Using Exit Time Predictions to Optimize Self-Automated Parking Lots

Rafael Ribeiro Nunes
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2014

Orientador
Michel Ferreira, Faculdade de Ciências da Universidade do Porto

Coorientador
Luís Moreira-Matias, Faculdade de Engenharia da Universidade do Porto

Rafael Ribeiro Nunes

Using Exit Time Predictions to Optimize Self Automated Parking Lots

U. PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Tese submetida à Faculdade de Ciências da Universidade do Porto para obtenção do grau de Mestre em Ciência de Computadores

Advisors: Professor Michel Ferreira and Luís Moreira-Matias

Departamento de Ciência de Computadores
Faculdade de Ciências da Universidade do Porto
Setembro de 2014

Contents

List of Tables

List of Figures

Acknowledgements

Gostaria de agradecer em primeiro ao meu orientador, Michel Ferreira, por ter tomado interesse no meu trabalho e me oferecer a oportunidade de trabalhar neste projeto. De seguida, ao meu coorientador Luís Moreira-Matias, pela sua orientação, exigência e disponibilidade, que foram fundamentais para o sucesso do meu trabalho.

Gostava ainda de agradecer ao LIAAD/INESC TEC e aos meus amigos e colegas do laboratório, por me terem acolhido e proporcionado um local de trabalho que me permitiu desenvolver este projeto bem como novas competências pessoais e interpessoais. Em especial gostaria de agradecer ao Fábio Pinto pela disponibilidade em me ajudar, ensinar e aturar.

Por fim, gostaria ainda de agradecer ao projeto I-CITY - ”ICT for Future Mobility”, financiado pelo fundo NORTE-07-0124-FEDER-000064.

Abstract

Private car commuting is heavily dependent on the subsidisation that exists in the form of available free parking. However, the public funding policy of such free parking has been changing over the last years, with a substantial increase of meter-charged parking areas in many cities. To help to increase the sustainability of car transportation, a novel concept of a self-automated parking lot has been recently proposed, which leverages on a collaborative mobility of parked cars to achieve the goal of parking twice as many cars in the same area, as compared to a conventional parking lot. This concept, known as self-automated parking lots, can be improved if a reasonable prediction of the exit time of each car that enters the parking lot is used to try to optimize its initial placement, in order to reduce the mobility necessary to extract blocked cars.

By using Machine Learning techniques a vehicle's exit time can be predicted by using historical data from its previous visits to the parking lot, creating a profile. Vehicles with similar profiles can be grouped together to create more accurate predictions. Given the stochasticity of parking time habits, an interval estimation of the vehicle's exit time is then used to determine which should be the initial placement that reduces the collaborative mobility necessary to create an exit path to the vehicle. Two case studies, of two faculties of the University of Porto in Portugal are used to validate this works methodology.

In this dissertation we show that the exit time prediction can be done with a relatively small error, and that this prediction can be used to reduce the collaborative mobility in a self-automated parking lot.

Keywords: intelligent transportation systems, intelligent parking lots, electric vehicles, vehicular communication, machine learning

Resumo

Viagens diárias em carros privados estão profundamente dependentes da subsidiação que existe em forma de estacionamento grátis disponível. No entanto, as políticas de financiamento público de tal estacionamento grátis tem vindo a mudar ao longo dos anos, com um aumento substancial de áreas de estacionamento com parquímetros em muitas cidades. Para ajudar o aumento da sustentabilidade da transportação por carro, um conceito inovador de um parque de estacionamento autónomo foi recentemente proposto, que toma proveito de uma mobilidade colaborativa dos carros estacionados para atingir o objetivo de estacionar o dobro dos carros na mesma área, comparado com um parque de estacionamento convencional. Este conceito, conhecido como parques de estacionamento autónomos, pode ser melhorado se uma previsão razoável do tempo de saída de cada carro que entra o parque de estacionamento for usada para tentar otimizar a sua posição inicial, reduzindo a mobilidade necessária para extrair carros bloqueados.

Usando técnicas de Machine Learning o tempo de saída de um veículo pode ser previsto através do uso de dados históricos das suas visitas anteriores ao parque de estacionamento, criando um perfil. Veículos com perfis semelhantes podem ser agrupados para criar previsões mais precisas. Dada a estocasticidade dos hábitos de tempo de estacionamento, uma estimação de um intervalo de tempo de saída de um veículo é então usada para determinar qual será a posição inicial que reduz a mobilidade colaborativa necessária para criar um caminho de saída para o veículo. Dois casos de estudo, de duas faculdades da Universidade do Porto em Portugal foram usados para validar a metodologia neste trabalho.

Nesta dissertação mostramos que a previsão do tempo de saída pode ser feita com um erro relativamente pequeno, e que esta previsão pode ser utilizada para reduzir a mobilidade colaborativa num parque de estacionamento autónomo.

Palavras-chave: sistemas de transporte inteligentes, parques de estacionamento inteligentes, veículos elétricos, comunicação veicular, *machine learning*

Introduction 1

Nowadays, urban planning is facing major challenges due to the overwhelming amount of population living in cities. World population has grown by 4 thousand million since 1960, and is currently growing at a rate of around 1.14% per year [wor]. In addition 54% of the world's population are concentrated in urban areas and it is expected to be 66% in 2050 [unu]. Consequently, mobility is evolving from a need to a daily struggle. Today, the car transportation still represents a significant role on it. However, the number of cars in many cities has reached a level where the road infrastructure is unable to avoid systematic traffic congestions. In addition, the high cost of fossil fuels and pollutant emission levels are creating significant challenges for the sustainability of private car commuting in major cities [GW97]. Tolls and prohibition of circulation in one or two week days for a given vehicle are already in place in some of our cities [AG07]. Technology is trying to mitigate these issues by 1) zero-emissions electric propulsion [Cha07] and 2) connected navigation [tom] are two examples of technologies that can help making car transportation more sustainable by 1) reducing emissions and 2) spending less time in the road, avoiding traffic congestion.

Parking is a major problem of car transportation, with important implications in traffic congestion and urban landscape. It has been shown that parking represents 75% of the variable costs of automobile commuting [SA$^+$05], supported by a major public subsidisation of the space devoted to car parking, where the user does not pay in more than 95% of the occasions [Sho06]. Such free parking has high costs to the municipalities, not only for the area occupied by the vehicle and on the pollution generated through the route it adopted to get there, but also by the surrounding area, which allows passengers to get in and out of the car or even other vehicles to get in and out of the park [Fea14].

1.1 Problem Statement

Self-automated parking lots represent a promising approach to solve the issue of car parking. In this approach vehicles are parked in a very compact way, without space devoted to access ways or even inter-vehicle space that allows opening doors. When a vehicle that is blocked by others wants to leave the parking lot, the other vehicles are moved in order to create an exit path. An example of a conventional parking lot and a self-automated one can be seen in Figures 1.1 and 1.2, respectively. The compact way the vehicles are parked in Figure 1.2 permits a much larger number of cars to be parked simultaneously. When the vehicle in red wishes to exit the park all vehicles behind it must be moved in order to create an exit path.

Hereby, the main idea consists on developing a method able to produce exit time predictions for each vehicle in real time based on historical data on their entries/exits in the parking lot. Based on such prediction, the vehicles can be placed using an order that reflects their expected exit times. Such approach aims to minimize the number of vehicles blocked by other ones who will leave the parking lot at a later time.

Formally, it is possible to represent the Self-Automated Parking Lot as a set of N independent First In First Out (FIFO) queues $Q = \{q_1, ..., q_N\}$, where each queue contains an ordered subset of the set of C parked cars $P = \{p_1, ..., p_C\}$. Given the arrival of a new car p_C+1, we need to 1) approximate the parking time function $f(p,t)$ - where p stands for a given vehicle and t for the time - using its historical values. Secondly, we use such approximation - $pr = \hat{f}(p,t)$ as parameter of the function *col* - which returns the number of *unnecessary car movements*[1]. Consequently, it is possible to define the following

$$\arg min_q col(q, pr, Q, Q') \qquad (1.1)$$

where q stands for the queue selected for the newly arrived vehicle p_{C+1}.

1.2 Motivation & Goals

In this dissertation we propose to explore both historical and real data on parking lot entries to build a prediction on the parking time of each vehicle. Using this information to improve the original placement of the car in order to reduce manoeuvring mobility. Our goal is not to obtain a precise exit time for each vehicle, but rather a interval-based estimation on the parking time that can be used in conjunction with the parking lot

[1]movements required to let cars located in further queue positions exit the park

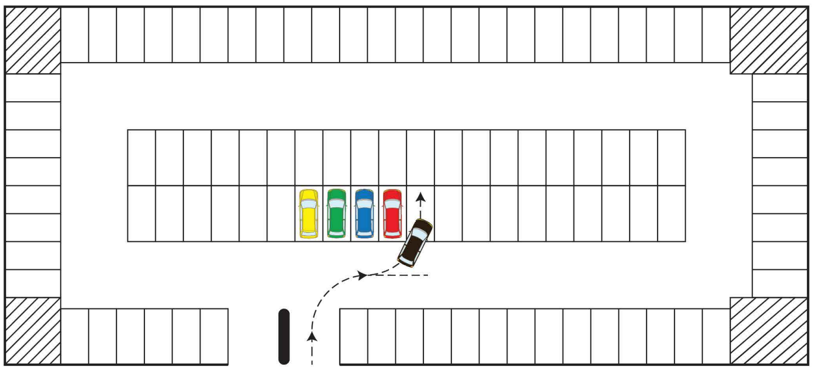


Figure 1.1: Conventional parking lot with access ways and inter-vehicle space. Adapted from [Fea14].

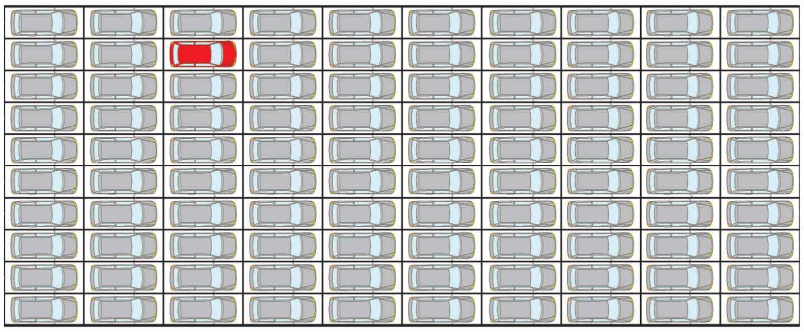


Figure 1.2: Self-automated parking lot in which the vehicles are parked in a very compact way. If the vehicle in red wants to exit the park, all vehicles behind it must be moved in order to create an exit path. Adapted from [Fea14].

layout (e.g. number of lanes) to reduce the likelihood of having to move parked vehicles to create exit paths for blocked vehicles. This methodology should be applicable to any real world park that keeps records on its vehicles. By doing so, we expect to increase the parking lot capacities in the mid-future. Such increase will provoke an improvement on the parking lot capacity and on its profits (in case of its entrance be paid). As more and more data is collected from a variety of sources the effectiveness of Machine Learning (ML) techniques grows and with the increasing use of sensors and other information gathering systems being used in Intelligent Transportation Systems (ITS), the use of these techniques are trending in ITS [ZWW$^+$11]. We propose then the following research goals, to 1) build a predictive model on the parking time using real parking data, able to return an interval-based estimation on its future value using ML techniques, and to 2) develop a function able to use the abovementioned intervals to select the most convenient lane to park a vehicle.

1.3 Dissertation Structure

This dissertation is divided as follows: First, an overview of the problem is given in Chapter 2, describing how the Automated Parking Lots works, as well as some issues concerning the Parking Lot Design and some research works related wit this topic. Next, a summary of the foundational concepts of ML is presented in Chapter 3. The state of the art of important methodologies later employed such as Supervised/Unsupervised Learning, Feature Selection and Incremental Learning are briefly revised. Chapter 4 explains the methodology used in this work in its four steps: Profile Generation, Parking Time Prediction, Incremental Interval Generation and Parking Lane Selection. The performed experiments are detailed in Chapter 5, including a detailed description of the Case Study, then the Experimental the setup used to conduct our experiments and the methods used to validate them. To conclude Chapter 5, the Results of the experiments are shown and discussed. Lastly, in Chapter 6 the conclusions are presented.

Problem Overview 2

2.1 Automated Parking Lots

Technology has been focusing in moving cars, disregarding the parked period of these cars, which represents 95% of the vehicle existence. Recently, a simple proposal that leverages on technology such as electric propulsion or wireless vehicular connectivity has addressed the issue of car parking, arguing that through a collaborative approach to the parking of cars, the area per car could be reduced to nearly half, when compared to the area per car in a conventional parking lot. This approach, known as *self-automated parking lots* [Fea14], works as follows. An electric vehicle (EV) is left at the entrance of a parking lot by its driver. This EV is equipped with vehicular communications that establish a protocol with a Parking Lot Controller (PLC). The EV is also based on Drive-by-Wire (DbW) technology, where in-vehicle Electronic Control Units (ECUs) manage signals sent by the acceleration and braking pedal, and steering wheel. The Vehicle-to-Infrastructure (V2I) communication protocol allows the PLC to control the mobility of the EV in the parking lot. The PLC remotely drives the EV to its parking space, using in-vehicle positioning sensors (e.g. rotation per wheel), magnet-based positioning, or some other type of positioning system (e.g. camera-based). Alternatively to a fully-automated system, a scenario of human-based tele-operated driving could also be used [SGL14]. In this concept of self-automated parking lots the cars are parked in a very compact way, without space devoted to access ways or even inter-vehicle space that allows opening doors. As a new vehicle enters the parking lot, the PLC sends wireless messages to move the vehicles in the parking lot to create space to accommodate the entering vehicle. If a blocked car wants to leave the parking lot, the PLC also sends messages to move the other vehicles, in order to create an exit path. In [Fea14] it was shown that this concept could reduce the area per vehicle to nearly half, as well as reduce the overall mobility of cars in the parking lot, when compared to a conventional parking lot. However, in the original paper, a first-fit strategy was used to initially park each vehicle. Clearly, the initial placement can be improved if some knowledge about the expected exit time of each car is used.

The basic idea is that a car should not be blocked by another car that will leave the parking lot later. If the cars in the parking lot are placed using an order that reflects their expected exit times, then the overall mobility in the parking lot to create exit paths can be reduced.

2.2 Parking Lot Design

The geometric design of the parking lot is an important issue in a self-automated parking lot. In conventional parking lots there are a number of considerations that have to be taken into account when designing them. For instance, width of parking spaces and access ways, one-way or two-way use of the access ways, entry angle in the parking bays ($90°, 60°, 45°$), pedestrian paths, visibility to find an available parking space, etc. In a self-automated parking lot, many of these considerations do not apply. Manoeuvring is done autonomously by the car following the instructions of the PLC, pedestrian access is not allowed, and the assigned parking space is determined by the PLC. The main design issue is defining a geometric layout that maximises parking space, leveraging on minimal buffer areas to make the necessary manoeuvres that allow the exit from any parking space under all occupancy configurations. This geometric design is ultimately determined by the shape of the space of the parking lot. The parking lot architecture also defines the trajectories and associated manoeuvres to enter and exit each parking space.

The parking lot has a V2I communication device which allows the communication between the vehicles and the PLC. In theory, this infrastructure equipment could be replaced by a vehicle in the parking lot, which could assume the function of PLC while parked there, handing over this function to another car upon exit, similarly to the envisioned functioning of a V2V Virtual Traffic Light protocol [FFC+10]. Note, however, that the existence of the actual infrastructure, which could be complemented with a video-camera offering an aerial perspective of the parking lot to improve the controller perception of the location and orientation of vehicles, could simplify the protocol and improve reliability.

Reducing and simplifying such trajectories and manoeuvres is also an important design issue, as they affect the reliability of the system and allow faster storage and retrieval of cars. Note also that the parking lot architecture can take advantage of the fact that the passenger does not enter the parking lot, and thus the inter-vehicle distances do not need to allow for space to open doors. To optimise and simplify manoeuvres, these self-automated parking lots will require specific minimum turning radius values for vehicles. Only vehicles that meet the turning radius specified by each parking lot will be allowed to enter it.

The geometric layout of the parking lot and its buffer areas can assume very different configurations for the self-automated functioning. One possibility is to have parallel lanes with minimal space between them, as illustrated in Fig. 2.1. In this type of layout, the PLC starts by assigning a lane to a vehicle. This initial decision is critical, as it should minimise the need to move a vehicle from one lane to another. Note that if the red vehicle in Fig. 2.1 needs to leave under the current configuration, then the vehicle behind it needs to be moved to another lane. If we could predict that the exit of the red vehicle would happen before the exit of the vehicle behind it, then this last vehicle would be better placed in a different lane. Our goal in this work is exactly to be able to predict an exit-interval for each vehicle, and design a lane selection methodology that reduces the mobility needed to create exit paths.

Note that parking lots will not be able to be completely full, as buffer space needs to exist to allow the exit of each vehicle under all possible configurations. The minimum number of empty spaces, configuring buffer areas, depends on the parking lot layout. In the layout presented in Fig. 2.1, with a lane depth of 7, we need a buffer area with a minimum of 6 empty spaces.

2.3 Related Work

Intelligent Transportation Systems use advanced communication and technology to solve various transportation problems, such as traffic congestion, transport efficiency, parking space management and driver assistance. One component of ITS are intelligent vehicles (IV), these systems through the sensing of vehicle's surrounding environment can provide its user with detailed information or even, partially or totally, control the vehicle.

The applications of IV can be divided in the following categories [Bis00], 1) *collision warning systems*, 2) systems that take partial control of the vehicle, either for *driving assistance* or *collision avoidance* and 3) systems that take total control of the vehicle (*vehicle automation*).

Collision warning systems are ones that advise or warn the driver of potential collisions through visual ou audible indications, either by the use of sensors or dedicated short-range communications [JD08] (DSRC), either vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I). The example of forward collisions can be solved by a vision based

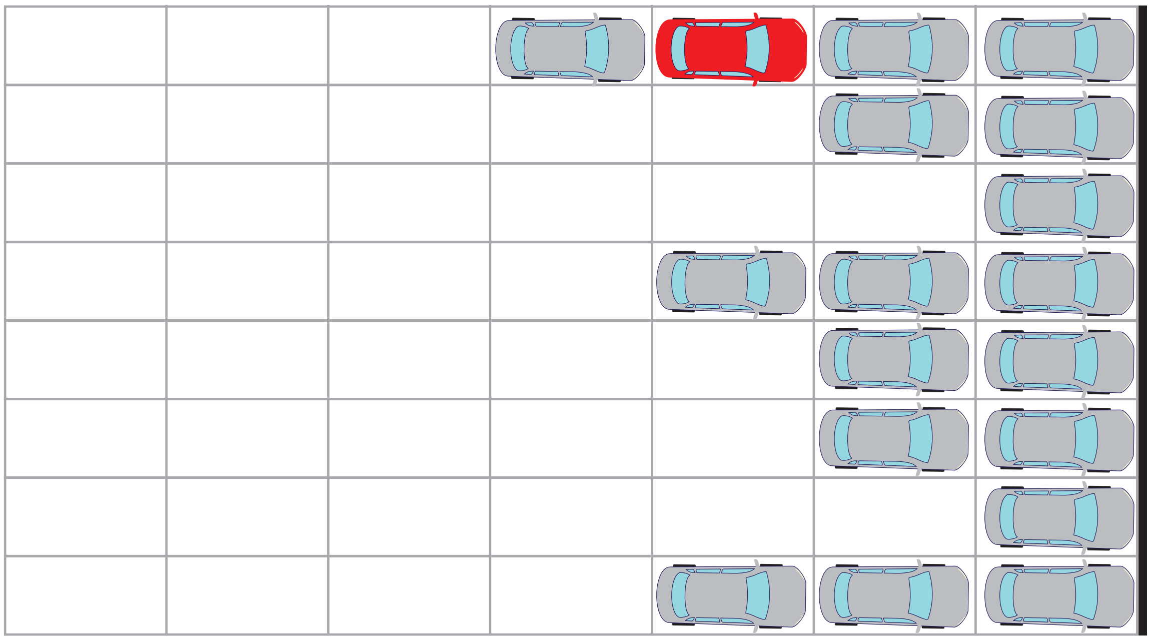


Figure 2.1: An example layout for a self-automated parking lot. The parking lot can never be completely full, as buffer areas are necessary to be able to allow the exit of each vehicle under all possible configurations. In this example, a minimum of 6 empty spar are necessary.

approach through the use of a camera, calculating the *time to contact* and the possible collision course through the size and position of the vehicles obtained in the camera image [DMSS04]. Sensors can be used for blind spot detection [MP87], dual sensors are commonly used in order to ensure accuracy, one first sensor initially detects the vehicle which triggers a second one to confirm the vehicle's presence. Another situation handled by collision warning systems is the lane-departure, which is when a vehicle unintentionally deviates from the center of its lane. This situation usually arises due to temporary or permanent loss of vision of the driver. One solution for the lane-departure is to employ machine vision, measuring the orientation of the lane marks [Lee02]. A final example is the rear-end impact occurs when another vehicle approaches the driver's vehicle from behind, whether it is in the same lane or an adjacent one. A radar is positioned on the vehicles blind spot, it sends two waves and then compares the phase of the two reflected signals to determine the distance [FH72].

A particular case are systems that monitor the drivers in order to detect dangerous levels of drowsiness or other conditions that can impair it's ability to handle the vehicle [IMK+02]. When such situations arise the collision avoidance system takes control of the vehicle or parts of it. An example of such technology is *drive-by-wire* (DbW) [ISS02] where the mechanical systems are replaced by electronic ones improving response times and allowing computer controlled intervention. Replaced systems can be the steering column (steer-by-wire [DAL$^+$01]), the throttle (throttle-by-wire) and the breaks (break-by-wire). Another example is *adaptive cruise control* (ACC) [CJL95], which automatically limits the speed of the vehicle in order to maintain a safety range from nearby vehicles by the use of a sensor.

Systems that take full vehicle control are denominated by vehicle automation. These systems accomplish the same functions as traditional cars but without the driver's actions. It does so by sensing the surrounding environment with technologies such as radars [CDW98], GPS [SNDW99] and computer vision [DZ88]. Advantages of such level of automation are the higher safety, more efficient traffic flow and higher convenience to the drivers.

Intelligent parking lots are a solution for the parking space management problem in ITS. Intelligent parking systems take advantage of advanced technology and communication systems in ITS to provide drivers with parking information and assistance.

Intelligent parking reservation systems allow drivers to remotely reserve a parking space in a desired parking lot removing the search for a free space. In such systems, parking spaces can be reserved through a variety of ways, such as internet [ISN$^+$01] and SMS [HBD10]. In order to determine which parking spaces are free, technologies such as image processing and sensors are used. Image processing techniques use a camera as a sensor, capturing the image and processing it to determine if a figure drawn at each parking lot is visible or hidden by a parked vehicle [YNB12]. A typical sensor-based space detection system consists of wireless sensors placed in the parking space that detect its occupancy state [BDFT09], alternatively ultrasonic sensors can be used [LYG08]. Revenue management systems are an extension to reservation systems by considering the parking price as a new variable. A parking space is reserved with a variable tariff (price paid per hour) and accepting or rejecting new reservations by the ratio between the park capacity and the total number of drivers that desire to park [TL06]. Sensor-based solutions can be expensive and sensors can have a small life time, becoming inaccurate or stop functioning over time. Vehicle Ad Hoc Networks (VANETs) can also be used to track parking space occupancy and for driver guidance [LLZS09].

Automatic parking uses either autonomous vehicles that calculate and perform the precise manoeuvres needed to park or a set of lifts that automatically parks the vehicles. Free space detection can again be used to first detect the available space as well as detecting obstacles through the use of image processing, then a path to the parking space is planned and executed [XCX00]. *Valet parking* systems are an automatic parking system in which the vehicle autonomously looks for a free parking space before parking it, whether it is on-road or off-road. These systems continuously drive around a pre-determined area until finding a free space [CKGC$^+$07]. Another solution are the *robotic parking systems*, in which the driver leaves the vehicle at the entrance of the park where computerized machinery and lifts pick up the vehicle and place it in a shelving-like parking system. This method not only automatically parks the vehicle but also occupies less space than a conventional parking lot [rob].

The work in this dissertation follows a novel approach, proposed in [Fea14], and explained in section 2.1 that differs from all other approaches in the literature. It consists in a intelligent parking lot of self-driven electric vehicles (EV) equipped with vehicular communications with the addition of a valet parking system. These EV communicate are controlled by a Parking Lot Controller that manages the parking lot. The vehicles are parked in a manner that maximises space usage. Due to the lower complexity of this approach compared with robotic parking, the costs of this solution can result in much lower costs for the end user. In addition, the existence of a valet parking system and the fact that the parking is achieved without moving platforms are strong points for this approach.

Foundational Concepts 3

Machine learning is a branch of Artificial Intelligence that deals with systems that can learn from data. It focuses on prediction based on known variables from the dataset available to the system, this data is known as training data. One of the main properties of such systems is the ability to perform with accuracy on new instances after having learned from a dataset.

ML scenarios differ by type, order and method of the *training data*, and the dataset used to evaluate the algorithm, known as *testing data* [MRT12]. Examples of both the training and testing data can be seen in Tables 3.1 and 3.2 respectively. A good example of a state of art ML algorithm could be the *k-nearest neighbours*. It starts by calculating the distance (with an user defined metric) between the instance to predict in testing data and all the instances in the training data. The k training instances with smaller distance to the testing one, are selected as neighbours. Then a majority vote is made to choose the predicted class (see section 3.1.6 for more detailed information on this algorithm).

Such systems have a variety of applications for instance enhanced medical diagnosis, prediction of the incoming number of calls in a call centre, weather prediction and recommendation systems. An example can be seen on Tables 3.1 and 3.2 from the *Iris* dataset [BL13]. Binary classification can be applied on the variable *Species* with the remaining columns as features. The prediction model would learn from the training data on Table 3.1 and test its accuracy with the data on Table 3.2. Table 3.3 shows an example of the k-nearest neighbours algorithm for $k = 3$, the Euclidean distance (Equation 3.9) is calculated between each training instance and the selected testing instance, the k training instances most similar instances are selected as neighbours. The concept of similarity is given by an user defined distance metric (see section 3.1.6 to see more on this). In Table 3.3 the neighbours are displayed in blue. From the k chosen instances a majority vote chooses the predicted class from the chosen instance's classes, since the 3 neighbours in the example are of the class *setosa*, the testing instance's class is predicted as *setosa*.

Table 3.1: Example of training data using the Iris dataset [BL13].

Sepal.L	Sepal.W	Petal.L	Petal.W	Species
5.7	2.6	3.5	1.0	versicolor
4.9	3.1	1.5	0.1	setosa
4.7	3.2	1.6	0.2	setosa
5.0	3.3	1.4	0.2	setosa
5.2	2.7	3.9	1.4	versicolor
5.1	3.8	1.5	0.3	setosa
5.6	2.5	3.9	1.1	versicolor
5.5	2.5	4.0	1.3	versicolor
5.1	3.4	1.5	0.2	setosa
5.7	2.8	4.1	1.3	versicolor

Table 3.2: Example of testing data using the Iris dataset [BL13]. The purpose is to predict the class with the samples on Table 3.1.

Sepal.L	Sepal.W	Petal.L	Petal.W	Predicted	Real
4.7	3.2	1.3	0.2	?	setosa
5.2	3.4	1.4	0.2	?	setosa
5.5	2.4	3.8	1.1	?	versicolor
5.8	2.6	4.0	1.2	?	versicolor

Table 3.3: Example of the *k*-nearest neighbours with the training data of Table 3.1 with the first test instance from Table 3.2 for *k*=3. In blue, the neighbours of the test instance.

Sepal.L	Sepal.W	Petal.L	Petal.W	Species	Euclidean Dist.
4.9	3.1	1.5	0.1	setosa	0.32
5.1	3.8	1.5	0.3	setosa	0.75
4.7	3.2	1.6	0.2	setosa	0.3
5.1	3.4	1.5	0.2	setosa	0.49
5.0	3.3	1.4	0.2	setosa	0.33
5.2	2.7	3.9	1.4	versicolor	2.95
5.6	2.5	3.9	1.1	versicolor	2.98
5.7	2.6	3.5	1.0	versicolor	2.62
5.5	2.5	4.0	1.3	versicolor	3.10
5.7	2.8	4.1	1.3	versicolor	3.20

This chapter intends to be an easy-following introduction to ML by revising its more basic concepts with a brief overview on 1) Supervised and 2) Unsupervised Learning methods. Then, some related research fields such as 3) Feature Selection and 4) Incremental Learning are also approached. This review follows closely the chapters 6 and 7 in [WF05].

3.1 Supervised Learning

In supervised learning the system receives a labelled dataset, the training data, and makes predictions for incoming new data points. The task is to deduce a function

$$y = f(\mathrm{x}, \beta) + \epsilon. \tag{3.1}$$

from the training data $\mathcal{D} = \{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_\mathrm{n}, y_n)\}$ that represents the hidden pattern in the data. The input variables x are commonly d dimensional vectors $\mathrm{x}_\mathrm{i} = [x_{i,1}, ..., x_{i,d}] \in \mathbb{R}^d$, β are the unknown regression parameters to be estimated and ϵ an associated random error. If y is numeric, $y \in \mathbb{R}$, it is named as a **regression** problem and if y is categorical it is named as a **classification** problem [Fig03].

Supervised learning methods can be defined as regression or classification. In this dissertation, we focused more on the regression task due to the nature of our target variable (i.e. parking time). Throughout this section, different types of supervised learning algorithms are revised.

3.1.1 Decision trees

Decision trees [Qui86] are tree-like graphs of decisions and their respective consequences. At first the whole training dataset is placed at the root, then the data is split accordingly to a splitting criterion, dividing the data in many branches (subsets). This process is done recursively for each branch, until each one has only members of the same class or is sufficiently small. The tree is then pruned to prevent *overfitting* and to enhance the tree's performance. The splitting criterion is determined by choosing the attribute that best divides the data in a branch into individual classes. The chosen attribute then becomes the decision attribute in that node [Bad07].

3.1.2 Model trees

The *M5 model tree* [Q$^+$92] are decision trees where the leaves from the grown tree are linear regression models. This model is built using standard regression techniques, however it uses only the attributes used in tests or other linear models in its subtree. The tree is then pruned, selecting for a given node the model that has the lower estimated error, either the model above them or the model subtree. Finally, when a value is predicted by a model tree, the value given by a leaf is adjusted by the predicted values of the nodes in the path from the root to that leaf.

3.1.3 Linear regression

This method is applicable when both the target variable and the attributes are numeric. It expresses the model as a linear combination of the attributes with weights calculated from the training data,

$$y = \mathrm{x}\beta + \epsilon \tag{3.2}$$

where x is a vector of training observations, β are the unknown regression parameters and ϵ an associated random error. We can further represent this as

$$y = w_0 + w_1 x_1 + w_2 x_2 + ... + x_k x_k + \epsilon \tag{3.3}$$

where y is the target variable, $x_1, x_2, ..., x_k$ are the attribute values and $w_0, w_1, ..., w_k$ are the calculated weights. Let $y^{(1)}$ be the class of the first instance and $x_1^{(1)}, x_2^{(1)}, ..., x_k^{(1)}$ be the values of the attributes for the same instance. The predicted value for the first instance class can be expressed as

$$w_0 + w_1 x_1^{(1)} + w_2 x_2^{(1)} + ... + w_k x_k^{(1)} = \sum_{j=0}^{k} w_j x_j^{(1)} \tag{3.4}$$

Linear regression intends to choose the weights $w_0, w_1, ..., w_k$, to minimize the sum of the squares of the differences between the real and the predicted values. The sum of the squares of the differences is written as

$$\sum_{i=1}^{n} (y^{(i)} - \sum_{j=0}^{k} w_j x_j^{(i)})^2 \tag{3.5}$$

where the expression inside the parenthesis is the difference between the ith instance's real and predicted value. This method is better used for functions with linear dependencies because it finds the best-fitting straight line. In the case of a non-linear dependency, this method is not likely the best solution for the data. Linear regression can also be used for classification by using a numeric domain for each class.

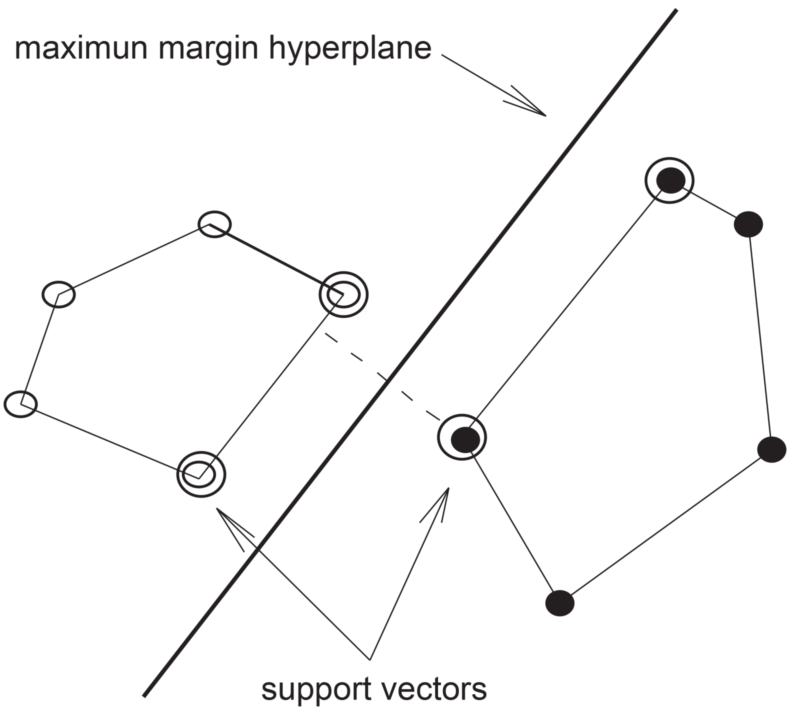


Figure 3.1: A maximum margin hyperplane.

Another family of linear regression methods are the *additive models*. These are non-parametric methods that are based on p-dimensional local averages. Additive models create an estimation of the regression surface by a combination of one-dimensional functions, avoiding the sparsity of high dimensional datasets (*curse of dimensionality*). *Project pursuit regression* [FS81] adapts additive models by first projecting the data matrix of the independent variables in the optimal direction before applying the smoothing functions.

3.1.4 Support vector machines

Support vector machines [CV95] are extended linear models. This method finds a *maximum margin hyperplane* or a set of them. A hyperplane is a linear model that better separates the classes. The instances closer to the maximum margin hyperplane are the *support vectors*. The support vectors define the hyperplane, given the two classes and their respective support vectors their maximum margin hyperplane can be constructed with the rest of the points of the class being irrelevant. An example is shown in Figure 3.1, with the two classes represented by open and full circles respectively.

Kernel functions can be used to extend support vector machines to non-linearly

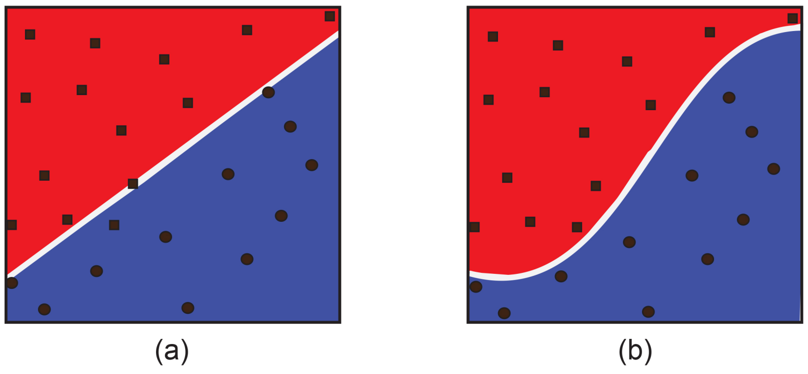


Figure 3.2: In (a) no hyperplane can separate the two classes. In (b) a non-linear mapping is used.

Table 3.4: Examples of kernel functions of the support vector machines.

kernel	formula	parameters
linear	$\mathbf{u}^{\mathrm{T}}\mathbf{v}$	(none)
polynomial	$\gamma(\mathbf{u}^{\mathrm{T}}\mathbf{v} + c_0)^d$	γ, d, c_0
radial	exp-$\gamma\lvert\mathbf{u}\text{-}\mathbf{v}\rvert^2$	γ
sigmoid	tanh$\gamma\mathbf{u}^{\mathrm{T}}\mathbf{v} + c_0$	γ, c_0

separable cases. A way to define a non-linear boundary is to use a non-linear mapping ϕ from the input space $\mathcal{X}$ to a higher-dimensional space, where linear separation is possible. The solution is to use a kernel function K that for two points $x, y \in \mathcal{X}$,

$$K(x, y) = \langle \phi(x), \phi(y) \rangle \tag{3.6}$$

where $\phi : \mathcal{X} \to \mathbb{H}$ is a mapping function to a higher-dimensional space $\mathbb{H}$. K is used as a similarity between elements of the input space $\mathcal{X}$. The kernel function is cheaper to calculate than ϕ and an inner product in $\mathbb{H}$ [MRT12]. In Figure 3.2 an example of a non-linearly separable case is shown, where the two classes are represented by squares and circles respectively. In (a) no hyperplane can separate the two classes, so a non-linear mapping is used instead, separating the two classes (b). Examples of some kernel functions are shown in Table 3.4.

3.1.5 Naive Bayes

Naive Bayes is a method based on Bayes's rule of conditional probability, using all attributes and naively assuming that they are independent and equally important. Even though this is not true in real life data, it shows surprisingly good results. The application of the Bayes theorem is as follows, let $P(C_i|E)$ be the probability that the instance E is of class C_i, then according to Bayes theorem,

$$P(C_i|E) = \frac{P(C_i)P(E|C_i)}{P(E)} \tag{3.7}$$

since $P(E)$ is the same for all classes, it can be ignored since it does not change the value of their probabilities. Considering the attributes independent given the class, $P(C_i|E)$ can be translated into,

$$P(C_i|E) = P(C_i)\prod_{j=1}^{a} P(A_j = v_{jk}|C_i). \tag{3.8}$$

where v_{jk} is the value of the attribute A_j in the instance [DP97].

3.1.6 Instance-based learning

This method compares the training instances with the test ones, by the means of a distance function, to determine which members of the training data are closer to the test instance. Most instance-based learners use Euclidean distance, which is defined as

$$\sqrt{(a_1^1 - a_1^2)^2 + (a_2^1 - a_2^2)^2 + ... + (a_k^1 - a_k^2)^2} \tag{3.9}$$

where $a_1^1, a_2^1, ..., a_k^1$ are the values of the k attributes of an instance and $a_1^2, a_2^2, ..., a_k^2$ of another instance. Because different attributes have different scales, the attribute values are normalized, so that some attributes are not considered less important just because of a smaller scale. The nearest neighbour is found by calculating the distance from the test instance to every member of the training set and selecting the smallest. Such calculations are usually slow since the time to make a single prediction is proportional to the number of training instances. This calculations can be optimized by the use of a binary tree, called *kD-tree.* This tree divides the input space recursively. The tree stores a set of points in k-dimensional space, k being the number of attributes. This strategy is called *k-nearest-neighbours* [Alt92]. To locate the nearest neighbour of a given instance in the tree, it follows the tree down to find the region containing the instance. This method is often much faster than comparing the test instance to every training instance, having a complexity, given by the number of nodes, of $\log_2 n$.

3.1.7 Ensemble learning

Ensemble learning consists on a combination of multiple individual predictive models in order to improve performance [MMSJS12]. Some well known examples of ensemble learning are described in this section.

One example of this techniques is *boosting* [Sch90], which consists in training weak learners and converting them into strong learners. A weak learner is one that only lightly improves a random guess, like a one level decision tree, while a strong learner is ideally one that approximates perfect performance. A set of week learners is trained sequentially and combined, where later trained learners focus more on the previously trained learners mistakes [Zho12]. An example of boosting is the *cubist* R package, that creates M5 models trees iteratively, creating new trees with adjusted versions of the training set outcome. The number of iterations is controlled by the *committees* parameter.

Bagging [Bre96] is a method to generate multiple models and aggregating them into one. In case of a regression problem is an average of all the versions and in case of classification it is made a plurality vote. Given a training data $\mathcal{D} = \{(\mathrm{x}_1, y_1), ..., (\mathrm{x}_\mathrm{n}, y_n)\}$ of size n, where x are the input variables and y are either the class labels or the numerical value, a predictor $\phi(\mathrm{x}, \mathcal{D})$ can be created through some process. Bagging, through a sequence of training sets $\mathcal{D}_k$ of size n, creates a new predictor from the set $\{\phi(\mathrm{x}, \mathcal{D}_k)\}$. If y is numerical, an average of $\{\phi(\mathrm{x}, \mathcal{D}_k)\}$ over k is made,

$$\phi_A(\mathrm{x}) = E_{\mathcal{D}}\phi(\mathrm{x}, \mathcal{D}) \tag{3.10}$$

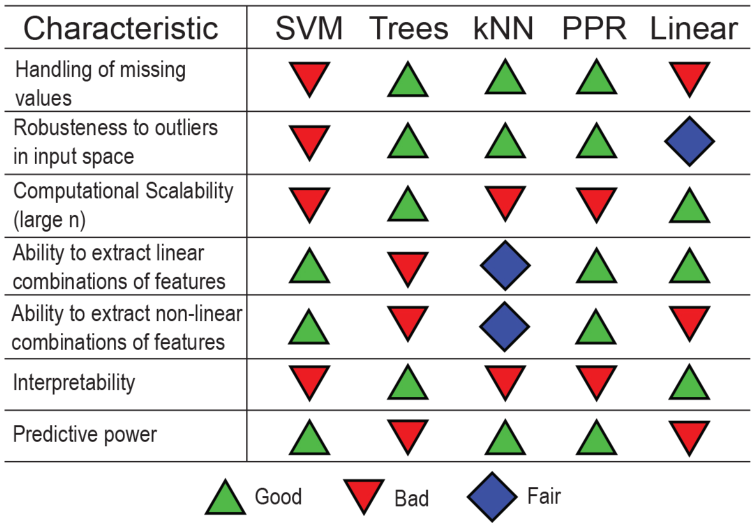
where $E_{\mathcal{D}}$ is the expectation over $\mathcal{D}$ and the subscript A in ϕ_A means aggregation. If the goal is to predict a class, then the aggregation can be made by voting, let $N_j = nr\{k; \phi(\mathrm{x}, \mathcal{D}_k) = j\}$ then

$$\phi_A(\mathrm{x}) = argmax_j N_j \tag{3.11}$$

where j is the index where N is maximum.

However, usually there is only one available training set, instead of multiple ones. In this case, repeated *bootstrap* [ET94] samples $\{\mathcal{D}^{(B)}\}$, drawn with random but with replacement are taken from $\mathcal{D}$. This way the set $\{\phi(\mathrm{x}, \mathcal{D}^{(B)})\}$ is obtained, enabling the process described above.

Random forests [Bre01] are an ensemble of decision trees constructed at training time. In random forests, bagging is used with random attribute selection. Whenever a there is a split in one of the decision trees a random sample of m attributes are selected from the full set of p attributes, and then one of the m attributes is chosen to split the tree. A new sample m is taken whenever there is a split. Usually it is considered $m \approx \sqrt{p}$, which means that the number of randomly chosen attributes is the squared root of the total number of attributes. The trees then vote for the most popular class, which will finally be attributed to the tested instance [Zho12].



Characteristic	SVM	Trees	kNN	PPR	Linear
Handling of missing values	▼	▲	▲	▲	▼
Robusteness to outliers in input space	▼	▲	▲	▲	◆
Computational Scalability (large n)	▼	▲	▼	▼	▲
Ability to extract linear combinations of features	▲	▼	◆	▲	▲
Ability to extract non-linear combinations of features	▲	▼	◆	▲	▼
Interpretability	▼	▲	▼	▼	▲
Predictive power	▲	▼	▲	▲	▼

▲ Good ▼ Bad ◆ Fair

Figure 3.3: Summary of a number of supervised learning algorithms. Based on Table 10.1 in [HTF$^+$09]

Figure 3.3 shows a summary of a number of supervised learning algorithms described above. The next section describes unsupervised learning and some well known algorithms in this category, focusing especially on *clustering* techniques.

3.2 Unsupervised Learning

Unsupervised learning is the task of finding significant hidden patterns in unlabelled data. Unsupervised learning differs from supervised learning in that, since there are no labels associated with the data, there is no relative error to evaluate the solution. Examples of such techniques are *association rules* [AIS93], which is a method to discover relationships between variables in databases using various *interestingness measures* [GH06], *sequential pattern mining* [AS95], which is analogous to association rules but considering the order of the data items, *hidden Markov models* [RJ86], in these models it is assumed that the system is a Markov process [Dyn] with hidden states, and finally *clustering techniques* which are described in more detail along the next section.

3.2.1 Clustering

Clustering is the task of dividing a dataset into groups, called *clusters*. These clusters should show the hidden patterns in the data, since elements in the same clusters should reflect a strong similarity between each other and differ from elements in other clusters.

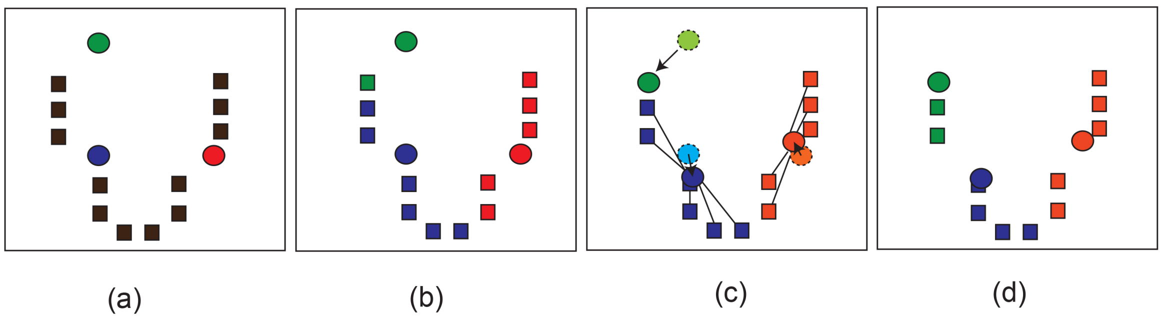


Figure 3.4: Example of the standard k-means algorithm for k=3. Data points are represented by squares while the centres of the clusters are the circles.

Depending on the clustering techniques, elements can be in more than one group or be exclusive to one. Groups can be probabilistic where each element belongs to one with a certain probability or even hierarchical, where there is a division of the elements by a hierarchy of groups. To decide which instances to agglomerate in a cluster a metric of dissimilarity between the clusters is used, an example of such metric is the Euclidean distance (Equation 3.9).

The most common clustering technique is called *k-means* [M+67]. The method works by first choosing k random points as cluster centres, all instances of the data are then assigned to their closest centre, according to the Euclidean distance (see Equation 3.9). For each cluster, the *centroids* (a centroid is the arithmetic mean of all points in a designated region) are calculated and taken as the new cluster centres. The assignment process is repeated until the same instances are assigned to each cluster in consecutive cycles. The number of clusters k is a user selected parameter. An example of the standard k-means algorithm is shown in Figure 3.4, in (a), $k = 3$ random centres are selected, in (b) k clusters result from associating the data points to the closest centre. The centroid of each cluster becomes the new centre (c). In (d) the steps (b) and (c) are repeated until the clusters stabilize.

Another clustering technique is the *expectation-maximization (EM) algorithm* [DLR77]. This algorithm differs with k-means in that EM finds clusters by the use of a mixture of Gaussians fitting the dataset. First the expectation step is performed, which is responsible to estimate the probability of each element belonging to each cluster, and then the maximization step, which is the calculation of the parameters of the probability distribution function of each class. The iteration ends with the algorithm performing a convergence test, which verifies if the difference between the averages of the class distribution of the current iteration and the previous is smaller than a defined threshold. In the case of being bigger than the threshold, the algorithm

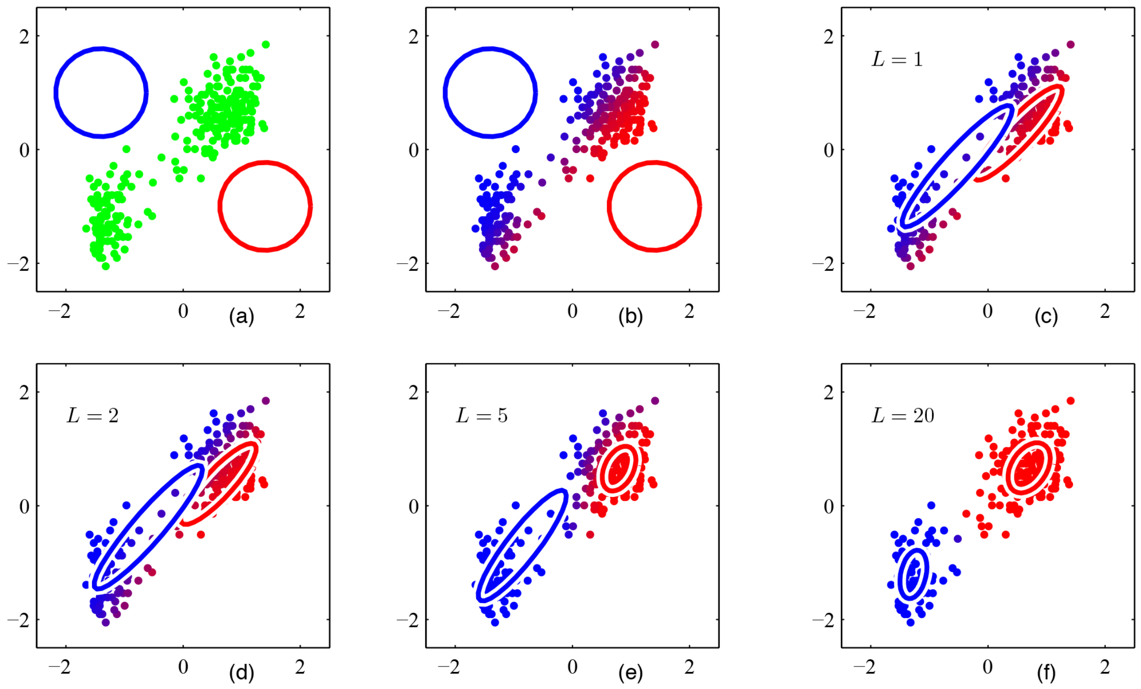


Figure 3.5: Example of the Expectation-Maximization Algorithm [B+06].

continues iterating, otherwise it stops. Figure 3.5 shows an example of the algorithm. In (a) the data points are in green and the initial centres (the circles represent the standard deviation of the Gaussian components) in green and blue. After an initial expectation step (b) each data point is associated with each centre by the probability of belonging to them, red and blue for each centre and purple for belonging to either. Plot (c) shows the state after the first maximization step, the mean of both clusters moved to the mean of the dataset (centre of mass). (d), (e) and (f) show the results after 2, 5 and 20 cycles (L). In (f) the algorithm is close to finish [B+06].

Hierarchical clustering [Joh67] is a method that builds a hierarchy of clusters. Hierarchical methods can be classified as *agglomerative*, where each observation starts in its own cluster and the clusters are merged (moving up in hierarchy), or *divisive* where all observations initially form one cluster and are recursively divided (moving down in the hierarchy). This method differs with the previously mentioned ones in that it determines the number of clusters k automatically.

There are many techniques to determine the number of clusters k, in which the data is grouped. In the next subsection such techniques are described, as silhouette, information theory or bayesian models.

3.2.1.1 Determining the number of clusters

Silhouette [Rou87] is a graphical method to represent and validate clusters. Each silhouette represents a cluster and its tightness and separation. The average silhouette is used to choose the most suited number of clusters to the designated problem. Let $k \geq 2$ be the number of clusters obtained by any clustering technique, let i be an object in the data and A be the cluster in which i is comprised, $i \in A$. If the number of elements in A is larger than 1, then we can compute $a(i)$ as the average dissimilarity between the i and all other objects in A. This dissimilarity can be interpreted as how well does the object i fit in its current cluster A. Let C be another cluster, with $C \neq A$, and $d(i, C)$ be the dissimilarity of i to all objects in C. For all clusters $C \neq A$, $d(i, C)$ is calculated. After that, $b(i) = \text{minimum}\{d(i, C)\}$ is selected. The selected cluster B, is called the *neighbour* of i. The silhouette, $s(i)$, is calculated as,

$$\begin{cases} 1 - \frac{a(i)}{b(i)} & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ \frac{b(i)}{a(i)} - 1 & \text{if } a(i) > b(i) \end{cases} \tag{3.12}$$

This equation can also be written as,

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{3.13}$$

When a cluster A contains only one object, $s(i) = 0$. If $s(i)$ is close to 1 then $a(i)$ is much smaller than $b(i)$, which means that the object is well placed in its cluster, since the dissimilarity from the members in its own cluster is small and the dissimilarity to the members of the nearest group is large. When $s(i)$ is close to 0, then $a(i)$ and $b(i)$ are approximately equal, being not clear whether the object i should have been assigned to A or B. If $s(i)$ is close to -1 then a(i) is much larger than $b(i)$, therefore i must have been misplaced in cluster A.

The *Akaike information criterion* (*AIC*) is a information based method to measure the information lost by using a certain model in a given a dataset. For any model, *AIC* is expressed as,

$$AIC = 2p - 2ln(\mathcal{L}) \tag{3.14}$$

where p is the number of parameters and $\mathcal{L}$ is the maximum-likelihood of the model. Given a set of models for the data, the model with the minimum value of *AIC* is the model closest to the process representing the data. Let g_1 and g_2 be two candidate models to represent the process f, the information loss from using g_1 to represent f would be $D_{KL}(f||g_1)$ using the *Kullback-Leibler divergence* (Equation 3.17), same calculations are made for g_2. The candidate model that minimizes the information loss would be chosen. But since f is not known, an estimation of the information lost must be made through AIC. Let M be the set of $AIC = \{AIC_1, AIC_2, ..., AIC_R\}$ values of the R candidate models and AIC_{min} the minimum value in M. Then $e^{\frac{AIC_{min}-AIC_j}{2}}$ is the relative probability that the jth model minimizes the estimated information loss [BA02]. $AICc$ [HT89] adds a correction for finite sample sizes to AIC, decreasing the probability of overfitting (selecting models that have too many parameters) [CH08]. $AICc$ can be defined as:

$$AICc = AIC + \frac{2p(p+1)}{n-p-1} \tag{3.15}$$

When all the models have the same p, then AICc and AIC give identical results.

The *Bayesian information criterion* (BIC) [S+78] is defined, for large n, as,

$$BIC = -2ln(\mathcal{L}) + p \cdot log(n) \tag{3.16}$$

where $\mathcal{L}$ is the maximum value of the likelihood function of the model, p is the number of parameters and n is the size of the dataset. BIC tries to find the true model among its candidates, a lower BIC means that a certain model is the more likely to be the true model. The penalization of parameters in BIC is stronger than in AIC, decreasing the probability of overfitting but increasing the underfitting one.

3.3 Feature Selection

Feature selection is a process to eliminate redundant or irrelevant attributes, due to the negative effect that these attributes can cause in the model. Reducing the dimensionality of the data makes the interpretation of the target variable easier and can improve the performance of the algorithms. These techniques can be used in combination with both supervised and unsupervised learning methods in order to enhance their performance. In this section it is described three methods of feature selection, *principal component analysis*, *factor analysis* and *information gain*.

Principal component analysis (PCA) [Jol05] is a feature generation method that uses a different orthogonal coordinate system to represent the data, influenced by the points of each specific dataset. The first axis is placed in the direction of the greatest variance of points, the second axis is perpendicular to it. In two dimensions, there is only one way for the second axis to be placed, but in three dimensions the axis should be in the direction that maximizes the variance. The axis are continuously placed in the direction of the maximum remaining variance. Each axis is called *component*, all components can be used for the computing or just the first components that

Table 3.5: Loadings for the principal component analysis of Table 3.1.

	Comp.1	**Comp.2**	**Comp.3**	**Comp.4**
Sepal.L	0.205	-0.489	0.823	0.204
Sepal.W	-0.266	-0.847	-0.370	-0.272
Petal.L	0.866		-0.133	-0.479
Petal.W	0.371	-0.198	-0.411	0.809

represent the highest variance, these are called the *principal components*. The results of PCA are evaluated with respect to the **scores** (the values of a transformed variable related to a certain instance) and the **loadings** (the value each original feature must be multiplied to obtain the component score). Table 3.5 shows the loadings of the principal component analysis of Table 3.1. The attribute with lowest impact is the Sepal.L as shown by the first component (its absolute value is the lowest).

Factor analysis [Har60] is a method closely related to principal component analysis but not identical. This method considers a hypothesis in which is not the attribute X causing Y or Y causing X, but a common causal model. Factor analysis searches variance in correlated variables to reduce data dimensionality. The variables are modelled as linear combinations of the potential factors plus error terms derived from regression techniques.

Information gain (or Kullback–Leibler divergence [KL51]) is a measure of the information lost between two probability functions f and g. This approach can be modelled as follows,

$$I(f,g) = \int f(x)log(\frac{f(x)}{g(x|\theta)})dx \tag{3.17}$$

where $I(f,g)$ is the information lost when the function g is used to approximate f [BA02].

3.4 Incremental learning

At times the constraints of a problem require a change of approach to the data, it is necessary to move away from finite training sets and static models to a dataset with unlimited data streams that add new elements over time. This new approach reveals newly added complexity to the previous mentioned ML techniques, since the irregularities of the data may now evolve over time and no longer being considered independent and identically distributed [GRSdC10].

One example of an algorithm that allows for such constraints is the *perceptron.* This algorithm is a linear classifier that allows for online learning by processing one training instance at a time. Given a training instance $\mathcal{D}_k = (x_k, y_k)$, let $w = (w_1, w_2, ..., w_n)$ be the current values of the weights and φ a threshold. For x_k we calculate a potential neuron:

$$\varepsilon_k = \sum_{i=1}^{n} w_i x_i k + \varphi \tag{3.18}$$

The output of such neuron is given by

$$\hat{y}_k = \begin{cases} 1 & (if \varepsilon \geq 0) \\ 0 & (otherwise) \end{cases}$$

The weights and threshold are then updated with the given instance,

$$w_{new} = w_{old} = +\lambda(y_k - \hat{y}_k)x_k$$

$$\varphi_{new} = \varphi_{old} + \lambda(y_k - \hat{y}_k)$$

where λ is a parameter called the **learning rate**. The process is repeated for all instances of the training dataset [Ros58]. *Multilayer perceptron* is an extension of the linear perceptron to be able to distinguish data that is not linearly separable, this is done through *backpropagation* [RHW88] or the use of the *delta rule.* The delta rule is one of the most commonly used learning rules, this method works by comparing the output of a prediction model with the real answer, if the difference is null then no learning resulted, otherwise the weights are adjusted in order to reduce the difference. This difference is minimized through *gradient descent* by moving the weight vector through the weight space (consisting of all possible values of all the neuron's weights) by the gradient of the error function with respect to each weight.

Some of the methods described along this chapter were used to solve the problem in this dissertation. This methodology is described on the next chapter. First, a general look at the methodology is given, next the generation of profiles is explained, followed by the parking time prediction and the incremental interval generation methods. Finally we take a look at the parking lane selection strategy.

Methodology 4

Our methodology consists on the following four steps: it starts by (A) dividing the original dataset in k smaller ones, containing users with similar parking habits between each other in each of the k datasets; then, (B) data driven regression is performed over the newly created sub-datasets. Thirdly, a parking time interval is generated (C) based on such predictions and on the residuals of previous predictions (difference between a predicted value ($\hat{y}$) and its real one, y). Finally the selected lane (D) will be the one which minimizes the likelihood of performing *unnecessary* vehicle movements [1]. This methodology is summarized in Fig. 4.1 and explained in detail throughout this chapter.

4.1 Profile Generation

Let $\mathbb{X} = \{X_1, X_2, ..., X_n\}$ be n timestamped data records on the parking lot entries describing the entry/exit behaviours of ρ distinct users. Let $U_i \subseteq \mathbb{X}$ denote the records of an individual user i (i.e. $U_{i=1}^{\rho} \equiv \mathbb{X}$) and Ψ_i describe the sample-based probability density function (*p.d.f.*) of its parking time habits. An example of the p.d.f. of a given user can be seen in Figure 4.2, the axis represents the parking time in seconds, this p.d.f shows a bimodal distribution, with the two highest probabilities falling on the 14000 seconds ($\approx$ 4 hours) and 32000 seconds ($\approx$ 9 hours). A clustering process is firstly made on $\mathbb{X}$ based on the extracted Ψ_i. The resulting k clusters can be defined as $\Pi = \{\pi_1, \pi_2, ..., \pi_k\}$. They will comprise sub-datasets containing data records on users having similar **profiles** (i.e. parking time-habits). Consequently, $\mathbb{X} \equiv \bigcup_{i=1}^{k} \pi_i$.

[1]Whenever a given vehicle c exits, all its lane's vehicles standing between c and the parking lot exit, have to be moved to a buffer zone. Such movements could be avoided by an exit-oriented sorting of each lane's vehicles.

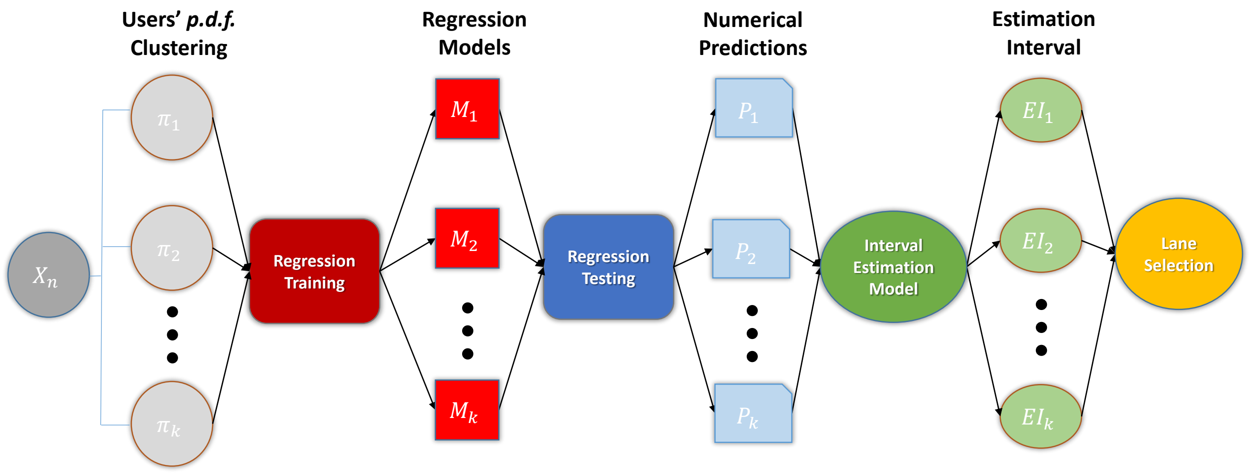


Figure 4.1: An illustration on the different steps of the proposed methodology.

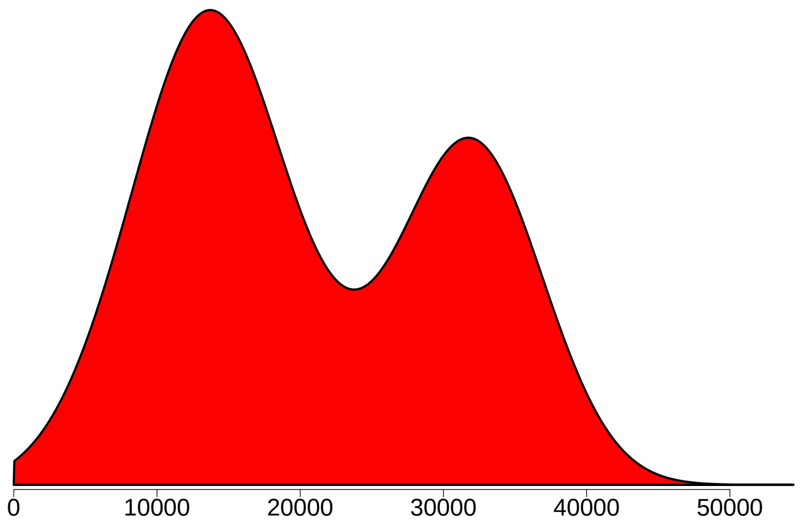


Figure 4.2: Example of the probability density function of a given user. The axis represents the parking time in seconds.

4.2 Parking Time Prediction

To perform the parking time prediction, we propose to use **data driven regression**. In regression, the goal is to determine a function $f(Z, \theta)$, given the input independent variables, Z, and the real values of the dependent variables, θ. The output of the model is not necessarily equal to the real value, due to noise in the data and/or limited number of entries. Consequently, a regression model commonly comprises an error e. The function f can be expressed as follows:

$$Y \approx f(Z, \theta) + e \tag{4.1}$$

Let $\mathbb{M} = \{M_{\pi_1}, M_{\pi_2}, ..., M_{\pi_k}\}$ be the set of k regression models and p_{j,π_i} denote the parking time prediction for a given timestamped user entrance with the profile π_i. $\mathbb{M}$ results of applying an induction method of interest to the datasets in Π. By doing so, the authors expect to approximate the real vehicles parking time given a set of describing variables (i.e.: Z). Alternatively, regression was applied to the ρ users (an individual approach), creating ρ distinct regression models. The predictions produced by $\mathbb{M}$ revealed better results, since within a group with more than one user, regression can be enhanced by having available data of all users within said group, instead of only one in an individual approach.

4.3 Incremental Interval Generation

Given a prediction for the parking time of an user timestamped entrance (i.e. p_{j,π_i}), it is possible to estimate an **interval** for this value based on the residuals produced by its regression model. Both the individual and the group residuals e_{π_i} were initially considered, but the interval generation through the use of the group residuals outperformed the one using the individual ones. Additionally, we propose to do the interval estimation by employing the group residuals' *quantiles*. A quantile is a point taken from a cumulative distribution function of a variable. The first quantile represents the point that is greater than 25% of the data, while the third quantile the point that is greater than 75%. Through the use of the quantiles, residuals' outliers are removed, which showed to allow for a more reliable interval estimation compared to the non-reduced residuals. Let $e_{1,i}$ and $e_{3,i}$ denote the first and third quantiles of the regression *residuals* produced by a given model M_{π_i} on the previously tested data records in π_i. Our baseline interval I is given by the following equation:

$$I_{j,\pi_i} = [p_{j,\pi_i} - e_{1,\pi_i}, p_{j,\pi_i} + e_{3,\pi_i}] \tag{4.2}$$

Let a **hit** occur every time the real parking time is contained within the interval estimated. Otherwise, we consider the occurrence of a **miss**. Our goal is to produce intervals in order to maximize the number of hits and, at the same time, to minimize its **width**. To do so, we propose to extend the baseline described in eq. (4.2) by employing a *self-adaptive* strategy. Such strategy consists on multiplying the quantile-based interval width by a $0 \leq \beta \leq 2$ (starting on $\beta = 1$). This value is *incrementally* updated whenever an user of π_i leaves the parking lot (i.e. each time a newly real parking time is known on π_i). Let α_{π_i} denote the number of *consecutive* misses/hits of our interval prediction method in π_i. Whenever $\alpha_{\pi_i} > \alpha_{th}$, the value of β is incremented/decremented by τ. α_{th} and τ are two user-defined parameters setting how **reactive** the interval prediction model should be. Consequently, it is possible to re-write the Eq. (4.2) into the following one:

$$I_{j,\pi_i} = [p_{j,\pi_i} - \Delta, p_{j,\pi_i} + \Delta], \Delta = (e_{3,\pi_i} - e_{1,\pi_i}) \times \beta \tag{4.3}$$

Every time that a sequence miss/hit or hit/miss occurs, the respective α value is set to 0. The β ends up by controlling the interval width: the described algorithm aims to **adapt itself** to the current scenario by *narrowing* the intervals width whenever it is getting multiple hits or by *stretching* itself on the opposite scenario. In comparison with the baseline (see Equation 4.2), the adaptive method shows higher hit percentage and smaller intervals.

4.4 Parking Lane Selection

In this work, the parking lot is assumed to follow a rectangular layout where the entrance and the exit are the same. It is possible to represent it as a $l \times r$ matrix, where l, r sets the number of **lanes** and the maximum number of vehicles in each lane, respectively. When a vehicle enters the parking lot,it is necessary to select a lane κ to park it in. Such selection should minimize the number of unnecessary vehicle movements (i.e. ϑ_κ). Consequently, each lane has an associated **score** W_κ. It can be faced as a likelihood of that selection force unnecessary movements given the i) current interval prediction for the newly arrived user (I_{j,π_i}) and ii) the vehicles already parked in κ. The lane with lowest score is predicted to be the one that minimizes ϑ_κ.

Empty lanes have a predefined score of $W = 1$ while a full one has $W = \infty$. Let h be the *last* vehicle in κ (i.e. the vehicle most recently parked in κ), I^U_{j,π_i} be the upper limit and I^L_{h,π_b} be the lower limit of the estimated interval (note that the vehicle's j profile, π_i, may be (or not) the same of the vehicle h, π_b). If $I^U_{h,\pi_b} < I^L_{j,\pi_i}$, it is expected that the vehicle j of profile π_i exits the parking lot first than h (e.g.: Fig. 4.3-c). In this case, $W_\kappa = \infty$. If $I^U_{j,\pi_i} < I^L_{h,\pi_b}$, then it is expected that j and h can leave the parking lot provoking no unnecessary movements (i.e.: $\vartheta_\kappa = 0$; e.g.: Fig. 4.3-a). Consequently, the score is then $W_\kappa = 0$ on this case. Otherwise, W_κ can be computed as follows

$$W_\kappa = \frac{I^U_{j,\pi_i} - I^L_{h,\pi_b}}{I^U_{h,\pi_i} - I^L_{h,\pi_b}} + \frac{(N_\kappa - 1)^4}{r} \tag{4.4}$$

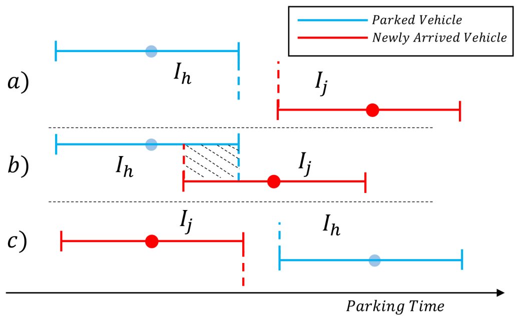


Figure 4.3: In a), the upper limit of I_h is lower than the lower limit of I_j, so h is expected to leave the parking lot first than j. In b) there is an overlap between the two intervals. Its width is used to compute the lane's score. Finally, c) is the opposite scenario of a).

where N_κ stands for the number of vehicles currently in κ. This approach is inspired on the typical *p-value* statistical test considering a null hypothesis by setting the *extreme* data point as I^U_{j,π_i} and I_{h,π_i} as a *rough* approximation on the parking time distribution function for the parked vehicle h. The second term of eq. (4.4) is an exponential weight which aims to express the possible cost of having unnecessary vehicle movements caused by assigning the newly arrived vehicle j to the lane κ.

The weight W_κ can then be summarized in the equation,

$$W_\kappa = \begin{cases} 0 & \text{if } \kappa \text{ is empty} \\ \infty & \text{if } I^U_{h,\pi_b} < I^L_{j,\pi_i} \\ 0 & \text{if } I^U_{j,\pi_i} < I^L_{h,\pi_b} \\ \frac{I^U_{j,\pi_i} - I^L_{h,\pi_b}}{I^U_{h,\pi_i} - I^L_{h,\pi_b}} + \frac{(N_\kappa - 1)^4}{r} & \text{otherwise} \end{cases}$$

The next chapter shows the performed experiments with the described methodology along this chapter. First presents the two case studies used for the experiments, next the experimental setup is described, followed by an explanation of how the evaluation of the experiments was made. Then, the results of the performed experiments and finally a discussion of the results.

Experiments 5

In this chapter we present the performed experiments. Firstly, the real world case study used to test our methodology is presented, then the experimental setup is described which is followed by the evaluation methods used to validate our results. To conclude the chapter, the results of the experiments are shown followed by a discussion of this results.

5.1 Case Study

The case study used on this work consists on two parking lots, one from the Faculty of Science of University of Porto, Portugal (case study A) and other from the Faculty of Engineering of University of Porto, Portugal (case study B).

Each data record has the following features: (i) an user ID, (ii, iii) two timestamps for the parking entry/exit, (iv) type of day (e.g.: Monday), (v) holiday/not-holiday boolean and, finally, the (vi) department, (vii) sex and (viii) job role (e.g. Full Professor).

For the case study A, the data of 309 users during the year of 2013 was used to validate our methodology, while in case study B the data of 323 users. The parking lot, for case study A, has the capacity to hold up to 100 vehicles. Since 96.4% of the data entries are in week days, only the workdays are considered in this study.

Ideally all data entries would have their entry and exit times properly labelled. However, it does not happen in this case because the parking entries/exits are not fully monitored. Consequently, there are entries without exits and vice-versa. To tackle such issue, a preprocessing task to pair the entries with the exits was performed. All the resulting data records with parking time smaller than 10 minutes or higher than 16 hours were removed. For the same reasons, we have also filtered the parking lot users by using the data records of the top-75%, regarding their number of parking entries.

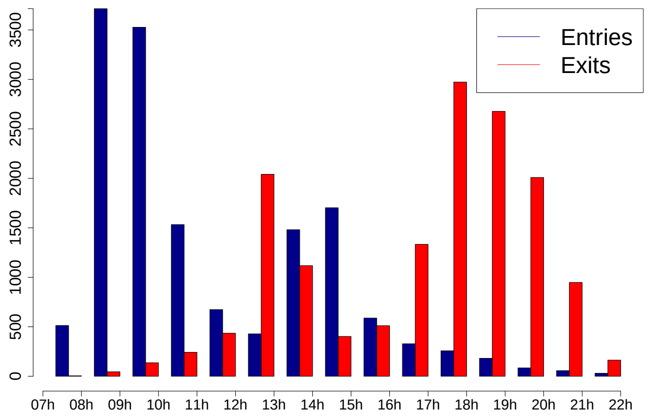


Figure 5.1: Case study B: barplot chart representing histograms for the Entry/Exit times between 7am and 10pm.

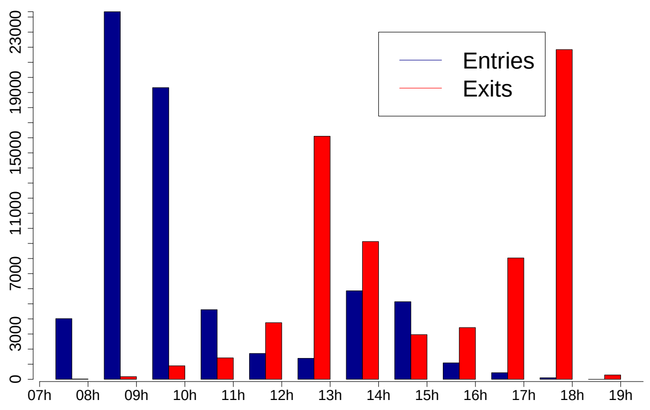


Figure 5.2: Case study B: barplot chart representing histograms for the Entry/Exit times between 7am and 7pm.

In the resulting dataset, for case study A, the average parking time is 5 hours and 25 minutes and with a standard deviation of 3 hours and 8 minutes, while for case study B, the average parking time is of 5 hours and the standard deviation of 2 hours and 30 minutes. Fig. 5.1 exhibits two histograms representing the hourly frequencies on the entry and exit times of case study A and Fig. 5.2 of case study B. It is possible to observe that the main entry times are between 8am and 10am and the main exit times between 5pm and 8pm. The vehicle's exits from the parking lot follows a bimodal distribution, with the modes at lunch time (between 12am and 2pm) and at late afternoon (between 5pm and 7pm).

5.2 Experimental Setup

The initial dataset was divided in a training set (January to October) and a test set (November). All experiments were conducted using R Software [Tea12]. The algorithms used were the k-Nearest Neighbours (kNN) [Alt92], the Random Forests (RF) [Bre01], the Projection Pursuit Regression (PPR) [FS81], the Support Vector Machines (SVM) [CV95] and the Cubist [KWKC12] from the R packages [`kknn`], [`randomForests`], [`stats`], [`e1071`] and [`Cubist`].

Regarding the feature selection, a well-known state-of-the-art technique was used: Principal Component Analysis (PCA) [Jol05]. The tested features were type of day, holiday/not-holiday boolean variable and the user's department, sex and job role. For clustering we used the Expectation-Maximization algorithm with the R package [`MClust`]. This algorithm was chosen due to being able to determine the optimal number of clusters automatically based on Bayesian Information Criterion [DLR77].

The last 2 weeks of the training set were used for model selection. In this stage, the following parameters were tested for each algorithm: for kNN, $distance = [1..5]$, $kMax = [2..15]$ and the kernels: rectangular, triangular, epanechnikov, gaussian, rank and optimal, for RF $mtry = \{3, 4, 5\}$ and $ntrees = \{500, 750, 1000\}$, for PPR $nterms = \{2, 3, 4\}$ and $max.terms = \{5, 6, 7, 8\}$ and for SVM the kernels: linear, radial, polynomial and sigmoid. The best pair (algorithm,parameter setting) was selected to perform the numerical prediction in the test set.

Finally, the reactiveness parameters on the interval estimation model (τ, α_{th}) were set for the values 0.1 and 3, respectively.

To evaluate our method performance, we considered a baseline naive strategy. It consists on directing the newly arrived vehicle to the leftmost lane κ with an empty space. A series of simulations were conducted to compare the parking lot behaviour using the aforementioned lane selection strategies (i.e. *naive* and *smart*). Multiple parking layouts were considered on this series of simulations. It aimed to demonstrate that the strategies behaviour is **independent** on the parking layout. The averaged maximum number of parked cars on a daily basis on the considered dataset is 50 for case study A and 210 for case study B. Consequently, every parked layouts with a capacity between 50 and 80 vehicles (i.e.: the 1^{st} quantile) containing, at least, 8 lanes, were considered on our experiences in case study A and capacity between 196 and 256 while containing at least 13 lanes for case study B.

5.3 Evaluation

The root-mean-squared-error (RMSE) and the mean absolute error (MAE) were the metrics used to evaluate the predictions. They can be defined as follows:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{g}(\hat{y}_t - y_t)^2}{g}} \tag{5.1}$$

$$MAE = \frac{\sum_{t=1}^{g}|\hat{y}_t - y_t|}{g} \tag{5.2}$$

where $\hat{y}$ is the predicted value, y the real one and g is the number of samples.

The parking time estimation interval is evaluated in two forms, a percentage of hits and a ratio between the hits and its width. If for a sample s there is a hit, then $hit_s = 1$, otherwise $hit_s = 0$. The ratio can be defined as:

$$ratio = \sum_{s=1}^{g} hit_s \times \frac{1}{\delta_I \times g} \tag{5.3}$$

where δ_I is the width of the estimation interval and g is the number of considered samples.

The evaluation criteria employed in the simulation was the total number of unnecessary vehicle movements forced by a given strategy (i.e., UM). Let us consider a exiting vehicle c, parked in a lane κ with g vehicles, in position i. The unnecessary number of movements UM caused for c to exit the parking lot can be computed as:

$$UM = \sum_{j=1}^{g-i} j \tag{5.4}$$

Let us consider a lane with $g = 5$ vehicles where the vehicle on the position $i = 2$ is requested to exit as an exemplification for the calculus of MU. In this case, $MU = 3 + 2 + 1 = 6$.

5.4 Results

The obtained results are three fold: (1) the PCA results has recommended to remove the user's sex and the holiday feature from the original set in both case studies. Table 5.1 and 5.2 show the loadings for the PCA of both the case study A and B, respectively. In Fig. 5.3 and 5.4 it can be seen the plot of the PCA for both case studies. (2) Table 5.3 exhibits the results of the numerical prediction using the remaining feature set for each profile π_i, by pointing the number of users contained in each group and the (RMSE,MAE) obtained in each one of them. (3) Table 5.5 shows the results from

Table 5.1: Loadings for the principal component analysis for the case study A.

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.5
Holiday					1.0
DayType		-0.99			
Department	0.16		0.98		
Sex				-1.0	
Job Role	0.98		-0.17		

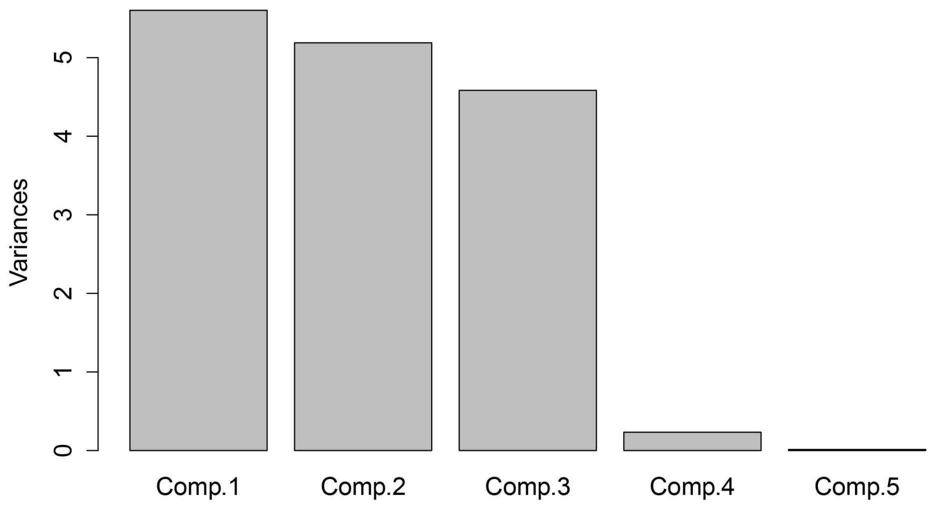


Figure 5.3: Plot of the principal component analysis for the case study A.

Table 5.2: Loadings for the principal component analysis for the case study B.

	Comp.1	Comp.2	Comp.3	Comp.4
Holiday				1.0
DayType		-1.0		
Sex			-1.0	
Job Role	-1.0			

the parking simulation in every tested configurations, with the number of unnecessary vehicle movements, μ for both strategies. The intervals generated had **65%** hits and an average interval width of $\approx$ **11000** seconds. The *smart* strategy overcomes the *naive* one in all the considered configurations.

5.5 Discussion

Both Table 5.3 and Table 5.4 exhibit a large variation on RMSE/MAE produced by the models of the different groups. The groups size is also different from group to group.

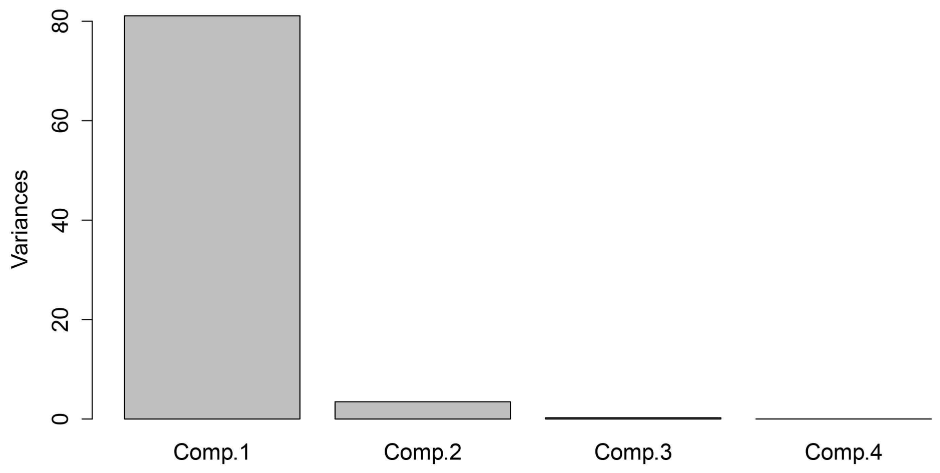


Figure 5.4: Plot of the principal component analysis for the case study B.

Table 5.3: Results from the numeric prediction for case study A.

Group	# of Individuals	RMSE	MAE	Hit %	Interval
1	11	5124	3320	63	8942
2	9	4804	3255	66	3862
3	3	7047	5235	68	9584
4	6	4644	4047	78	9764
5	1	7716	5458	82	3482
6	1	376	340	72	3504
7	5	3968	3317	68	9196
8	7	7618	6101	58	11738
9	11	9106	7628	53	11900
10	6	8244	7403	55	12560
11	4	2609	2058	72	5255
12	10	7871	5436	67	9583
13	6	8901	5789	72	9558
14	7	8595	6883	54	11228
15	4	5981	4804	50	6258
16	10	6682	5356	70	10293
17	1	361	298	50	3158
	W.Average	**6601**	**5076**	**65**	**11188**

These groups can be faced as **profiles** which describe the *typical* parking behavior of the users within. It is possible to observe that some groups contain only one user (i.e. 5,6,17 in case study A and 8,14 in case study B) which indicates that they have a completely different profile than the remaining ones. So far, such profiles are only based on each user's parking time (namely, by using the Euclidean Distance over their *p.d.f.*). However, some users can experience large variations on their parking time

Table 5.4: Results from the numeric prediction for case study B.

Group	# of Individuals	RMSE	MAE	Hit %	Interval
1	9	3266	1884	65	1481
2	20	2880	1924	63	3257
3	4	2854	2088	62	4379
4	3	1823	1300	63	2448
5	15	2902	1983	68	2751
6	2	1687	1092	65	2103
7	2	628	503	60	1151
8	1	1031	688	61	1488
9	7	1984	1229	64	1836
10	24	5370	3670	64	5951
11	5	4753	3012	63	4701
12	61	7683	6371	65	14170
13	3	2471	1346	71	1633
14	1	278	244	50	566
15	7	2985	2134	69	4393
16	13	3349	2230	66	3953
17	25	5705	4284	69	9364
18	5	2346	1529	61	2070
19	21	5956	4289	66	7613
20	8	4871	3874	68	9199
21	46	6801	5382	63	12082
22	3	3287	2289	75	5942
23	15	4166	2637	67	3677
24	14	5776	3931	66	5389
25	10	3324	1972	67	3178
26	2	608	257	95	2062
27	11	2512	1554	69	3037
28	38	7014	5018	64	8630
	W.Average	**5263**	**3903**	**65**	**7635**

depending on some subsets of feature values (i.e. to enter the parking lot at morning or at afternoon). This fact can partially explain the above mentioned RMSE/MAE variability.

The averaged hits percentage (65% in both case studies) and its large width uncover the stochasticity of the parking time variable given the current feature set. In fact, it is reasonable to admit that we may need other features to improve our prediction model such as weather or event-based ones (e.g. a sunny day or a special soccer match may reduce/increase the parking time). However, we cannot sustain these insights on

Table 5.5: Simulation results with the number of unnecessary vehicle movements for both strategies for case study A.

Config.	Naive	Smart	Config.	Naive	Smart
10x05	1665	**1379**	*05x10*	7799	**7540**
11x05	1482	**1205**	*05x11*	7817	**7615**
12x05	1255	**1074**	*05x12*	7817	**7633**
13x05	1074	**914**	*05x13*	7817	**7633**
14x05	937	**813**	*05x14*	7817	**7633**
15x05	811	**771**	*05x15*	7817	**7633**
09x06	2234	**2032**	*06x09*	5596	**5423**
10x06	1819	**1583**	*06x10*	5596	**5444**
11x06	1510	**1282**	*06x11*	5596	**5453**
12x06	1255	**1139**	*06x12*	5596	**5453**
13x06	1074	**930**	*06x13*	5596	**5453**
08x07	2808	**2520**	*07x08*	3818	**3545**
09x07	2248	**2116**	*07x09*	3818	**3545**
10x07	1819	**1616**	*07x10*	3818	**3551**
11x07	1510	**1303**	*07x11*	3818	**3551**
07x08	3818	**3545**	*08x07*	2808	**2520**
09x08	2248	**2116**	*08x09*	2808	**2535**
10x08	1819	**1617**	*08x10*	2808	**2535**
08x08	2808	**2535**			

Table 5.6: Simulation results with the number of unnecessary vehicle movements for both strategies for case study B.

Config.	Naive	Smart	Config.	Naive	Smart
14x13	36552	**35413**	*13x14*	42052	**40719**
15x13	32394	**31426**	*13x15*	42305	**41044**
16x13	28539	**27750**	*13x16*	42305	**41106**
15x14	32394	**31511**	*14x15*	36890	**36004**
16x14	28539	**27760**	*14x16*	36890	**36005**
16x15	28539	**27760**	*15x16*	32394	**31521**
13x13	39653	**38607**	*14x14*	36890	**35928**
15x15	32394	**31521**	*16x16*	28539	**27760**

the present results.

The *naive* strategy is clearly benefited by configurations with more lanes, where the UM can be naturally minimized by underusing the total lane's capacity by filling first the empty ones. In fact, this strategy is already focused on minimizing UM by maintaining the maximum number of vehicles parked on a lane as **low** as possible.

Such behaviour can explain some of the lower gain margins presented by the *smart* strategy on some configurations (check Table 5.5 and 5.6). In fact, as shown in Table 5.6, case study B presents a much lower gain margin, in percentage, than case study A even though it presents better results in terms or RMSE, MAE and interval width as seen in Table 5.4. It is possible this lower margins are caused by the high number of lanes used in the case study B, due to the high number of vehicles in this parking lot. These results can probably denote that our parking lane selection strategy over-penalizes the number of existing cars in a lane, therefore negatively affecting bigger configurations like the ones used in case study B. Obviously, the UM could also be minimized by moving vehicles from one lane to another. However, the discussions about the optimal parking layout for each case study and on the parked vehicle's self-arrangements are out of this work's scope.

Even considering the abovementioned drawbacks, it should be highlighted that the **proposed methodology overcomes the *naive* strategy for all the presented parking layouts in both case studies**. The aim with this work is to demonstrate that is possible to **mine** both the historical and the real-time data of a parking lot entrances/exits to improve the lane selection on a self-automated parking lot. This stepwise framework takes advantage of *off-the-shelf* ML algorithms to do so. In our opinion, this proof of concept represents a consistent **breakthrough** on this relevant topic by opening promising research lines to be explored by other researchers.

Conclusions 6

Throughout this dissertation, a ML framework to predict the exit times on a self-automated parking lot is proposed. It consists on using historical data on the entries/exits of the parking lot to uncover user's profiles able to explain their parking habits. Users are then grouped according to their profiles, with users with similar profiles being aggregated into the same group. Then the vehicle's exit time is predicted and used to estimate an exit time interval based on its residuals. Lastly, this interval estimation is used to choose a lane from the parking lot in which to park the user's vehicle.

Our final goal is to optimize the vehicle's initial placement by improving the lane selection using the exit time predictions. The experiments demonstrated that our method can overcome a naive strategy by **reducing the collaborative mobility needs on roughly 10% in case study A and 3% in case study B**. As discussed in Section 5.5 the parking lane selection strategy used in this work may over-penalize bigger configurations, since case study B presents considerably better results in RMSE, MAE and interval width. This can mean that the strategy may have to be optimized for each case study. From such conclusions, we hope to open new research lines on this topic.

As future work, we propose to first optimize the parking lane selection strategy to be equally effective in both smaller and bigger configurations, next to explore the inter-lane vehicle movements to re-arrange their placements. Such movements aim to *react* to the parking current status by a) updating the exit time predictions while the vehicles are still parked or by b) moving the blocking vehicles to their neighbour lanes instead of using the buffer. The validity of such hypothesis comprise open research questions.

A paper based on this dissertation was published in a leading conference on intelligent transportation systems [NMMF14].

References

[AG07] Miguel-Ángel Alva-Gonzáles. *Environmentally unfriendly consumption behaviour: theoretical and empirical evidence from private motorists in Mexico City.* Cuvillier Verlag, 2007.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[Alt92] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE, 1995.

[B^+06] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[BA02] Kenneth P Burnham and David R Anderson. *Model selection and multimodel inference: a practical information-theoretic approach.* Springer, 2002.

[Bad07] Laviniu Aurelian Badulescu. The choice of the attribute selection measure in decision tree induction. *Annals of the University of Craiova-Mathematics and Computer Science Series*, 34:88–93, 2007.

[BDFT09] Zhang Bin, Jiang Dalin, Wang Fang, and Wan Tingting. A design of parking space detector based on video image. In *Electronic Measurement & Instruments, 2009. ICEMI'09. 9th International Conference on*, pages 2–253. IEEE, 2009.

REFERENCES

[Bis00] Richard Bishop. A survey of intelligent vehicle applications worldwide. In *Intelligent Vehicles Symposium, 2000. IV 2000. Proceedings of the IEEE*, pages 25–30. IEEE, 2000.

[BL13] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[Bre96] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[Bre01] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[CDW98] Steve Clark and Hugh Durrant-Whyte. Autonomous land vehicle navigation using millimeter wave radar. In *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, volume 4, pages 3697–3702. IEEE, 1998.

[CH08] Gerda Claeskens and Nils Lid Hjort. *Model selection and model averaging*, volume 330. Cambridge University Press Cambridge, 2008.

[Cha07] CC Chan. The state of the art of electric, hybrid, and fuel cell vehicles. *Proceedings of the IEEE*, 95(4):704–718, 2007.

[CJL95] William J Chundrlik Jr and Pamela I Labuhn. Adaptive cruise control, October 3 1995. US Patent 5,454,442.

[CKGC$^+$07] David C Conner, Hadas Kress-Gazit, Howie Choset, Alfred A Rizzi, and George J Pappas. Valet parking without a valet. In *IROS*, pages 572–577, 2007.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[DAL$^+$01] Elmar Dilger, Peter Ahner, Herbert Lohner, Peter Dominke, Chi-Thuan Cao, Ngoc-Thach Nguyen, Helmut Janetzke, Thorsten Allgeier, Wolfgang Pfeiffer, Bo Yuan, et al. Steer-by-wire steering system for motorized vehicles, April 17 2001. US Patent 6,219,604.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[DMSS04] Erez Dagan, Ofer Mano, Gideon P Stein, and Amnon Shashua. Forward collision warning with a single camera. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 37–42. IEEE, 2004.

[DP97] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[Dyn] Evgeniĭ Dynkin. *Markov processes.*

[DZ88] Ernst D Dickmanns and Alfred Zapp. Autonomous high speed road vehicle guidance by computer vision. In *International Federation of Automatic Control. World Congress (10th). Automatic control: world congress.*, volume 1, 1988.

[ET94] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*, volume 57. CRC press, 1994.

[Fea14] Michel Ferreira and et al. Self-automated parking lots for autonomous vehicles based on vehicular ad hoc networking. In *Proc IEEE Intelligent Vehicles Symp. - IV*, Dearborn, Michigan , United States, June 2014.

[FFC$^+$10] Michel Ferreira, Ricardo Fernandes, Hugo Conceição, Wantanee Viriyasitavat, and Ozan K Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pages 85–90. ACM, 2010.

[FH72] William R Faris and William P Harokopus. Rear end warning system for automobiles, October 10 1972. US Patent 3,697,985.

[Fig03] Mário AT Figueiredo. Adaptive sparseness for supervised learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(9):1150–1159, 2003.

[FS81] Jerome H Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

[GH06] Liqiang Geng and Howard J Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, 38(3):9, 2006.

[GRSdC10] Joao Gama, Pedro Pereira Rodrigues, Eduardo J Spinosa, and André Carlos Ponce Leon Ferreira de Carvalho. *Knowledge discovery from data streams.* Chapman & Hall/CRC Boca Raton, 2010.

[GW97] David L Greene and Michael Wegener. Sustainable transport. *Journal of Transport Geography*, 5(3):177–190, 1997.

[Har60] Harry H Harman. Modern factor analysis. 1960.

[HBD10] Noor Hazrin Hany Mohamad Hanif, Mohd Hafiz Badiozaman, and Hanita Daud. Smart parking reservation system using short message services (sms). In *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*, pages 1–5. IEEE, 2010.

[HT89] Clifford M Hurvich and Chih-Ling Tsai. Regression and time series model selection in small samples. *Biometrika*, 76(2):297–307, 1989.

[HTF$^+$09] Trevor Hastie, Robert Tibshirani, Jerome Friedman, T Hastie, J Friedman, and R Tibshirani. *The elements of statistical learning*, volume 2. Springer, 2009.

[IMK$^+$02] Takehiro Ito, Shinji Mita, Kazuhiro Kozuka, Tomoaki Nakano, and Shin Yamamoto. Driver blink measurement by the motion picture processing and its application to drowsiness detection. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 168–173. IEEE, 2002.

[ISN$^+$01] K Inaba, M Shibui, T Naganawa, M Ogiwara, and N Yoshikai. Intelligent parking reservation service on the internet. In *Applications and the Internet Workshops, 2001. Proceedings. 2001 Symposium on*, pages 159–164. IEEE, 2001.

[ISS02] Rolf Isermann, Ralf Schwarz, and Stefan Stolzl. Fault-tolerant drive-by-wire systems. *Control Systems, IEEE*, 22(5):64–81, 2002.

[JD08] Daniel Jiang and Luca Delgrossi. Ieee 802.11 p: Towards an international standard for wireless access in vehicular environments. In *Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE*, pages 2036–2040. IEEE, 2008.

[Joh67] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.

[Jol05] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2005.

[KL51] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, pages 79–86, 1951.

[KWKC12] Max Kuhn, Steve Weston, Chris Keefer, and Nathan Coulter. Cubist models for regression, 2012.

[Lee02] Joon Woong Lee. A machine vision system for lane-departure detection. *Computer vision and image understanding*, 86(1):52–78, 2002.

[LLZS09] Rongxing Lu, Xiaodong Lin, Haojin Zhu, and Xuemin Shen. Spark: a new vanet-based smart parking scheme for large parking lots. In *INFOCOM 2009, IEEE*, pages 1413–1421. IEEE, 2009.

[LYG08] Sangwon Lee, Dukhee Yoon, and Amitabha Ghosh. Intelligent parking lot application using wireless sensor networks. In *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on*, pages 48–57. IEEE, 2008.

[M+67] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. California, USA, 1967.

[MMSJS12] João Mendes-Moreira, Carlos Soares, Alípio Mário Jorge, and Jorge Freire De Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys (CSUR)*, 45(1):10, 2012.

[MP87] Brett A Miller and Daniel Pitton. Vehicle blind spot detector, September 15 1987. US Patent 4,694,295.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of machine learning.* MIT press, 2012.

[NMMF14] Rafael Nunes, Luis Moreira-Matias, and M. Ferreira. Using exit time predictions to optimize self automated parking lots. *Intelligent Transportation Systems Conference*, 2014.

[Q+92] John R Quinlan et al. Learning with continuous classes. In *Proceedings of the 5th Australian joint Conference on Artificial Intelligence*, volume 92, pages 343–348. Singapore, 1992.

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[RHW88] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 1988.

[RJ86] Lawrence Rabiner and Biing-Hwang Juang. An introduction to hidden markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

REFERENCES

[rob] http://www.roboticparking.com/.

[Ros58] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[Rou87] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[S$^+$78] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[SA$^+$05] Donald C Shoup, American Planning Association, et al. *The high cost of free parking*, volume 206. Planners Press Chicago, 2005.

[Sch90] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

[SGL14] Tito Tang Sebastian Gnatzig, Frederic Chucholowski and Markus Lienkamp. A system design for teleoperated road vehicles. In *ICINCO 2013 - Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics*, pages 231–238, 2014. 10th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2013.

[Sho06] Donald C Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[SNDW99] Salah Sukkarieh, Eduardo Mario Nebot, and Hugh F Durrant-Whyte. A high integrity imu/gps navigation loop for autonomous land vehicle applications. *Robotics and Automation, IEEE Transactions on*, 15(3):572–578, 1999.

[Tea12] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2012.

[TL06] Dušan Teodorović and Panta Lučić. Intelligent parking systems. *European Journal of Operational Research*, 175(3):1666–1681, 2006.

[tom] http://automotive.tomtom.com/en/connected-navigation-system, retrieved september 9, 2014.

[unu] World population increasingly urban with more than half living in urban areas, retrieved september 9, 2014.

[WF05] Ian H Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[wor] http://www.worldometers.info/world-population/, retrieved september 9, 2014.

[XCX00] Jin Xu, Guang Chen, and Ming Xie. Vision-guided automatic parking for smart car. In *IEEE intelligent Vehicles symposium*, volume 2000, pages 725–730, 2000.

[YNB12] R Yusnita, Fariza Norbaya, and Norazwinawati Basharuddin. Intelligent parking space detection system based on image processing. *International Journal of Innovation, Management and Technology*, 3(3), 2012.

[Zho12] Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms.* CRC Press, 2012.

[ZWW$^+$11] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. Data-driven intelligent transportation systems: A survey. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1624–1639, 2011.

Implementation Code A

```
regression.grouped<-function()
{
  libraries2()
  library(Cubist)
  library(gbm)
  vars<-c("ID", "Gender", "Funcao", "Date", "DayType",
          "Hour", "Minute", "Second", "Metodo", "Real_
             ParkingTime", "Predicted_ParkingTime")
  methods<-c("REG.gbm", "REG.svm", "REG.cubist", "REG.PPR")

  load("dados_original.RData")
  dados<-dados_original
  dados$ID<-as.character(dados$ID)
  data<-dados
  data<-data[which(data$Date<"2013/11/01"),]
  ids<-unique(data$ID)

  dat<-clustering.mclust(data)
  cent<-dat$G

  clust<-dat$classification
  filw<-sprintf("clust.csv", cent)
  write.csv2(getStats.mclust(dat), filw, row.names=FALSE)

  ids1<-which(clust==1)
  ids2<-which(clust==2)
  ids3<-which(clust==3)
  ids4<-which(clust==4)
  ids5<-which(clust==5)
```

```
ids6<-which(clust==6)
ids7<-which(clust==7)
ids8<-which(clust==8)
ids9<-which(clust==9)
ids10<-which(clust==10)
ids11<-which(clust==11)
ids12<-which(clust==12)
ids13<-which(clust==13)
ids14<-which(clust==14)
ids15<-which(clust==15)
ids16<-which(clust==16)
ids17<-which(clust==17)
ids18<-which(clust==18)
ids19<-which(clust==19)
ids20<-which(clust==20)
ids21<-which(clust==21)
ids22<-which(clust==22)
ids23<-which(clust==23)
ids24<-which(clust==24)
ids25<-which(clust==25)
ids26<-which(clust==26)
ids27<-which(clust==27)
ids28<-which(clust==28)
ids29<-which(clust==29)
ids30<-which(clust==30)

group1<-ids[ids1]
group2<-ids[ids2]
group3<-ids[ids3]
group4<-ids[ids4]
group5<-ids[ids5]
group6<-ids[ids6]
group7<-ids[ids7]
group8<-ids[ids8]
group9<-ids[ids9]
group10<-ids[ids10]
group11<-ids[ids11]
```

```
group12<-ids[ids12]
group13<-ids[ids13]
group14<-ids[ids14]
group15<-ids[ids15]
group16<-ids[ids16]
group17<-ids[ids17]
group18<-ids[ids18]
group19<-ids[ids19]
group20<-ids[ids20]
group21<-ids[ids21]
group22<-ids[ids22]
group23<-ids[ids23]
group24<-ids[ids24]
group25<-ids[ids25]
group26<-ids[ids26]
group27<-ids[ids27]
group28<-ids[ids28]
group29<-ids[ids29]
group30<-ids[ids30]

data<-dados
data<-data[which(data$Date>="2013/10/01"),]

vars1<-length(data$ID[which(data$ID%in%group1)])
vars2<-length(data$ID[which(data$ID%in%group2)])
vars3<-length(data$ID[which(data$ID%in%group3)])
vars4<-length(data$ID[which(data$ID%in%group4)])
vars5<-length(data$ID[which(data$ID%in%group5)])
vars6<-length(data$ID[which(data$ID%in%group6)])
vars7<-length(data$ID[which(data$ID%in%group7)])
vars8<-length(data$ID[which(data$ID%in%group8)])
vars9<-length(data$ID[which(data$ID%in%group9)])
vars10<-length(data$ID[which(data$ID%in%group10)])
vars11<-length(data$ID[which(data$ID%in%group11)])
vars12<-length(data$ID[which(data$ID%in%group12)])
vars13<-length(data$ID[which(data$ID%in%group13)])
vars14<-length(data$ID[which(data$ID%in%group14)])
```

```
vars15<-length(data$ID[which(data$ID%in%group15)])
vars16<-length(data$ID[which(data$ID%in%group16)])
vars17<-length(data$ID[which(data$ID%in%group17)])
vars18<-length(data$ID[which(data$ID%in%group18)])
vars19<-length(data$ID[which(data$ID%in%group19)])
vars20<-length(data$ID[which(data$ID%in%group20)])
vars21<-length(data$ID[which(data$ID%in%group21)])
vars22<-length(data$ID[which(data$ID%in%group22)])
vars23<-length(data$ID[which(data$ID%in%group23)])
vars24<-length(data$ID[which(data$ID%in%group24)])
vars25<-length(data$ID[which(data$ID%in%group25)])
vars26<-length(data$ID[which(data$ID%in%group26)])
vars27<-length(data$ID[which(data$ID%in%group27)])
vars28<-length(data$ID[which(data$ID%in%group28)])
vars29<-length(data$ID[which(data$ID%in%group29)])
vars30<-length(data$ID[which(data$ID%in%group30)])

m<-matrix(0,30,1)
m[1,]<-vars1
m[2,]<-vars2
m[3,]<-vars3
m[4,]<-vars4
m[5,]<-vars5
m[6,]<-vars6
m[7,]<-vars7
m[8,]<-vars8
m[9,]<-vars9
m[10,]<-vars10
m[11,]<-vars11
m[12,]<-vars12
m[13,]<-vars13
m[14,]<-vars14
m[15,]<-vars15
m[16,]<-vars16
m[17,]<-vars17
m[18,]<-vars18
m[19,]<-vars19
```

```
  m[20,]<-vars20
  m[21,]<-vars21
  m[22,]<-vars22
  m[23,]<-vars23
  m[24,]<-vars24
  m[25,]<-vars25
  m[26,]<-vars26
  m[27,]<-vars27
  m[28,]<-vars28
  m[29,]<-vars29
  m[30,]<-vars30
  dfxa<-as.data.frame(m)
  avg<-sprintf("avgentries.csv", cent)
  write.csv(dfxa, avg)

  grupos<-list(group1, group2, group3, group4, group5, group6,
      group7, group8, group9, group10,
                 group11, group12, group13, group14, group15,
                    group16, group17, group18, group19, group20,
                 group21, group22, group23, group24, group25,
                    group26, group27, group28, group29, group30)

  for(i in 7:30){
    m<-matrix(0, 50*length(methods)*500, length(vars))
    data.reg<-data[which(data$ID%in%grupos[[i]]) ,]
    data.reg$ID<-as.factor(data.reg$ID)
    m<-doRegression(data.reg, m, methods)
    df<-as.data.frame(m)
    names(df)<-vars
    idc<-which(as.character(df$ID)=="0")
    df<-df[-idc,]
    filw<-sprintf("predictions_reg_groups%d.csv", i)
    write.csv2(df, filw, row.names=FALSE)
  }
}

doRegression<-function(data, m, methods)
```

```
{
  idx<-1
  for(method in methods){
    low.split.day<-"2013/10/01"
    split.day<-"2013/11/03"
    while(split.day<="2013/11/29"){
      data.reg<-data

      new.split.day<-incrementa_data(split.day)
      id.train<-which(data.reg$Date<=split.day & data.reg$Date
         >=low.split.day)
      id.test<-which(data.reg$Date==new.split.day)

      data.reg<-data.reg[-4]
      data.reg<-data.reg[-7]

      data.train<-data.reg[id.train,]
      data.test<-data.reg[id.test,]

      if(method=="REG.PPR" && length(data.train$HORA)>6)
      {
        nts<-2:3
        mts<-4:5
        for(nt in nts){
          for(mt in mts){
            if(length(levels(data.train$ID))==1){
              model<-ppr(Parking_Time ~ ., data=data.train
                  [-1], nterms=nt, max.terms=mt) #
              res<-round(predict(model,data.test[-1]))
            }
            else{
              tryCatch({
                model<-ppr(Parking_Time ~ ., data=data.train,
                    nterms=nt, max.terms=mt) #
                res<-round(predict(model,data.test))
              }, warning = function(w) {
                print(w)
          }, error = function(e) {
            print(e)
            res<-c()
          }, finally = {
            print("f")
          })
        }
        method2 = paste("REG.PPR", "nt", sep="")
        method2 = paste(method2, nt, sep="")
        method2 = paste(method2, "mt", sep="")
        method2 = paste(method2, mt, sep="")
        len<-length(res)
        if(len>0){
          for(i in c(1:len)){
            id<-as.character(data.test$ID[i])
            gender<-as.character(data.test$SEXO[i])
            funcao<-as.character(data.test$CICLO_INI[i])

            hora<-floor(data.test$HORA[i]/3600)
            minuto<-floor((data.test$HORA[i]-hora*3600)/
               60)
            segundo<-data.test$HORA[i]-hora*3600-minuto*60
            values<-c(id, gender, funcao, new.split.day,
              DAY_OF_WEEK(new.split.day), hora, minuto,
               segundo, method2, data.test$Parking_Time[i
               ],res[i])
           m[idx,]<-values
            idx<-idx+1
          }
        }
      }
    }
  }
  if(method=="REG.svm" && length(data.train$HORA)>6)
  {
    kernel<-c("linear", "radial") #
    for(k in kernel){
```

```
if(length(levels(data.train$ID))==1){
  model<-svm(Parking_Time ~ ., data=data.train[-1],
     kernel=k)
  res<-round(predict(model,data.test[-1], interval="
     prediction"))
}
else{
  tryCatch({
    model<-svm(Parking_Time ~ ., data=data.train,
       kernel=k)
    res<-predict(model,data.test)
  }, warning = function(w) {
    print(w)
  }, error = function(e) {
    print(e)
    res<-c()
  }, finally = {
    print("f")
  })
}
len<-length(res)
if(len>0){
  for(i in c(1:len)){
    id<-as.character(data.test$ID[i])
    gender<-as.character(data.test$SEXO[i])
    funcao<-as.character(data.test$CICLO_INI[i])

    hora<-floor(data.test$HORA[i]/3600)
    minuto<-floor((data.test$HORA[i]-hora*3600)/60)
    segundo<-data.test$HORA[i]-hora*3600-minuto*60
    values<-c(id, gender, funcao, new.split.day, DAY
       _OF_WEEK(new.split.day), hora, minuto,
       segundo, paste("SVM", k, sep=""), data.test$
       Parking_Time[i],res[i])
    m[idx,]<-values
    idx<-idx+1
  }
```

```
      }
    }
  }
  if(method=="REG.cubist" && length(data.test$HORA)>0)
  {
    committees<-1:10
    for(com in committees){
      if(length(levels(data.train$ID))==1){
        data.train.x<-data.train[-1]
        model<-cubist(x=data.train.x[-5], y=data.train.x$
            Parking_Time, committees=com)
        res<-predict(model,data.test[-1])
      }
      else{
        model<-cubist(x=data.train[-6], y=data.train$
            Parking_Time, committees=com)
        res<-predict(model,data.test)
      }
      len<-length(res)
      if(len>0){
        for(i in c(1:len)){
          id<-as.character(data.test$ID[i])
          gender<-as.character(data.test$SEXO[i])
          funcao<-as.character(data.test$CICLO_INI[i])

          hora<-floor(data.test$HORA[i]/3600)
          minuto<-floor((data.test$HORA[i]-hora*3600)/60)
          segundo<-data.test$HORA[i]-hora*3600-minuto*60
          values<-c(id, gender, funcao, new.split.day, DAY
              _OF_WEEK(new.split.day), hora, minuto,
              segundo, paste("cubist", com, sep=""), data.
              test$Parking_Time[i],res[i])
         m[idx,]<-values
          idx<-idx+1
        }
      }
    }
```

```
}
if(method=="REG.gbm" && length(data.train$HORA)>100)
{
  if(length(levels(data.train$ID))==1){
    data.train<-data.train[-1]
    model<-gbm(Parking_Time ~ ., data=data.train[-1],
        distribution="gaussian", n.trees=1000)
    res<-predict(model,data.test[-1], n.trees=1000)
  }
  else{
    model<-gbm(Parking_Time ~ ., data=data.train,
        distribution="gaussian", n.trees=1000)
    res<-predict(model,data.test, n.trees=1000)
  }
  len<-length(res)
  if(len>0){
    for(i in c(1:len)){
      id<-as.character(data.test$ID[i])
      gender<-as.character(data.test$SEXO[i])
      funcao<-as.character(data.test$CICLO_INI[i])

      hora<-floor(data.test$HORA[i]/3600)
      minuto<-floor((data.test$HORA[i]-hora*3600)/60)
      segundo<-data.test$HORA[i]-hora*3600-minuto*60
      values<-c(id, gender, funcao, new.split.day, DAY_
         OF_WEEK(new.split.day), hora, minuto, segundo,
         "gbm", data.test$Parking_Time[i],res[i])
     m[idx,]<-values
      idx<-idx+1
    }
  }
}
if(method=="REG.RF")
{
  if(length(data.train$Second)>0){
    mts<-3:5
    nts<-c(500,750,1000)
```

```
for(mt in mts){
  for(nt in nts){
    if(length(levels(data.reg$ID))==1){
      model<-randomForest(Parking_Time ~ ., data=
          data.train[-1]) #, mtry=mt, ntrees=nt
      res<-round(predict(model,data.test[-1]))
    }
    else{
      model<-randomForest(Parking_Time ~ ., data=
          data.train) #, mtry=mt, ntrees=nt
      res<-round(predict(model,data.test))
    }
    #method2="REG.RF"
    method2 = paste("REG.RF", "nt", sep="")
    method2 = paste(method2, nt, sep="")
    method2 = paste(method2, "mt", sep="")
    method2 = paste(method2, mt, sep="")

    len<-length(res)
    if(len>0){
      for(i in c(1:len)){
        id<-as.character(data.test$ID[i])
        acronym<-info$sigla[which(info$id==id)]
        gender<-info$sexo[which(info$id==id)]
        department<-info$departamento[which(info$id
           ==id)]
        funcao<-info$funcao[which(info$id==id)]

        hora<-floor(data.test$Second[i]/3600)
        minuto<-floor((data.test$Second[i]-hora*
           3600)/60)
        segundo<-data.test$Second[i]-hora*3600-
           minuto*60
        values<-c(id, acronym, gender, department,
           funcao, new.split.day, is.holiday(new.
           split.day), DAY_OF_WEEK(new.split.day),
           hora, minuto, segundo, method2, data.test
```

```
                    $Parking_Time[i],res[i])
                  m[idx,]<-values
                  idx<-idx+1
                }
              }
            }
          }
        }
      }
      split.day<-incrementa_data(split.day)
      low.split.day<-incrementa_data(low.split.day)
    }
  }
  return(m)
}

clustering.mclust<-function(data)
{
  ids<-unique(data$ID)

  m<-matrix(0, length(unique(as.character(data$ID))), 402)
  indice<-1
  for(id in ids){
    data.reg<-data[which(data$ID==id),]
    x <- data.reg$Parking_Time
    est <- bkde(x)

    val<-c(id, est$y)
    m[indice,]<-val
    indice<-indice+1
  }
  df<-as.data.frame(m)
  for(i in 2:402)
  {
    df[,i]<-as.character(df[,i])
    df[,i]<-as.numeric(df[,i])
  }
  dat<-Mclust(as.matrix(df), G=2:30)
  return(dat)
}

rmse.per.group<-function(j)
{
  options(OutDec= ",")
  filename<-sprintf("predictions_reg_groups%d.csv", j)
  data<-read.csv2(filename,sep=";")
  vars<-c("id", "nentries", "method","rmse","mad")
  data$ID<-as.character(data$ID)
  data$Metodo<-as.character(data$Metodo)
  ids<-unique(data$ID)
  metodos<-unique(data$Metodo)
  m<-matrix(0, length(ids)*length(metodos), length(vars))
  data$Predicted_ParkingTime<-as.numeric(gsub(",", ".",as.
     character(data$Predicted_ParkingTime)))

  idx<-1
  for(id in ids){
    for(metodo in metodos){
      previsto <- data$Predicted_ParkingTime[which(data$ID==id
          & data$Metodo==metodo)]
      real <- data$Real_ParkingTime[which(data$ID==id & data$
         Metodo==metodo)]

      nentries<-length(previsto)

      rmse <- sqrt((sum((real - previsto)**2))/length(previsto
         ))
      mad<-mean(abs(real - previsto))

      m[idx,]<-c(id, nentries, metodo, rmse, mad)
      idx<-idx+1
    }
  }
  df<-as.data.frame(m)
  names(df)<-vars
  df<-df[order(df$rmse), ]
  filenamew<-sprintf("rmse_group%d.csv", j)
  write.table(df, filenamew,sep=";",dec=",", row.names=FALSE)
}

average.rmse.groups<-function(n){
  filename<-sprintf("rmse_group%d.csv", n)
  data<-read.csv2(filename, sep=";")
  data$method<-as.character(data$method)
  m<-matrix(0,length(unique(data$method)),3)
  idx<-1
  for(met in unique(data$method)){
    media_rmse<-0
    media_mad<-0
    total_entries<-0
    for(id in unique(data$id)){
      data.reg<-data[which(data$method==met & data$id==id),]
      media_rmse<-media_rmse+data.reg$nentries[1]*data.reg$
         rmse
      media_mad<-media_mad+data.reg$nentries[1]*data.reg$mad
      total_entries<-total_entries+data.reg$nentries[1]
    }
    media_rmse<-media_rmse/total_entries
    media_mad<-media_mad/total_entries

    m[idx,]<-c(met, media_rmse, media_mad)
    idx<-idx+1
  }
  df<-as.data.frame(m)
  names(df)<-c("Metodo", "rmse", "mad")
  filw<-sprintf("average_rmse_%d.csv", n)
  write.table(df,filw,sep=";",dec=",", row.names=FALSE)
}

all.group.rmse<-function()
{
  for(i in 1:28){
    rmse.per.group(i)
    average.rmse.groups(i)
    rmse.detailed(i)
  }
}

getResults.regression.groups<-function()
{
  options(OutDec= ",")
  vars<-c("K","Grupo","Metodo", "rmse", "mad")
  m<-matrix(0, length(8:30)*4, length(vars))
  idx<-1
  for(i in 28){
    grp<-sprintf("clust.csv", i)
    data2<-read.csv2(grp, sep=";")
    avgFile<-sprintf("avgentries.csv", i)
    avgE<-read.csv(avgFile)
    testes<-1:i
    for(j in testes){
      filw<-sprintf("average_rmse_%d.csv", j)
      data<-read.csv2(filw, sep=";")

      data<-data[order(data$rmse), ]
      data$Metodo<-as.character(data$Metodo)

      rmse_1<-data$rmse[1]
      mad_1<-data$mad[1]

      val<-c(i, j, data$Metodo[1], rmse_1, mad_1)
      m[idx,]<-val
      idx<-idx+1
    }
  }
  df<-as.data.frame(m)
  names(df)<-vars
  write.table(df, "resultados.csv",sep=";",dec=".", row.names=
      FALSE)
}

intervalo.grupo.sliding.adapt.qt<-function()
{
  options(OutDec= ",")

  alphai_p<-0
  alphai_n<-0
  betai<-1.0
  deltai<-0.1

  vars<-c("metodo", "grupo1", ",grupo2", "id", "date", "valor_
      real",
          "valor_previsto", "mad", "hit", "intervalo")
  idx<-1

  rmse_file<-sprintf("resultados.csv",i)
  rmse<-read.csv2(rmse_file, sep=";")
  rmse$Metodo<-as.character(rmse$Metodo)

  load("dados_original.RData")
  dados<-dados_original
  data<-dados
  data<-data[which(data$Date<"2013/11/01"),]
  ids<-unique(data$ID)

  dat<-clustering.mclust(data)
  cent<-dat$G

  clust<-dat$classification

  ids1<-which(clust==1)
  ids2<-which(clust==2)
  ids3<-which(clust==3)
  ids4<-which(clust==4)
  ids5<-which(clust==5)
```

```
ids6<-which(clust==6)
ids7<-which(clust==7)
ids8<-which(clust==8)
ids9<-which(clust==9)
ids10<-which(clust==10)
ids11<-which(clust==11)
ids12<-which(clust==12)
ids13<-which(clust==13)
ids14<-which(clust==14)
ids15<-which(clust==15)
ids16<-which(clust==16)
ids17<-which(clust==17)
ids18<-which(clust==18)
ids19<-which(clust==19)
ids20<-which(clust==20)
ids21<-which(clust==21)
ids22<-which(clust==22)
ids23<-which(clust==23)
ids24<-which(clust==24)
ids25<-which(clust==25)
ids26<-which(clust==26)
ids27<-which(clust==27)
ids28<-which(clust==28)
ids29<-which(clust==29)
ids30<-which(clust==30)

group1<-ids[ids1]
group2<-ids[ids2]
group3<-ids[ids3]
group4<-ids[ids4]
group5<-ids[ids5]
group6<-ids[ids6]
group7<-ids[ids7]
group8<-ids[ids8]
group9<-ids[ids9]
group10<-ids[ids10]
group11<-ids[ids11]
```

```
group12<-ids[ids12]
group13<-ids[ids13]
group14<-ids[ids14]
group15<-ids[ids15]
group16<-ids[ids16]
group17<-ids[ids17]
group18<-ids[ids18]
group19<-ids[ids19]
group20<-ids[ids20]
group21<-ids[ids21]
group22<-ids[ids22]
group23<-ids[ids23]
group24<-ids[ids24]
group25<-ids[ids25]
group26<-ids[ids26]
group27<-ids[ids27]
group28<-ids[ids28]
group29<-ids[ids29]
group30<-ids[ids30]

m<-matrix(0, 50000, length(vars))
for(i in 1:cent){
  pred_file<-sprintf("predictions_reg_groups%d.csv",i)
  pred<-read.csv2(pred_file , sep=";")
  pred$ID<-as.character(pred$ID)
  pred$Date<-as.character(pred$Date)
  pred$Metodo<-as.character(pred$Metodo)
  pred$Predicted_ParkingTime<-as.numeric(gsub(",", ".",as.
     character(pred$Predicted_ParkingTime)))

  rmse_id_file<-sprintf("rmse_group%d.csv",i)
  rmse_id<-read.csv2(rmse_id_file , sep=";")
  rmse_id$id<-as.character(rmse_id$id)

  met<-rmse$Metodo[which(rmse$Grupo==i)]

  for(id in unique(rmse_id$id)){
```

```
alpha_p<-alphai_p
alpha_n<-alphai_n
beta<-betai
delta<-deltai

if(id %in% group1){
  grupo<-1
}
else if(id %in% group2){
  grupo<-2
}
else if(id %in% group3){
  grupo<-3
}
else if(id %in% group4){
  grupo<-4
}
else if(id %in% group5){
  grupo<-5
}
else if(id %in% group6){
  grupo<-6
}
else if(id %in% group7){
  grupo<-7
}
else if(id %in% group8){
  grupo<-8
}
else if(id %in% group9){
  grupo<-9
}
else if(id %in% group10){
  grupo<-10
}
else if(id %in% group11){
      grupo<-11
    }
    else if(id %in% group12){
      grupo<-12
    }
    else if(id %in% group13){
      grupo<-13
    }
    else if(id %in% group14){
      grupo<-14
    }
    else if(id %in% group15){
      grupo<-15
    }
    else if(id %in% group16){
      grupo<-16
    }
    else if(id %in% group17){
      grupo<-17
    }
    else if(id %in% group18){
      grupo<-18
    }
    else if(id %in% group19){
      grupo<-19
    }
    else if(id %in% group20){
      grupo<-20
    }
    else if(id %in% group21){
      grupo<-21
    }
    else if(id %in% group22){
      grupo<-22
    }
```

```
rmse_detail_file<-sprintf("rmse_detailed%d.csv", grupo)
rmse.reg<-read.csv2(rmse_detail_file , sep=";")
rmse.reg$Metodo<-as.character(rmse.reg$Metodo)
rmse.reg$Date<-as.character(rmse.reg$Date)
rmse.reg<-rmse.reg[which(rmse.reg$Metodo==met) ,]

pred.reg<-pred[which(pred$ID==id & pred$Metodo==met) ,]

if(length(pred.reg$ID)>0){
  tempo_inicial<-pred.reg$Hour*3600+pred.reg$Minute*60+
     pred.reg$Second
  valor_real<-tempo_inicial+pred.reg$Real_ParkingTime

  for(j in 1:length(valor_real)){
    novo_mad<-0
    if(j==1){
      qt<-quantile(rmse.reg$mad)
      intervalo_baixo<-round(c((tempo_inicial+pred.reg$
         Predicted_ParkingTime)-qt[2]))
      intervalo_cima<-round(c((tempo_inicial+pred.reg$
         Predicted_ParkingTime)+qt[4]))

      dista<-(intervalo_cima-intervalo_baixo)*beta
      dista<-dista/2

      intervalo_baixo<-round(c((tempo_inicial+pred.reg$
         Predicted_ParkingTime)-dista))
      intervalo_cima<-round(c((tempo_inicial+pred.reg$
         Predicted_ParkingTime)+dista))
    }
    else{
      rmse.reg.tmp<-rmse.reg[which(rmse.reg$Date>=
         decrementa_data_2week(pred.reg$Date[j])) ,]
      pred.rmse<-pred[which(pred$Date>=decrementa_data_2
         week(pred.reg$Date[j]) & pred$Date<pred.reg$
         Date[j]  & pred$Metodo==met) ,]
    novo_mad<-abs(pred.rmse$Predicted_ParkingTime-pred
        .rmse$Real_ParkingTime)

    mades<-c(rmse.reg.tmp$mad, novo_mad)

    qt<-quantile(mades)
    intervalo_baixo<-round(c((tempo_inicial+pred.reg$
        Predicted_ParkingTime)-qt[2]))#*beta
    intervalo_cima<-round(c((tempo_inicial+pred.reg$
        Predicted_ParkingTime)+qt[4]))#*beta

    dista<-(intervalo_cima-intervalo_baixo)*beta
    dista<-dista/2

    intervalo_baixo<-round(c((tempo_inicial+pred.reg$
        Predicted_ParkingTime)-dista))
    intervalo_cima<-round(c((tempo_inicial+pred.reg$
        Predicted_ParkingTime)+dista))
  }
  ida<-which(intervalo_baixo<=(10*60))
  intervalo_baixo[ida]=10*60
  if(valor_real[j]<=intervalo_cima[j] && valor_real[j
      ]>=intervalo_baixo[j]){
    hit<-1
    alpha_n<-0
    alpha_p<-alpha_p+1
  }
  else{
    hit<-0
    alpha_p<-0
    alpha_n<-alpha_n-1
  }
  m[idx,]<-c(met, i, grupo, pred.reg$ID[j], pred.reg$
      Date[j], valor_real[j], tempo_inicial[j]+pred.reg
      $Predicted_ParkingTime[j], dista[j]*2, hit,
      intervalo_cima[j]-intervalo_baixo[j])
```

```
            idx<-idx+1
            if(alpha_p>=3){
              beta<-beta-delta
              if(beta<1){
                beta<-1
              }
            }
            else if(alpha_n<=-1){
              beta<-beta+delta
              if(beta>2){
                beta<-2
              }
            }
          }
        }
      }
    }
    df<-as.data.frame(m)
    for(i in 1:4){
      df[,i]<-as.character(df[,i])
    }
    idx<-which(df$V1=="0")
    df<-df[-idx,]
    names(df)<-vars
    write.table(df,'id_intervalo_grupo_sliding_adapt_qt.csv',sep
      =";",dec=",", row.names=FALSE)


    df$valor_real<-as.numeric(df$valor_real)
    df$valor_previsto<-as.numeric(as.character(gsub(",", ".",df$
      valor_previsto)))
    df$mad<-as.numeric(as.character(gsub(",", ".",df$mad)))
    df$hit<-as.numeric(as.character(df$hit))
    df$intervalo<-as.numeric(as.character(gsub(",", ".",df$
      intervalo)))

    vars<-c("intervalo")
```

```
  m<-matrix(0, 1, length(vars))
  idx<-1
  media<-mean(df$intervalo)
  m[idx,]<-c(media)
  idx<-idx+1
  df<-as.data.frame(m)
  names(df)<-vars
  write.table(df,'intervalo_grupo_sliding_adapt_qt.csv',sep=";
     ",dec=",", row.names=FALSE)
}

rmse.detailed<-function(j)
{
  options(OutDec= ",")
  filename<-sprintf("predictions_reg_groups%d.csv", j)
  data<-read.csv2(filename,sep=";")
  if(length(data$Real_ParkingTime)>0){
    data$ID<-as.character(data$ID)
    data$Metodo<-as.character(data$Metodo)
    nentries<-c()
    for(i in 1:length(data$ID)){
      nentries<-c(nentries, (length(data$ID[which(data$ID==
         data$ID[i])])/length(unique(data$Metodo)) ))
    }
    data$Predicted_ParkingTime<-as.numeric(gsub(",", ".",as.
       character(data$Predicted_ParkingTime)))
    mad<-abs(data$Real_ParkingTime - data$Predicted_
       ParkingTime)
    data<-cbind(data, nentries=(nentries))
    data<-cbind(data, mad=(mad))
    data<-data[-2:-3]
    data<-data[-3:-6]
    filenamew<-sprintf("rmse_detailed%d.csv", j)
    write.table(data, filenamew,sep=";",dec=",", row.names=
      FALSE)
  }
}
```

```
interval.stats<-function()
{
  vars<-c("metodo", "hits", "total", "percent", "Hit_VI")
  m<-matrix(0, 1000, length(vars))
  idx<-1
  ##########
  data<-read.csv2("id_intervalo_grupo_sliding_adapt_qt.csv",
     sep=";")
  data$metodo<-as.character(data$metodo)
  data$intervalo<-as.numeric(as.character(gsub(",", ".",data$
     intervalo)))
  hit<-length(data$hit[which(data$hit==1)])
  total<-length(data$hit)
  percent<-(hit*100)/total
  valor<-hit/mean(data$intervalo)
  m[idx,]<-c("Grupo_Sliding_Adapt_Qt", hit, total, percent,
     valor)
  idx<-idx+1

  df<-as.data.frame(m)
  names(df)<-vars
  df$percent<-as.numeric(as.character(gsub(",", ".",df$percent
     )))
  df$Hit_VI<-as.numeric(as.character(gsub(",", ".",df$Hit_VI))
     )
  df$metodo<-as.character(df$metodo)
  ida<-which(df$metodo=="0")
  df<-df[-ida,]
  write.table(df,'hits.csv',sep=";",dec=",", row.names=FALSE)
}

data.preprocessing.feup<-function()
{
  resultados<-read.csv2("resultados.csv")
  resultados$Metodo<-as.character(resultados$Metodo)
  vars<-c("ID", "Gender", "Funcao", "Date", "DayType", "Hour",
    "Minute", "Second", "Metodo", "Real_ParkingTime", "
    Predicted_ParkingTime")
m<-matrix(0, 70000, length(vars))
idx<-1
for(i in 1:22){
  best.method<-resultados$Metodo[which(resultados$Grupo==i)]
  filename<-sprintf("predictions_reg_groups%d.csv", i)
  data<-read.csv2(filename, sep=";")
  data$Metodo<-as.character(data$Metodo)
  data<-data[which(data$Metodo==best.method),]
  data$Predicted_ParkingTime<-as.numeric(gsub(",", ".",as.
     character(data$Predicted_ParkingTime)))
  if(length(data$ID)>0){
    for(j in 1:length(data$ID)){
      linha<-c(data[j,1], data[j,2], data[j,3], as.character
          (data[j,4]), as.character(data[j,5]), as.character(
          data[j,6]), as.character(data[j,7]), as.character(
          data[j,8]), data[j,9], data[j,10], data[j,11])
      m[idx,]<-linha
      idx<-idx+1
    }
  }
}
data<-as.data.frame(m)
names(data)<-vars
data<-data[-which(as.character(data$ID)=="0"),]
data$ID<-as.numeric(as.character(data$ID))
data$Date<-as.character(data$Date)
data$DayType<-as.character(data$DayType)
data<-data[-2:-3]
data$Hour<-as.numeric(as.character(data$Hour))
data$Minute<-as.numeric(as.character(data$Minute))
data$Second<-as.numeric(as.character(data$Second))
data$Second<-data$Hour*3600+data$Minute*60+data$Second
data<-data[-4:-5]
names(data)[4]<-"Hora_entrada"
data<-data[-5]
```

```
  data$Real_ParkingTime<-as.numeric(gsub(",", ".",as.character
     (data$Real_ParkingTime)))
  data$Predicted_ParkingTime<-as.numeric(gsub(",", ".",as.
     character(data$Predicted_ParkingTime)))
  data$Real_ParkingTime<-data$Real_ParkingTime+data$Hora_
     entrada
  data$Predicted_ParkingTime<-data$Predicted_ParkingTime+data$
     Hora_entrada

  rmse<-read.csv2("id_intervalo_grupo_sliding_adapt_qt.csv",
     sep=";")
  ids_rmse<-unique(rmse$id)
  for(id in unique(data$ID)){
    if(!(id %in% ids_rmse)){
      idx.id<-which(data$ID==id)
      data<-data[-idx.id,]
    }
  }
  return(data)
}

park.cars.smart.2<-function(rows, columns)
{
  dados<-data.preprocessing()
  vars<-c("dia", "tamanho", "misses")

  resultados<-matrix(0, length(unique(dados$Date)), length(
     vars))
  idx<-1
  for(dia in unique(dados$Date)){
    m<-matrix(0, rows, columns)
    m_realtime<-matrix(0, rows, columns)
    m_parkingtime<-matrix(0, rows, columns)
     dados.reg<-dados[which(dados$Date==dia),]
     dados.entrada<-dados.reg[order(dados.reg$Hora_entrada), ]
     dados.saida<-dados.reg[order(dados.reg$Real_ParkingTime),
        ]
```

```
misses<-0

while(length(dados.entrada$ID)>0){
  if(isFull(m)){
    dados.entrada<-dados.entrada[-1,]
    index<-match(dados.entrada$ID[1], dados.saida$ID)
    dados.saida<-dados.saida[-index,]
  }
  else{
    best.fila<-bestStack.2(m, dados.entrada[1,], m_
       realtime, m_parkingtime)
    linha<-nextFreeSpot.Stack(m, best.fila, rows)
    m[linha,best.fila]<-dados.entrada$ID[1]
    m_realtime[linha,best.fila]<-dados.entrada$Real_
       ParkingTime[1]
    m_parkingtime[linha,best.fila]<-dados.entrada$
       Predicted_ParkingTime[1]
  }
  dados.entrada<-dados.entrada[-1,]
  while(nextEvent(dados.entrada, dados.saida)=="saida"){
    saida.id<-dados.saida$ID[1]
    pos<-find.car(m, saida.id)
    m[pos[[1]],pos[[2]]]<-0
    m_realtime[pos[[1]],pos[[2]]]<-0
    m_parkingtime[pos[[1]],pos[[2]]]<-0
    dados.saida<-dados.saida[-1,]
    if(length(pos)>0){
      nmoves<-check.movement(m, pos[[1]],pos[[2]])
      misses<-misses+nmoves
      m<-rearrange(m, pos[[1]],pos[[2]])
      m_realtime<-rearrange(m_realtime, pos[[1]],pos[[2]])
      m_parkingtime<-rearrange(m_parkingtime, pos[[1]],pos
         [[2]])
    }
  }
}
while(length(dados.saida$ID)>0){
        saida.id<-dados.saida$ID[1]
        pos<-find.car(m, saida.id)
        m[pos[[1]],pos[[2]]]<-0
        m_realtime[pos[[1]],pos[[2]]]<-0
        m_parkingtime[pos[[1]],pos[[2]]]<-0
        dados.saida<-dados.saida[-1,]
        if(length(pos)>0){
          nmoves<-check.movement(m, pos[[1]],pos[[2]])
          misses<-misses+nmoves
          if(nmoves>0){
            m<-rearrange(m, pos[[1]],pos[[2]])
            m_realtime<-rearrange(m_realtime, pos[[1]],pos[[2]])
            m_parkingtime<-rearrange(m_parkingtime, pos[[1]],pos
                [[2]])
          }
          mask.id<-match(saida.id, mascaras_id)
          mask.print<-mascaras[mask.id]
        }
      }
      resultados[idx,]<-c(dia, paste(rows, columns, sep="x"),
          misses)
      idx<-idx+1
    }
    df<-as.data.frame(resultados)
    names(df)<-vars
    filew<-sprintf("misses_metodo_adv_%dx%d.csv", rows, columns)
    write.table(df,filew,sep=";",dec=",", row.names=FALSE)
  }

  bestStack.2<-function(m, linha_id_novo, m_realtime, m_
     parkingtime) #recebo o id do novo carro
  {
    nrows<-length(m[,1])
    ncolumns<-length(m[1,])
    melhor.coluna<-0
    melhor.ncarros<-0
    melhor.custo<-100000
```

```
novo_previsao<-0
novo_intervalo_baixo<-0
novo_intervalo_cima<-0
linha_previsao<-c()
linha_intervalo_baixo<-c()
linha_intervalo_cima<-c()
linha_score<-c()

for(i in 1:ncolumns){
  linha<-nextFreeSpot.Stack(m, i, nrows)
  if(linha!=-1 && linha != 1){
    id_ultimo<-m[linha-1, i]
    lista<-calc.Weight.2(id_ultimo, linha_id_novo, m, m_
        realtime, m_parkingtime, ncolumns, (linha-1)) ##mudar
         aqui depois

    novo_previsao<-lista[[1]]
    novo_intervalo_baixo<-lista[[2]]
    novo_intervalo_cima<-lista[[3]]
    linha_previsao<-c(linha_previsao, lista[[4]])
    linha_intervalo_baixo<-c(linha_intervalo_baixo, lista
        [[5]])
    linha_intervalo_cima<-c(linha_intervalo_cima, lista
        [[6]])
    linha_score<-c(linha_score, lista[[7]])

    peso<-lista[[7]]
    if(peso<melhor.custo){
      melhor.coluna<-i
      melhor.ncarros<-linha-1
      melhor.custo<-peso
    }
    else if(peso==melhor.custo){
      if(melhor.ncarros>(linha-1)){
        melhor.coluna<-i
        melhor.ncarros<-linha-1
```

```
            melhor.custo<-peso
          }
        }
      }
      else if(linha==1){
        peso<-1
        if(peso<melhor.custo){
          melhor.coluna<-i
          melhor.ncarros<-0
          melhor.custo<-peso
        }
        linha_previsao<-c(linha_previsao, linha_previsao[length(
           linha_previsao)-1])
        linha_intervalo_baixo<-c(linha_intervalo_baixo, linha_
           intervalo_baixo[length(linha_intervalo_baixo)-1])
        linha_intervalo_cima<-c(linha_intervalo_cima, linha_
           intervalo_cima[length(linha_intervalo_cima)-1])
        linha_score<-c(linha_score, peso)
      }
      else{
        linha_previsao<-c(linha_previsao, "-")
        linha_intervalo_baixo<-c(linha_intervalo_baixo, "-")
        linha_intervalo_cima<-c(linha_intervalo_cima, "-")
        linha_score<-c(linha_score, "-")
      }
    }
    return(melhor.coluna)
  }

  calc.Weight.2<-function(id, linha_id_novo, m, m_realtime, m_
     parkingtime, nfilas, ncarros)
  {
    rmse<-read.csv2("id_intervalo_grupo_sliding_adapt_qt.csv",
       sep=";")
    rmse$mad<-as.numeric(as.character(gsub(",", ".",rmse$mad)))
    rmse$metodo<-as.character(rmse$metodo)
    rmse$date<-as.character(rmse$date)
```

```
rmse_ultimo<-rmse[which(rmse$id==id),]
rmse_novo<-rmse[which(rmse$id==linha_id_novo$ID),]

pos<-find.car(m, id)
line_ultimo_real<-m_realtime[pos[[1]], pos[[2]]]
line_ultimo_predicted<-m_parkingtime[pos[[1]], pos[[2]]]
line_novo<-linha_id_novo

rmse_ultimo<-rmse_ultimo[which(round(rmse_ultimo$valor_real)
    ==round(line_ultimo_real) & round(rmse_ultimo$valor_
    previsto)==round(line_ultimo_predicted)),]
rmse_novo<-rmse_novo[which(round(rmse_novo$valor_real)==
    round(line_novo$Real_ParkingTime) & round(rmse_novo$valor
    _previsto)==round(line_novo$Predicted_ParkingTime) & rmse
    _novo$date==line_novo$Date),]

rmse_novo<-rmse_novo[1,]
rmse_ultimo<-rmse_ultimo[1,]

intervalo_ultimo_baixo<-line_ultimo_predicted-(rmse_ultimo$
    intervalo/2)
intervalo_ultimo_cima<-line_ultimo_predicted+(rmse_ultimo$
    intervalo/2)
intervalo_novo_baixo<-line_novo$Predicted_ParkingTime-(rmse_
    novo$intervalo/2)
intervalo_novo_cima<-line_novo$Predicted_ParkingTime+(rmse_
    novo$intervalo/2)

if(intervalo_ultimo_cima<intervalo_novo_baixo){
  custo<-20
}
else if(intervalo_novo_baixo<intervalo_ultimo_baixo &&
    intervalo_novo_cima>intervalo_ultimo_cima){
  sobreposicao<-1
  custo<-(sobreposicao)+(((ncarros-1)^(4))/nfilas)
}
else if(intervalo_ultimo_baixo>intervalo_novo_cima){
    custo<-0
  }
  else if(intervalo_novo_baixo<=intervalo_ultimo_baixo &&
      intervalo_novo_cima<intervalo_ultimo_cima){
    sobreposicao<-(intervalo_novo_cima-intervalo_ultimo_baixo)
        /(intervalo_ultimo_cima-intervalo_ultimo_baixo)
    custo<-(sobreposicao)+(((ncarros-1)^(4))/nfilas)
  }
  else if(intervalo_novo_cima>=intervalo_ultimo_cima){
    sobreposicao<-(intervalo_ultimo_cima-intervalo_novo_baixo)
        /(intervalo_ultimo_cima-intervalo_ultimo_baixo)
    custo<-(sobreposicao)+(((ncarros-1)^(4))/nfilas)
  }
  else{
    sobreposicao<-(intervalo_novo_cima-intervalo_novo_baixo)/(
        intervalo_ultimo_cima-intervalo_ultimo_baixo)
    custo<-(sobreposicao)+(((ncarros-1)^(4))/nfilas)
  }

  return(list(line_novo$Predicted_ParkingTime, intervalo_novo_
      baixo, intervalo_novo_cima,
              line_ultimo_predicted, intervalo_ultimo_baixo,
                 intervalo_ultimo_cima, custo))
}

print.parking<-function(m)
{
  row.size<-length(m[,1])
  column.size<-length(m[1,])
  m<-change.matrix(m)
  for(i in 1:row.size){
    linha<-""
    for(j in 1:column.size){
      if(m[i,j]!=0){
        linha<-paste(linha, "[ x ]", sep="")
      }
      else{
        linha<-paste(linha, "[   ]", sep="")
      }
    }
  }
}

change.matrix<-function(m)
{
  nrows<-length(m[,1])
  ncolumns<-length(m[1,])
  new_m<-matrix(0, nrows, ncolumns)
  for(i in 1:ncolumns){
    ncars<-(nextFreeSpot.Stack(m, i, nrows)-1)
    if(ncars==-2){
      new_m[,i]<-rep(1, nrows)
    }
    else if(ncars>0){
      for(j in nrows:(nrows-ncars+1)){
        new_m[j,i]<-1
      }
    }
  }
  return(new_m)
}

nextEvent<-function(dados.entrada, dados.saida)
{
  if(length(dados.entrada$ID)>0){
    if(dados.entrada$Hora_entrada[1]>dados.saida$Real_
        ParkingTime[1]){
      return("saida")
    }
  }
  return("entrada")
}

isFull<-function(m)
```

```
{
  nrows<-length(m[,1])
  ncolumns<-length(m[1,])
  for(i in 1:ncolumns){
    if(m[nrows, i]==0){
      return(FALSE)
    }
  }
  return(TRUE)
}

nextFreeSpot.Stack<-function(m, column, nrows)
{
  for(i in 1:nrows){
    if(m[i, column]==0){
      return(i)
    }
  }
  return(-1)
}

there.is.FreeSpace<-function(m)
{
  for(i in 1:length(m[1,])){
    if(m[1,i]==0){
      return(i)
    }
  }
  return(0)
}

find.car<-function(m, id)
{
  nrows<-length(m[,1])
  ncolumns<-length(m[1,])
  for(i in 1:nrows){
    for(j in 1:ncolumns){
      if(m[i,j]==id){
        return(list(i,j))
      }
    }
  }
  return(NULL)
}

check.movement<-function(m, nr, nc)
{
  nrows<-length(m[,1])
  nmoves<-0
  if(nr!=nrows){
    while(m[nr+1,nc]!=0 && (nr+1)!=nrows){
      nmoves<-nmoves+1
      nr<-nr+1
    }
    if((nr+1)==8)
      nmoves<-nmoves+1
  }
  if(nmoves>0){
    return(sum(1:nmoves))
  }
  else{
    return(0)
  }
}

rearrange<-function(m, nr, nc)
{
  nrows<-length(m[,1])
  while(nr!=nrows){
    m[nr, nc]<-m[nr+1,nc]
    nr<-nr+1
  }
  m[nrows, nc]<-0
  return(m)
```

```
}


bestStack.naive<-function(m, id_novo, data.reg)
{
  nrows<-length(m[,1])
  ncolumns<-length(m[1,])
  melhor.coluna<-0
  melhor.ncarros<-100000
  for(i in 1:ncolumns){
    linha<-nextFreeSpot.Stack(m, i, nrows)
    if(linha!=0){
      if(linha<melhor.ncarros){
        melhor.coluna<-i
        melhor.ncarros<-linha
      }
    }
  }
  return(melhor.coluna)
}

park.cars.naive<-function(rows, columns)
{
  dados<-data.preprocessing()
  vars<-c("dia", "tamanho", "misses")

  resultados<-matrix(0, length(unique(dados$Date)), length(
     vars))
  idx<-1
  for(dia in unique(dados$Date)){
    m<-matrix(0, rows, columns)
    dados.reg<-dados[which(dados$Date==dia),]
    dados.entrada<-dados.reg[order(dados.reg$Hora_entrada), ]
    dados.saida<-dados.reg[order(dados.reg$Real_ParkingTime),
       ]
    misses<-0
```

```
while(length(dados.entrada$ID)>0){
  freeSpace<-there.is.FreeSpace(m)
  if(freeSpace!=0){
   m[1,freeSpace]<-dados.entrada$ID[1]
  }
  else if(isFull(m)){
    dados.entrada<-dados.entrada[-1,]
    index<-match(dados.entrada$ID[1], dados.saida$ID)
    dados.saida<-dados.saida[-index,]
  }
  else{
    best.fila<-bestStack.naive(m, dados.entrada$ID[1],
       dados.entrada)
    linha<-nextFreeSpot.Stack(m, best.fila, rows)
   m[linha,best.fila]<-dados.entrada$ID[1]
  }
  dados.entrada<-dados.entrada[-1,]
  while(nextEvent(dados.entrada, dados.saida)=="saida"){
    saida.id<-dados.saida$ID[1]
    pos<-find.car(m, saida.id)
    if(length(pos)>0){
      m[pos[[1]],pos[[2]]]<-0
      nmoves<-check.movement(m, pos[[1]],pos[[2]])
      misses<-misses+nmoves
      if(nmoves>0){
        m<-rearrange(m, pos[[1]],pos[[2]])
      }
    }
    dados.saida<-dados.saida[-1,]
  }
}
while(length(dados.saida$ID)>0){
  saida.id<-dados.saida$ID[1]
  pos<-find.car(m, saida.id)
  if(length(pos)){
   m[pos[[1]],pos[[2]]]<-0
    nmoves<-check.movement(m, pos[[1]],pos[[2]])
```

```
        misses<-misses+nmoves
        if(nmoves>0){
          m<-rearrange(m, pos[[1]],pos[[2]])
        }
      }
      dados.saida<-dados.saida[-1,]
    }
    resultados[idx,]<-c(dia, paste(rows, columns, sep="x"),
       misses)
    idx<-idx+1
  }
  df<-as.data.frame(resultados)
  names(df)<-vars
  filew<-sprintf("misses_metodo_naive_%dx%d.csv", rows,
     columns)
  write.table(df,filew,sep=";",dec=",", row.names=FALSE)
}
```

Using Exit Time Predictions to Optimize Self Automated Parking Lots

Rafael Nunes, Luis Moreira-Matias and Michel Ferreira

Abstract— **Private car commuting is heavily dependent on the subsidisation that exists in the form of available free parking. However, the public funding policy of such free parking has been changing over the last years, with a substantial increase of meter-charged parking areas in many cities. To help to increase the sustainability of car transportation, a novel concept of a self-automated parking lot has been recently proposed, which leverages on a collaborative mobility of parked cars to achieve the goal of parking twice as many cars in the same area, as compared to a conventional parking lot. This concept, known as self-automated parking lots, can be improved if a reasonable prediction of the exit time of each car that enters the parking lot is used to try to optimize its initial placement, in order to reduce the mobility necessary to extract blocked cars. In this paper we show that the exit time prediction can be done with a relatively small error, and that this prediction can be used to reduce the collaborative mobility in a self-automated parking lot.**

I. Introduction

Parking is a major problem of car transportation, with important implications in traffic congestion and urban landscape. It has been shown that parking represents 75% of the variable costs of automobile commuting [1], supported by a major public subsidisation of the space devoted to car parking, where the user does not pay in more than 95% of the occasions [2].

The sustainability of car transportation is nowadays facing several challenges. The number of cars in many cities has reached a level where the road infrastructure is unable to avoid systematic traffic congestions. In addition, the high cost of fossil fuels and pollutant emission levels are creating significant challenges for the sustainability of private car commuting in major cities. Tolls and prohibition of circulation in one or two week days for a given vehicle are already in place in some of our cities. Technology is trying to mitigate these challenges faced by car transportation. Zero-emissions electric propulsion and connected navigation are two examples of technologies that can help making car transportation more sustainable.

Technology has been focusing however in moving cars, disregarding the parked period of these cars, which represents 95% of the vehicle existence. Recently, a simple proposal that leverages on technology such as electric propulsion or wireless vehicular connectivity has addressed the issue of car parking, arguing that through a collaborative approach to the parking of cars, the area per car could be reduced to nearly half, when compared to the area per car in a conventional parking lot. This approach, known as *self-automated parking lots* [3], works as follows. An electric vehicle (EV) is left at the entrance of a parking lot by its driver. This EV is equipped with vehicular communications that establish a protocol with a Parking Lot Controller (PLC). The EV is also based on Drive-by-Wire (DbW) technology, where in-vehicle Electronic Control Units (ECUs) manage signals sent by the acceleration and braking pedal, and steering wheel. The Vehicle-to-Infrastructure (V2I) communication protocol allows the PLC to control the mobility of the EV in the parking lot. The PLC remotely drives the EV to its parking space, using in-vehicle positioning sensors (e.g. rotation per wheel), magnet-based positioning, or some other type of positioning system (e.g. camera-based). Alternatively to a fully-automated system, a scenario of human-based tele-operated driving could also be used [4]. In this concept of self-automated parking lots the cars are parked in a very compact way, without space devoted to access ways or even inter-vehicle space that allows opening doors. As a new vehicle enters the parking lot, the PLC sends wireless messages to move the vehicles in the parking lot to create space to accommodate the entering vehicle. If a blocked car wants to leave the parking lot, the PLC also sends messages to move the other vehicles, in order to create an exit path. In [3] it was shown that this concept could reduce the area per vehicle to nearly half, as well as reduce the overall mobility of cars in the parking lot, when compared to a conventional parking lot. However, in the original paper, a first-fit strategy was used to initially park each vehicle. Clearly, the initial placement can be improved if some knowledge about the expected exit time of each car is used. The basic idea is that a car should not be blocked by another car that will leave the parking lot later. If the cars in the parking lot are placed using an order that reflects their expected exit times, then the overall mobility in the parking lot to create exit paths can be reduced.

In this paper we use an entire year of entries and exits in a parking lot, where each vehicle uses a unique identifier, to be able to derive its expected exit time, using this information to improve the original placement of the car in order to reduce manoeuvring mobility. Our goal is not to obtain a precise exit time for each vehicle, but rather a time-interval that can be

Manuscript submitted June 22, 2014.
Manuscript reviewed August 19, 2014.

Rafael Nunes is with the Faculdade de Engenharia, U.Porto, 4200-465 Porto - Portugal (e-mail: rafael.nunes[at]dcc.fc.up.pt).
Luis Moreira-Matias is with the Instituto de Telecomunicações, 4200-465 Porto, Portugal and with Faculdade de Engenharia, U.Porto, 4200-465 Porto - Portugal (phone: 00351-91-4221647; e-mail: luis.matias[at]fe.up.pt).
Michel Ferreira is with the Instituto de Telecomunicações, U.Porto, 4169-007 Porto - Portugal (e-mail: michel[at]dcc.fc.up.pt).

This work was supported by the project I-CITY - "ICT for Future Mobility", as part of Grant NORTE-07-0124-FEDER-000064.

used in conjunction with the parking lot layout (e.g. number of lanes) to reduce the probability of having to move parked vehicles to created exit paths for blocked vehicles.

The remainder of this paper is organised as follows: in the next section we present some considerations regarding parking lot design, and further describe our optimisation goal based on a typical layout for a self-automated parking lot. We then present our methodology to predict an exit time interval for each vehicle, and how this interval is used to select the original lane to park each vehicle. We then present our dataset set used as case study and present experimental results in the next section, including a discussion of these results. Finally, we end with some conclusions.

II. Parking Lot Design

The geometric design of the parking lot is an important issue in a self-automated parking lot. In conventional parking lots there are a number of considerations that have to be taken into account when designing them. For instance, width of parking spaces and access ways, one-way or two-way use of the access ways, entry angle in the parking bays ($90°, 60°, 45°$), pedestrian paths, visibility to find an available parking space, etc. In a self-automated parking lot, many of these considerations do not apply. Manoeuvring is done autonomously by the car following the instructions of the PLC, pedestrian access is not allowed, and the assigned parking space is determined by the PLC. The main design issue is defining a geometric layout that maximises parking space, leveraging on minimal buffer areas to make the necessary manoeuvres that allow the exit from any parking space under all occupancy configurations. This geometric design is ultimately determined by the shape of the space of the parking lot. The parking lot architecture also defines the trajectories and associated manoeuvres to enter and exit each parking space.

The parking lot has a V2I communication device which allows the communication between the vehicles and the PLC. In theory, this infrastructure equipment could be replaced by a vehicle in the parking lot, which could assume the function of PLC while parked there, handing over this function to another car upon exit, similarly to the envisioned functioning of a V2V Virtual Traffic Light protocol [5]. Note, however, that the existence of the actual infrastructure, which could be complemented with a video-camera offering an aerial perspective of the parking lot to improve the controller perception of the location and orientation of vehicles, could simplify the protocol and improve reliability.

Reducing and simplifying such trajectories and manoeuvres is also an important design issue, as they affect the reliability of the system and allow faster storage and retrieval of cars. Note also that the parking lot architecture can take advantage of the fact that the passenger does not enter the parking lot, and thus the inter-vehicle distances do not need to allow for space to open doors. To optimise and simplify manoeuvres, these self-automated parking lots will require specific minimum turning radius values for vehicles.

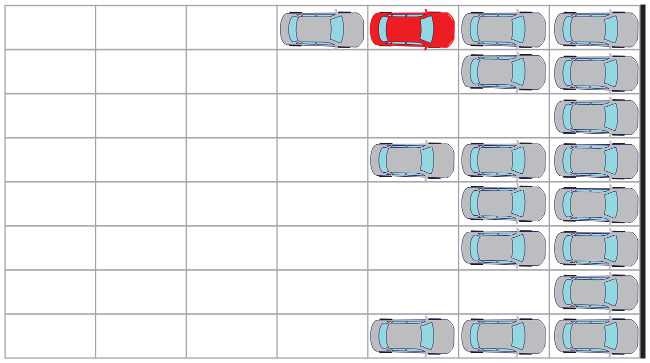


Fig. 1. An example layout for a self-automated parking lot. The parking lot can never be completely full, as buffer areas are necessary to be able to allow the exit of each vehicle under all possible configurations. In this example, a minimum of 6 empty spar are necessary.

Only vehicles that meet the turning radius specified by each parking lot will be allowed to enter it.

The geometric layout of the parking lot and its buffer areas can assume very different configurations for the self-automated functioning. One possibility is to have parallel lanes with minimal space between them, as illustrated in Fig. 1. In this type of layout, the PLC starts by assigning a lane to a vehicle. This initial decision is critical, as it should minimise the need to move a vehicle from one lane to another. Note that if the red vehicle in Fig. 1 needs to leave under the current configuration, then the vehicle behind it needs to be moved to another lane. If we could predict that the exit of the red vehicle would happen before the exit of the vehicle behind it, then this last vehicle would be better placed in a different lane. Our goal in this paper is exactly to be able to predict an exit-interval for each vehicle, and design a lane selection methodology that reduces the mobility needed to create exit paths.

Note that parking lots will not be able to be completely full, as buffer space needs to exist to allow the exit of each vehicle under all possible configurations. The minimum number of empty spaces, configuring buffer areas, depends on the parking lot layout. In the layout presented in Fig. 1, with a lane depth of 7, we need a buffer area with a minimum of 6 empty spaces.

III. Methodology

Our methodology consists on the following four steps: it starts by (A) dividing the original dataset in k smaller ones, containing users with similar parking habits; then, (B) data driven regression is performed over the newly created sub-datasets. Thirdly, a parking time interval is generated (C) based on such predictions and on their previous residuals (difference between a predicted value ($\hat{y}$) and its real one, y). Finally the selected lane (D) will be the one which minimizes the likelihood of performing *unnecessary* vehicle movements

[1]. This methodology is summarized in Fig. 2 and explained in detail throughout this section.

A. *Profile Generation*

Let $\mathbb{X} = \{X_1, X_2, ..., X_n\}$ be n timestamped data records on the parking lot entries describing the entry/exit behaviours of ρ distinct users. Let $U_i \subseteq \mathbb{X}$ denote the records of and individual user i (i.e. $U_{i=1}^{\rho} \equiv \mathbb{X}$) and Ψ_i describe the sample-based probability density function (*p.d.f.*) of its parking time habits. A clustering process is firstly made on $\mathbb{X}$ based on the extracted Ψ_i. The resulting k clusters can be defined as $\Pi = \{\pi_1, \pi_2, ..., \pi_k\}$. They will comprise sub-datasets containing data records on users having similar **profiles** (i.e. parking time-habits). Consequently, $\mathbb{X} \equiv \bigcup_{i=1}^{k} \pi_i$.

B. *Parking Time Prediction*

To perform the parking time prediction, we propose to use **data driven regression**. In regression, the goal is to determine a function $f(Z, \theta)$, given the input independent variables, Z, and the real values of the dependent variables, θ. The output of the model is not necessarily equal to the real value, due to noise in the data and/or limited number of entries. Consequently, a regression model commonly comprises an error e. The function f can be expressed as follows:

$$Y \approx f(Z, \theta) + e \tag{1}$$

Let $\mathbb{M} = \{M_{\pi_1}, M_{\pi_2}, ..., M_{\pi_k}\}$ be the set of k regression models and p_{j,π_i} denote the parking time prediction for a given timestamped user entrance with the profile π_i. $\mathbb{M}$ results of applying an induction method of interest to the datasets in Π. By doing so, the authors expect to approximate the real vehicles parking time given a set of describing variables (i.e.: Z).

C. *Incremental Interval Generation*

Given a prediction for the parking time of an user timestamped entrance (i.e. p_{j,π_i}), it is possible to estimate an **interval** for this value based on the residuals produced by its regression model. Hereby, we propose to do so by employing the residuals' *quantiles*. A quantile is a point taken from a cumulative distribution function of a variable. The first quantile represents the point that is greater than 25% of the data, while the third quantile the point that is greater than 75%. Let $e_{1,i}$ and $e_{3,i}$ denote the first and third quantiles of the regression *residuals* produced by a given model M_{π_i} on the previously tested data records in π_i. Our baseline interval I is given by the following equation:

$$I_{j,\pi_i} = [p_{j,\pi_i} - e_{1,\pi_i}, p_{j,\pi_i} + e_{3,\pi_i}] \tag{2}$$

Let a **hit** occur every time the real parking time is contained within the interval estimated. Otherwise, we consider the occurrence of a **miss**. Our goal is to produce intervals in order to maximize the number of hits and, at the same time, to minimize its **width**. To do so, we propose to extend the baseline described in eq. (2) by employing a *self-adaptive* strategy. Such strategy consists on multiplying the quantile-based interval width by a $0 \leq \beta \leq 2$ (starting on $\beta = 1$). This value is *incrementally* updated whenever an user of π_i leaves the parking lot (i.e. each time a newly real parking time is known on π_i). Let α_{π_i} denote the number of *consecutive* misses/hits of our interval prediction method in π_i. Whenever $\alpha_{\pi_i} > \alpha_{th}$, the value of β is incremented/decremented by τ. α_{th} and τ are two user-defined parameters setting how **reactive** the interval prediction model should be. Consequently, it is possible to re-write the eq. (2) into the following one:

$$I_{j,\pi_i} = [p_{j,\pi_i} - \Delta, p_{j,\pi_i} + \Delta], \Delta = (e_{3,\pi_i} - e_{1,\pi_i}) \times \beta \tag{3}$$

Everytime that a sequence miss/hit or hit/miss occurs, the respective α value is set to 0. The β ends up by controlling the interval width: the described algorithm aims to **adapt itself** to the current scenario by *narrowing* the intervals width whenever it is getting multiple hits or by *stretching* itself on the opposite scenario.

D. *Parking Lane Selection*

In this paper, the parking lot is assumed to follow a rectangular layout where the entrance and the exit are the same. It is possible to represent it as a $l \times r$ matrix, where l, r sets the number of **lanes** and the maximum number of vehicles in each lane, respectively. When a vehicle enters the parking lot,it is necessary to select a lane κ to park it in. Such selection should minimize the number of unnecessary vehicle movements (i.e. ϑ_κ). Consequently, each lane has an associated **score** W_κ. It can be faced as a likelihood of that selection force unnecessary movements given the i) current interval prediction for the newly arrived user (I_{j,π_i}) and ii) the vehicles already parked in κ. The lane with lowest score is predicted to be the one that minimizes ϑ_κ.

Empty lanes have a predefined score of $W = 1$ while a full one have $W = \infty$. Let h be the *last* vehicle in κ (i.e. the vehicle most recently parked), I_{j,π_i}^{U} be the upper limit and I_{h,π_b}^{L} be the lower limit of the estimated interval (note that the vehicle's j profile, π_i, may be (or not) the same of the vehicle h, π_b). If $I_{h,\pi_b}^{U} < I_{j,\pi_i}^{L}$, it is expected that the vehicle j of profile π_i exits the parking lot first than h (e.g.: Fig. 3-c). In this case, $W_\kappa = \infty$. If $I_{j,\pi_i}^{U} < I_{h,\pi_b}^{L}$, then it is expected that j and h can leave the parking lot provoking no unnecessary movements (i.e.: $\vartheta_\kappa = 0$; e.g.:

[1]Whenever a given vehicle c exits, all its lane's vehicles standing between c and the parking lot exit, have to be moved to a buffer zone. Such movements could be avoided by an exit-oriented sorting of each lane's vehicles.

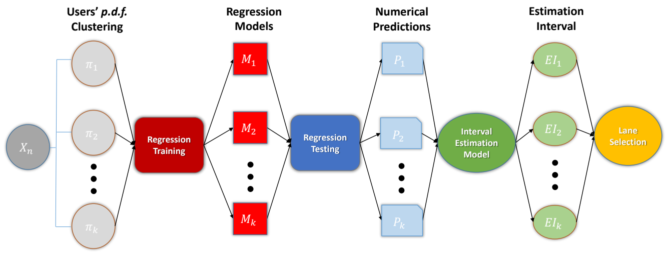


Fig. 2. An illustration on the different steps of the proposed methodology.

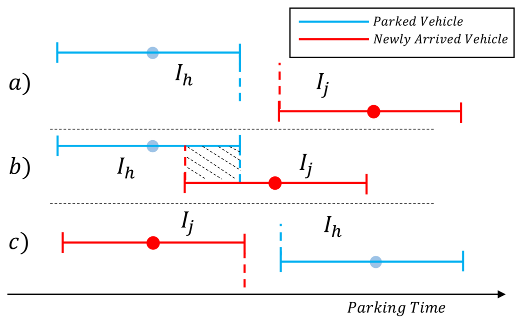


Fig. 3. In a), the upper limit of I_h is lower than the lower limit of I_j, so h is expected to leave the parking lot first than j. In b) there is an overlap between the two intervals. Its width is used to compute the lane's score. Finally, c) is the opposite scenario of a).

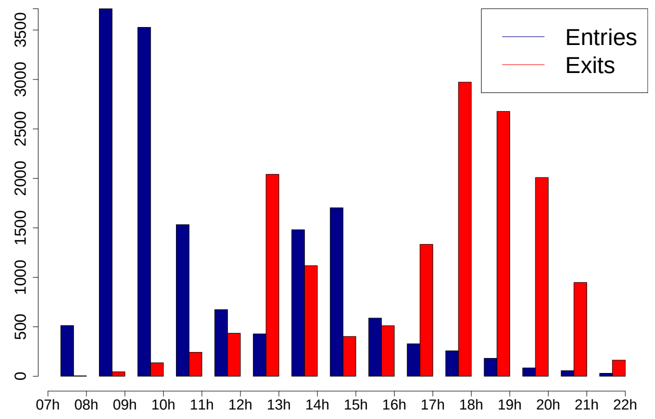


Fig. 4. Barplot chart representing histograms for the Entry/Exit times between 7am and 10pm.

Fig. 3-a). Consequently, the score is then $W_\kappa = 0$ on this case. Otherwise, W_κ can be computed as follows

$$W_\kappa = \frac{I^U_{j,\pi_i} - I^L_{h,\pi_b}}{I^U_{h,\pi_i} - I^L_{h,\pi_b}} + \frac{(N_\kappa - 1)^4}{r} \tag{4}$$

where N_κ stands for the number of vehicles currently in κ. This approach is inspired on the typical *p-value* statistical test considering a null hypothesis by setting the *extreme* data point as I^U_{j,π_i} and I_{h,π_i} as a *rough* approximation on the parking time distribution function for the parked vehicle h. The second term of eq. (4) is an exponential weight which aims to express the possible cost of having unnecessary vehicle movements caused by assigning the newly arrived vehicle j to the lane κ.

IV. Case Study

This case study consists on the parking lot of the Faculty of Science of University of Porto, Portugal. The data of 309 users during the year of 2013 was used to validate our methodology. This parking lot has the capacity to hold up to 100 vehicles. Since 96.4% of the data entries are in week days, only the workdays are considered in this study.

Each data record has the following features: (i) an user ID, (ii, iii) two timestamps for the parking entry/exit, (iv) type of day (e.g.: Monday), (v) holiday/not-holiday boolean and, finally, the (vi) department, (vii) sex and (viii) job role (e.g. Full Professor).

Ideally all data entries would have their entry and exit times properly labelled. However, it does not happen in this case because the parking entries/exits are not fully monitored. Consequently, there are entries without exits and vice-versa. To tackle such issue, a preprocessing task to pair the entries with the exits was performed. All the resulting data records with parking time smaller than 10 minutes or higher than 16 hours were removed. For the same reasons, we have also filtered the parking lot users by using the data records of the top-75%, regarding their number of parking entries.

In the resulting dataset, the average parking time is 5 hours and 25 minutes and with a standard deviation of 3 hours and 8 minutes. Fig. IV exhibits two histograms representing the hourly frequencies on the entry and exit times. It is possible to observe that the main entry times are between 8am and 10am and the main exit times between 5pm and 8pm. The vehicle's exits from the parking lot follows a bimodal distribution, with the modes at lunch time (between 12am and 2pm) and at late afternoon (between 5pm and 7pm).

V. Experimental Results

In this section, we start by describing the experimental setup used in our experiments and the evaluation metrics used to validate our methodology. Then, we present some experimental results and a brief discussion on their insights.

A. Experimental Setup

The initial dataset was divided in a training set (January to October) and a test set (November). All experiments were conducted using R Software [6]. The algorithms used were the k-Nearest Neighbours (kNN) [7], the Random Forests (RF) [8], the Projection Pursuit Regression (PPR) [9] and the Support Vector Machines (SVM) [10] from the R packages [`kknn`], [`randomForests`], [`stats`] and [`e1071`].

Regarding the feature selection, a well-known state-of-the-art technique was used: Principal Component Analysis (PCA) [11]. The tested features were type of day, holiday/not-holiday boolean variable and the user's department, sex and job role. For clustering we used the Expectation-Maximization algorithm with the R package [`MClust`]. This algorithm was chosen due to being able to determine the optimal number of clusters automatically based on Bayesian Information Criterion [12].

The last 2 weeks of the training set was used for model selection. In this stage, the following parameters were tested for each algorithm: for kNN, $distance = [1..5]$, $kMax = [2..15]$ and the kernels: rectangular, triangular, epanechnikov, gaussian, rank and optimal, for RF $mtry = \{3, 4, 5\}$ and $ntrees = \{500, 750, 1000\}$, for PPR $nterms = \{2, 3, 4\}$ and $max.terms = \{5, 6, 7, 8\}$ and for SVM the kernels: linear, radial, polynomial and sigmoid. The best pair (algorithm,parameter setting) was selected to perform the numerical prediction in the test set.

Finally, the reactiveness parameters on the interval estimation model (τ, α_{th}) were set for the values 0.1 and 3, respectively.

To evaluate our method performance, we considered a baseline naive strategy. It consists on directing the newly arrived vehicle to the leftmost lane κ with an empty space. A series of simulations were conducted to compare the parking lot behavior using the aforementioned lane selection strategies (i.e. *naive* and *smart*). Multiple parking layouts were considered on this series of simulations. It aimed to demonstrate that the strategies behavior is **independent** on the parking layout. The averaged maximum number of parked cars on a daily basis on the considered dataset is 50. Consequently, every parked layouts with a capacity between 50 and 80 vehicles (i.e.: the 1^{st} quantile) containing, at least, 8 lanes, were considered on our experiences.

B. Evaluation

The root-mean-squared-error (RMSE) and the mean absolute error (MAE) were the metrics used to evaluate the predictions. They can be defined as follows:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{g}(\hat{y}_t - y_t)^2}{g}}, MAE = \frac{\sum_{t=1}^{g}|\hat{y}_t - y_t|}{g} \tag{5}$$

where $\hat{y}$ is the predicted value, y the real one and g is the number of samples.

The parking time estimation interval is evaluated in two forms, a percentage of hits and a ratio between the hits and its width. If for a sample s there is a hit, then $hit_s = 1$, otherwise $hit_s = 0$. The ratio can be defined as:

$$ratio = \sum_{s=1}^{g} hit_s \times \frac{1}{\delta_I \times g} \tag{6}$$

where δ_I is the width of the estimation interval and g is the number of considered samples.

The evaluation criteria employed in the simulation was the total number of unnecessary vehicle movements forced by a given strategy (i.e., UM). Let us consider a exiting vehicle c, parked in a lane κ with g vehicles, in position i. The unnecessary number of movements UM caused for c to exit the parking lot can be computed as:

$$UM = \sum_{j=1}^{g-i} j \tag{7}$$

Let us consider a lane with $g = 5$ vehicles where the vehicle on the position $i = 2$ is requested to exit as an exemplification for the calculus of MU. In this case, $MU = 3 + 2 + 1 = 6$.

C. Results

The obtained results are three fold: (1) the PCA results have recommended to remove the user's sex and the holiday feature from the original set. (2) Table I exhibits the results of the numerical prediction using the remaining feature set for each profile π_i, by pointing the number of users contained in each group and the (RMSE,MAE) obtained in each one of them. (3) Table II shows the results from the parking simulation in every tested configurations, with the number of unnecessary vehicle movements, μ for both strategies. The intervals generated had **65%** hits and an average interval width of $\approx$ **11000** seconds. The *smart* strategy overcomes the *naive* one in all the considered configurations.

TABLE I
RESULTS FROM THE NUMERIC PREDICTION.

Group	# of Individuals	RMSE	MAE	Hit %	Interval
1	11	5124	3320	63	8942
2	9	4804	3255	66	3862
3	3	7047	5235	68	9584
4	6	4644	4047	78	9764
5	1	7716	5458	82	3482
6	1	376	340	72	3504
7	5	3968	3317	68	9196
8	7	7618	6101	58	11738
9	11	9106	7628	53	11900
10	6	8244	7403	55	12560
11	4	2609	2058	72	5255
12	10	7871	5436	67	9583
13	6	8901	5789	72	9558
14	7	8595	6883	54	11228
15	4	5981	4804	50	6258
16	10	6682	5356	70	10293
17	1	361	298	50	3158
	W.Average	**6601**	**5076**	**65**	**11188**

TABLE II
SIMULATION RESULTS WITH THE NUMBER OF UNNECESSARY VEHICLE MOVEMENTS FOR BOTH STRATEGIES.

Config.	Naive	Smart	Config.	Naive	Smart
10x05	1665	**1379**	*05x10*	7799	**7540**
11x05	1482	**1205**	*05x11*	7817	**7615**
12x05	1255	**1074**	*05x12*	7817	**7633**
13x05	1074	**914**	*05x13*	7817	**7633**
14x05	937	**813**	*05x14*	7817	**7633**
15x05	811	**771**	*05x15*	7817	**7633**
09x06	2234	**2032**	*06x09*	5596	**5423**
10x06	1819	**1583**	*06x10*	5596	**5444**
11x06	1510	**1282**	*06x11*	5596	**5453**
12x06	1255	**1139**	*06x12*	5596	**5453**
13x06	1074	**930**	*06x13*	5596	**5453**
08x07	2808	**2520**	*07x08*	3818	**3545**
09x07	2248	**2116**	*07x09*	3818	**3545**
10x07	1819	**1616**	*07x10*	3818	**3551**
11x07	1510	**1303**	*07x11*	3818	**3551**
07x08	3818	**3545**	*08x07*	2808	**2520**
09x08	2248	**2116**	*08x09*	2808	**2535**
10x08	1819	**1617**	*08x10*	2808	**2535**
08x08	2808	**2535**			

D. Discussion

Table I exhibits a large variation on RMSE/MAE produced by the models of the different groups. The groups size is also different from group to group. These groups can be faced as **profiles** which describe the *typical* parking behavior of the users within. It is possible to observe that some groups contain only one user (i.e. 5,6,17) which indicates that they have a completely different profile than the remaining ones. So far, such profiles are only based on each user's parking time (namely, by using the Euclidean Distance over their *p.d.f.*). However, some users can experience large variations on their parking time depending on some subsets of feature values (i.e. to enter the parking lot at morning or at afternoon). This fact can partially explain the above mentioned RMSE/MAE variability.

The averaged hits percentage (65%) and its large width uncover the stochasticity of the parking time variable given the current feature set. In fact, it is reasonable to admit that we may need other features to improve our prediction model such as weather or event-based ones (e.g. a sunny day or a special soccer match may reduce/increase the parking time). However, we cannot sustain these insights on the present results.

The *naive* strategy is clearly benefited by configurations with more lanes, where the UM can be naturally minimized by underusing the total lane's capacity by filling first the empty ones. In fact, this strategy is already focused on minimizing UM by maintaining the maximum number of vehicles parked on a lane as **low** as possible. Such behaviour can explain some of the lower gain margins presented by the *smart* strategy on some configurations (check Table II). Obviously, the UM could also be minimized by moving vehicles from one lane to another. However, the discussions about the optimal parking layout for each case study and on the parked vehicle's self-arrangements are out of this paper's scope.

Even considering the abovementioned drawbacks, the authors want to highlight that the **proposed methodology overcomes the *naive* strategy for all the presented parking layouts**. The aim with this work is to demonstrate that is possible to **mine** both the historical and the real-time data on the parking lot entrances/exits to improve the lane selection on a self-automated parking lot. This stepwise framework takes advantage of *off-the-shelf* Machine Learning algorithms to do it so. In our opinion, this proof of concept represents a consistent **breakthrough** on this relevant topic by opening promising research lines to be explored by other researchers.

VI. Final Remarks

Throughout this paper, a Machine Learning framework to predict the exit times on a self-automated parking lot is proposed. It consists on using historical data on the entries/exits on the parking lot to uncover user's profiles able to explain their parking habits. Our goal is to optimize the vehicle's initial placement by improving the lane selection using such predictions. The experiments demonstrated that our method can overcome a naive strategy by **reducing the collaborative mobility needs on roughly 10%**. By doing so, we hope to open new research lines on this topic.

As future work, we propose to explore the inter-lane vehicle movements to re-arrange their placements. Such movements aim to *react* to the parking current status by a) updating the exit time predictions while the vehicles are still parked or by b) moving the blocking vehicles to their neighbour lanes instead of using the buffer. The validity of such hypothesis comprise open research questions.

References

[1] D. C. Shoup, A. P. Association, *et al.*, *The high cost of free parking*. Planners Press Chicago, 2005, vol. 206.

[2] D. C. Shoup, "Cruising for parking," *Transport Policy*, vol. 13, no. 6, pp. 479–486, 2006.

[3] M. Ferreira et al., "Self-automated parking lots for autonomous vehicles based on vehicular ad hoc networking," in *Proc IEEE Intelligent Vehicles Symp. - IV*, Dearborn, Michigan , United States, June 2014.

[4] T. T. Sebastian Gnatzig, Frederic Chucholowski and M. Lienkamp, "A system design for teleoperated road vehicles," in *ICINCO 2013 - Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics*, 2014, pp. 231–238, 10th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2013.

[5] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," in *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*. ACM, 2010, pp. 85–90.

[6] R. C. Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2012. [Online]. Available: http://www.R-project.org

[7] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, 1992.

[8] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[9] J. H. Friedman and W. Stuetzle, "Projection pursuit regression," *Journal of the American statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[10] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[11] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[12] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 1–38, 1977.