes for such is their comprehensibility, and that they can be expressed in the form of a set of logical rules describing the decision functions. These aspects played an important role in the choice of this method for this work.

A decision tree is a considered a method which is both nonparametric and efficient, and this method allows for both regression and classification.(Chikalov, 2011) If the output of the Decision Tree is continuous, then it is a regression tree, if the output is discrete then it is considered a classification tree.(Suthaharan, 2015)

A Decision Tree is composed of nodes that form a Rooted Tree, i.e. a Directed Tree which has a root node with no incoming edges. Every node but the root has exactly a single incoming edge. Besides the root node there are two more node types, these are the test

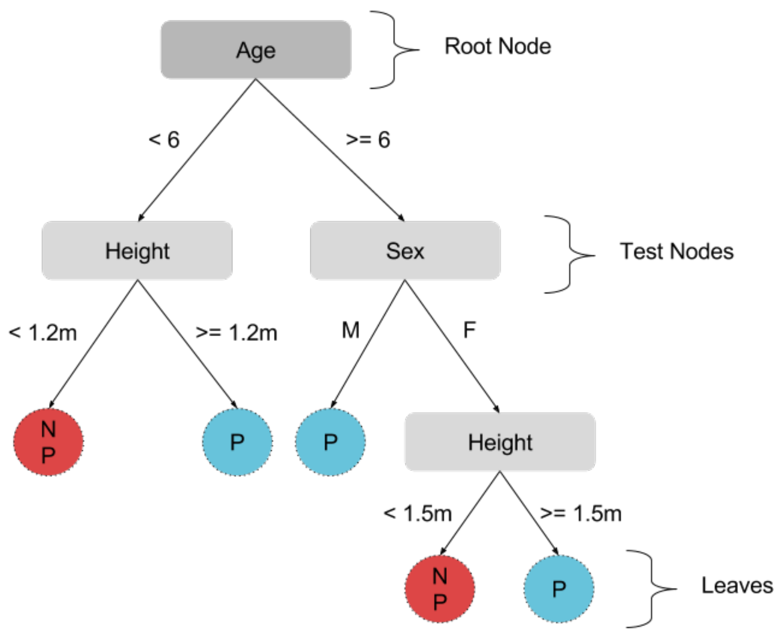


Figure 11.: Illustration of a Classification Tree determining whether someone has ever played basketball.

nodes and leaves, and can be distinguished by whether or not they have outgoing edges. If they have such outgoing edges then they are a test or internal node, and otherwise they are leaves or terminal nodes.(Rokach and Maimon, 2014) An illustration of a Decision Tree and this distinction of nodes can be seen in Figure 11.

As a classifier a decision tree works by recursively partitioning the domain, each test node splits the domain of a feature or set of features into a set of sub-domains that is both complete and whose members are mutually exclusive. These two properties are very important to ensure that every new instance can be classified, and that it is classified at exactly one node alone. Thus avoiding ambiguous results or lack of results when running an instance through the classifier.(Rokach and Maimon, 2014)

After explaining the form of a Decision Tree model and some of its properties it is of relevance to further explore how such model is learned from data, Hyafil and Rivest (1976) demonstrated that constructing an optimal binary tree is an NP-complete problem.

Consequently because the search space for the optimal tree is large, in practice Decision Trees are constructed with heuristic search methods.(Grabczewski, 2013)

As can be seen in Figure 12 the steps to growing a decision tree can be summarized as:

Decision trees are hierarchical and as can be seen from Figure 11 easily interpretable. They also allow for one to trace back any prediction to a set of rules or conditions corre-

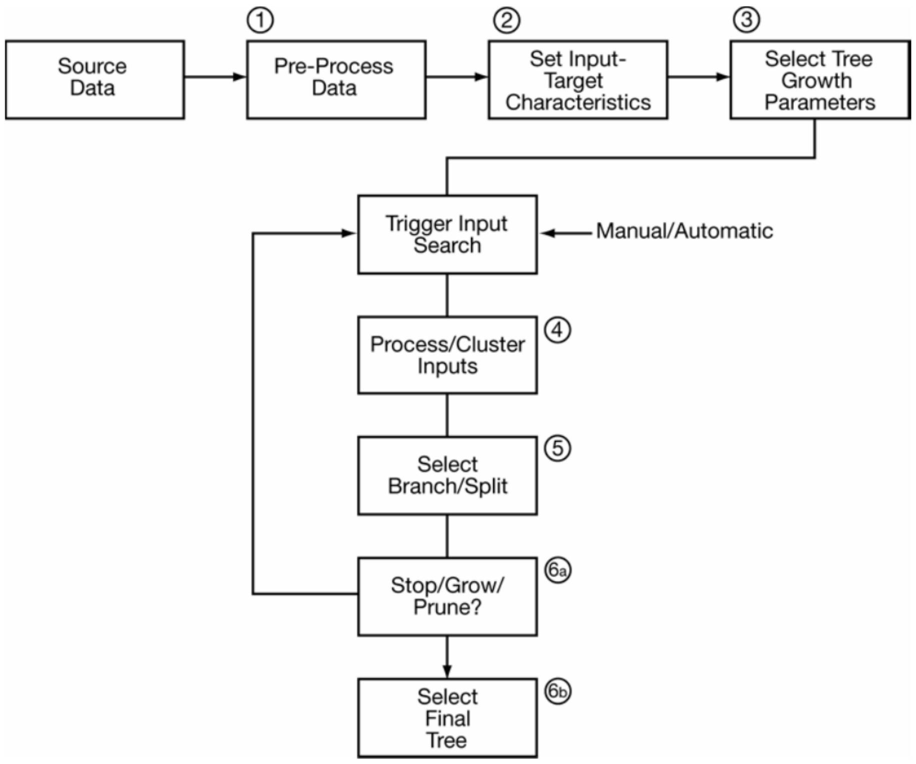


Figure 12.: Decision Tree growth sub processes according to De Ville (2006).

sponding to the splits of the tree up to the leaf where the prediction value can be found. An example would be in Figure 11, imagining that John Doe is 8 years old a male and is 1.3m high, that we can predict he has played basketball and explain the reasons for this conclusion in a very approachable way by naming the two conditions that lead to this prediction, that his age is 6 or higher and he is a male.

This ended up being the method of choice during this work because it fits all of the criteria, and was readily available in the tool chain being used. The performance and results of this method both on its own as well as compared to some of the others are presented in the Tests and Results section.

2.3.5 *Naive Bayes*

According to Amor et al. (2004), Naive Bayes are very simple Bayes Networks, which means they are composed of a directed acyclic graph. The Naive Bayes classifier is considered to be fast and easy to implement(Rennie et al., 2003). Zhang (2004) states that the Naive Bayes classifier for two classes can be formulated as 9. Where E is an example tuple of attribute values of the form $E = (x_1, x_2, ..., x_n)$, and $C = +$ is the positive class and $C = -$ is the negative class. And is obtained from the Bayesian Rule showcased in 6, while assuming that E is only classified as $C = +$ given that 7 is the case, and assuming the attribute variables are independent so that equation 8 is the case.

$$p(c|E) = \frac{p(E|c) * p(c)}{p(E)} \tag{6}$$

$$f_b(E) = \frac{p(C = +|E)}{p(C = -|E)} \geq 1 \tag{7}$$

$$p(E|c) = p(x_1, x_2, ..., x_n|c) = \prod_{i=1}^{n} p(x_i|c) \tag{8}$$

$$f_{nb}(E) = \frac{p(C = +)}{p(C = -)} \prod_{i=1}^{n} \frac{p(x_i|C = +)}{p(x_i|C = -)} \tag{9}$$

In Amor et al. (2004) the authors compare the performance of Naive Bayes and Decision Tree, concluding that Decision Trees perform generally better, but that Naive Bayes is generally 7 times faster at learning and classifying then Decision Trees.

This method can be represented in a fairly intuitive manner by showcasing how much each attribute contributes to the probability, but it wasn't chosen because it performed worse than Decision Trees and the presentation of the Decision Tree was still deemed a more intuitive representation, and therefore a better fit to the project being developed. Nevertheless, because it did fit all the criteria and because it was feasible, a comparison of performance was made and will be presented in the results section.

2.3.6 *Evaluation Metrics*

After choosing a method, acquiring some form of result is trivial, but in order to obtain some guarantee of usefulness one has to explore how to evaluate the model generated by the chosen machine learning method.

A model can be evaluated from many perspectives, some forms of evaluation are more objective and others are more subjective, in choosing a method so far the aspects taken into account throughout this document are of a subjective evaluation nature, the focus being on their intelligibility. This was a requirement of the work being done and prioritized over other aspects or results.

In this section the focus will be on evaluation metrics that describe how well the model predicts, there are also other objective aspects like, for example, the training and the prediction performance in both time and memory space.

Error is a relevant metric for evaluation and can be calculated in many forms. Two error metrics used to estimated the deviation from expected values are the Mean Absolute

Percentage Error(MAPE) and Root Mean Squared Error(RMSE), they can be formulated as equations 10 and 11. (Saiprasert and Pattara-Atikom, 2013)

$$e_{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{s_o - s_p}{s_o}\right| \tag{10}$$

$$e_{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n}(s_o - s_p)^2} \tag{11}$$

There are other means of evaluating the results, within our scope it is relevant to speak of those that target a binary classifier. The data is processed into two classes in this work, this process is mentioned in Chapter 4.3.

Since the data is now classified as problematic or normal, the evaluation can be made by comparing percentages of key statistics. For this there are some key concepts, when evaluating a binary classifier one can designate one of the output categories to be positive and the other to be negative, in this work the most important task is to find issues, therefore the problematic category is considered a positive or a hit, the normal category is considered a negative or a miss. Given this, there are four possibilities for any classified entry, it is a true positive(TP) if it was classified as positive and it is also positive in actuality, it is a true negative(TN) if it was classified as negative and it is also a negative in actuality, it is a false positive(FP) if it was classified as positive but in actuality it is not positive, it is a false negative(FN) if it was classified as negative but in actuality it is not negative. These are illustrated in Table 1.

Table 1.: Confusion matrix.(Davis and Goadrich, 2006)

	Actual Positive	**Actual Negative**
Predicted Positive	TP	FP
Predicted Negative	TN	FN

There a few key metrics under this concept, in Powers (2011) an overview of these metrics as well as some of the common names for them is made. One such metric which was considered as having a pivotal role in this work is Precision, this is also known as Confidence in Data Mining, can be described as the True Positive Accuracy(Powers, 2011; Davis and Goadrich, 2006) and is referred in Jiao et al. (2014) as Selectivity. Precision assesses the predictive power of the algorithm.(Sokolova et al., 2006)

Another commonly used metric is Recall, in the literature this can be named as True Positive Rate or as Sensitivity(Jiao et al., 2014; Powers, 2011; Davis and Goadrich, 2006). Recall approximates the probability of the positive prediction being correct.(Sokolova et al., 2006)

Thirdly is the notion of Accuracy(Sokolova et al., 2006) or Agreement(Jiao et al., 2014) which is the probability of a prediction being correct(Sokolova et al., 2006).

Specificity, also referred in literature as Inverse Recall or True Negative Rate is the proportion of actual negative instances that are correctly predicted as negative.(Powers, 2011)

Inverse Precision or True Negative Accuracy, is the proportion of predicted negative instances that are actual negative instances.(Powers, 2011)

The F-score or F-measure is a value that benefits algorithms with higher Recall and challenges algorithms with higher Specificity.(Sokolova et al., 2006)

These values can be formulated as follows:

$$
\begin{aligned}
Precision &= \frac{TP}{TP+FP} \\
Recall &= \frac{TP}{TP+FN} \\
Accuracy &= \frac{TP+TN}{TP+FP+FN+TN} \\
Specificity &= \frac{TN}{FP+TN} \\
InversePrecision &= \frac{TN}{FN+TN} \\
F-measure &= \frac{(\beta^2+1)*Precision*Recall}{\beta^2*Precision+Recall}
\end{aligned}
$$

The Receiver Operator Characteristic(ROC) is considered a way of obtaining comprehensive evaluation classifier performance, it is a curve that shows the relation between Recall and Specificity.(Sokolova et al., 2006) The Area Under the Curve(AUC) is a single scalar which can be calculated using the trapezoidal areas created between each ROC point(Davis and Goadrich, 2006).

According to Sokolova et al. (2006) Balanced Accuracy can be formulated as shown in 12.

$$BalancedAccuracy = \frac{Recall+Specificity}{2} \tag{12}$$

Another metric is the Youden's index, introduced by Youden (1950), according to Sokolova et al. (2006) it attempts to evaluate the algorithm's ability to avoid failure and is given by either the equation 13 or 14.

$$\gamma = Recall-(1-Specificity) \tag{13}$$

$$\gamma = 2*(BalancedAccuracy)-1 \tag{14}$$

Another way to evaluate the classifier is by using Likelihoods which can be positive or negative. A higher positive likelihood means a better performance on the positive class while a lower negative likelihood means a better performance on the negative class. The positive and negative Likelihoods can respectively be described as the equations 15 and 16.(Sokolova et al., 2006)

$$\rho_{+} = \frac{Recall}{1 - Specificity} \tag{15}$$

$$\rho_{-} = \frac{1 - Recall}{Specificity} \tag{16}$$

Lastly is the Discriminant Power, this measure summarizes Recall and Specificity, it attempts to evaluate how well an algorithm distinguishes between positive and negative examples. The formula is given in Equation 17.(Sokolova et al., 2006)

$$DP = \frac{\sqrt{3}}{\Pi} * (\log\left(\frac{Recall}{Recall - 1}\right) + \log\left(\frac{Specificity}{1 - Specificity}\right)) \tag{17}$$

A classifying algorithm is considered a poor discriminant if the value of the Discriminant Power(DP) can be characterized as $DP < 1$, limited if $DP < 2$, fair if $DP < 3$, and good otherwise.(Sokolova et al., 2006)

3

STATE OF THE ART

One of the central topics for this work is Driving and Road Traffic Expression, this section will go over several approaches focusing in the context of information technologies including an overview of the PHESS Driving platform which is one of the most important sources used in the system created.

3.1 DRIVER STATE

There are many ways in which to judge driving performance, it can be related to how a specific resource is consumed like money or time, or other aspects like what characteristics belong to the trip in question like pleasantness or safety of the passengers.

In this area there are several studies that focus on the state of the driver, one such study is Tango and Botta (2013) which focuses on driver visual distraction and its real time detection using machine learning techniques.

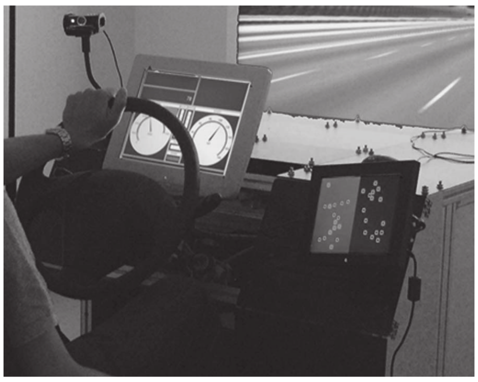


Figure 13.: Surrogate Visual Research Task on the right part of the driving simulator cockpit in Tango and Botta (2013).

This study, Tango and Botta (2013), presents the notion that driver distraction even if not always consistently defined throughout the literature, has been found to be a leading cause of vehicle crashes and incidents in different studies. Given that notion it attempts to model driver distraction based on the following features: speed, time to collision, steering angle, lateral position, position of the accelerator pedal, and position of the brake pedal. Using SVM, the same method explained in Section2.3.3, both feed forward and layer recurrent Neural Networks, the method presented in Section2.3.2, and adaptive neurofuzzy inference systems (ANFIS) introduced by Jang (1993). Visual data was used solely for labelling purposes by the experimenter, i.e. it was not used as a feature on any of the machine learning models and the setup can be seen in Figure 13. The SVM model using a radial basis function (RBF) as a kernel, explained in Equation 4, was found to produce results with better performance on the evaluation metrics considered which were the Correct Rate which is the percentage of correct classifications, Sensitivity as discussed previously in this document, and Specificity which is the percentage of correctly classified negative instances, scoring over 90% in each of those metrics in average.

Human error was also the object of the study presented in Jiao et al. (2014), more specifically the target issue in this study is Driver Fatigue recognition through Slow Eye Movement (SEM). The study starts by recognizing that SEM has been proved to be a reliable indicator of sleep in a number of other studies using data obtained from an electro-oculogram in a driving simulator for over two hours. Using the test environment shown on Figure 14 the authors collected data and compared three machine learning techniques, more specifically, SVM, GELM (discriminative graph regularized Extreme Learning Machine), and KNN (K-Nearest Neighbours). The different techniques had for various sets of features and differing samplings (over and under sampling) varying relative performances among themselves on the evaluation metrics of Agreement, Sensitivity, and Selectivity.

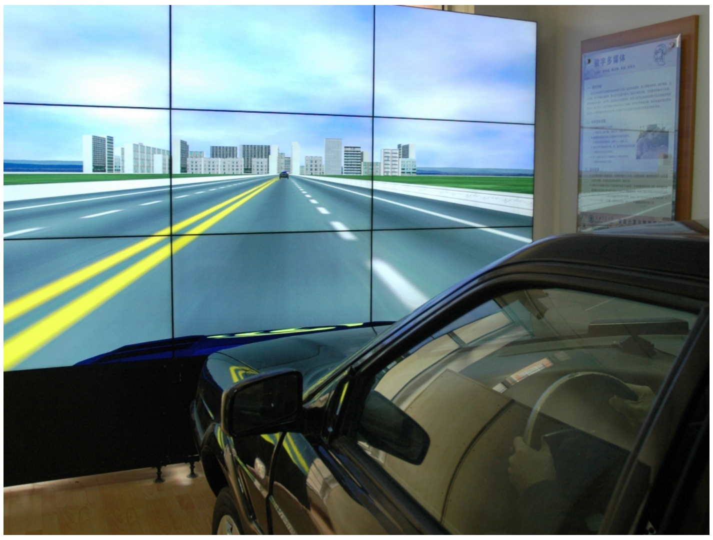


Figure 14.: Driving Simulation Environment in Jiao et al. (2014).

In a similar vein Yoshida et al. (2014) attempts to characterize a driver's cognitive state in real driving situations. The prediction of the cognitive condition, is made using a grouping of parameters deemed the driver's action and state of the car:

- longitudinal and lateral control information, i.e. changes in acceleration, braking, or turning
- gear selection information, referring to changes e.g. 1 up, or stationary
- The person's gaze target information, like the duration of the gaze and what is being gazed, e.g. rearview mirror
- The car's state, which is comprised of items like speed, acceleration rate, steering angle, and gear number.

Vehicle information was gathered using a Controlled Area Network(CAN), which is an in-vehicle LAN used to gather driving data.

This study also used eye movement tracking for classification purposes, this includes the variance of eye position, the target of the gaze concerning road elements such as traffic lights or other vehicles or pedestrians, and the attribute of the target for gazing, such as average x and y coordinates and continuation time of a gaze. This was done using an eye tracking device, and labelling the gazing targets was done manually.

Steering Entropy and Task Cognition are the two measurements that the study attempts to predict as indicative of cognitive state, their explanation is beyond the scope of this document but can be found in Yoshida et al. (2014).

The used techniques are the Support Vector Machine(SVM) with a RBF kernel and the Random Forest(RF). The chosen evaluation metrics were the accuracy and recall which were defined previously in this document. Both techniques were able to outperform in most cases a default evaluation based on classifying all instances equally. There was no obvious best technique among those two since their relative performances varied depending on the interval being evaluated as well as with the cost parameter of the SVM.

Jabon et al. (2010) presents a study made using a driving simulator and video feed from two cameras as shown on the right picture of Figure 15. The goal was to analyse facial expression and predict unsafe driving behaviour, and image was the chosen source of information because it does not require special markers or users intervention.

In this study each frame of video was analysed in order to determine the coordinates of 22 different facial points, the openness of the mouth and eyes, and the movement of the head, these points are illustrated in the left picture of Figure 15. This information was then synchronized with the simulator data which concerns the road conditions, steering wheel angle, lane tracking information, car speed, longitudinal acceleration due to pedal presses and braking, braking information, number of accidents, and type of accident. After dividing the data in intervals and filtering out results where the confidence in the face tracking was lower than 60%, the authors found that the facial features which are most predictive of major and minor accidents differed greatly.

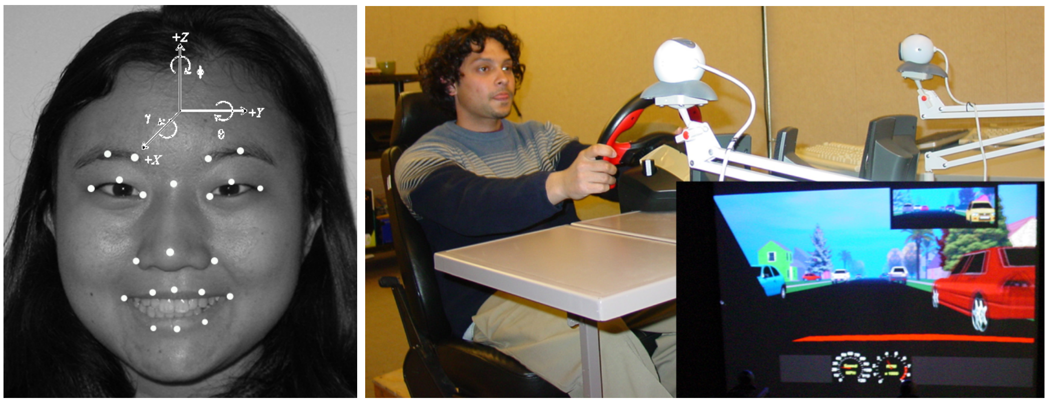


Figure 15.: Facial tracking points(left) and simulation environment(right) in Jabon et al. (2010).

In order to predict minor accidents different classifiers were experimented with, namely Bayesian Nets, Decision Tables, Decision Trees, Support Vector Machines, Regressions, and LogitBoost simple decision stump classifiers. These classifiers were evaluated according to Cohen's kappa coefficient. The LogitBoost classifier provided the best performance of those classifiers concerning minor accidents. The results presented a trend where car features were more useful predicting minor accidents close to the accident, and face features were more predictive longer before a minor accident.

Similar work was done for major accidents, this time using the classifiers: SVM with a poly-kernel, LogitBoost with the weak classifier to be a simple decision stump, a Multilayer Perceptron Neural Net, a simple Decision Table, and Logistic Regression. Once more LogitBoost had the best performance according to the same evaluation metrics. But with major accidents the trend of higher facial predictive power with longer distance from the accident could no longer be found. In all types of accident integrating the environment, car, and facial features tended to provide better results.

3.2 SMARTPHONE AS A SENSOR PLATFORM

A trend with the advent of the smartphone is its use as a sensing, relaying, and/or computing mobile platform, this observation is supported by the researched studies presented next. The smartphone presents many advantages over other means, including cost, being easily used in multiple vehicles, embedded battery, and a wide range of sensors.

3.2.1 *Driver Behaviour*

Some studies approach the matter of Road Traffic analysis from a personal perspective by relating their assessment to the Driver's behaviour, in this section some of those studies will be explored.

One such study was performed in Johnson et al. (2011), this study presents a mobile sensor platform for intelligent recognition of aggressive driving. The authors separate driver style as being either typical or aggressive.

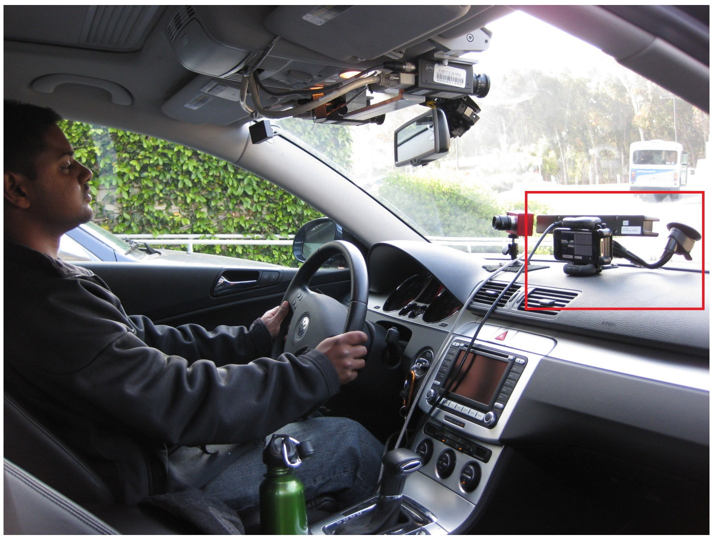


Figure 16.: Position of mounted mobile device in Johnson et al. (2011).

The system created, the MIROAD, focuses on a rear-facing camera, accelerometer, gyroscope, and GPS as data sources, these are part of an iPhone 4 device which is mounted in the center of the vehicle wind-shield as seen in Figure 16. The system detects the following types of events:

- Right and Left turns (90^o)
- U-turns (180^o)
- Aggressive right and left turns (90^o)
- Aggressive U-turns (180^o)
- Aggressive braking
- Right and left aggressive lane changes(swerving)
- Device removal

- Excessive speed

In order to detect when events begin the MIROAD uses the Simple Moving Average (SMA) described as:

$$SMA = \frac{g_x(i)^2 + g_x(i-1)^2 + ... + g_x(i-k-1)^2}{k}$$

If SMA is greater than an upper threshold t_u then $g_x(i-k-1)$ is the beginning of the event and the event lasts until SMA is less than a lower threshold t_L, with an upper bound of 15 seconds per event before it is discarded.

Manoeuvre classification was then made using a Dynamic Time Warping(DTW) algorithm. The DTW algorithm finds the closest match between the captured event and the style of the pre-recorded template signals.

The study goes on to conclude that combining the accelerometer and gyroscope sensors allows for a better detection of events, the DTW algorithm was able to detect 97% of aggressive events using the combined sensor set described above.

Castignani et al. (2013) approaches this topic by providing a single score concerning driving inefficiency and aggressiveness. This study uses mobile sensors from an android to collect information that leads to this evaluation. The sensors used were the GPS, accelerometer, magnetometer, and gravity sensor.

The data was then transformed into input variables to be used in a classifier. The magnitude of acceleration vector was obtained from the accelerometer after removal of the gravity component, this acceleration was then used to define two variables, the number of moderate acceleration events per kilometre and the number of aggressive acceleration events per kilometre. An acceleration event would be considered moderate when the magnitude of acceleration was larger than $1.5m/s^2$ and aggressive when it was larger than $3m/s^2$. Acceleration was also the focus of the transformations on the GPS signal, calculating the linear acceleration, i.e. the speed variation rate between GPS samples. This resulted in four variables, the first two were the maximum positive and maximum negative linear accelerations, and the second two were analogous to the values extracted from the accelerometer magnitude, the number of moderate and aggressive events per kilometre. The thresholds of the GPS based linear acceleration are for the absolute value and are hit at $1m/s^2$ and $2.5m/s^2$ for moderate and aggressive events respectively. Speed played the role to determine overspeeding, this corresponds to the difference between the vehicle speed at each location and the speed limit in that specific location. Three variables came from this notion, the first is the normalized amount of time the driver incurs in overspeed between 1 and 0 where 1 is the whole trip and 0 indicates no overspeed in the trip. The other two variables are the average overspeed and the maximum overspeed per driver. The last property taken into consideration was the steering, measure through the bearing angle provided by the GPS. This is a relative angle to the north. The gyroscope and magnetometer were not used because the authors found them to have a high level of noise. The steering rate is the variation of the bearing angle for two consecutive GPS updates in degrees per second. This originated three input variables, the number of moderate events per kilometre, the number of aggressive events per kilometre, with $10^o/s$ and $40^o/s$ respectively as thresholds, and the last variable was the maximum steering rate for each driver.

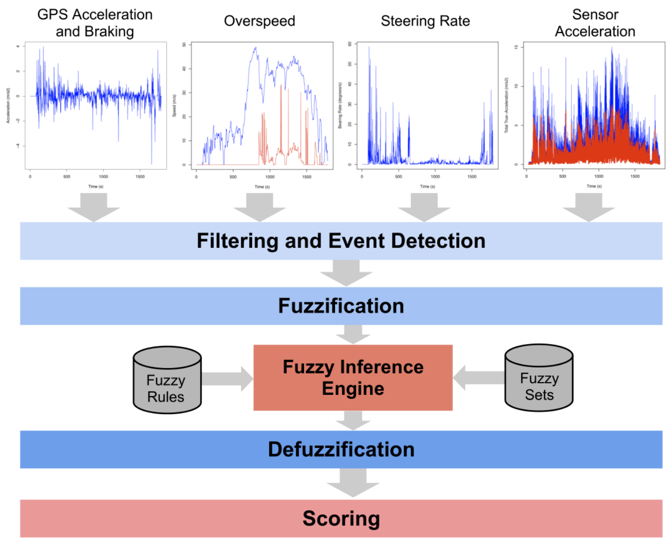


Figure 17.: View of the system in Castignani et al. (2013).

Using a Fuzzy Inference Engine and inference rules, with a COG(Centre of Gravity) algorithm in the defuzzification an output score is obtained. The score is a combination of three other scores relating to urban areas, suburban areas, and extra-urban areas, in which the highest weight is given to the urban areas due to aggressive driver behaviour being more risky in those environments. The results of tests made in a real environment using an Android smartphone show that in urban areas drivers tend to drive aggressively with higher overspeed and acceleration events.

Dange et al. (2015) presents a study about building a social gaming platform which awards virtual coins for favourable driving behaviour, the authors approached the matter of assessing said behaviour in a multi-faceted manner. This approach consisted of a score, this score had 5 sources equally contributing to its value(20% each), these sources are described in Table 2.

The approach can be summarized as follows, the Kohonen Neural Networks are used to derive a measure of harshness based on the brake or acceleration signals. The K-Nearest Neighbours serves to map the relativity between the speed and break signals, penalizing when there's a larger conflict between them, for example applying a high pressure brake

Table 2.: Evaluators and Signals Evaluated in Dange et al. (2015).

Evaluator	Signals Evaluated
Linear Distances	Acceleration and RPM
Linear Distances	Speed and Fuel Consumption
Kohonen Neural Networks	Acceleration and brake
K-Nearest Neighbours	Speed and brake
Dynamic Sliding Window	Speed

when in high speed should be penalized. Linear Distances evaluates harshness with a linear equation connecting the threshold boundaries for penalty and award regions of given metrics, this is illustrated in Figure 18.

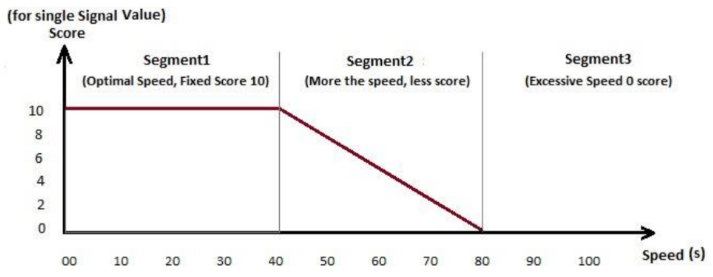


Figure 18.: Evaluation criterion for Linear Distances with representation of sements involving variation of speed signal values in Dange et al. (2015).

Lastly the Dynamic Sliding Window is event-based and scores differently depending on the size of the window and deviation from optimal parameters of the metrics being evaluated, namely the Speed. Penalizing speeds above the optimal window.

In Silva et al. (2014) and Silva et al. (2015) the authors introduce the PHESS Driving platform, this system uses smartphone sensors to collect data. This data is then relayed to a web service and processed for analysis.

Based on the limitation of the smartphone sensors the authors chose the following indicators to gather information about driving patterns:

- Average velocity(Silva et al., 2014)
- Average fuel consumption(Silva et al., 2014)
- Intensity of acceleration and braking(Silva et al., 2014, 2015)
- Number of braking and accelerating events per time unit(Silva et al., 2014, 2015)
- Standard deviation of velocity, intensity of braking and acceleration and accelerations(Silva et al., 2014)

- Number of turn events based on curvature detection(Silva et al., 2014)
- Intensity of force exerted in the vehicle during turns(Silva et al., 2014, 2015)

The value of some of those indicators were calculated through simple statistical procedures on the recorded data such as the average speed, number of brakes, or standard deviations.

The numbers of braking and accelerating events are measured in time windows, allowing for a comparison between different time windows. The average fuel consumption is based on user input data describing the vehicle average consumption and the smartphone recorded data which allows to obtain the distance travelled.

Curve and turn detection is made by tracking the angle difference in the direction between two points using the formula:

$$Dir = \tanh(\sin(\delta_2 - \delta_1) * \cos(\varphi_2), \cos(\varphi_1) * \sin(\varphi_2)) - \sin(\varphi_1) * \cos(\varphi_2) * \cos(-(\varphi_2 - \varphi_1))$$

The intensity of an acceleration in this study is given by the magnitude of the acceleration vector in all three dimensions, the following formula applies:

$$Intensity = \sqrt{Acc_x^2 + Acc_y^2 + Acc_z^2}$$

The classification on of each metric is done based on thresholds obtained from the quantiles of the data when ordered, namely the top 5%(>95%) were considered Red events, the following 15%(80-95%) were considered Yellow events, and the rest Green events. Green being the most desirable type of event followed by Yellow and then Red, respectively.

More than the above a macro scenario was also taken into consideration, in which the platform tries to assess external conditions of traffic. The main indicators regarding this aspect are Road Congestion and Road high speed(Silva et al., 2015). To accomplish this, the map is divided into a grid of squared regions, the results in each squared cell of the grid are then aggregated and averaged the ensuing result is illustrated in Figure 19. The data is then attributed one of two classifications, high speed squares, and low speed, congested, squares. This classification is based on whether the average speed is that which is above the 80% quantile and the inverse is a low speed square, an approach similar in nature to that of individual assessment.

The trip data and scoring used for contextualization and training respectively, in the system developed along this dissertation originate in the PHESS Driving platform introduced by the authors in Silva et al. (2014) and Silva et al. (2015).

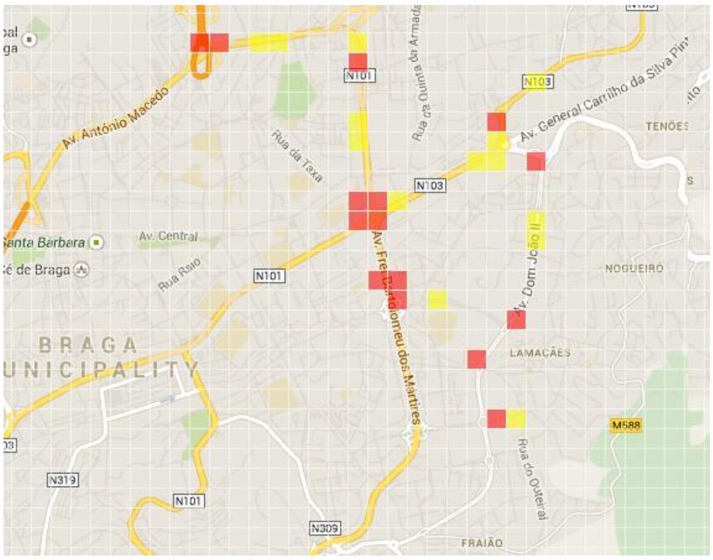


Figure 19.: Aggregated User Data on Community Map in Silva et al. (2015).

3.2.2 *Traffic Flow and Road Conditions*

Evaluation through regions is also adopted by Dolui et al. (2013) creating bounding rectangles as illustrated in Figure 20, in this case the authors attempt to determine the status of traffic of a particular region using real time data. This work uses the GPS coordinates and respective time stamps to calculate the speed of the vehicle at the instant of each such record.

Data is aggregated in a server and stored in a database, requiring therefore internet connectivity. The area covered by the smartphone networks, is then divided into rectangular regions, these are defined by a set of four boundary points each with two coordinates. Just like in Silva et al. (2015), Dolui et al. (2013) does not account for altitude when defining regions.

Dolui et al. (2013) defines instantaneous speed as:

$$S_{ins}(i) = \frac{\sqrt{(X_{i+1} - X_i)^2 + (Y_{i+1} - Y_i)^2}}{t_{i+1} - t_i}$$

This was deemed sufficient to detect the traffic scenario, but not accurate enough to describe the vehicle's actual speed since such a formulation has unnecessary spikes of both positive and negative acceleration. The cumulative speed was calculated through the formula:

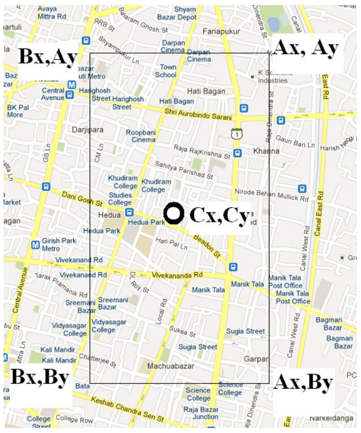


Figure 20.: A sample region in Dolui et al. (2013).

$$S_{cum}(i) = \frac{\sum_{k=i}^{k=i-j} v_{ins}(i)}{j+1}$$

Where the window $j = 3$ was in this case deemed to produce the smoother line describing speed.

The instant acceleration between two instants was defined by the instant speed in:

$$A_{ins}(i) = \frac{v_{i+1} - v_i}{t_{i+1} - t_i}$$

Direction was formulated qualitatively and depends on a series of rules comparing the coordinates of two consecutive points. The caveat of the rules applied is that they fail near the International Date Line.

Accelerometer values are used to determine braking due to potholes or speed breakers, and is estimated based on the change of accelerometer values compared to a frame of reference.

Just like with instant acceleration there are two methods of estimating this deviation in accelerometer value, the initial value method and the cumulative method. The first is described by:

$$\Delta Accl = \sqrt{(x_i - a_x)^2 + (y_i - b_y)^2 + (z_i - c_z)^2}$$

Where the vector(a_x, b_y, c_z) is a calibrated initial value that serves as frame of reference. And the second, the cumulative method, is described by:

$$\Delta Accl(i) = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2 + (z_i - z_{i-1})^2}$$

Traffic congestion can be calculated from the actual speed of the vehicles, as defined by the cumulative speed formula. For a region R_i the average speed of n vehicles in the time span of m instances is calculated using:

$$S_{av}(R_i) = \frac{\sum_{k=1}^{k=n} \frac{\sum_{j=1}^{j=m}(S_{cum}(j))}{m}}{n}$$

Which is considered a measure of congestion, another measure of congestion presented in Dolui et al. (2013) is the number of halts, considered to be when $S_{cum} < 1km/hr$, defined by the halting frequency H_{freq} and halting time H_t, the parameter taken into consideration is then the average halting time of a car over a time span defined by:

$$H_{av}(R_i) = \frac{H_t}{H_{freq}}$$

Given the above described parameters, then the authors regard retardation of the vehicle in two ways, either it is due to a pothole or speed breaker, or it is due to growing congestion in a road. This distinction is made based on the aforementioned accelerometer values, where the deviation of acceleration previously described as $\delta Accl$ is substantial then it can be interpreted as there being a pothole or speed breaker.

Based on a sequence of rules comparing the parameters among themselves or with simple statistics such as estimated average and deviation, the authors classify both traffic(Open, Normal, Slow, Congested, or Impending) and road status(NULL, or Pothole). Combining both to explain slow traffic as being due to a pothole when they are detected simultaneously.

In Akhtar et al. (2014) the authors opted for fuzzy inference for evaluation of data, this data has as a source a smartphone. In this work accelerometer reads are considered meaningful based on the axes according to the following Table 3.

Table 3.: Type of motion based on the axis used by Akhtar et al. (2014).

Axis	Direction	Type of Motion
X	Left/Right	Lane Change
Y	Front/Rear	Acceleration/Deceleration
Z	Up/Down	Road Anomalies

Combining GPS and accelerometer data in a fuzzy system, the presented work produces a safety value.

In Saiprasert and Pattara-Atikom (2013) a GPS based approach is made to detect speeding. The elements retrieved from the recorded GPS data are a measure of instantaneous speed of the vehicle at each data sample, the position composed by the latitude and longitude of each data sample and the heading which is the direction in degrees East of True North. The authors take a simplistic approach to detecting speed by checking when a consecutive instantaneous speed values exceed the speed limit during a specified amount of time. This study also focused in comparing the GPS receivers of different smartphones as well as the speed of recorded in the car's speedometer(Figure 21), their findings suggest a fluctuation in instantaneous speed of approximately 4km/h. The study concludes that due to such a constant speed offset on data from smartphone GPS, that it is as accurate as the values from a car's speedometer.

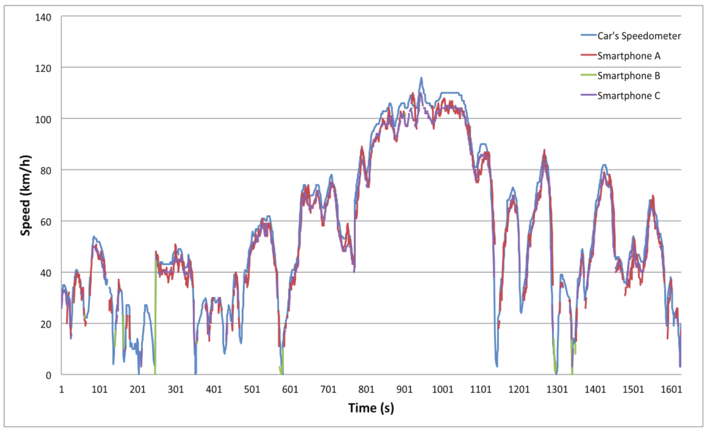


Figure 21.: Speed recorded in a journey presented in Saiprasert and Pattara-Atikom (2013).

In Kovacheva et al. (2013) the authors present the Luxembourg based project LuxTraffic with the goal of providing real time traffic information using smartphone sensor data. This approach segments roads and then aggregates data by segment. Using the GPS and smartphone location services on a static map, the project uses several mapped checkpoints and is able to determine the segment of the map the vehicle is in. The average speed is calculated with distance over time, and the attribution of a traffic flow state is dependent on the average speed of the segment it characterizes. The validity of data is decided based on an algorithm that depends on the day of the week, the hour of the day, as well as the speed captured, elements like whether it happened on rush hour or on a Sunday or Saturday are considered. For segments with absent data, since the platform works in real time, the authors use a the CITA(Cita.lu, 2016) service which has traffic reports based on the amount of vehicles detected, to complement the information of the platform. This allows the authors to create a platform that uses both mobile and stationary sources of data to map traffic flow status.

Goncalves et al. (2014) presents a system that assesses both traffic conditions and driving performance. Traffic was measured using speed calculated with distance over time using the GPS as a data source. If in a time window the average of the vehicles is equal or above the speed limit then it signifies that traffic congestion is low, and the lower the vehicle velocity meant the higher the traffic congestion. Driver performance presented was solely in terms of whether or not the driver went over the speed limit.

While driver-centric approaches are the topic of plenty of studies found in the literature as was showcased above, there are also studies where road conditions are the focus. Astarita et al. (2014) was one such study where the smartphone accelerometer z-axis, sound sensor, and GPS were used to detect road anomalies using a simple formula(found in 18) based on difference of vertical acceleration(m/s^2), difference in sound level(dB) and difference in time(s) with thresholds obtained in the experimental results. When only acceleration or sound are out of the provided thresholds it is considered a state of probable road anomaly.

$$\frac{d\Delta Acc_Z}{dt} > 3 \lor \frac{d\Delta Acc_Z}{dt} < 10 \lor \frac{ddB}{dt} > 2 \tag{18}$$

In Douangphachanh and Oneyama (2013) the authors attempt to estimate the International Roughness Index(IRI) using a smartphone in a more natural scenario, that is, without affixing its position on the vehicle, in locations where drivers would be more likely to put their smartphone while driving. For this they use the Vehicular Intelligent Monitoring System(VIMS) to establish what the actual IRI is. The study compares sampling data at different frequencies and concludes that data gathered with a frequency in the range of 40-50Hz is better at expressing road roughness condition.

3.3 SUMMARY

From analysing the state of the art one can conclude that there is an abundance of different approaches, both in terms of techniques used, which range from simple methods to more complex machine learning methods, as well as in terms of what data is worth analysing.

The approaches are also varied in perspective, with some works depending on stationary sensing points, while others depend on smartphones or other mobile devices.

Even the testing context does not appear to be overwhelmingly biased towards one specific setting, where some studies prefer a controlled environment, others gave preference for a more realistic approach. And while some were tested in a physical context others were tested in a virtual context using simulations.

Because the perspectives of each of the approaches was so different, it's not possible to make a fair performance comparison between them.

4

DESIGN AND IMPLEMENTATION

Contemplating the State of the Art and research done in this area, it became apparent that the current focus of research is on the driver or the vehicle motion when making an assessment of issues occurring on the road. Visual observation of the PHESS Driving results on the website seems to suggest some relation between problematic driving behaviour and neighbouring road objects such as roundabouts and road junctions.

These observations lead to a research into whether contextual factors play a role in road traffic issues.

With that as a motivation some research was made into that matter, it was found that according to Khattak et al. (1998) weather plays a significant role in driving accidents, and that according to Lin et al. (1994) time is also a significant factor. This presented both a problem and an opportunity, the problem being to find out whether contextual information can be used to identify situations that lead to problematic driving, and the opportunity to try to model contexts which are prone to leading to issues without depending on prior knowledge concerning the vehicle or driver's state.

While, that by itself is a worthy problem to solve and an opportunity to explore, the initial purpose of creating an intelligible result was still a concern, it is the author's belief that there is inherent value in having some comprehension of the contributing factors that lead to problematic behaviour.

This work was developed as an add-on module to the PHESS Driving platform, making use of the platform developed by Silva et al. (2014) and Silva et al. (2015). This meant access to both the datasets collected using the mobile application, and the results of the analysis performed, namely the dangerous driving events detected during each trip.

With access granted to the data processed by the PHESS Driving platform, two main points of interest were identified, the first was that of enriching the platform with contextual data, and the second was to find out whether the ensuing contextual information can stand useful and lead to intelligible results on its own.

The development plan was then to divide the solution into two stages, the first was to be an aggregation platform that created a fuller picture of a trip, and the second was to be a means to analyse the data and create a means to model problematic driving based on contextual aspects, furthermore it should be capable of delivering an intelligible report of the results obtained.

4.1 PROPOSED APPROACH

Instead of creating a single monolithic piece of software that encompassed the whole solution it was decided that two components would be created, each one dealing with a different concern, that of enriching the data and that of creating an analysis and subsequent reports.

This proposed two part system would as such enable each part to be used independently of the other.

Next it was necessary to define how to enrich the present data, weather and time were two obvious candidates, to those it was added the notion of neighbouring locations. These correspond to points of interest such as a neighbouring Coffee Shop or ATM, as well as areas of interest such as a neighbouring University Campus or Park.

Because it is a widely adopted means of communication and since it offers very interesting possibilities, it was decided the two platforms that made the system were to communicate any results to the outside over http. Furthermore JSON was chosen as the format of delivering the results, the three candidates contemplated for such were JSON(4.1), XML(4.2), and YAML(4.3).

```
{
  "trip": {
    "tripId": 1,
    "averageSpeed": 50,
    "entries": [1,2]
  }
}
```

Listing 4.1: Example of JSON.

```
<trip>
  <tripId>1</tripId>
  <averageSpeed>50</averageSpeed>
  <entries>
    <id>1</id>
    <id>2</id>
  </entries>
<trip>
```

Listing 4.2: Example of XML.

```
trip:
  tripId:1
```

```
    averageSpeed:50
    entries:
      - 1
      - 2
```

Listing 4.3: Example of YAML.

A format was chosen by taking different aspects into consideration, firstly every format contemplated had the capability of correctly representing the information. This meant there was a need for a hierarchical structure which all three allow, and an ability to portray lists of entries also present in all three. Secondly the relative size among formats, it's apparent just by looking at the examples that the overhead incurred in using XML is larger, and it appears to be specially noticeable in list elements, some of the results, namely on the enriched data side, were bound to contain large lists of elements that could easily number up to the thousands, for this reason XML was deemed to have too large an overhead for the solution intended. This left YAML and JSON, on one hand the first appears to have less overhead, even if only slightly so. On the other hand JSON appears to be very pervasive on the web and can be easily used with standard Javascript, which would later become useful while building a demo web page in the final stages of the project. More than that JSON was also the data format used by the existing PHESS Driving API endpoints, thus choosing JSON allowed for an uniform format of communication.

ENRICHING DATA Data enrichment was chosen to be implemented in a reactive way, this element is to be a web server waiting for a http request to enrich a trip with contextual data, after consulting with the maintainers of the PHESS Driving platform it was decided that this result is then to be posted to the PHESS Driving platform using a http endpoint specifically created for this purpose.

ANALYSING DATA Data analysis was also planned to work in a reactive way, answering requests for a model or a predictive report on a specific situation. Later on a demo page was added for the purpose of demonstrating the system at work with a web client.

In conformity with the theme of this work the project was given the handle IDM which stands for Intelligible Data Metrics.The IDM system was planned to encompass the two logic components above described, as well as a database, and a cache of files with results of some of the raw requests to diminish the total amount of requests as much as possible since third party services usually have limits on the amount of requests over a period of time or even, as a whole, caching the responses helps to avoid duplicate requests.

4.2 System Architecture

The architecture of the IDM system consists of two main components, the aggregator and the analyst, these serve the functions their naming implies, the first aggregates data from various sources and paints a fuller picture of trip entries, the second performs data analysis in order to create the desired model, as well as, produce a few choice evaluation metrics regarding the model. These two components are connected in the sense that they expect and operate on a similarly structured database. The overview of the system can be found in Figure 22.

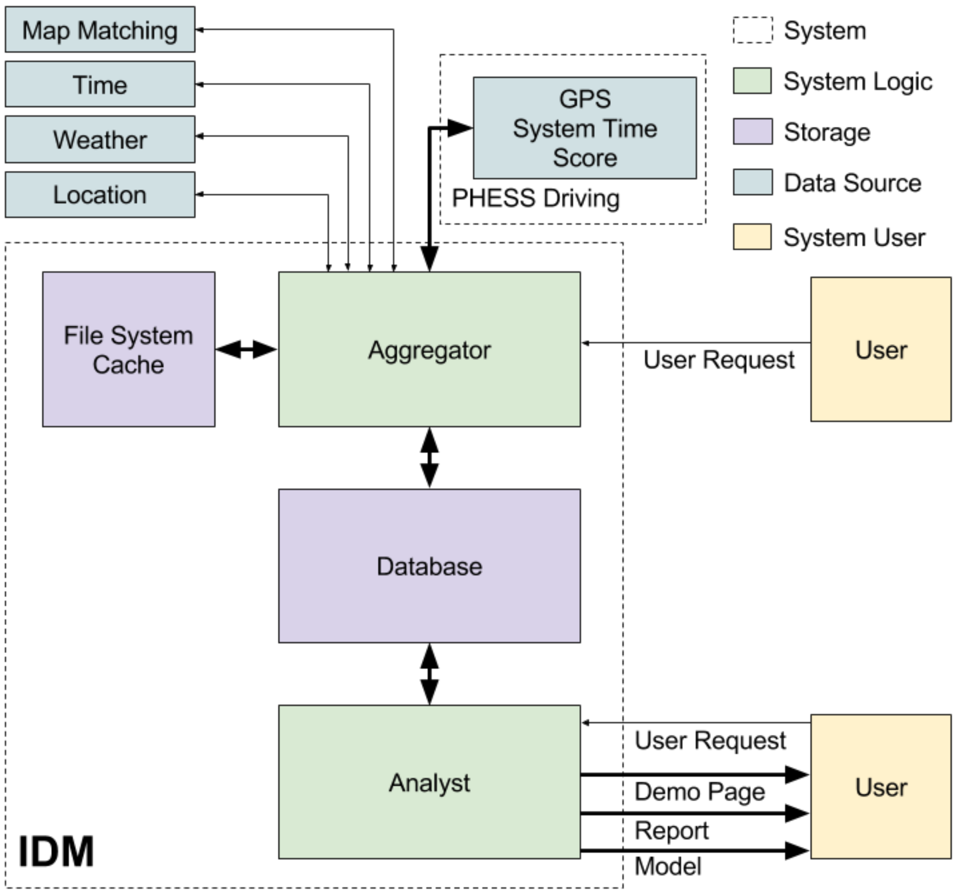


Figure 22.: The IDM system architecture.

As can be seen the architecture does not demand a direct dependence between logical components, in the development chapter(Chapter 4.3) each component will be further explored and detailed. The map matching addition as a data source while not directly a contextual feature used in the end analysis, is an important data fusion step and an integral part of enriching the information. Map matching was explored early on, and its purpose is to connect the GPS entries to a point in a given road path, thus providing a context of where in the road network each entry occurred including information such as street name, points of interest, and areas of interest surrounding the path the vehicle was travelling in. The map matching happening in an outside system was decided during development and only later added to the system architecture.

It is observable in the scheme shown in Figure 22 that the component responsible for enriching the data, in that scheme labelled as *Aggregator*, demands little interaction with a system user, which in this case can refer to any client capable of submitting the appropriate http GET request. The decision to have a manual lever, so to speak, capable of triggering the enrichment of data from a batch of trips was made with the PHESS Driving maintainer, allowing him to choose an opportune schedule for the PHESS Driving platform to receive the enriched data resulting from the request and aggregation as well as picking which trips are to be enriched and then further analysed.

The enriched data goes to two different components from the *Aggregator*, as previously discussed it is sent back to the PHESS Driving platform, but it is also stored in a database belonging to the IDM system, this is the information that will then serve as a source for the features analysed in the *Analyst* component.

4.3 DECISIONS

This section will go over the major decisions related to the development of the solution proposed. It will encompass technologies, data sources, the models, and evaluation metrics.

4.3.1 *Technologies and Models*

The development of the system described in the proposed solution required a varied array of technologies and tools with different scopes and purposes. One of the earlier decisions was that of the programming language of the logic components, the choice was to use the Scala language which like Java runs on the JVM and can seamlessly make use of Java libraries, thus carrying the advantages of using its large ecosystem. Ultimately the choice came down to personal preference of the developer.

The database of choice was MongoDB, it stores information in collections of documents and it is NoSQL. There are many interesting features about it regarding the intended use case, it allows for complex documents instead of rows, it can be distributed, and it works well with JSON formats. But it was the geo-indexing feature that made this not only viable but also the chosen database for the project at hand. The geo-indexing feature allows for the creation of collection's index based on a given document's field which contains a GeoJson object. When this type of index is present complex geometric queries can be made over a collection. The discovery of neighbouring locations was implemented using this feature. MongoDB only guarantees ACID properties at the document level though, not at collection or database level. This issue was not problematic to the specific use case of this work since the data analysis is fault tolerant and will simply discard any entry that does not have all the expected values.

Data analysis and processing was performed using the Apache Spark engine, the choice came down to the distributed nature of Apache Spark which can allow future scaling. The reason for focus on the scaling nature of the data processing engine was that after collecting all data and creating the enriched dataset, the amount of features numbered in several hundreds and the amount of entries was over fifty thousand, this encompassed only a few dozen trips and as such a limited area where trips happened. Having the capability to distribute workload may therefore prove vital to continued operation. Furthermore, Apache Spark has a machine learning library with some common methods including Decision Trees. An extension to plug MongoDB data into Spark was also found and used.

The HTTP server is made using Scala's Spray and using the actor model to dispatch requests in an asynchronous manner. The choice was mostly made due to simplicity of implementation and how the actor paradigm abstracts some issues of concurrency. Furthermore the Akka actor library that was used in implementing the server was also in use for purposes of request scheduling in the project so making use of it was convenient.

One of the important decisions to be made was how to model the data so that predictions could be made concerning problematic road traffic situations. As mentioned before, not only that, but the factors of a problematic situation were also an important aspect to be taken into consideration, with the intent of helping the user comprehend the prediction.

Of those tested, the chosen method to model the data was the Decision Tree, otherwise known as Classification Tree, this method is capable of performing a qualitative classification which was a requirement, it also allows a description of the factors that had a role in the prediction, those can be obtained by finding the branches that lead to said result. The overview of these methods can be found in Section 2.3.4.

Another attractive aspect is the depiction of the model itself, the tree is easy to comprehend and visualize, it can also provide a fuller picture by detailing what happens with the variation of the factors that lead to one of its leafs, which represent classifications.

So as to establish how well models work at predicting a baseline was needed to be established, for this three models were created. The coin toss model is a simple 50% chance model implemented manually to serve as one of three naive approaches that establish a baseline, the other two are the all-in and all-out approaches, these consist of assuming all classifications to be problematic or all classifications to be not problematic. This is obviously an ultimately unrealistic and useless approach, but does provide points of reference from which to improve on various evaluation metrics.

Choosing other models for comparison came down to availability and variety, both chosen were readily available in the machine learning library being used, Apache Spark ML, and were of varying nature one being the Support Vector Machine and the other being Naive Bayes, an overview of these methods is made in Sections 2.3.3 and 2.3.5 respectively.

In the end six models were measured against each other, the three baseline models, along with the Decision Tree, the Naive Bayes, and the SVM models. Chapter 5 exposes the details of the tests run with these models along with the results obtained.

4.3.2 *Data Sources*

This section will present the data sources used in this work. These sources are external to the work used, and connected to via http requests. Some of these sources required the creation of accounts and the use of keys to access the API, and some of the APIs impose limits on the amount and magnitude of requests depending on the used license, for the purpose of this work all licenses obtained were non-commercial and free. Depending on how the workload evolves and how the nature of the PHESS Driving platform grows, different licenses may be required in the future. The software was created with this in mind and allows for the use of different access keys by manipulating its configuration file.

PHESS Driving

People Help Energy Savings and Sustainability, or PHESS, is a platform developed with the intent of being useful to people concerning the matter of energy saving and sustainability. PHESS Driving is the component that deals with creating driver profiles and providing feedback to the user in order to help them perform better. As presented in Silva et al. (2014) and Silva et al. (2015) this platform creates a score for various types of events, the web server also allows for the retrieval of trip information containing records of GPS, Accelerometer, Sound, and Luminosity given the appropriate credentials. All data is retrieved in JSON format, and the IDM system makes use of acceleration, and GPS information, as well as the time stamps and scoring attributed to events happening at the time of those records.

Location information using OpenStreetMap

OpenStreetMap, henceforth referred to as OSM, is a project aiming to create a map of the world distributed as Open Data. This map is collaboratively created and maintained by a volunteer community of mappers with an emphasis on local knowledge. This includes information such as roads, trails, cafés, railway stations, and more.

This service was chosen over alternatives such as Bing Maps, or Google Maps, precisely for being an open data project, having little in the way of service and data usage limitations other than fair use and physical constraints.

This service is used both directly and indirectly by the Aggregation Server, the indirect usage is done by the map matching service.

The direct use of this service was done as calls to three endpoints on the api, using GET methods and coordinates or way identifiers as parameters. The responses retrieved were structured as XML documents.

OSM served as a means to gather data regarding streets, such as their name and geometry, as well as data concerning points of interest (PoI) and areas of interest (AoI) surrounding the streets. This is some of the contextual information to be added to the raw information collected from mobile sensors.

Weather Underground

The Weather Underground service has existed since 1995, it was chosen due to the ease of access to historical weather information.

The data retrieved consists of Humidity, Temperature, Wind Speed, Wind Direction, and Precipitation. It comes structured in a JSON document that describes the weather throughout the day.

Track Matching

Track Matching is a service that performs map-matching of location data on the OpenStreetMap road network. This service is used to match the coordinates of a trip retrieved from the PHESS Driving server to the road network. The data retrieved consists of geometries as GeoJSON, a format of JSON documents for the description of geometric features, as well as projections of the raw GPS data into points on the road itself.

While the matching does appear to be accurate to the route taken, the projections themselves seem to be done in naive manner to the closest point in the route detected.

Google Timezone API

The Google Maps Timezone API is part of Google Maps and allows for the retrieval of the timezone of coordinates, together with the Joda Time library this information permits an accurate estimate of what was the local time of the day of some entry, accounting for daylight savings time and position on the globe.

4.3.3 *Initial developments*

The map matching method required some exploration, it was found that there weren't many readily available tools for the job, with the top contenders being GraphHopper, running an

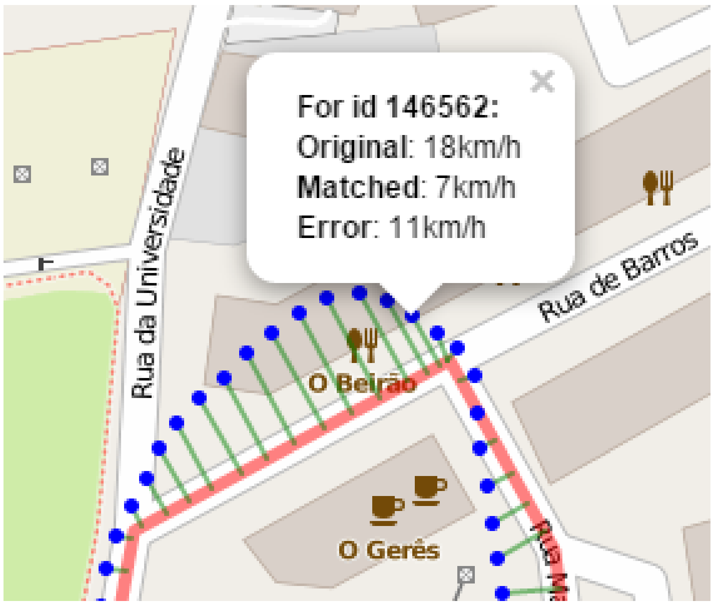


Figure 23.: Visual analysis of GPS points(blue) in comparison to the calculated road path(red).

Open Source Routing Machine(OSRM) server, use the Track Matching service, or use some of the services that offer access to the OSRM capabilities like Mapbox.

GraphHopper can be used through Scala, and so made for a compelling alternative, but upon visual analysis it turned out that its matched paths were not coinciding with the trip's path.

Between using the Track Matching service and running an instance of an OSRM server, the choice came down to implementation where the Track Matching service only demanded the implementation of a client to the necessary API endpoints and the OSRM demanded setting up a server to have running. The Mapbox service has a matching API as well, but is very limited in the amount of entries allowed and demands that the Mapbox map be used in the same project. In the end the chosen method of obtaining a match of the GPS entries on the road network was to use the Track Matching API.

Of course that once chosen its adequacy still had to be evaluated, this consisted mostly of visual comparison and interpretation. To analyse visually the leaflet.js [1] Javascript library was used, this allowed the drawing of the path and GPS points on a real map in order to ascertain whether the GPS entries belong to the calculated paths.

The most important characteristic, that of matching a path, seemed to be visually consistent as can be seen in Figure 23. In this analysis there is a comparison of calculated speed based on GPS distance over time. This comparison was made to showcase the differences in speed between the original GPS entry and the projected GPS point on the road path.

1 The leaflet.js library can be found at `http://leafletjs.com/`

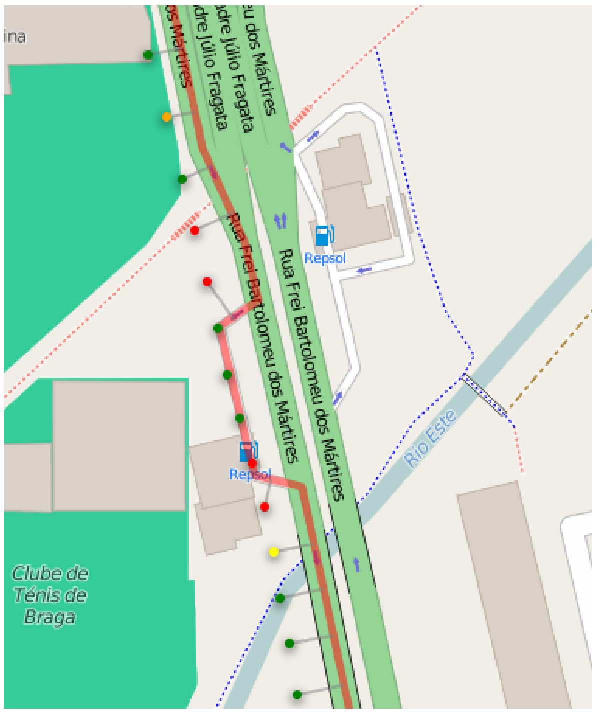


Figure 24.: Case where the map matching inferred incorrectly the path.

The projections are showcased through green lines connecting the point on the path to the original GPS point which is represented by a marker in blue.

In Figure 24 a situation where the matching algorithm failed to correctly predict the path is illustrated. The GPS points are colored according to the magnitude of speed error between the raw GPS and the matched projections from less error which is green to gradually more error from yellow to orange to red. So while most of the visual analysis showed that the algorithm was accurate in calculating the travelled path, there are limits to it's matching capabilities. In the presented Figure the raw GPS was shifted to the left of the road it took place in, and so when there was a path parallel at that same distance it incorrectly assumed the driver changed to that path. In this case it was a path that went into a gas station parallel to the main road. This analysis suggests there may be some correlation between speed error and incorrect matching.

Further analysis shows that the speed error between the matched projections and the raw GPS can also point to other issues, namely that of incorrect point projections as can be seen in Figure 25, in this case it appears the projections are simply being made to the nearest point in the path and as such in some situations such as the illustrated one, several points will cluster into a single corner leading to a speed of 0 within the clustered sequence of points. Even when a more complex path is correctly inferred the algorithm tended to make

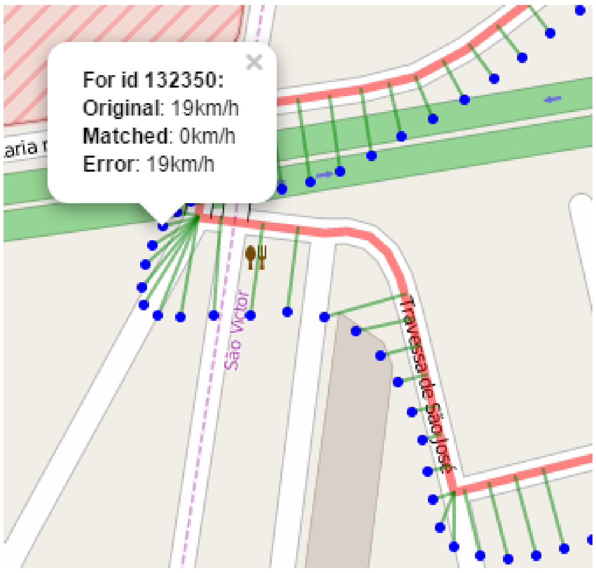


Figure 25.: Projection of several GPS points to the same point in a curve leading to speed error using the map matched points.

some problematic projections of the GPS points into that path as can be seen in Figure 26, where it is visible that speed error can also correlate to incorrect point projections.

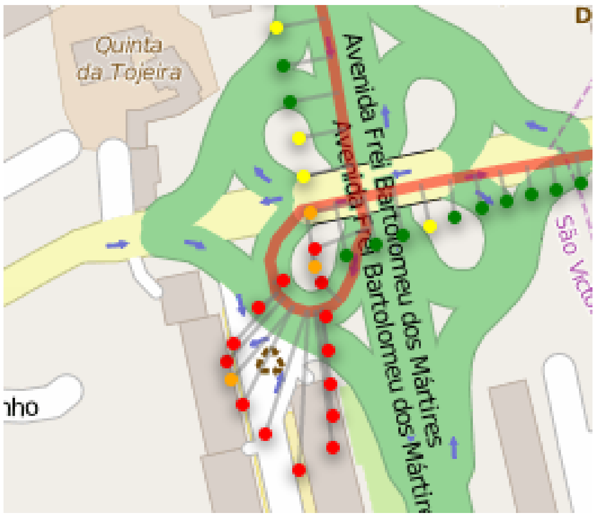


Figure 26.: Complex path correctly map matched, but incorrect point projection.

In the end this solution was considered good enough since obtaining the path itself is the fulcrum of finding the street name and other contextual information regarding said street and nearby locations of interest.

4.4 IMPLEMENTATION

This section will expose the Implementation of the IDM System, from the system's logic composed of the two components of aggregation and analysis of data and information, to the database and other storage concerns.

4.4.1 *Storage scheme*

Information was stored both in a DB and in files. Files were used to save a cache of requests, this was done to deal with limitations on the amount of requests or resources in each of the services used. Any request to the PHESS Driving platform or to the TrackMatching is cached in files. This caching was implemented at the get go in order to limit the requests made to both services. Because it was implemented so early it does not use the database, the responses are saved directly into JSON files.

Weather requests are not cached because there is no advantage due to the uniqueness of the requests, depending on specific coordinates and place. For those same reasons the Google Timzeone API responses are not cached in file.

All information that will be relevant to the analysis performed, is maintained in the MongoDB Database.

The data is divided into different collections, according to use and content. Each collection is composed of documents and indexes. The documents contain structured data easily accessible as JSON objects. The indexes indicate fields or more complex keys to search through. These are not inherently necessary, but they were observed to increase the performance of queries greatly. Besides performance, certain types of indexes provide logical advantages, this is the case of one of the used indexes, the geoIndex which indexes GeoJSON in a geo-spatial manner, allowing for queries such as geometrical intersections, which are used in this work.

The various collections will now be described, as well as a brief explanation for why they were chosen in this manner. Since MongoDB does not have foreign keys like SQL Databases each collection will be presented on its own in pseudo-json format.

```
{
  "_id" : Long,
  "tripId" : Long,
  "timeStamp" : Long,
  "rawGPS" : {
    "lat" : Double,
    "lon" : Double
  },
  "rawSpd" : Double,
  "matchedGPS" : {
    "lat" : Double,
    "lon" : Double
  },
  "matchedSpd" : Double,
  "matchId" : Long,
  "osmId" : Long,
  "weatherSampleTime" : Long,
  "minute" : Int,
  "hour" : Int,
  "dayOfWeek" : Int,
  "dayOfMonth" : Int,
  "monthOfYear" : Int,
  "year" : Int
}
```

Listing 4.4: The tripEntries collection

The *tripEntries* collection, described in 4.4 contains information relating to a few different aspects, these are grounded on a GPS record from the PHESS Driving platform. The *_id*, *tripID*, *timeStamp*, and *rawGPS* are all values retrieved from the PHESS system, and are respectively, the id of the GPS record, the epoch time in milliseconds when the record was saved, the GPS values retrieved. The *matchedGPS* are the coordinates of the map matching projection of the original record, *matchedSpd* and *rawSpd* were calculated using a sliding window of size two, based on the time between records and the distance of the GPS points, respectively for matched GPS and the original values obtained from the PHESS server. The *matchID* is an id attributed by the TrackMatching platform and has an equivalent value in the *matchEntries* collection in the field *_id*. The *osmId* is the id of the way in the OpenStreetMap platform where the matched GPS point rests. The *weatherSampleTime* is the epoch time in milliseconds of when the weather information was sampled. The *minute*, *hour*, *dayOfWeek*, *dayOfMonth*, *monthOfYear*, and *year* are the corresponding values in the calendar taking into account the timezone of the GPS.

```
  {
    "_id" : Long,
    "tripId" : Long,
    "entryIds" : [Long, ...],
    "path" : {
      "type" : "LineString",
      "coordinates": [[Double, Double], ...]
    },
    "wayName" : String,
    "wayInfo" : {
      String : String,
      ...
    },
    "pois" : [Long, ...],
    "aois" : [Long, ...]
  }
```

Listing 4.5: The matchEntries collection

A matched entry contains the map matched version of a path in the road network, those are store in the *matchEntries* collection, see 4.5, the *_id* is the id attributed to such path on the TrackMatching server, the *entryIds* is the list of ids of GPS records on the PHESS Driving platform that constitute the current path, the *tripId* is the id attributed by the PHESS platform to the trip of the GPS records in question. The *path* is a GeoJSON string of lines described by a list of GPS coordinate pairs(Latitude and Longitude), the *wayName* is the name of the way this path is in retrieved using the OpenStreetMap API. The *wayInfo* is the tags associated with said way in the OpenStreetMap platform, each tag is a pair of two strings. The *pois* and *aoi*s fields refer to the ids of Points of Interest and Areas of Interest which are within a perimeter surrounding the path, this process will be described bellow in this document.

```
{
  "_id" : ObjectId,
  "id" : Long,
  "tripId" : Long,
  "evType" : Long,
  "evValue" : Int
}
```

Listing 4.6: The tripEvents collection

The *tripEvents* collection, represented in 4.6 stores all events found in the PHESS Driving server, *_id* is an automatic field done by MongoDB for indexing purposes, the *id* field is the id of a trip entry where the event occurred, the *tripId* is the id of the trip where the event occurred, *evType* and *evValue* are respectively the event type and event value as defined by the PHESS Driving API.

```
{
  "_id" : Long,
  "validGPSPercent" : Double,
  "avgFrequency" : Double,
  "stdevFrequency" : Double,
  "avgDistance" : Double,
  "stdevDistance" : Double,
  "avgMatchingError" : Double,
  "stdevMatchingError" : Double
}
```

Listing 4.7: The trips collection

The *trips* collection, see 4.7 has statistics and information regarding each whole trip, the *_id* field is the id attributed to the trip by the PHESS Driving platform. The field *validGPSPercent* describes the percentage of GPS records that had map matched coordinates. Due to the map matching algorithm used by the TrackMatching system always finding a match, this was observed to be 100% of the time in the available trips. Since the algorithm itself is blackboxed, this remains only a candidate explanation for that fact. The values of *avgFrequency* and *stdevFrequency* refer, respectively, to the average and standard deviation of the frequency of entries during a trip. Analogously the *avgDistance* and *stdevDistance* refer to the average and standard deviation of distance between the original GPS entries. The values of *avgMatchingError* and *sdevMatchingError* respectively describe the average and standard deviation of the Euclidean distance between the map matched projection of each entry and its original GPS reading on the PHESS Driving platform.

```
{
  "_id" : ObjectId,
  "id" : Long,
  "tripId" : Long,
  "timeInMillisUTC" : Long,
  "entryIds" : [Long, ...],
  "temperature" : Double,
  "windSpd" : Double,
  "windDir" : Double,
  "humidity" : Double,
  "precipitation" : Double
}
```

Listing 4.8: The weatherEntries collection

The *weatherEntries* collection, described in 4.8, stores data related to the weather, its *_id* is automatically added by MongoDB for indexing purposes, but is not used. At the moment the *id* field corresponds to a time stamp in milliseconds when the weather was recorded, the *tripId* corresponds to the trip where that record takes place. The field *entryIds* corresponds to a list of ids of trip entries where the current weather information was deemed valid. The *temperature* field is the temperature in Celsius, the *windSpd* is the wind speed in km/h and *windDir* is the wind direction in degrees. The *humidity* is represented as a percentage and *precipitation* is in mm.

```
{
  "_id" : ObjectId,
  "label" : Double,
  "nFeatures" : Int,
  "indexOfFeatures" : [Int, ...],
  "valueOfFeatures" : [Double, ...]
}
```

Listing 4.9: The aggregated collection

The *aggregated* collection, see 4.9, is a collection of aggregated data used as a direct feed to the Spark framework for analysis and distributed operations. As in other collections the *_id* field is automatically created by MongoDB. The *label* field is a number that corresponds to the classification of an entry, the entry is composed by that label and a feature vector. Since it is a sparse vector three fields of information are relevant, the list of values, the total number of features in the vector, and a list of indices whose positions map directly to the list of values. In these lists the order is therefore important. Together those lists indicate the index of a feature and its value, and are on average much smaller than the total number of features (1-2%). Using sparse vectors when sparsity is so common among features allows for large savings of storage space, it may in some cases even be possible for algorithms to take advantage of this sparsity for performance purposes.

```
      {
        "_id" : ObjectId,
        "nodeId" : Long,
        "tags" : [[String, String], ...],
        "loc" : {
          "type" : "Point",
          "coordinates" : [Double,Double]
        }
    }

    {
        "_id" : ObjectId,
        "wayId" : Long,
        "tags" : [[String, String], ...],
        "loc" : {
          "type" : "Polygon",
          "coordinates" : [[[Double, Double],...]...]
        }
      }
```

Listing 4.10: pointsOfInterest and areasOfInterest collections

The collections *osm_pointsOfInterest* and *osm_areasOfInterest*, described in 4.10, are analogous, they have a MongoDB attributed *_id* field. An element id field, *nodeId* and *wayId* which corresponds to the ids provided for the same elements in the OpenStreetMap platform. A list of *tags*, each of which is a key-value pair, these are the characterizing labels found in the OpenStreetMap system for these elements, and a geoJson in the field *loc*, being that areas of interest are polygons while points of interest correspond to points. These collections are geoIndexed to allow for complex geometric queries such as intersection between polygons and other polygons or points.

```
{
  "_id" : Long
}
```

Listing 4.11: The regionsCached collection

The *osm_regionsCached* collection, described in 4.11, is a collection of simple documents containing only the id of a region that has been cached. These are virtual regions calculated by the Analysis software for caching purposes, the process is more detailed in this document in the Analysis Software section.

```
{
  "_id" : Long,
  "tag" : [ String, String ]
}
```

Listing 4.12: The tags collection

The *osm_tags* collection, described in 4.12, maps each tag to an index number, this is later used for the purposes of indexing values in the sparse vectors, as well as part of encoding the textual labels into the feature vector. The *_id* field represents the index number, and the *tag* field is a pair of strings.

4.4.2 *The System Logic*

Aggregation

The aggregation module works as a web service capable of servicing batches of requests concurrently, making use of scala Futures, as well as, using Promises and Actors to control request throttling and asynchronous requests.

The API endpoint of the aggregation component of the IDM system receives a batch of ids from trips to be processed, each trip then goes through the illustrated process numbered from 1 to 6 in Figure 27.

These ids are handed to an Akka Actor, this actor receives a message for each request, parses it and starts the process of enriching the trip data, this is identified as step 1.

Using each identifier, information about trip entries is retrieved from the PHESS Driving platform as can be seen on step 2. Of this information the timestamp of an entry, its GPS

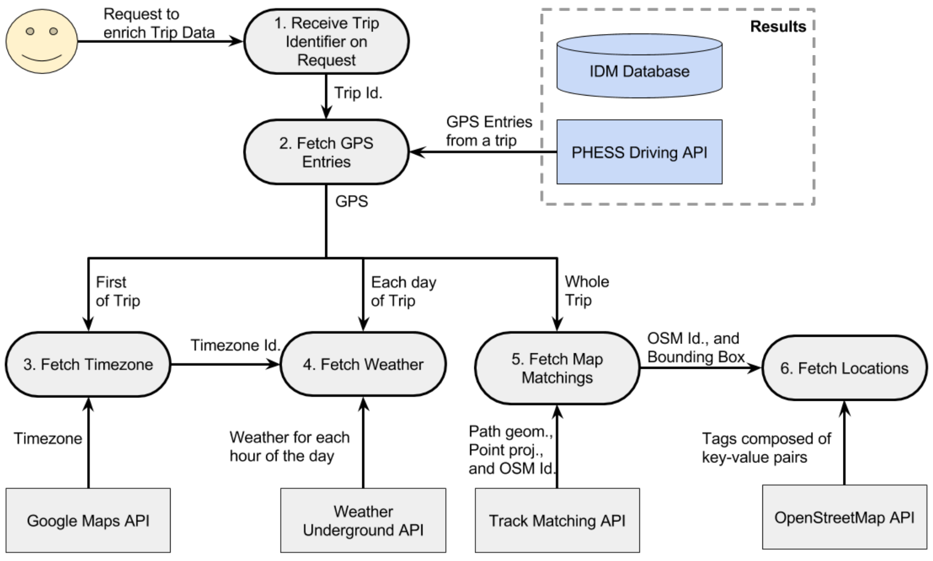


Figure 27.: Data flow in the aggregation component.

coordinates, and its driving behaviour score as determined by the PHESS Driving platform, are used.

Next in the pipeline is the retrieval of date time information, represented by step 3, using the Google Maps Time Zone API data is collected concerning the time zone code of the trip, for each trip only the initial timezone code is considered. While this is something that can be made to update with some frequency along the trip, it was deemed unnecessary, since it is not very likely that a trip will significantly change its timezone. Nevertheless, if the system were to be taken into a production setting then this is a detail that should be pondered.

Following this process, in step number 4, using the joda time library for java, which has compatibility with the Scala language, the day, month, and year are computed from the time zone code retrieved earlier and the timestamp from the PHESS Driving entries. This information together with the GPS location allows for the retrieval of a weather context from the Weather Underground service.

Having the raw GPS and the date time context, the time stamped GPS is passed through a map matching algorithm using the TrackMatching service, this results in a projection of the raw points gathered from a smartphone's GPS into the OpenStreetMap road network, this is the 5th step in the illustrated scheme.

Using the map matched information and the OSM API, in step 6, all the map information is retrieved within a bounding box surrounding a route. This information is then filtered both in type, as well as in value so as to create a set of meaningful locations of interest, without elements such as roads, or values like information about an address. This way

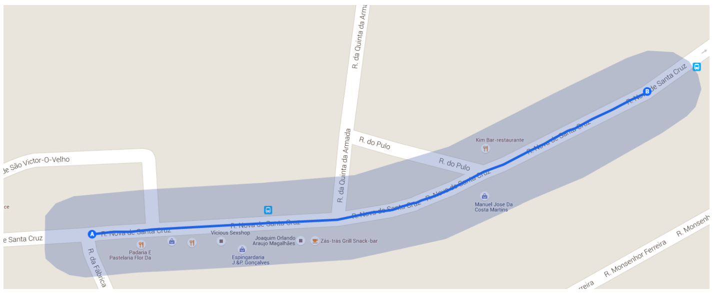


Figure 28.: Example polygon surrounding a path.

only elements which are points or areas of interest like an ATM or a shopping mall, will be used.

With the map matched elements and the locations of interest gathered, a process of calculating a neighbouring area is initiated. This area is a polygon or multiple polygons, this geometry is calculated and is then used to query against the database for locations of interest intersecting that geometry using database functionality with geoIndexes. An illustration of what would such a polygon look like can be found in Figure 28.

For each GPS entry, a driver's behavioural score is retrieved from the PHESS Driving platform, reducing the various aspects of this score to a single value containing only the highest danger indicator among all possible indicators, and an indicator of zero if there's no value or event attributed to the entry. This score is only used internally for analysis and isn't returned as part of the enriched data.

Once all information is collected and saved, a small statistics report is generated regarding the trip, with elements like the average error of map matching among others.

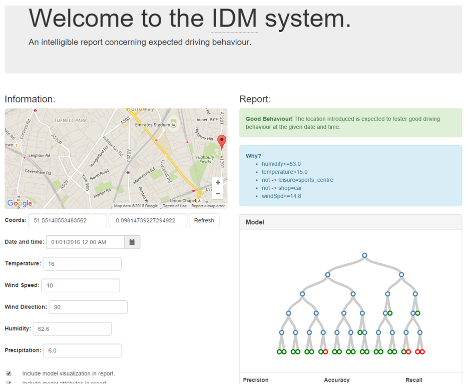
All the results are saved to the database, and then sent to the PHESS Driving platform so as to enrich the data available. The local database serves for the purposes of further analysis and processing of the data.

Analysis

The analysis component picks up where the aggregation software left, it connects Spark to MongoDB and uses Spark to manipulate data through its distributed operations such as map, reduce, filter, count, etc. as well as to leverage the presence of machine learning algorithms in the Spark engine to model the data.

The software runs as a server with a simple API as well as a web page for illustrative purposes where the decision tree model can be visualized.

There are several manners of approaching analysis, the most direct manner is merely to try to check whether there's a correlation between a location and specific conditions and


Welcome to the IDM system.

An intelligible report concerning expected driving behaviour.

Information:

Coords: 51.55140553483562 -0.09814739227294922 Refresh

Date and time: 01/01/2016 12:00 AM

Temperature: 16

Wind Speed: 10

Wind Direction: 90

Humidity: 62.6

Precipitation: 6.0

☑ Include model visualization in report.

Report:

Good Behaviour! The location introduced is expected to foster good driving behaviour at the given date and time.

Why?
- humidity<=83.0
- temperature>15.0
- not -> leisure=sports_centre
- not -> shop=car
- windSpd<=14.8

Model

Figure 29.: View of the web page's structure.

a given driving behaviour, the chosen manner however was to frame this as a problem of problematic location and conditions to a driver's behaviour.

Driver behaviour comes coded into three classes green, yellow, and red respectively in order of crescent danger or unwelcome behaviour as established by the PHESS Driving platform. The PHESS platform classifies it as such based on the amount and intensity of accelerations.

In the IDM project the goal was to delve beyond that and understand whether based on such we can find relations to other aspects, namely proximity to locations and facilities, and conditions such as the weather or time of day.

The data was reduced into two classes that describe a complete domain of possibilities, namely whether an entry refers to a problematic behaviour, or whether it does not.

The created analysis prototype uses data gathered by the Aggregation Software, the strategy starts by parsing all location tags. These tags are recovered from OSM data as pairs of strings, for example a Sports Centre may be described as ("leisure", "sports_centre") as can be found in 31. Using Spark operations such as distinct and map a list of all such unique existing pairs is created. This list is then filtered leaving out some of the information that is not of relevance to the Analysis, such as a building's address. This effectively reduces the noise of the data and modelling concerning such specifics would be regarded as overfitting for our purposes of generalizing the characteristics of a location to classify other locations whose navigation data is not yet available.

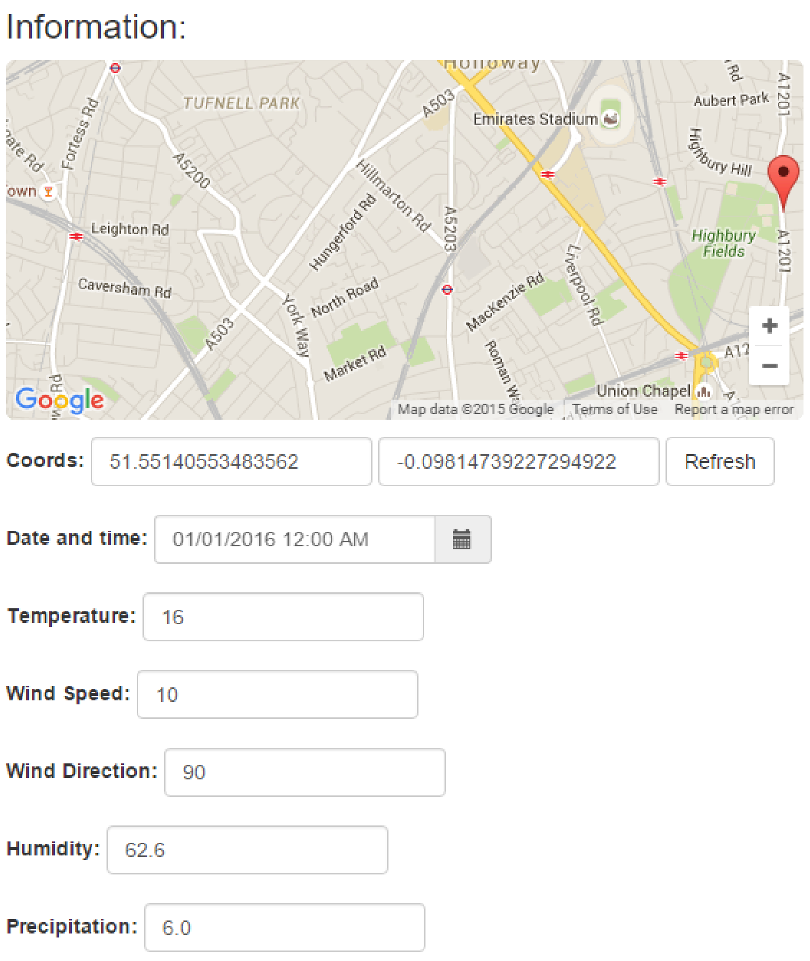
With a list of what are considered the interesting tags a map is created from index to tag and vice-versa. In the current state of the database as of this writing this accounts for 920 such pairings.

These pairs are put together in a feature vector, using Spark's sparse vector to take advantage of the sparsity of these tags throughout the datasets, this allows for much smaller vectors to contain the required information by assuming that omission equates to the tag not being present in the entry.

The feature vector treats each tag pair as a binary value where one means the presence of the tag, and zero means the absence of the tag.

Entries are then also filled with other data, namely weather data, which comes in a continuous numeric form properly conveying the values of each of the weather attributes. And date time data, which are integers on the due range for each of the characteristics, for example a month is an integer between one and twelve.

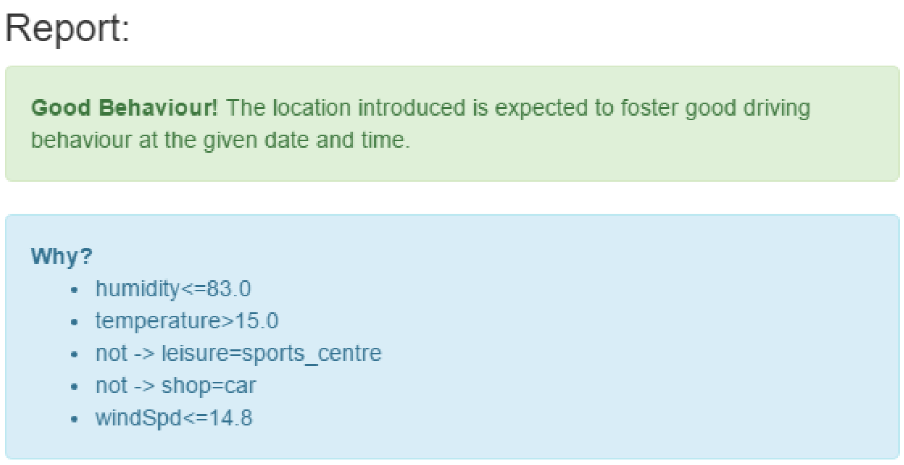
While conceptually the features differ in type as described above, due to the spark implementation of the decision tree classification model they are represented as doubles in the coded Sparse Vector.


Information:

Coords:	51.55140553483562	-0.09814739227294922	Refresh
Date and time:	01/01/2016 12:00 AM		
Temperature:	16		
Wind Speed:	10		
Wind Direction:	90		
Humidity:	62.6		
Precipitation:	6.0		

Figure 30.: Form of input features on the web page.

A report page was created for demonstration purposes, the web page structure can be seen in Figure 29. This page makes use of the API endpoint that returns a classification and report about it given a set of features. It also makes use of the API endpoint that returns the tree model currently in use and presents it to the user.



Report:

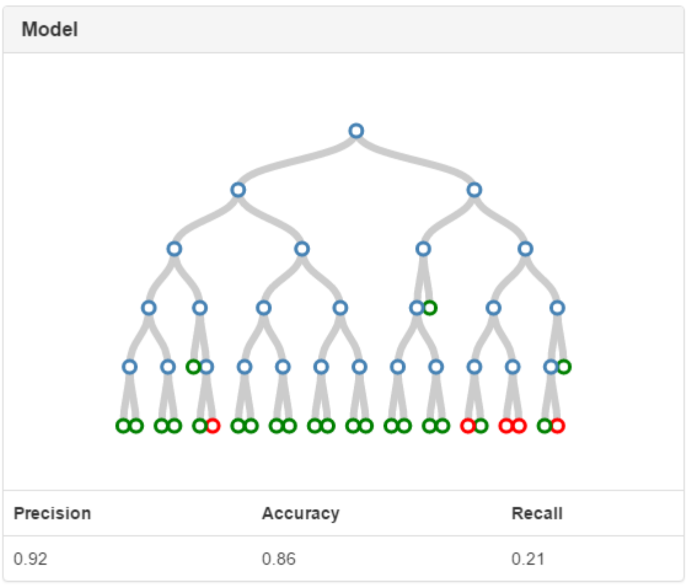
Good Behaviour! The location introduced is expected to foster good driving behaviour at the given date and time.

Why?

- humidity<=83.0
- temperature>15.0
- not -> leisure=sports_centre
- not -> shop=car
- windSpd<=14.8

Figure 31.: Report presented on the web page.

The page presents an easy to use form to fill with information regarding a single feature vector, this input is then enriched with other data on the server side and ran through the model, a list of a combination of factors is then presented as the justification for the classification, as seen in 31. This form can be seen in Figure 30 and the model representation can be seen in Figure 32.



Model

Precision	Accuracy	Recall
0.92	0.86	0.21

Figure 32.: Model and its evaluation metrics presented on the web page.

There's also the possibility of directly downloading a model, as well as information about some evaluation parameters of this model, namely precision, accuracy, and recall.

4.5 OUTCOMES

The resulting system offers various services, the first is the batch processing of trips in order to enrich their data, the results are then sent to the PHESS Driving server instead of the request source.

The request for data enrichment demands a trip identifier, this is a number, and it serves to retrieve the whole trip GPS from the PHESS Driving platform, the shape of such GPS data is showcased in 4.13.

```
{
  "tripID": Long,
  "GPSRecords": [
    {
      "id": Long,
      "timestamp": Long,
      "latitude": Double,
      "longitude": Double,
      "altitude": Double,
      "velocity": Double
    }, ...
  ]
}
```

Listing 4.13: Raw GPS data from PHESS Driving platform

That data is then enriched for each trip, and is returned as a JSON document with the structure described in 4.14. This returned document's fields are analogous in meaning to those stored in the IDM database and described in Section 4.4.1.

```
{
  {
  "statistics" : {
    "id" : Long,
    "validGPSPercent" : Double,
    "avgFrequency" : Double,
    "stdDevFrequency" : Double,
    "avgDistance" : Double,
    "stdDevDistance" : Double,
    "avgMatchingError" : Double,
    "stdDevMatchingError" : Double
  },
  "entries" : [
    {
      "id" : Long,
      "tripId" : Long,
      "timeStamp" : Long,
      "rawGPS" : {
        "lat" : Double,
        "lon" : Double
      },
      "rawSpd" : Double,
      "matchedGPS" : {
        "lat" : Double,
        "lon" : Double
      },
      "matchId" : Long,
      "osmId" : Long,
      "weatherSampleTime" : Long,
      "minute" : Int,
      "hour" : Int,
      "dayOfWeek" : Int,
      "dayOfMonth" : Int,
      "monthOfYear" : Int,
      "year" : Int
    }
  ],
  "matches" : [
    {
      "id" : Long,
      "tripId" : Long,
      "entryIds" : [ Long ],
      "path" : [
        {
          "lat" : Double,
          "lon" : Double
        }
      ],
      "wayName" : String,
      "wayInfo" : {
        String : String,
        ...
        String : String
      },
      "nearbyPOI" : [ Long ],
      "nearbyAOI" : [ Long ]
    }
  ],
  "weathers" : [
    {
      "id" : Long,
      "tripId" : Long,
      "timeInMillisUTC" : Long,
      "entryIds" : [ Long ],
      "temperature" : Double,
      "windSpd" : Double,
      "windDir" : Double,
      "humidity" : Double,
      "precipitation" : Double
    }
   ],
  "pointsOfInterest" : [
    {
      "nodeId" : Long,
      "node" : {
        "lat" : Double,
        "lon" : Double
      },
      "tags" : {
        String : String,
        ...
        String : String
      }
    }
  ],
  "areasOfInterest" : [
    {
      "wayId" : Long,
      "nodes" : [
        {
          "lat" : Double,
          "lon" : Double
        }
      ],
      "tags" : {
        String : String,
        ...
        String : String
      }
    }
  ]
}
```

Listing 4.14: Enriched Trip Data

As can be seen there is a considerable amount of context added to each data point as well as trip statistics concerning the map matching, and the original GPS data.

On the analysis component there are a four endpoints to speak of, one of them is the benchmark endpoint that does not produce a result to the client, but is instead a trigger call for the server to train the several different models and evaluate them using various metrics, the result is output to the server logger which is currently printing to the standard output. In production this can be configured to other settings.

Another important part of the server is the tree model endpoint, this creates a JSON document that describes the current Decision Tree model used by the server, a demonstration of its use is done in the web page where this endpoint is used to generate the tree model representation.

There are two more endpoints in the analysis server API, the demo web page itself which has already been exposed in the Section 4.4.2, and the report endpoint which creates a classification and a report on the reasons for said classification based on a set of provided features. These features are the weather, time and date, and GPS position. Information concerning the neighbouring locations is then automatically retrieved by the server and processed in order to make use of the Decision Tree classifier. This last endpoint is also used by the demo web page for the purposes of retrieving and presenting a report. As was its purpose, the demo web page ends up serving as a proof of concept use of some of the API.

4.6 SUMMARY

This section explains the challenges, how they were reached from an initial exploration to a more mature view of the problem based on prior research. It then goes on to explain what was proposed as an approach to solve the identified challenges. The result is a system, denominated IDM for Intelligible Data Metrics, the architecture is presented in Section 4.2.

As was identified, decisions were made regarding the data sources, namely the services which served as data sources for the system, and the storage of information related to the various features as well as some of the caching options. Decisions also encompassed the technologies to be used, this meant the system's logic and the storage technologies, and the tools to perform data analysis.

The following section, Section 4.4, explains the implementation, how the aggregation of data occurs, as well as how the analysis is performed. Lastly an overview of the outcomes of such a system was made, including which services it provides, the resulting enriched data, and how the results of the analysis are offered. This included the responses to data enrichment requests and the analysis requests.

5 CASE STUDY AND EXPERIMENTS

In this chapter the case study will be presented, this case study regards trips belonging to the PHESS Driving platform. An overview of the setup for any experimentation will be done, including software and hardware details. Two sets of tests will be presented, the first regarding the temporal performance of training the chosen model of a Decision Tree, and the second set of tests will concern the evaluation of the model in comparison to others with different sets of features.

5.1 EXPERIMENT SETUP

The tests were ran using data captured by the PHESS Driving platform obtained between July 2014 and March 2015. This entailed several drivers and devices used for capture, and an average GPS recording rate of roughly a record per second with no restrictions imposed on the drivers. Roughly 23.7% of recorded driving happened before midday and the remaining 76.3% after midday. Trips were heterogeneous in time with the average trip lasting 16 minutes and 47 seconds, and a standard deviation of 20 minutes and 14 seconds. The shortest trip taking 1 minute and 16 seconds, and the longest trip taking 54 minutes and 10 seconds.

Another important aspect to mention is the specifications of the machine and environment where experimentation was performed. All experiments were ran on Windows 10 64-bit Operating System, using Java Runtime Environment (JRE) 1.8 for 64-bit. The hardware was an Asus N550JV laptop, with 8GB DDR3 ram and a Samsung SSD 850 EVO with 500GB as the sole hard drive, the CPU was an intel core i7-4700HQ at 2.4GHz.

Two sets of tests were made, the first performed were the training performance tests. The goal is to help discern how the training of the Decision Tree model scales with more or less entries, hopefully giving a reasonable snapshot of how it may perform in the future. It is important to note though that this does not account for distribution over different machines using spark, which should create significant overhead, so the reader is advised to take the test results at face value and not extrapolate too far about the performance. It should remain on the same trend so long as it is done under a single machine and within the memory limits, while spark is made to be able to use disk space as well as memory, it is expected that having to resort to disk space during the training will lead to slower run times.

The second set of tests consisted in creating different classification models using different methods in order to contextualize the solution created with the Decision Tree.

A baseline was established with naive classifiers. Two of these classifiers simply consider every entry to be problematic or every entry to be non-problematic, it bears no practical usefulness on its own, but will serve to establish a sort of baseline over which to improve upon. Another classifier used was the coin toss, this represents a random choice of tossing a coin in order to determine whether a situation should be classified as problematic or otherwise. Results will be compared to these naive approaches, so as to provide some measure of how useful they are.

As reviewed in the contextual Chapter 2 in Section 2.3.6 there are many ways to evaluate the classifier performance, ultimately no single metric appears to fit all scenarios and therefore a choice must be made according to the demands of the problem. To enable doing such, the problem was formulated in a simple manner, the system needs to be able to find problematic combinations of features in such way that it could be useful in a real life scenario.

The envisioned scenario was that of allocating resources to avoid contexts that are prone to problematic driving behaviour. From the point of view of the driver, avoiding a single point in the road is often fairly inexpensive, yet going through such a point can have significant consequences. But from the city's point of view, planning for and fixing possible issues demands significant resources in terms of planning the road network or even which sorts of buildings and spaces are allowed and where. Since this work is done in the context of smart cities, as referenced in Section 2.1, the point of view of the city was given priority, and therefore importance was given to lessen the waste of resources.

Identifying the combination of factors that lead to a problematic driving event, has been addressed by the choice of the Decision Tree as the means to model the data patterns leading to such events. The Decision Tree allows for the creation of a list of conditions that lead to a problem when a problem is found. With that in mind it comes down to finding an evaluation metric that would make it useful in a real life scenario, such as was proposed above. The information used to discern issues is of three types, weather, time, and neighbouring locations.

One of the things that is important to take into account on this matter is that the system proposed is not supposed to find every single possible issue on the road, in fact it has a specific scope. Firstly because it can not be assumed that the score attribution given by the PHESS Driving, takes into account all types of issues, and secondly location, time, and weather conditions should not account for all that is bound to cause an issue. As was

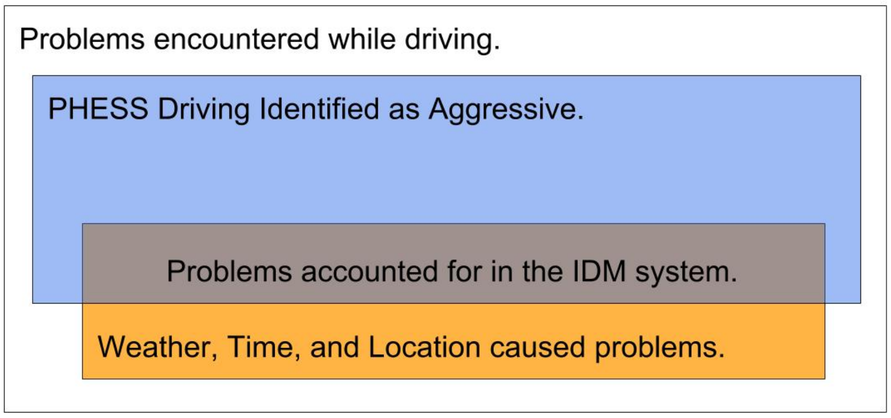

Problems encountered while driving.
PHESS Driving Identified as Aggressive.
Problems accounted for in the IDM system.
Weather, Time, and Location caused problems.

Figure 33.: Scope of problems the IDM System attempts to predict.

presented in Chapter 3 some of the problems may be caused by the driver's own condition (e.g. fatigue). This sort of physiological cause is not accounted in the model, and issues deriving from that and other unaccounted for sources are not expected to be able to be modelled using external factors alone. In fact Lin et al. (1994) found that the amount of time one has been driving is a significant factor in accidents, and that 10 or more years of driver experience consistently meant lower accident risk. None of which is generalizable to the external factors proposed by this work.

Another thing that is of importance regarding the matter of choice of evaluation metrics, is that not all categories are equal in importance. The negative category of the performed binary classification, which is the non problematic situation, isn't very informative itself. This is clear, once we acknowledge, that it is not an indication of there being no problems in actuality. This is because it only indicates that there are no predicted yellow or red events as designated by the PHESS Driving platform. Consequently a situation that is marked as non-problematic is only predicted to not have certain types of problems. The scope of problems the IDM systems deals with is illustrated in Figure 33.

The choice then comes down to identifying whichever problems are identifiable with the types of information taken into account, even if those problems end up being of a limited sort. The primary metric of evaluation chosen was precision, this metric represents the percentage of correct predictions made of there being a problem, this presents a great advantage to a real life scenario, the better the precision the more one is justified in spending resources to investigate and consequently alleviate the problem found. Accuracy isn't a very useful metric by itself, specially in a case such as this where the categories are unbalanced and, naturally, a higher volume of instances of normal entries and a lower volume of problematic entries is the case. Accuracy can lead to bad results since a naive classifier that predicts all entries as normal would have a relatively large accuracy of roughly 83%.

Nevertheless accuracy is an indication of how relevant the classifier is on the whole, a small accuracy means that a lot of the time the classifier can not be trusted to provide an accurate estimate, as such Accuracy was considered a secondary element of evaluation to contextualize the whole dataset in relation to the classifier. The other chosen secondary metric was recall, this choice was based on providing context for the precision metric itself, a smaller recall means that less problems are found. It is important to note that this metric serves as context, for it is expected that the recall will be low seeing as the classifier is not intended to find all sorts of issues but only a subset of them which relate to weather, time, and location.

5.2 RESULTS

The data set resulting from enriching the data of the trips recorded by the PHESS Driving platform contained 64 trips and 144 weather entries, with 54,915 trip entries pivoted on the GPS, 7,446 points of interest, 14,280 areas of interest, with 942 different tags for both points and areas. The total amount of scored event entries retrieved from the PHESS Driving platform was 121,653.

Temporal Performance of Training a Decision Tree Model

The results of the performance tests, concerning the training of the Decision Tree model, can be seen in Figure 34.

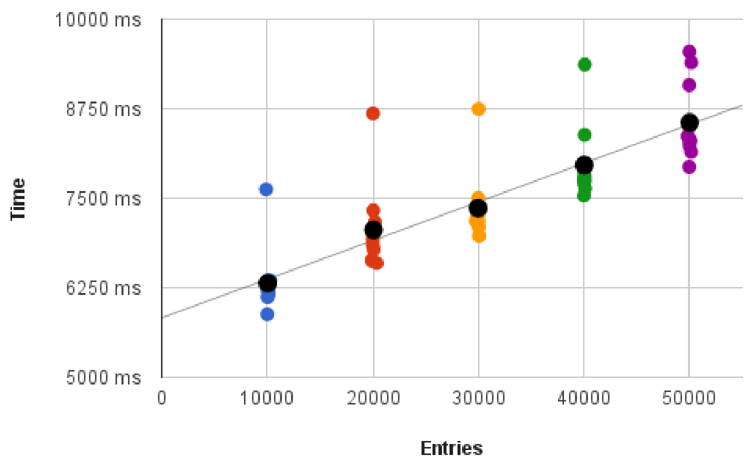


Figure 34.: Evolution of time spent training the model with increasing numbers of input entries.

This shows an apparently linear evolution of time spent training with the amount of entities provided, this result was expected as it is claimed in Spark.apache.org (2015) that the training time should evolve linearly with the number of entries, features, and maximum bins into which a feature is split, the last two remained constant during the tests. Unfortunately while the data provided is sparse, the algorithm is not at the moment optimized for such.

Comparison of the various Models and Features

Tests were made comparing the following algorithms, Coin Toss, Decision Tree using Gini, Decision Tree using Entropy, Naive Bayes, and SVM. All implementations were from the Apache Spark Machine Learning package except the Coin Toss which was created over Spark but which was implemented by mapping each actual label to a random classification, 0 or 1. Other metrics were calculated besides the ones chosen and those metrics are logged by the server to the standard output, but not saved into memory nor are they provided in response to requests. Some of those results will be in an appendix to this document but will not be discussed in this section.

While under sampling the training data, precision behaved differently from the usage of the whole training set. Using 10,000 training entries the precision tended to vary, with the lowest being a 77% precision and the highest 96% precision.

Method	Precision	Recall	Accuracy
All Positive	0.169	1	0.169
All Negative	0	0	0.831
Coin Toss	0.166	0.511	0.484
DT(gini)	0.977	0.182	0.861
DT(entropy)	0.991	0.133	0.853
Naive Bayes	0.222	0.519	0.612
SVM	0.275	0.168	0.785

Table 4.: Results of models obtained with different methods.

In Table 4 the results using location, weather, and time are showcased. It is important to note that due to random split of the data some fluctuation was observed from run to run. In most runs there was no noticeable difference between the Decision Tree created using Gini or Entropy as impurity measures. There appears to be a clear trade-off where some methods were stronger in the recall and worse in precision. The decision trees were able to obtain a very high precision at a cost of a low recall, whether this lower number is due to there not being a relation between the used factors and a large part of the problematic situations or not is unknown. The difference in performance may also be related to the size of the feature vector and sparsity found.

For comparison of different aspects there was a test run using only weather elements. The results can be found in Table 5, the major difference was in the outcome of the Decision Trees, some slight variations in the rest can be accounted by the use of random splits for a training and test set. Nevertheless it is important to point out that the Decision Tree performance was worse.

Method	Precision	Recall	Accuracy
Coin Toss	0.162	0.5	0.488
Decision Tree(Gini)	0.7	0.2	0.854
Decision Tree(Entropy)	0.97	0.116	0.853
Naive Bayes	0.229	0.535	0.624
SVM	0.3	0.189	0.793

Table 5.: Results of models obtained with different methods using only weather data.

Another test was run using only time features, the results can be found in Table 6. As can be seen the results are in general significantly worse, with recall taking the highest loss by comparison in most of the methods employed. As was the case in the rest, Coin Toss remains fairly neutral to the data itself since it is a naive approach. It is worth noting that the Decision Tree built using Entropy as impurity was able to reach a 100% precision, but that such is accounted for by a significant loss of recall. Similarly the SVM method was able to increase the precision in comparison to other runs, but it has a drop in recall. In this case the SVM recall is so low that it makes the model be of little use, it predicted only 27 problems in a total of 15451 entries with 2661 problematic instances, and even so it missed that estimate for almost half, that is, 12 of those 27 predictions.

Method	Precision	Recall	Accuracy
Coin Toss	0.173	0.507	0.491
Decision Tree(Gini)	0.788	0.186	0.851
Decision Tree(Entropy)	1	0.099	0.845
Naive Bayes	0.231	0.239	0.732
SVM	0.556	0.006	0.828

Table 6.: Results of models obtained with different methods using only temporal data.

Lastly a test using only the neighbouring locations' features was performed, the results of this test are in Table 7. These results are in general inferior to the rest with the exception of the Coin Toss model. The only model that has some, even if limited, predictive power using the neighbouring locations' features is the Naive Bayes Model. This may indicate that the neighbouring locations are not very indicative of problematic driving situations on their own, or it may be due to the limited amount of locations being taken into account.

From the above it is visible that the aggregation of all aspects provides a better performance, especially in the case of Decision Trees, which are the target model in this work.

Method	Precision	Recall	Accuracy
Coin Toss	0.173	0.503	0.502
Decision Tree(Gini)	0.615	0.009	0.83
Decision Tree(Entropy)	0.615	0.009	0.83
Naive Bayes	0.265	0.073	0.807
SVM	0	0	0.829

Table 7.: Results of models obtained with different methods using only neighbouring location data.

The tests and evaluation show that this approach has some usefulness in discerning problematic cases, but they also show that this approach has a limited scope of problems it can detect, the latter was expected since the literature seems to point out factors of other nature as also being significant towards risk and safety measures.

The results seem to indicate clearly that on their own, both weather and time can be used to make predictions with some adequacy, as measured by precision. On the other hand locations seem to perform very poorly on their own.

Curiously most methods seem to not perform as well as the Decision Tree method, this result may warrant future study paying special attention to the sparsity of the data and whether there is a correlation.

5.3 SUMMARY

In this chapter the experiments and results were described, firstly was an exposition of the experimental setup, this included a description of the machine, operating system and other relevant details concerning the context of the experiments.

After explaining the setup two experiments were described, that of training algorithm performance with varying numbers of cardinality of the training set, and that of model performance when predicting problematic events in driving as defined by the PHESS Driving platform.

The results of the first experiment were the expected thus confirming the performance asserted for the used implementation of the Decision Tree training algorithm.

The second experiment brought about some unexpected results, namely the better performance of the Decision Tree over the Support Vector Machine in terms of Accuracy and Precision, but depending on the feature vectors mixed results in relation to the Recall measure.

6 CONCLUSION

In this final chapter a summary and some conclusions regarding the work and research performed will be made, there will be some suggestions for future work, both in the sense of expanding the created platform as well as performance and production concerns. Related work developed parallel to this thesis will also be presented.

6.1 WORK SYNTHESIS

This work had several stages, in this section a brief synthesis will be made from the beginning and describing how it evolved into its current state.

At the start the work performed was mostly that of exploration, a study of what information was accessible, how to collect it, and what to do with it was the first stage. Several services were compared, and not all aspects made the final version, for example holidays were left out, as the amount of trips collected were not sufficient to make a proper analysis of the holidays.

The limitations and demands of several services were taken into account and the resulting decisions were the five data sources currently in use. After some research a more defined vision of the system began to form, at this point there was an exploration of map matching and its relation to the raw GPS, this information later served to enrich the data and provide extra analysis and information to the PHESS Driving server.

The next stage was the choice of tools to perform analysis, with distributivity in mind the Apache Spark engine was chosen. Afterwards, since most of the research was done and there was a concrete vision of what was to be implemented, was linking enriched data to the PHESS Driving server in a feedback loop.

After the data retrieval prototype was implemented work began on creating the analysis server. The initial phase of this stage was concerned in determining what methods to use and what exactly was to be evaluated and which information should be used for that assessment. The decision to not focus on specific particulars of each vehicle and driver was made.

Once the prototype of the analysis server was implemented retrieving statistics and creating several models for comparison was the next stage, this allowed for the evaluation and validation of results as presented in Chapter 5. The writing of the dissertation occurred in parallel to the implementation of the analysis server prototype and the tests and result gathering.

Because the server was providing only machine-ready responses in json, it was decided that a web page would be implemented to demonstrate the work done. This required learning some part of a few technologies such as bootstrap[1] and jQuery[2] the first was to organize the web page and the second to handle page elements and http requests.

Lastly the intuitiveness of the model was decided to be showcased in the demo page, leading to learning of the d3.js[3] framework and implementation of the tree visualization using that library. The last step was to finish this document by adding information concerning all of the aforementioned.

Objectives

As was presented in Section 1.2, the main goal of this work was to create richer information about road traffic expression.

This objective was accomplished as presented in Section 5.2, where the results of the process explained in 4.3 were exposed, thus demonstrating that contextual data was fused with the mobile data to achieve richer perspective of the driving situations found in the recorded trips.

The achievement of the sub-objectives mentioned in the objectives section (1.2) was also shown and expounded upon throughout this document. The study of data fusion can be found in this document in Section 2.2, and so can an exploratory analysis of several Machine Learning techniques be found in Section 2.3. The system was inspired by several elements of the study that lead to it, these can be found in Chapter 3 and the Smart Cities contextual Section 2.1.

The design of the system, application of data fusion, and creation of the system are explained throughout Chapter 4.3, and the study of the system and its results can be found in Chapter 5.

It is the author's opinion that the objectives proposed were achieved, and that while there is still a lot that can be explored, that the current results satisfy the scope of this work.

1 More information on Bootstrap can be found in `https://getbootstrap.com/`
2 The homepage of the jQuery library is `https://jquery.com/`
3 More details concerning the d3.js library can be found in `https://d3js.org/`

6.2 RELEVANT WORK

During this dissertation other relevant work was performed in parallel to the study and development of the IDM system. A scientific contribution was made to a published article that was presented in the 9th edition of the IDC: International Symposium on Intelligent Distributed Computing in 2015 Guimarães. The article was published as:

- Artur Quintas, Jorge Martins, Marcos Magalhães, Fábio Silva, and Cesar Analide. *Intelligible data metrics for ambient sensorization and gamification.* In Intelligent Distributed Computing IX, pages 333–342. Springer, 2016.

Besides participating in Quintas et al. (2016), the author also worked on acquiring certification on two online edx courses offered by UC Berkeley [4], which were relevant to this work:

- "*Introduction to Big Data with Apache Spark*"[5]
- "*Scalable Machine Learning*"[6].

These played an important role in being able to perform the analysis component of the IDM system using Apache Spark. The first introduced the Apache Spark platform, its basic elements such as the Resilient Distributed Dataset, and how to operate over this type of dataset using distributed operations such as map and reduce, as well as the benefits of some operations over others in terms of performance and when data should be cached or variables broadcast. The second course focused on data analysis, going over the manual implementation of linear and logistic regression, as well as principal component analysis (PCA), and how to use and evaluate the machine learning methods provided by Apache Spark.

6.3 PROSPECT FOR FUTURE WORK

Future work could include some optimizations in the aggregation process, and an analysis of time versus storage so as to know whether advantage can be taken of larger storage space in order to save time in some computations.

As the platform grows and a variety of trips in different days of the year increases, features such as flagging holidays could be introduced, it's also expected that time of day and day of week will play a bigger role as more and more trips at different times of the day and different days of the week are recorded into the PHESS Driving server. This work may

4 More information at `https://www.edx.org/school/uc-berkeleyx`
5 Course page: `https://www.edx.org/course/scalable-machine-learning-uc-berkeleyx-cs190-1x`
6 Course page: `https://www.edx.org/course/introduction-big-data-apache-spark-uc-berkeleyx-cs100-1x`

be evolved in several ways, one is to fully integrate with the PHESS Driving platform and improve upon their mobile application to take into account many of the aspects discussed in the state of the art chapter, namely mapping potholes and rough road conditions.

Another way is to create a mobile application from the ground up to collect the required data, since this is the type of analysis that works better with more samples, it should be a priority to spread the mobile application use as well as to have minimal energy consumption which was a point mentioned in many of the works reviewed that used smartphones.

A proper mapping of speed limits would also be interesting to explore, although this may imply a limited boundary for testing only where speed limits are known, and may also incur in generalization problems since this information is not available everywhere.

In terms of data, the addition of several sources per type of data may present an interesting scenario for future study as well.

Lastly, a separation of concerns may be beneficial, attempting to predict different types of problems, namely on the three big sustainability aspects, economic, social, and environmental. This can range from energy and fuel savings, to accident prevention, to more fluid traffic flows. There's also the possibility of approaching other types of conditions such as pleasantness for the passengers.

BIBLIOGRAPHY

ITU-T Recommendation database, 2012. URL http://www.itu.int/ITU-T/recommendations/rec.aspx?rec=y.2060.

Naheed Akhtar, Kavita Pandey, and Swastik Gupta. Mobile application for safe driving. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pages 212–216. IEEE, 2014.

Nahla Ben Amor, Salem Benferhat, and Zied Elouedi. Naive bayes vs decision trees in intrusion detection systems. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 420–424. ACM, 2004.

ANSR, 2009. URL http://www.ansr.pt/SegurancaRodoviaria/PlanosdeSegurancaRodoviaria/Documents/Guia_Planos_Municiapais_Seguranca_Rodoviaria.pdf.

ANSR, 2016. URL http://www.ansr.pt/Estatisticas/RelatoriosDeSinistralidade/Documents/2015/INFORMA%C3%87%C3%83O%20PERI%C3%93DICA/Per%C3%ADodo%20(22%20a%2031)dez.pdf.

Vittorio Astarita, Demetrio Carmine Festa, Domenico Walter Edvige Mongelli, and Antonio Tassitani. New methodology for the identification of road surface anomalies. In *Service Operations and Logistics, and Informatics (SOLI), 2014 IEEE International Conference on*, pages 149–154. IEEE, 2014.

R.C. Barros, A.C.P.L.F. de Carvalho, and A.A. Freitas. *Automatic Design of Decision-Tree Induction Algorithms*. SpringerBriefs in Computer Science. Springer International Publishing, 2015. ISBN 9783319142319. URL https://books.google.pt/books?id=PFqEBgAAQBAJ.

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel, and Bernd Wiswedel. KNIME: The Konstanz Information Miner. In *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer, 2007. ISBN 978-3-540-78239-1.

Erik Blasch, Elio Bosse, and Dale A. Lambert. *High-Level Information Fusion Management and Systems Design*. Artech House, 2012. ISBN 9781608071517.

Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013, 2013.

German Castignani, Raphaël Frank, and Thomas Engel. An evaluation study of driver profiling fuzzy algorithms using smartphones. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–6. IEEE, 2013.

Bin Cheng, S. Longo, F. Cirillo, M. Bauer, and E. Kovacs. Building a Big Data Platform for Smart Cities: Experience and Lessons from Santander. In *2015 IEEE International Congress on Big Data (BigData Congress)*, pages 592–599, June 2015. doi: 10.1109/BigDataCongress.2015.91.

I. Chikalov. *Average Time Complexity of Decision Trees*. Intelligent Systems Reference Library. Springer Berlin Heidelberg, 2011. ISBN 9783642226618. URL `https://books.google.pt/books?id=pvIdeV3vNzYC`.

Cita.lu. Cita.lu | motorway network, 2016. URL `http://www.cita.lu/en`.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

E. Costa and A. Simões. *Inteligência artificial: fundamentos e aplicações*. FCA, 2008. ISBN 9789727222698. URL `https://books.google.pt/books?id=osluPgAACAAJ`.

Apache CouchDB. Apache CouchDB. URL `https://couchdb.apache.org/`.

Gautam R Dange, Pratheep K Paranthaman, Francesco Belloti, Marco Samaritani, Riccardo Berta, and Alessandro De Gloria. Assessment of driver behavior based on machine learning approaches in a social gaming scenario. 2015.

Belur V Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997.

Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240. ACM, 2006.

B. De Ville. *Decision Trees for Business Intelligence and Data Mining: Using SAS Enterprise Miner*. SAS Press series. SAS Institute, 2006. ISBN 9781599943107. URL `https://books.google.pt/books?id=ICqMItMBkQgC`.

Koustabh Dolui, Sayan Mukherjee, and Soumya Kanti Datta. Traffic status monitoring using smart devices. In *Intelligent Interactive Systems and Assistive Technologies (IISAT), 2013 International Conference on*, pages 8–14. IEEE, 2013.

Viengnam Douangphachanh and Hiroyuki Oneyama. Estimation of road roughness condition from smartphones under realistic settings. In *ITS Telecommunications (ITST), 2013 13th International Conference on*, pages 433–439. IEEE, 2013.

R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern classification*. Pattern Classification and Scene Analysis: Pattern Classification. Wiley, 2001. ISBN 9780471056690. URL `https://books.google.pt/books?id=YoxQAAAAMAAJ`.

Endesa. *Smartcity Malaga - A model of sustainable energy management for cities of the future*. 2014. URL `http://www.endesasmartgrids.com/images/pdfs/2014-02-14-white-paper-sm-malaga.pdf`.

Endesa.com. About endesa endesa.com, 2015a. URL `http://www.endesa.com/en/aboutEndesa/Subhome`.

Endesa.com. Smartcity barcelona endesa.com, 2015b. URL `http://www.endesa.com/en/sustainability/PoliticaSostenibilidad/CompromisoTecnologia/Barcelona_Smartcity`.

Enel. *MALAGA BECOMES TESTING GROUND FOR ENEL SMART GRID DEVELOPMENT*. 2013. URL `http://www.endesa.com/en/saladeprensa/noticias/Documents/Nota%20Smart%20City%20Malaga%20inglese.pdf`.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

J.A. Galache, T. Yonezawa, L. Gurgen, D. Pavia, M. Grella, and H. Maeomichi. ClouT: Leveraging Cloud Computing Techniques for Improving Management of Massive IoT Data. In *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications (SOCA)*, pages 324–327, November 2014. doi: 10.1109/SOCA.2014.47.

Joaquim Goncalves, Joao SV Goncalves, Rosaldo JF Rossetti, and Cristina Olaverri-Monreal. Smartphone sensor platform to study traffic conditions and assess driving performance. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2596–2601. IEEE, 2014.

K. Grabczewski. *Meta-Learning in Decision Tree Induction*. Studies in Computational Intelligence. Springer International Publishing, 2013. ISBN 9783319009605. URL `https://books.google.pt/books?id=TIO6BQAAQBAJ`.

David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997.

David Lee Hall and Sonya A. H. McMullen. *Mathematical Techniques in Multisensor Data Fusion*. Artech House, January 2004. ISBN 9781580533355.

M.A. Hardy and A. Bryman. *Handbook of Data Analysis*. SAGE Publications, 2009. ISBN 9781446203446. URL `https://books.google.pt/books?id=GMcK2KXHDh0C`.

M.H. Hassoun. *Fundamentals of Artificial Neural Networks*. A Bradford book. MIT Press, 1995. ISBN 9780262082396. URL `https://books.google.pt/books?id=Otk32Y3QkxQC`.

Laurent Hyafil and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information Processing Letters*, 5(1):15–17, 1976.

Maria E Jabon, Jeremy N Bailenson, Emmanuel Pontikakis, Leila Takayama, and Clifford Nass. Facial expression analysis for predicting unsafe driving behavior. *IEEE Pervasive Computing*, (4):84–95, 2010.

Jyh-Shing Roger Jang. Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, 1993.

Yingying Jiao, Yong Peng, Bao-Liang Lu, Xiaoping Chen, Shanguang Chen, and Chunhui Wang. Recognizing slow eye movement for driver fatigue detection with machine learning approach. In *Neural Networks (IJCNN), 2014 International Joint Conference on*, pages 4035–4041. IEEE, 2014.

Derick Johnson, Mohan M Trivedi, et al. Driving style recognition using a smartphone as a sensor platform. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, pages 1609–1615. IEEE, 2011.

Asad Khattak, Paula Kantor, and Forrest Council. Role of adverse weather in key crash types on limited-access: roadways implications for advanced weather systems. *Transportation Research Record: Journal of the Transportation Research Board*, (1621):10–19, 1998.

D.G. Kleinbaum and M. Klein. *Logistic Regression: A Self-Learning Text*. Statistics for Biology and Health. Springer New York, 2010. ISBN 9781441917423. URL `https://books.google.pt/books?id=FTVDAAAAQBAJ`.

Aleksandrina Kovacheva, Raphaël Frank, and Thomas Engel. Luxtraffic: A collaborative traffic sensing system. In *Local & Metropolitan Area Networks (LANMAN), 2013 19th IEEE Workshop on*, pages 1–6. IEEE, 2013.

Dana Lahat, Tulay Adali, and Christian Jutten. Multimodal data fusion: an overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9):1449–1477, 2015.

Tzuoo-Ding Lin, Paul P Jovanis, and Chun-Zin Yang. Time of day models of motor carrier accident risk. *Transportation Research Record*, (1467), 1994.

James Manyika, Michael Chui, Brad Brown, Jacques Bughin, Richard Dobbs, Charles Roxburgh, Angela Hung, and Byers Brown. Big data: The next frontier for innovation, competition, and productivity. 2011.

T.M. Mitchell. *Machine Learning*. McGraw-Hill international editions - computer science series. McGraw-Hill Education, 1997. ISBN 9780070428072. URL https://books.google.pt/books?id=xOGAngEACAAJ.

D.C. Montgomery, E.A. Peck, and G.G. Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012. ISBN 9780470542811. URL https://books.google.pt/books?id=0yR4KUL4VDkC.

A. Monzon. Smart cities concept and challenges: Bases for the assessment of smart city projects. In *Smart Cities and Green ICT Systems (SMARTGREENS), 2015 International Conference on*, pages 1–11, May 2015.

H. Pham. *Springer Handbook of Engineering Statistics*. Springer Handbook of Engineering Statistics. Springer, 2006. ISBN 9781852338060. URL https://books.google.pt/books?id=XDfP63NCGU0C.

David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

K.L. Priddy and P.E. Keller. *Artificial Neural Networks: An Introduction*. Tutorial Text Series. Society of Photo Optical, 2005. ISBN 9780819459879. URL https://books.google.pt/books?id=BrnHR7esWmkC.

Artur Quintas, Jorge Martins, Marcos Magalhães, Fábio Silva, and Cesar Analide. Intelligible data metrics for ambient sensorization and gamification. In *Intelligent Distributed Computing IX*, pages 333–342. Springer, 2016.

J.R. Rabuñal. *Artificial Neural Networks in Real-Life Applications*. Idea Group Pub., 2005. ISBN 9781591409045. URL https://books.google.pt/books?id=-RZDR0CypZkC.

Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

R. Rojas and J. Feldman. *Neural Networks: A Systematic Introduction*. Springer Berlin Heidelberg, 2013. ISBN 9783642610684. URL https://books.google.pt/books?id=4rESBwAAQBAJ.

L. Rokach and O. Maimon. *Data Mining with Decision Trees: Theory and Applications*. Series in machine perception and artificial intelligence. World Scientific Publishing Company Pte Limited, 2014. ISBN 9789814590075. URL https://books.google.pt/books?id=Gal-ngEACAAJ.

S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 2010. ISBN 9780136042594. URL `https://books.google.pt/books?id=8jZBksh-bUMC`.

Chalermpol Saiprasert and Wasan Pattara-Atikom. Smartphone enabled dangerous driving report system. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 1231–1237. IEEE, 2013.

G.A.F. Seber and A.J. Lee. *Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012. ISBN 9781118274422. URL `https://books.google.pt/books?id=X2Y6OkXl8ysC`.

Fábio Silva, Cesar Analide, and Paulo Novais. Assessing road traffic expression. *International Journal of Artificial Intelligence and Interactive Multimedia*, 3(1):20–27, 2014.

Fábio Silva, Cesar Analide, and Paulo Novais. Traffic expression through ubiquitous and pervasive sensorization-smart cities and assessment of driving behaviour. 2015.

SmartSantander.eu. SmartSantander. URL `http://www.smartsantander.eu/`.

A. Søgaard. *Semi-Supervised Learning and Domain Adaptation in Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2013. ISBN 9781608459865. URL `https://books.google.pt/books?id=nexcAQAAQBAJ`.

Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *AI 2006: Advances in Artificial Intelligence*, pages 1015–1021. Springer, 2006.

Apache Spark. Apache Spark^{TM} - Lightning-Fast Cluster Computing. URL `http://spark.apache.org/`.

Spark.apache.org. Decision trees - mllib - spark 1.5.2 documentation, 2015. URL `https://spark.apache.org/docs/latest/mllib-decision-tree.html`.

S. Suthaharan. *Machine Learning Models and Algorithms for Big Data Classification: Thinking with Examples for Effective Learning*. Integrated Series in Information Systems. Springer US, 2015. ISBN 9781489976413. URL `https://books.google.pt/books?id=ad3HCgAAQBAJ`.

Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

Fabio Tango and Miroslav Botta. Real-time detection system of driver distraction using machine learning. *Intelligent Transportation Systems, IEEE Transactions on*, 14(2):894–905, 2013.

Kenji Tei and L. Gurgen. ClouT : Cloud of things for empowering the citizen clout in smart cities. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 369–370, March 2014. doi: 10.1109/WF-IoT.2014.6803191.

J. Treboux, A.J. Jara, L. Dufour, and D. Genoud. A predictive data-driven model for traffic-jams forecasting in smart santader city-scale testbed. In *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 64–68, March 2015. doi: 10.1109/WCNCW.2015.7122530.

V. Vapnik. *The Nature of Statistical Learning Theory*. Information Science and Statistics. Springer New York, 2013. ISBN 9781475732641. URL `https://books.google.pt/books?id=EqgACAAAQBAJ`.

Vladimir N Vapnik. The nature of statistical learning theory. 1995.

Vladimir N Vapnik, Bernhard E Boser, and Isabelle M Guyon. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152. ACM, 1992.

Vodafone, 2016. URL `http://press.vodafone.pt/2016/02/23/fundacao-vodafone-apresenta-1a-aldeia-inteligente-de-montanha-3/`.

S. Weisberg. *Applied Linear Regression*. Wiley Series in Probability and Statistics. Wiley, 2005. ISBN 9780471704089. URL `https://books.google.pt/books?id=xd0tNdFOOjcC`.

B. Yegnanarayana. *ARTIFICIAL NEURAL NETWORKS*. PHI Learning, 2009. ISBN 9788120312531. URL `https://books.google.pt/books?id=RTtvUVU_xL4C`.

T. Yonezawa, J.A. Galache, L. Gurgen, I. Matranga, H. Maeomichi, and T. Shibuya. A citizen-centric approach towards global-scale smart city platform. In *2015 International Conference on Recent Advances in Internet of Things (RIoT)*, pages 1–6, April 2015. doi: 10.1109/RIOT.2015.7104913.

Yutaka Yoshida, Hayato Ohwada, Fumio Mizoguchi, and Hirotoshi Iwasaki. Classifying cognitive load and driving situation with machine learning. *International Journal of Machine Learning and Computing*, 4(3):210–215, 2014.

William J Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950.

L. Yu, S. Wang, and K.K. Lai. *Foreign-Exchange-Rate Forecasting with Artificial Neural Networks*. International Series in Operations Research & Management Science. Springer US, 2010. ISBN 9780387717203. URL `https://books.google.pt/books?id=T-a3BAAAQBAJ`.

Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

X. Zhu and A. Goldberg. *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009. ISBN 9781598295481. URL `https://books.google.pt/books?id=uY9hAQAAQBAJ`.

A

SUPPORT MATERIAL

The full results of the comparison between models using all features:

Method	Accuracy	Precision	Recall	Specificity	Inverse Precision
All Positive	0.169	0.169	1	0	N/A
All Negative	0.831	0	0	1	0.831
Coin Toss	0.484	0.166	0.511	0.479	0.828
DT(gini)	0.861	0.977	0.182	0.999	0.857
DT(entropy)	0.853	0.991	0.133	1	0.850
Naive Bayes	0.612	0.222	0.519	0.63	0.866
SVM	0.785	0.275	0.168	0.91	0.843

Method	Balanced Accuracy	F-Measure	+ Likelihood	- Likelihood
All Positive	0.5	0.4	1	N/A
All Negative	0.5	0	N/A	1
Coin Toss	0.495	0.289	0.989	1.022
DT(gini)	0.591	0.174	211.153	0.819
DT(entropy)	0.567	0.129	567.549	0.867
Naive Bayes	0.575	0.328	1.405	0.762
SVM	0.539	0.146	1.865	0.915

Method	Youden's Index	Discriminant Power	AUC ROC	AUC PR
All Positive	0	N/A	0.5	0.584
All Negative	0	N/A	0.5	0.584
Coin Toss	-0.011	-0.062	0.495	0.38
DT(gini)	0.181	6.227	0.591	0.649
DT(entropy)	0.133	7.323	0.567	0.636
Naive Bayes	0.15	0.577	0.575	0.411
SVM	0.078	1.431	0.539	0.292

Full results of the Decision Tree performance tests for different amounts of entries:

Entities	Training Time	Entities	Training Time	Entities	Training Time
958	7,739 ms	9883	7,625 ms	20023	8,685 ms
1019	4,704 ms	10111	6,168 ms	20047	7,330 ms
987	4,781 ms	10057	6,355 ms	20207	7,170 ms
988	4,867 ms	9993	5,880 ms	20372	6,594 ms
1013	5,083 ms	10075	6,182 ms	19991	6,922 ms
1042	6,146 ms	10023	6,117 ms	19991	6,850 ms
1030	5,845 ms	10294	6,351 ms	20163	7,075 ms
1046	4,599 ms	9965	6,310 ms	20115	7,034 ms
977	4,927 ms	9954	6,250 ms	19990	6,992 ms
1031	4,523 ms	10067	6,196 ms	19938	6,624 ms
977	5,321 ms	10083	6,197 ms	19899	6,637 ms
1032	5,153 ms	10039	6,129 ms	20090	6,783 ms
1008	5,307 ms	10045	6,313 ms	20069	7,058 ms

Entities	Training Time	Entities	Training Time	Entities	Training Time
30029	8,748 ms	40062	9,368 ms	50176	9,396 ms
29903	7,368 ms	40050	8,386 ms	49973	9,080 ms
30005	7,280 ms	40069	7,748 ms	49986	9,546 ms
29998	7,505 ms	39996	7,905 ms	49810	8,368 ms
29758	7,181 ms	39964	7,765 ms	49993	8,237 ms
30129	7,371 ms	40007	7,868 ms	50061	8,296 ms
29866	7,211 ms	40096	7,635 ms	50003	8,293 ms
29862	7,243 ms	40023	7,921 ms	50040	8,320 ms
30038	7,099 ms	39996	7,708 ms	49989	8,495 ms
29996	7,193 ms	40075	7,916 ms	50030	8,577 ms
29930	7,162 ms	39983	7,536 ms	50165	8,147 ms
30067	6,974 ms	40058	7,812 ms	49972	7,939 ms
29965	7,361 ms	40032	7,964 ms	50017	8,558 ms