Progress in Artificial Intelligence (2022) 11:367–396
https://doi.org/10.1007/s13748-022-00291-5

REGULAR PAPER

Exploring how independent variables influence parking occupancy prediction: toward a model results explanation with SHAP values

Hanae Errousso[1,2,3] · El Arbi Abdellaoui Alaoui[4] · Siham Benhadou[1,2] · Hicham Medromi[1,2]

Received: 18 February 2021 / Accepted: 16 August 2022 / Published online: 25 September 2022

Abstract
Finding a parking space is a difficult challenge that drivers face on a daily basis in urban neighborhoods around the world. They often report that desirable spaces near to their destination are either unavailable or very expensive, extending further the search time and congesting even more city centers. Intelligent parking solutions can integrally solve this ongoing problem by better managing existing resources. They allow drivers to access real-time information on parking space availability, collected with different detection techniques (crowdsourcing, parking meters, sensors). Some of these systems also encompass opportunistic services, such as forecasting, needed to adapt to unforeseen dynamic situations. Hence, we presented, in this paper, a methodology for predicting car park occupancy rates using four different machine learning algorithms. Each of these methods is trained with four feature sets to exemplify how information quality impacts prediction accuracy. In addition to achieving high accuracy, it is absolutely crucial to interpret model outputs and analyze each individual feature's importance. That's why we developed an explanation model based on SHAP values. We implemented our proposal exploiting five months of real-time parking data broadcast by Aarhus City Council. Results show that the best-obtained predictions are by far very accurate with a coefficient of determination (R^2) that achieves 0.988 and a mean absolute error that doesn't exceed 2.021%, while requiring a very low computing time that is only 5 s.

Keywords Parking occupancy · Prediction · Machine learning methods · SHAP values · Explanation model · Game theory

Abbreviations

T	Time
D	Day of week
w	Weather
h	Holiday
T	Temperature
l	Location
D	Distance
E	Event
AFE	Average forecast error
R^2	Coefficient of determination
P	Parking price
Pc	Parking capacity
Ro	Rate of vehicles occupying
Rl	Rate of vehicles leaving
Po	Previous observations
idA	Area name
idP	Parking identifier
POR	Parking occupancy rate
SDFE	Standard deviation forecast error
NAP	Number of available places
SP	Survival probability
Dt	Duration time
MAE	Mean absolute error
MAPE	Mean absolute percentage error
MSE	Mean square error
MNE	Mean normalized error
RMSE	Root mean square error
RRSE	Root relative squared error

✉ El Arbi Abdellaoui Alaoui
abdellaoui.e@gmail.com

[1] Research Foundation for Development and Innovation in Science and Engineering, 8118 Casablanca, Morocco

[2] National and High School of Electricity and Mechanic, HASSAN II University, 8118 Casablanca, Morocco

[3] EIGSI, 20410 Casablanca, Morocco

[4] Ecole Normale Supérieure, Moulay Ismail University, 3104 Meknes, Morocco

1 Introduction

The world population is experiencing an acceleration of its growth rate. It will reach around 8.5 billion in 2030, increase to 9.7 billion in 2050 and 11.2 billion in 2100 [1]. Cities today bring together half of humanity. Coupled with the massive rural–urban migration, urban infrastructure and services are pushed to their limits regarding environment, scalability and security. Municipalities are looking for a sustainable economy to improve energy efficiency, optimize natural resources, effectively manage urban infrastructure and minimize carbon emission levels. They are therefore obliged to capitalize on the fast progress of information and communication technologies and evolve toward a "smart city" [2].

This concept is defined as a city that performs prospectively in its fundamental components like governance, mobility, environment, lifestyle, etc. It connects the IT, physical, social and business infrastructures to leverage the city's collective intelligence [3]. Many cities in Asia, Europe and North America have led their own smart city initiatives, citing for example Yokohama Smart City, SmartSantander, LIVE Singapore and CITYKEYS. These and other projects were realized thanks to the emergence of the internet of things (IoT). It allows easy access and interaction with a wide variety of equipment including household appliances, surveillance cameras, monitoring sensors, parking meters, actuators, displays, vehicles and so on [4]. It generates, as a result, a huge amount of varied data that can be useful in the development of several applications like smart grids, mobile healthcare, traffic management and wearable.

Intelligent transportation systems (ITS) [89], a potential application of IoT, remain a pressing need to manage parking facilities as well as traffic and therefore avoid the challenges caused by their mishandling. ITS improve travel safety, increase transportation system effectiveness, reduce traffic congestion, ensure daily mobility sustainability and enhance drivers' experiences. They cover a wider range of technological solutions, the main ones being parking assistance, guidance systems and information service.

Finding a parking space in large cities is a real struggle that causes frustration, waste of money, air pollution and greenhouse gas emissions. Cruising for parking contributes to 30% of traffic on average [5] and to about 40% in peak demand [6]. Drivers spend between 3.5 and 14 min seeking for a curbside parking spot [5]. In a large city like Chicago, the number of curbside parking spaces is more than 35 000, resulting in 172 million vehicle miles traveled per year in search for parking [7]. This is equivalent to 8.37 million gallons of gasoline consumption and more than 129 000 tons of CO2 emissions. All these tricky problems have arisen due to missing detailed information on available parking spaces at a given time. This means that their resolution simply requires collecting necessary data from the already installed infrastructure (cameras, sensors, parking meters, internauts, connected vehicles), processing obtained records and communicating results to drivers. However, in order to help drivers identify appropriate parking spots at the beginning of their trip, availability prediction is the only practical way. It effectively taps into real-time data to produce short- or long-term forecasts anticipating available parking spaces.

1.1 Research questions

With our suggestion, we tried to tackle three questions: Does each predictive method require exactly the same feature set to achieve the best accuracy? If not, how does these attributes' contribution to get the outcome variable depend on the applied algorithm? What effect do their type (ensemble versus individual) and principle (trees, neurons or lines) have on the quality of the computed predictions? What elements render decisions made by a model more appropriate than those taken by another one?

1.2 Proposed solution

To answer clearly and precisely these matters, the parking occupancy rate is modeled depending on four different time-series feature sets. Our objective is to assess how the nature, number and precision of independent variables affect the performance of machine learning models. It is forecasted by four nonparametric algorithms belonging to two distinct categories in order to compare the efficiency of the ensemble methods recognized for their broad range of applications versus single ones characterized by their diversity. Furthermore, a simpler explanation model, based on SHAP values [90], is implemented to interpret outputs from the best-performing model, reveal the predominant features and clarify their individual contribution. This is possible because it allows investigating the decision-making process of Artificial intelligence algorithms.

1.3 Research contributions

Our major contribution is to parse the impact that the quality of insights, obtained from the features, has on the quality of predictions. We examined whether there is a dependency between the model type employed and the number of explanatory variables. We also tested if an excess of attributes can reduce the method's capabilities. We were the first construing results generated by a model-based parking occupancy prediction framework using SHAP values, to our best knowledge. We sought to assimilate the inner working of these algorithms perceived as black boxes difficult to explain. We focused on determining which factors most influence a parking lot's occupancy relative to its peers.

1.4 Outline

The reminder of this document is structured as follows: Sect. 2 summarizes many intelligent parking proposals published in the literature; Sect. 3 defines our overall system framework and lists its various components; Sect. 4 formalizes the problem, decrypts our prediction mechanism architecture and concisely presents the developed machine learning methods; Sect. 5 introduces the real database used, describes the occupancy trend of its car parks and analyzes the calculated error metrics values; Sect. 6 explains the permutation feature importance measurement and assesses explanatory variables' predictive power; Sect. 7 reviews in depth SHAP approach and outlines a comprehensive features interpretation; Sect. 8 briefly recaps our findings and discusses possible improvements.

2 Related works

Intelligent parking systems benefit fully from the technological development and therefore offer a variety of real-time parking space availability information. Parking solutions proposed in the literature can be categorized into two main types: "sensor-based" and "crowdsourcing-based" systems.

Most research works put forward intelligent parking platforms based on different types of sensor networks. The authors of [8] detected parking space occupancy using convolutional neural networks to directly process real-time pictures filmed by a Raspberry Pi camera. Adopting the same principle, the paper [9] developed a decentralized and efficient system for visual parking lot occupancy detection where each smart camera can simultaneously monitor up to fifty parking spaces. PGS is a parking guidance system based on wireless sensor network which guides a driver to an available parking lot [10]. Its architecture contains a Parking information server, T-Sensor nodes (magnetic sensor), T-Sink nodes and a TBS. The authors of [11] conceived a mobile sensing unit that is an ultrasonic sensor mounted on the passenger side of a car to measure the distance from the vehicle to the nearest roadside obstacle. They estimate roadside parking occupancy by a supervised learning algorithm which analyzes the structure of the sonar trace. A smart parking solution that makes use of a protected wireless network and sensor communication is proposed by Yan et al. [12]. It is designed as a continuous-time Markov chain, precisely as a birth–death stochastic process. It broadcasts new business promotions to all vehicles passing by the parking site through wireless networks. An integrated smart parking system is introduced in [13]. It brings multiple parking service providers together under a unified platform using blockchain technology. In [14], two different ways to manage availability information in parking facilities are evaluated: posting a warning on variable message sign panel to indicate that there are no free spaces when occupancy percentages are above 90% or 95% and zoning that places vehicle detection systems at intermediary points around the facility to separate it into internal zones.

Crowdsourcing is a model that consists to solve, in a distributed manner, a complicated problem by involving a crowd of indefinite size [15]. This practice proved its ability to help drivers find appropriate parking spaces through several successful experiences. ParkJam [16] is an Android application exploiting publicly accessible geographical data and car park availability information collected through crowdsourcing. PocketParker is a crowdsourcing system using smartphones to predict parking lot availability [17]. It detects arrivals and departures by leveraging existing activity recognition algorithms. UW-ParkAssist application [18] combines data collected passively using vector-based location mapping on users' phones with crowdsourced information actively supplied by users related to their observed density of parking. It offers real-time information about parking availability in UW-Parkside's campus lots. CroPark is a user-engaged crowdsourcing parking monitoring system that builds a parking occupancy map [19]. It employs ultrasonic sensors to measure the distance from the vehicle to roadside and uses a supervised learning algorithm to estimate the number of available parking spaces. Pick N Park application gathers and delivers real-time information about a number of car parks in crowded urban areas [20]. It is supplemented by a car park estimation model which provides a good approximation in terms of parking search time. CrowdPark [21] enables users to loosely reserve parking spots. It achieves parking reservation by crowdsourcing information about when parking resources will be available.

However, such collected real-time data is not practical enough as the target places may be occupied just before the drivers arrive. Consequently, they compete for the desired space and, out of obligation, behave violently by verbally arguing or parking in a forbidden space. With the aim to fill this gap, various articles model parking space occupancy by different approaches like Markov chain variants, regression methods and machine learning techniques (Table 1). The authors of [22] employed a discrete Markov chain model to demystify the future state of a parking lot, by the time a vehicle is expected to reach it. Similarly, the writers of [23] described the stochastic occupancy change of a parking facility by employing a continuous-time Markov queue. Such methods, however, can only be efficient for car parks equipped with access control. They also generate long-term forecasts with a high variance and are therefore reinforced by other methods. A hybrid approach combining agent-based with dynamic time varying Markov chain in [24] and with Markov chain Monte Carlo in [25] is proposed. In [26], a

Table 1 Intelligent parking solution for availability prediction

Reference	Algorithms	Data type	Inputs	Output	Database	Spatiotemporal analysis	Performance metrics	Performance achieved
Zheng et al. [35]	Regression tree	Historic	t and d	POR	San Francisco	No	MSE	0.000
	Support vector		Po		Melbourne		MAE	0.013
	Neural network		t, d and Po				R^2	0.986
Vlahogianni et al. [36]	Neural network	Historic	t	POR	Santander	No	MAE	0.004
							RMSE	0.006
	Weibull distribution			SP			MAPE	0.412
							RRSE	59.250
Li et al. [37]	LSTM network	Historic Latest data	t, w, T, h and Po	NAP	Beijing	No	RMSE	5.42
Errousso et al. [38]	Stacking with 2 levels	Historic	t, d and idA	NAP	Melbourne	No	RMSE	21.656
	Actuarial method			SP			MAE	2.166
							R^2	0.879
Stolfi et al. [39]	Polynomial fitting	Historic	t, d and idP	POR	Birmingham	No	MSE	–
	Fourier series							
	K-Means							
	KM-Polynomials							
	Shift & Phase							
	Time Series							
Camero et al. [40]	Recurrent Neural Network	Historic	d and idP	POR	Birmingham	No	MAE	0.067
Céderic et al. [41]	Bagging regressor	Historic	t, d and idP	POR	Birmingham	No	RMSE	0.001
	Random forest						MAE	0.000
	Adaboost regressor						R^2	0.999
	Gradient boosting							
Chirichigno et al. [42]	Linear Regression	Historic	t, d and idP	POR	Tandil	No	MSE	0.027
	Regression Trees	Google Maps					MAE	0.123
	Gradient Boost	G. Street View					R^2	0.577
Caicedo et al. [43]	Calibrated discrete choice model	Historic	–	NAP	Barcelona	No	AFE	0.12
		Real time					SDFE	0.24
Rajabioun et al. [44]	Probabilistic model	Historic Real time	t, d, l, Pc, Rl, Ro…	NAP	San Francisco	No	MNE	0.012
Rajabioun et al. [45]	Vector autoregressive model	Historic	t and idP	NAP	San Francisco	Yes	MAPE	0.14

Table 1 (continued)

Reference	Algorithms	Data type	Inputs	Output	Database	Spatiotemporal analysis	Performance metrics	Performance achieved
Shao et al. [46]	LSTM model	Historic	t, idA and Po	POR	Melbourne	Yes	MAE	0.018
	Temporal clustering						RMSE	0.022
	Nonlinear least square		t, d and POR	DT			MAPE	0.035
							RRSE	1.524
Chen et al. [47]	ARIMA	Historic	t, d, E, D and P	POR	San Francisco	Yes	MAPE	0.357
	Linear regression							
	Support vector							
	Neural network							
	K-means							

short-term prediction of parking space availability is suggested and relied on the wavelet neural network and the largest Lyapunov exponents method. The subsequent occupancies of parking spaces are forecasted, in [27], by the naïve Bayes, C5.0 decision tree, random forest and regression analysis founded on the spatial–temporal features extracted from parking data.

Some researchers seek, besides predicting parking availability, to reduce both the number of required models and training examples by studying the database's characteristics. The high temporal autocorrelation of the on-street parking dynamics is analyzed in [28] and non-Euclidean spatial autocorrelation among parking lots in [29]. For the first paper, recurring patterns in the collected data are exploited by means of clustering and training set reduction techniques. For the second one, global spatial dependencies between parking lots are captured by a contextual graph convolution block and a soft clustering graph convolution and incorporated by a recurrent neural network. Richter et al. [30] presented a back-end-based approach to learn historical models of parking availability per street by utilizing five spatiotemporal clustering strategies. Their goal is to significantly lower the space needed to store these models and it is shared with the authors of [31] who proposed a 2-step approach to predict parking space availability. In the first step, the raw parking data are smoothed by SVR in combination with a specifically defined technique to tune parameters. In the second, a multidimensional SVR model is trained on top of this smoothed trend curve.

Unlike previous research, ParkPGH is a smart parking application designed to supplement the predicted number of available parking spaces at a particular time by a binary dependent variable denoting whether the garage is full or not [32]. The authors in [33] investigated the extension of estimating parking information from areas equipped with sensors to areas that are missing them. They built a parking demand profile, which reflects the time of the day where parking occurs and its duration for a given area, by using complementary city data. Rong et al. [34] estimated the real-time parking availability throughout a city by proposing a deep-learning-based approach, called Du-Parking, that relies on the historical parking availability data and a variety of datasets (meteorology, events, map mobility trace data and navigation data).

According to the literature surveyed, there remain unexplored research topics in relation to parking availability prediction. There are only works dedicated to searching for or proposing very exact algorithms. Moreover, the interaction between a machine learning model's response and a feature combination is rarely approached. In this study, we experimented with several sets of inputs to find the optimal one for each model. We demonstrated that there is no optimal feature subset that works for every method. We aimed to build a general idea about how much input information is required by each model type to get the best performance. We deciphered, for the first time in this context, why certain predictions are made using Shapley additive explanation method. It provides an overview of possible improvements to a model as it clusters the data points depending on the feature importance.

3 Our system architecture

To predict parking occupancy, we adopted the intelligent parking system framework detailed in Fig. 1. It includes diverse data components and combines various information providers. As such, it aims to guarantee higher data precision, continual updating and diversified types of records. It comprises four layers: data collection, communication, middleware and user layer.

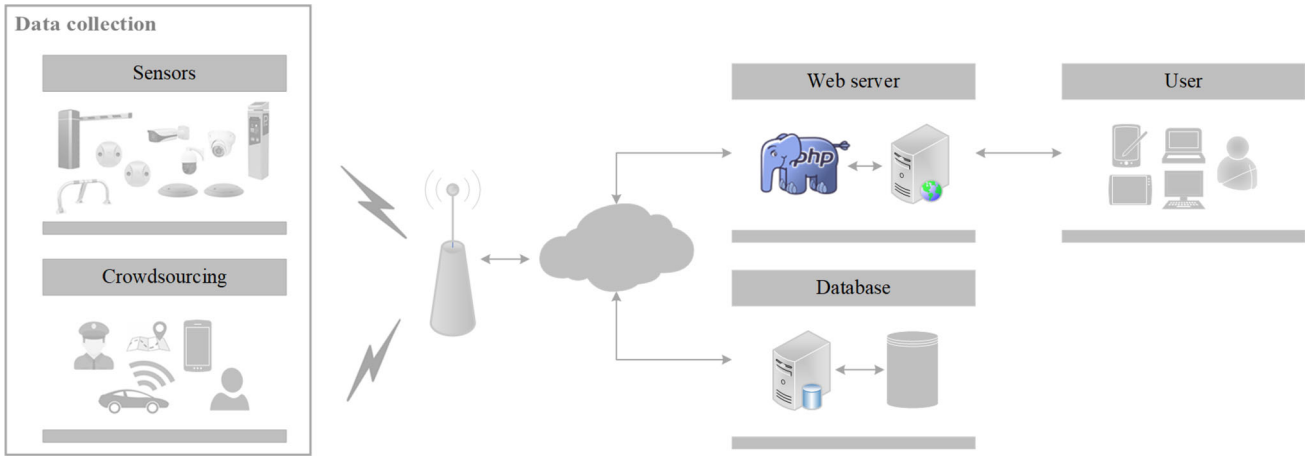

Fig. 1 Smart parking architecture

The first layer gathers information on parking occupancy using two different manners. It detects vehicle absence or presence thanks to several types of sensors installed at each parking space (magnetic, ultrasonic, infrared and acoustic) or cameras monitoring the entire car park. It determines the number of occupied spaces by means of access control barriers or parking meters. If a parking lot isn't equipped with these stationary sensors, it is possible to make use of mobile sensors as smart drivers, connected vehicles, smartphones and urban authorities. After parking or before moving off, drivers answer a question about parking availability based on their observations and with a manual entry on their device. Smartphones communicate to the server users' geographical position with the help of their geolocation system. These devices can also scan the road being traversed, identify vacant spots and distinguish parking places from other zones. Connected vehicles with all their numerous sensors allow to ascertain when they are running or stationary, to pinpoint unoccupied parking spaces and to recognize congested road stretches. They aid in defining street status by considering a road close to the driver's destination as occupied if it is crossed at low speed without finding a place to park. Urban authorities, in particular police forces and car park supervisors, may adjust the real-time records in the database according to their own data, e.g., patrol observations.

In communication layer, multiple communication protocols can be deployed in order to maintain message exchange between all system constituents: between sensors, from sensors to gateway and from gateway to web server as well as to database. All these entities interconnected or maintained in connection constitute a computer network. Its typology, as its topology, is chosen mainly according to the network size (number of machines), the required data transfer speed and its extent. Three computer network categories are perfectly adapted for this type of application, namely local area network (for parking lots spread over a few meters to a kilometer), a metropolitan area network (for car parks extended over a dozen kilometers) and a wide area network (for parking spaces dispersed over large geographical distances). Several physical configurations are considered to connect the network's different nodes, the relevant ones being the star, mesh, tree, ring, point-to-point and circular topologies.

Middleware layer is responsible for processing collected data of varied nature (coordinates, images, sentence, number, address, date…), filtering misleading detections and deducting consequently the parking availability in certain streets. It directs all the software intelligence embedded in the described parking solution utilizing machine learning algorithms, data mining models and powerful graphics techniques. It streamlines all captured data into a single database which is then hosted with the related servers.

The last layer concerns interaction with users and usage of all services provided via a mobile application or website. It displays online an interactive map showing the real-time occupancy of a city's parking spaces. It guides drivers to designated spots from their current position by sketching out the route to be taken or issuing voice instructions. It also lets users reserve one or more spaces while taking into account fees varying with location, time and parking duration. It proposes promotions to incite drivers for greater involvement depending on their participation rate in data collection.

4 Modeling parking availability

4.1 Methodology

Our real-time approach for forecasting parking space availability provides, as output, the occupancy rate calculated simply by dividing the number of vehicles parked in a car

Table 2 Correlation matrix

	Parking identifier	Year	Month	Day	Day of week	Weekday or weekend	Time	Occupancy rate
Parking identifier	1	0.007	0.000	1.970e–20	4.342e–20	– 1.024e–19	2.029e–20	0.94349
Year	0.007	1	1.843e–12	3.931e–18	2.731e–21	5.391e–07	– 1.043e–15	0.13284
Month	0.000	1.843e–12	1	– 2.626e–01	1.756e–02	1.261e–02	– 7.492e–03	0.61677
Day	1.970e–20	3.931e–18	– 2.626e–01	1	1.264e–02	– 2.856e–02	1.165e–02	– 0.51358
Day of week	4.342e–20	2.731e–21	1.756e–02	1.264e–02	1	7.915e–01	– 1.127e–03	0.78082
Weekday or weekend	– 1.024e–19	5.391e–07	1.261e–02	– 2.856e–02	7.915e–01	1	5.903e–03	– 0.40515
Time	2.029e–20	– 1.043e–15	– 7.492e–03	1.165e–02	– 1.127e–03	5.903e–03	1	0.81201
Occupancy rate	0.94349	0.13284	0.61677	– 0.51358	0.78082	– 0.40515	0.81201	1

park by its capacity. It considers, as explanatory variables for the algorithms, four sets containing different attributes:

- X0 = {Parking identifier, year, month, day, day of week, time}
- X1 = {Parking identifier, year, month, day, time}
- X2 = {Parking identifier, day of week, time}
- X3 = {Parking identifier, weekday or weekend, time}

Intuitively, these predictors represent the most influential factors in parking usage. But to demonstrate their statistical significance and to catch the pairwise degrees of relationship between each independent variable and the dependent variable, we elaborated the correlation matrix that refers to the symmetric array of Pearson correlation coefficients (Table 2). This coefficient is measured on a unitless scale and ranges from −1 to + 1 through 0 [48]. A negative sign in front of the coefficient means that the two variables vary inversely, and this is the case for Day and Weekday or weekend. As the coefficient is closer to 1 (in absolute value), the relationship between the variables is stronger. The variables Parking identifier, Day of week and Time explain, respectively, 94.349%, 78.082% and 81.201% of the variance of the occupancy rate variable.

Figure 2 depicts the computational architecture of our parking availability prediction methodology, which comprises four components:

- Data preparation which eliminates noise altering collected data, reduces their variability, divides the database into two separate sets (four-fifths for training against one-fifth for test) and partitions randomly the training set into 10 equally sized subsamples by means of k-fold cross-validation;
- Models training which runs, for every algorithm to be built, 10 iterations of training and validation so that in each iteration a different fold of the data is considered as the validation set while the remaining 9 folds make up the learning one;
- Model performance testing which generates predictions on the test set and measures the predictive accuracy of each machine learning algorithm by three error metrics that are R^2, MAE and RMSE;
- Model explanation which computes how much each predictor contributes, positively or negatively, to the target variable by plotting four different graphs (local interpretability, standard summary, summary and dependency plots).

4.2 Developed algorithms

To forecast the availability of car parks, four machine learning techniques are selected, namely multivariate adaptive regression splines (MARS), support vector regression (SVR), artificial neural network (ANN) and extremely randomized trees (ERT). These algorithms are chosen primarily because they are judged as better in the literature [35,38,41,47,52], their internal principles are largely dissimilar, their parameters are significantly different and their learning samples are noticeably distinct.

SVR is a nonlinear kernel-based regression method which locates a regression hyperplane with smallest structural risk in a so-called high dimensional feature space [49,50]. Given

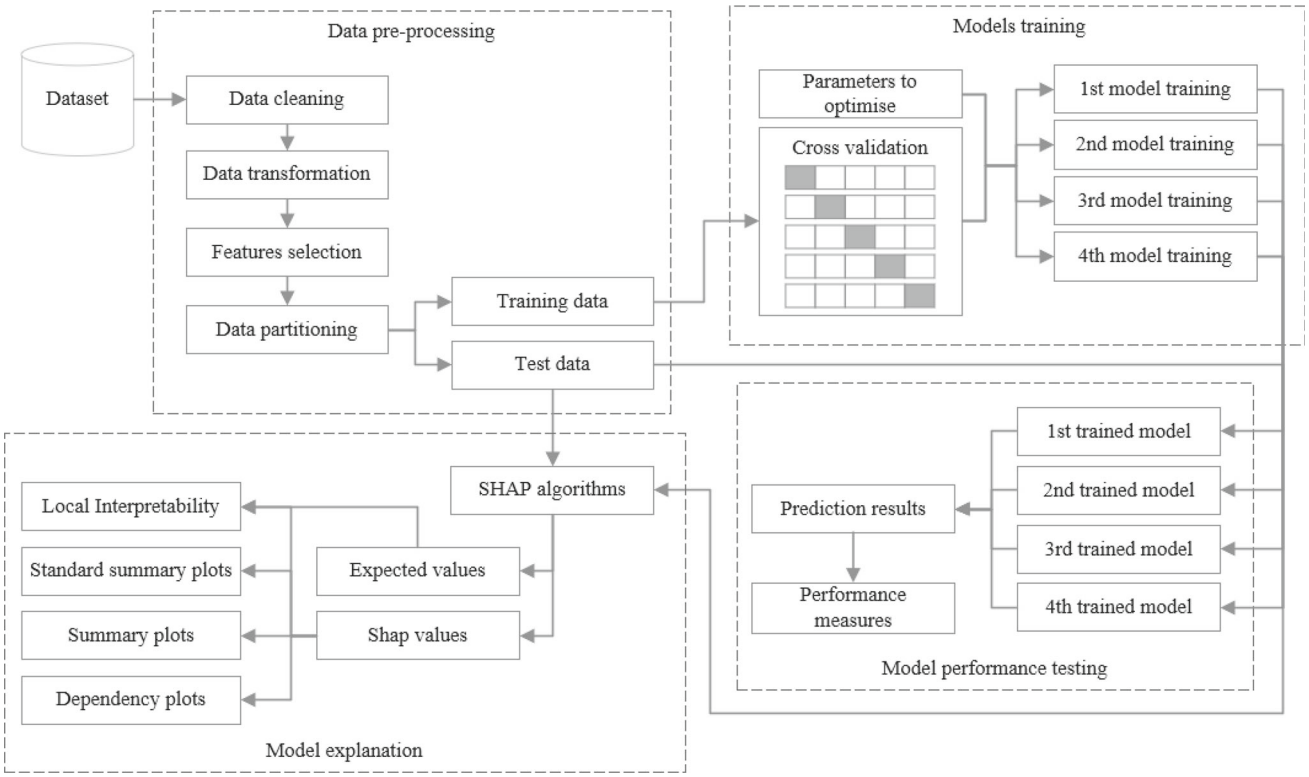

Fig. 2 The framework of our proposed parking availability prediction system

a set of training data $\{(x_1, y_1), \ldots, (x_p, y_p)\}$, where $x_i \subset \mathbb{R}^n$, $i = 1, \ldots, p$, denotes the input vector and $y_i \subset \mathbb{R}$, $i = 1, \ldots, p$, designates the corresponding target value, the SVR estimating function takes the following form:

$$f(x) = \sum_{i=1}^{p} (\alpha_i - \alpha_i^*) \langle \phi(x_i) \bullet \phi(x) \rangle + b = \sum_{i=1}^{p} (\alpha_i - \alpha_i^*) k(x_i, x) + b \quad (1)$$

where $b \subset \mathbb{R}$ is an offset, $k(x_i, x)$ is a kernel function which represents the inner product $\langle \phi(x_i) \bullet \phi(x) \rangle$, α_i and α_i^* are nonzero Lagrange multipliers. The most commonly employed kernel function is the radial basis function (RBF) defined, in [51], as:

$$k(x_i, x) = \exp\left(-\gamma \|x_i - x\|^2\right) \quad (2)$$

where γ is the width parameter of the RBF kernel function and chosen thanks to heuristics.

ANN is a computational model inspired by the organization and functioning of biological neurons primary to the parallel nature of the human brain [54,55]. The multi-layered perceptron, which is composed of one input layer, one or multiple hidden layers and one output layer, is its widely used variant [56]. Each layer comprises several neurons and each neuron i in a layer sends outgoing signals to every node j in the next layer. Neuron j gets an effective signal S_j resulting from the weighted sum of all received signals [57].

$$S_j = \sum_{i=0}^{p} w_{ji} x_i \quad (3)$$

where x_0 is the bias and w_{j0} is the bias weights. The output y_j of neuron j in the hidden layer is set by passing the signal S_j through an activation function that aims, furthermore to introduce nonlinearity into the neural network, to bound the value of the neuron so that the neural network is not paralyzed by divergent neurons [58]. The rectified linear unit function (Relu) is the most often utilized activation function and is given by the formula below.

$$y_j = f(S_j) = \max(0, S_j). \quad (4)$$

MARS is a nonparametric regression technique that Friedman developed in [59]. It models relationships implying interactions with less variables and those being relatively additive. It performs piecewise linear regressions for constructing basically flexible models [60]. Consequently, it fits distinct linear regression slopes within separate feature space intervals in order to estimate a model's nonlinearity. It operates a quick- but cost-consuming search process in identifying which variables to employ and the upper limits of their

intervals [61]. It further looks at the interactions between the variables to take their degrees into account [62]. The equation hereafter stands for the general MARS function.

$$f(x) = a_0 + \sum_{m=1}^{M} a_m \prod_{k=1}^{K_m} \left[s_{km} \left(x_{v(k,m)} - t_{km} \right) \right]_+ \tag{5}$$

where a_0 and a_m are parameters, M is the basis functions number, K_m is the nodes number, s_{km} specifies the corresponding step function sense by taking 1 for right and -1 for left, $x_{v(k,m)}$ is the dependent variable and t_{km} states the nodes location.

MARS builds the optimal model in two steps. Firstly, several basis functions are designed to overfit the data. Secondly, they are selected in order of highest contribution by means of the generalized cross-validation (GCV) criterion expressed as follows [63]. It addresses missing value problems with dummy variable capabilities.

$$\mathrm{LOF}(f_M) = \mathrm{GCV}(M) = \frac{1}{p} \sum_{i=1}^{p} \frac{[y_i - f_M(x_i)]^2}{[1 - C(M)/p]^2} \tag{6}$$

where $C(M)$ is the cost-penalty measures of a model containing M basis functions.

ERT is a tree-based ensemble method that creates a set of unpruned regression trees according to the classical top-down procedure [64]. It splits nodes by choosing cut-points fully at random rather than computing local optimal ones and grows the trees with the whole learning sample instead of a bootstrap replica [65]. It is parameterized regarding how many trees to build, their depth, number of features picked at random for every node and smallest sample size required to divide a node [66]. The score in ERT is measured by the relative variance reduction. For a sample S and a split s, it is calculated by the formula (7).

$$\mathrm{Score}(S, s) = \frac{\mathrm{var}\{y|S\} - \frac{|S_l|}{|S|}\mathrm{var}\{y|S_l\} - \frac{|S_r|}{|S|}\mathrm{var}\{y|S_r\}}{\mathrm{var}\{y|S\}} \tag{7}$$

where $\mathrm{var}\{y|S\}$ is the variance of the output y in the sample S, S_l and S_r represent the two subsets of cases from S corresponding to the two outcomes of a split s.

It compensates the randomization with a forest of trees and computes the average of the constituent tree outputs [67].

4.3 Hyperparameters of exploited algorithms

Table 3 summarizes the most relevant hyperparameters of the four algorithms that should be tuned to comparatively analyze their performance. It outlines their respective meanings and specifies the values used for the dataset implementation. Model parameters are those parameters updated during the data learning process, unlike hyperparameters that determine model architecture, are defined before training [4].

Hyperparameters demonstrate their interest by improving model performance, e.g., complexity or learning rate. Each model is characterized by a large number of hyperparameters but looking for the best combination leading to the best performance can be tackled as a search problem.

The best parameters for the four algorithms described above are obtained using the GridSearchCV library developed by Sklearn. This library is designed to identify the optimal parameters for any type of machine learning algorithm. It takes a dictionary of parameters to fine-tune and finds the best-performing ones according to the evaluation measure set at the computation time. Furthermore, in doing so, it leverages the k-fold cross-validation technique that divides a data set into K equal sets. Among these K sets, each set is utilized once as test data and the other sets are employed as training data. This approach determines the single most effective hyperparameter combination considering the $K(K-1)$ training sets and K test samples.

To enhance SVR model accuracy, several parameters need to be tuned. Three major ones are: Kernel, C and Gamma. The main role of the kernel is to transform a low dimensional input space into a higher dimensional space, it is always set to 'rbf' kernel. The penalty parameter C, or regularization, stands for the misclassification or error term. Gamma, the kernel coefficient, defines to what extent the calculation of the plausible separation line is influenced. For RBF kernel, the typical values of these parameters are ranging from 0.001 to 1000. The results of a tenfold cross-validation are illustrated in the figure above.

Figure 3 clearly shows that the lowest value of MAE is obtained with the combination of 100 and 1.0 for the parameters C and gamma, respectively: {'C': 100, 'gamma': 1}.

learning_rate, max_iter and hidden_layers_sizes are three hyperparameters that determine our MLP model convergence speed. They are optimized utilizing GridSearchCV with activation function set to "Relu" on account of its common use, implementation simplicity and effectiveness in overcoming limitations. Optimization will be run considering two learning rates (constant and adaptive), three values for max_iter (100, 200 and 400) and three different combinations of hidden_layers_sizes ((120,), (150,) and (200,)). These configurations are iterated with a tenfold cross-validation and evaluated by means of the mean absolute error score. Figure 4 represents all three plots for the three different values of the hidden_layers_sizes parameter.

Visualized results indicate that the best parameter combination leading to the best score consists of: {'hidden_layer_sizes': (150,), 'learning_rate': 'constant', 'max_iter': 200}.

Table 3 Hyperparameters of our four machine learning techniques

Algorithm	Parameter	Definition	Value
SVR	Kernel	Kernel type to be utilized	rbf
	Gamma	Kernel coefficient	1
	Tol	Tolerance for stopping criterion	1e-3
	c	Regularization parameter	100
ANN	Hidden_layer_sizes	Number of neurons in the hidden layer	150
	Activation	Activation function for the hidden layer	Relu
	Learning_rate	Learning rate schedule for weight updates	Constant
	Max_iter	Maximum number of iterations	200
	Batch_size	Size of minibatches for stochastic optimizers	Auto
	Solver	Solver for weight optimization	Adam
	Alpha	L2 penalty	0.0001
MARS	Max_terms	Maximum number of terms generated by the forward pass	$\min(2n + m//10, 400)$*
	Max_degree	Maximum degree of terms generated by the forward pass	1
	Penalty	Smoothing parameter	5
	Endspan	Number of extreme data values for each feature not eligible as knot locations	−1
	thresh	Parameter used when evaluating stopping conditions for the forward pass	0.001
ERT	N_estimators	Number of trees in the forest	50
	Min_samples_leaf	Minimum number of samples required to be at a leaf node	1
	Min_samples_split	Minimum number of samples required to split an internal node	2
	N_jobs	Number of jobs to run in parallel	1
	Max_depth	Maximum depth of the tree	None
	Criterion	Function to measure the quality of a split	Gini
	Max_features	The number of features to consider when looking for the best split	5

*Where n is the number of features and m is the number of rows.

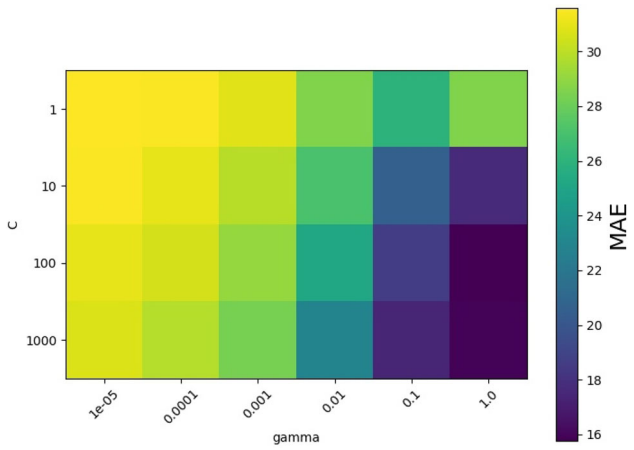

Fig. 3 Optimizing parameters of SVR

MARS as well has some parameters to tune, four of which are relevant: max_terms, max_degree, penalty and endspan. To avoid overfitting our model, only three of them are employed. The best-performing combination of the parameters is determined by exploiting once again the GridSearchCV optimizer and testing several values, from 1 to 4 for max_degree as penalty and ranging from 5 to 150 for max_terms. The results of a tenfold cross-validation are depicted in the following graphs (Fig. 5).

Recommended parameters for predicting parking availability with MARS are: {'max_degree': 1, 'max_terms': 4, 'penalty': 5}.

The main parameters to adjust when applying ERT method are n_estimators and max_features. The former is the number of trees in the forest. The larger the better, but also the

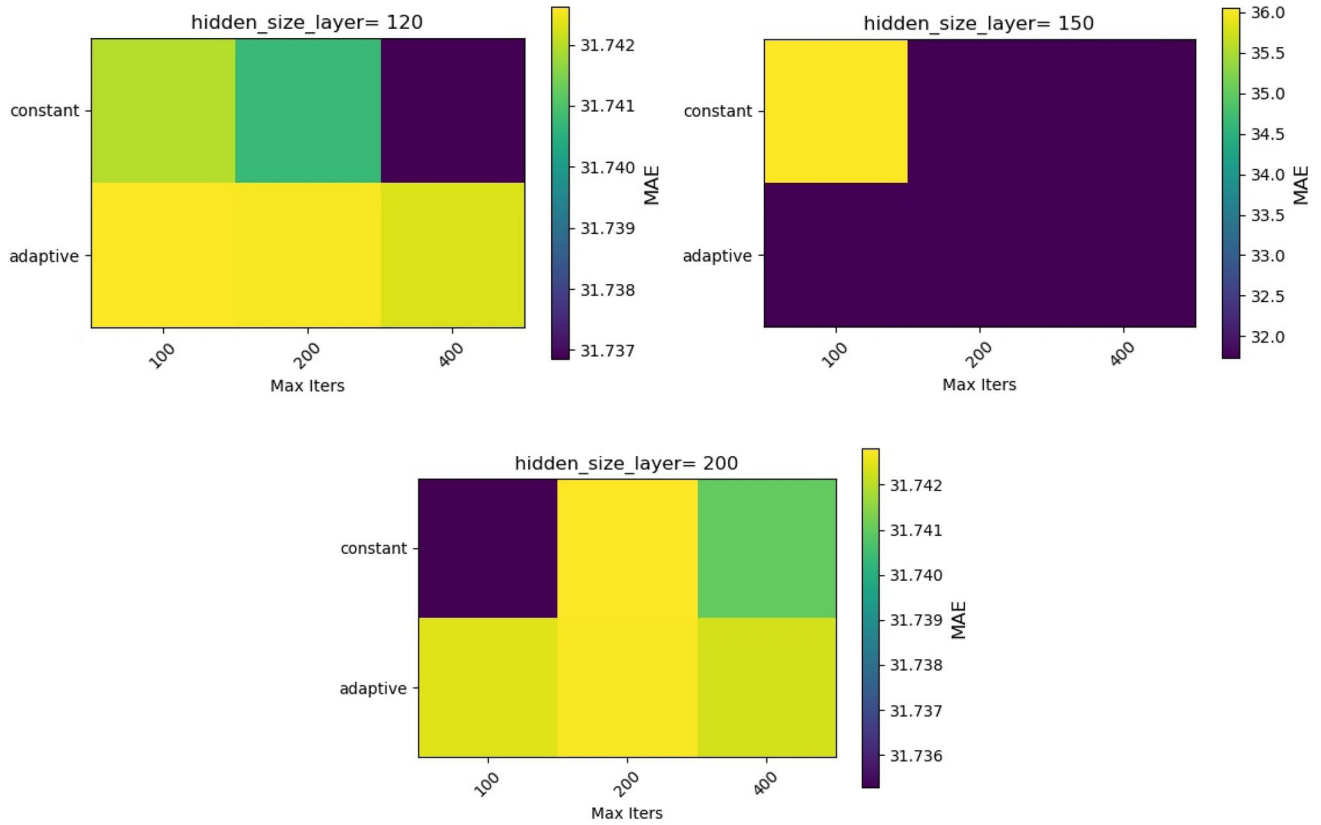

Fig. 4 Optimizing parameters of ANN

longer it will take to compute. The latter is the size of the feature random subsets to consider when splitting a node. The smaller the size, the greater the variance reduction, but also the greater the increase in bias. Good default empirical values range from 1 to $\sqrt{n\_features}$ for max_features and from 5 to 1000 for n_estimators. These values are generally not optimal and can result in models that require significant memory resources. The best parameter values would be cross-validated with the help of GridSearchCV, the findings of which are displayed in Fig. 6.

Given this graph, small values for max_features and n_estimators do not fit best, validating our assumed hypothesis: {'max_features': 5, 'n_estimators': 50}.

5 Implementation and results

The suggested framework is deployed with python 3 using JupyterLab web-based interactive environment. It is evaluated by leveraging historical parking information in the city of Aarhus, Denmark.

This interpretive language is very cross-cutting, with multiple modules covering a wide range of domains. These modules are Python programs that include functions to carry out an extraordinary number of tasks. To implement the selected machine learning algorithms, three libraries are exploited:

- Scikit-learn: A state-of-the-art tool for artificial intelligence problems incorporating large set of machine learning algorithms for supervised and unsupervised medium-scale problems [91,92]. It employs a high-level generic language with minimal dependencies. It features a clean, uniform and streamlined API while benefiting from helpful online documentation. Here, predictive models developed by SVM algorithm are implemented with the sklearn.svm module for regression (SVR). The MLPRegressor class implements a multi-layer perceptron (a type of artificial neural network) algorithm that is trained using backpropagation. The sklearn.ensemble module which contains ensemble-based learning methods addressing classification, regression and anomaly detection is utilized to deploy the ERT method. Sklearn's "pyearth" library is employed to roll out the MARS method. In addition, our models' performance measurements are performed by the sklearn.metrics module.
- Pandas: open-source data processing and analysis tool, widely chosen for its power, flexibility, practicality and especially for its speed [53].

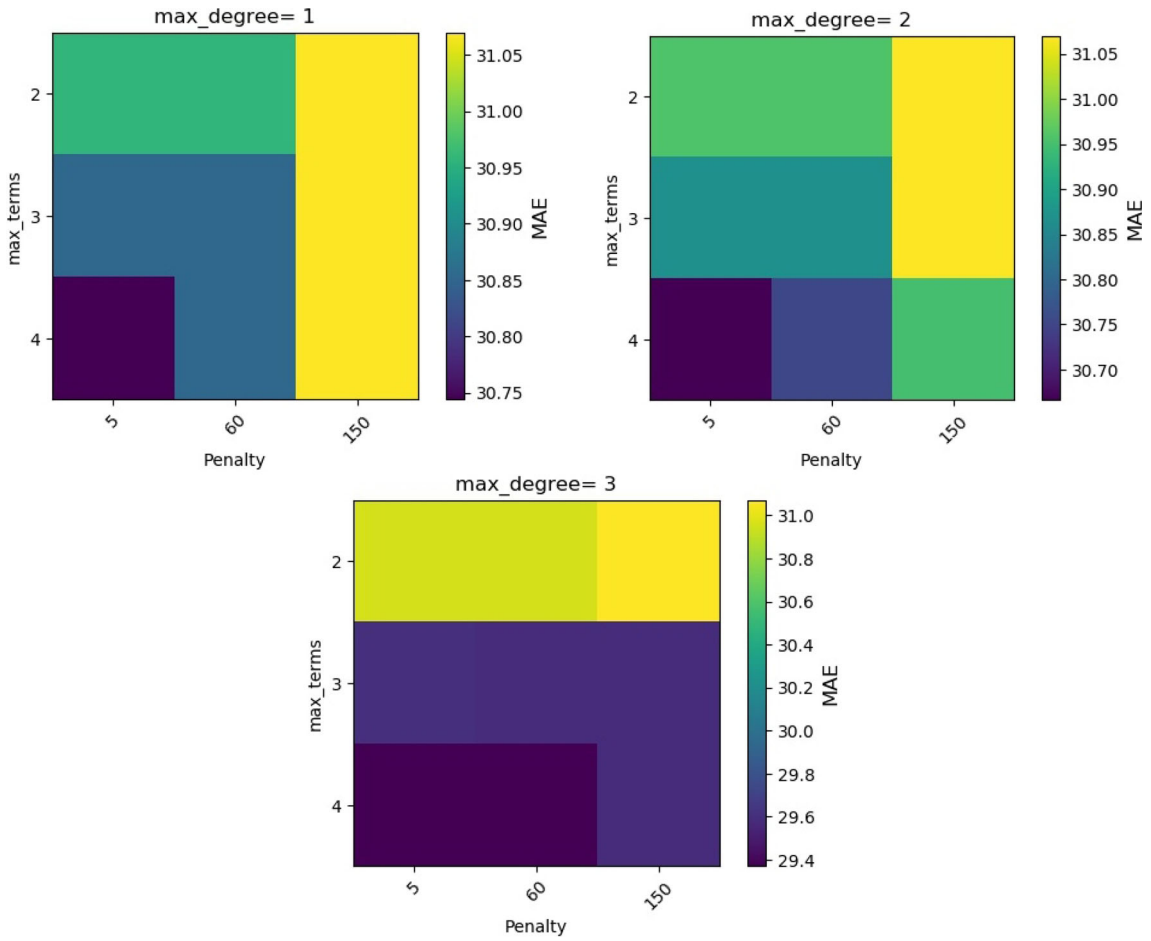

Fig. 5 Optimizing parameters of MARS

- SciPy: open-source library coded in Python principally dedicated to scientific or technical computing [93]. It provides algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations…

Source code and datafile can be downloaded from https://github.com/hanaeers/aarhus_parking.

5.1 Experimental dataset: Aarhus parking data stream

The city of Aarhus supplied a data stream with parking data for eight parking lots over a period of five months from May 22, 2014, to November 4, 2014. These parking lots are equipped with sensors and able to indicate the number of vacant spaces [68]. Data features comprise Garage code, total spaces, vehicle count, update time and stream time (Table 4).

In total, the database contains 55 264 data points and is available from [69]. Before developing the selected machine learning models, we treated the data to restructure values,

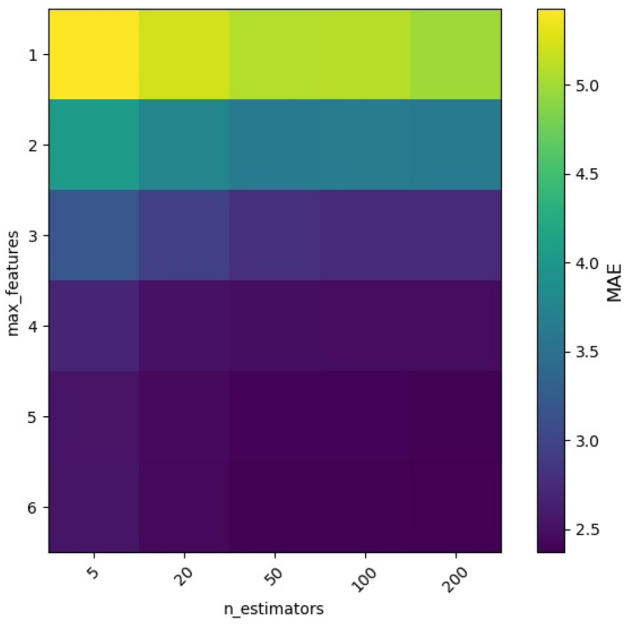

Fig. 6 Optimizing parameters of ERT

Table 4 Database attribute description

	Attribute	Description	Type
1	Garage code	The name and the geographical position of a car park	String
2	Total spaces	The total number of parking spaces in a car park (its maximum capacity)	Numerical
3	Vehicle count	The number of spaces occupied in a parking lot at a given time	Numerical
4	Update time	The time and date of the data update carried out twice an hour throughout the day	Datetime
5	Stream time	The time and date of the parking data release	Datetime

complete any missing ones and eliminate incoherent lines that distort the findings. For this purpose, we conducted multiple corrective operations:

- Delete rows with negative numbers of occupied parking spaces, remove useless columns (features) and aggregate data per parking lot;
- Attribute a distinct integer number, between 1 and 8, to every car park name;
- Get the day, month, year and time from the variable "Update time";
- Retrieve the day of week from the same variable and allocate an integer from 1 (if Monday) to 7 (if Sunday) for each day;
- Insert a column for a binary variable that indicates whether the day of week is a weekday or a weekend;
- Calculate the percentage of spaces occupied by dividing the number of vehicles parked in a parking lot by its capacity and multiplying this ratio by 100;
- Set 100% as the upper limit for parking lot occupancy rates that exceed this value;
- Replicate, if data is missing for a certain hour, the previous hour's value if not the previous week's value (same day of preceding week).

After data pre-processing and transformation, our database contained 11 053 records relating to 8 car parks covering 167 days. Figure 7 illustrates parking occupancy distribution over the days of a month as a line plot. We can clearly notice that the occupancy rate varies greatly during the month, is quite similar for all the car parks and peaks on days 3, 14 and 19. The first few days of the month are characterized by a high occupancy of the parking spaces in contrast to its end.

The bar chart (Fig. 8) shows the monthly occupancy distribution of the 8 parking lots in Aarhus. The car park occupancy

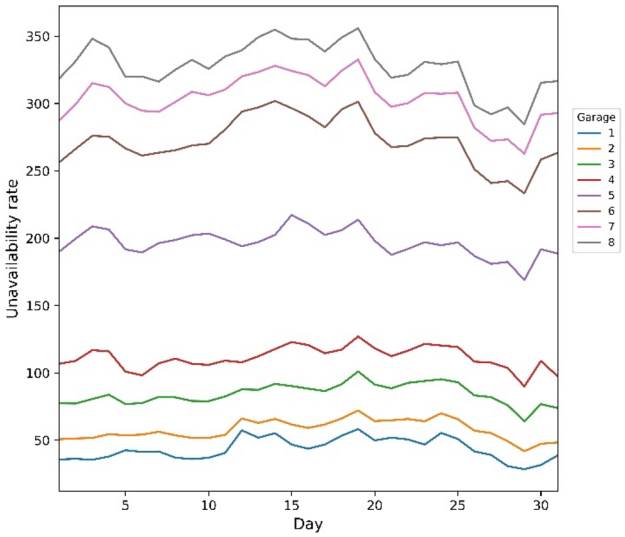

Fig. 7 Daily distribution of parking occupancy

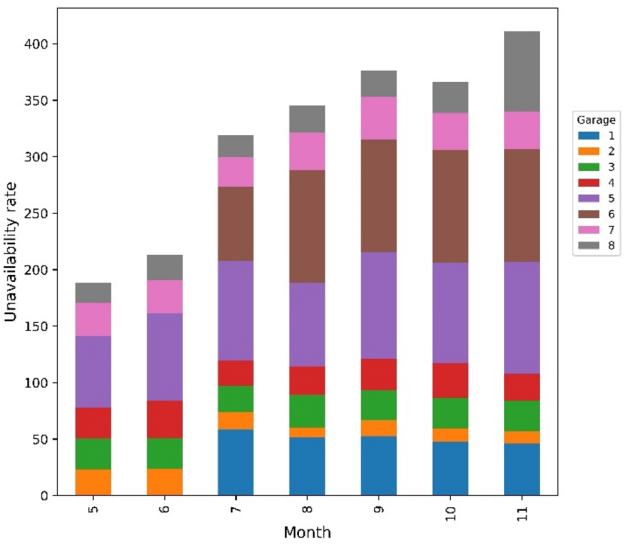

Fig. 8 Monthly availability of parking lot

differs from month to month. Parking lot 8 is more occupied in month 11 than in month 5, its availability is high and at least stationary. Parking lot 5 is less available in month 8 compared to the other months and its occupancy fluctuates by a margin of 50%. Parking lots 1 and 6 are fully available in months of May and June, therefore their unavailability rate in this period is zero.

In general, the occupancy rate of parking lots is higher on weekdays rather than weekends (Fig. 9). It is significantly dissimilar from one parking lot to another, e.g., parking lot 2 is distinguished by a low occupancy rate while parking lot 5 has a high percentage. Parking lots 3, 4, 7 and 8 are filled to within 25% on weekends.

The line chart (Fig. 10) hereunder presents the parking

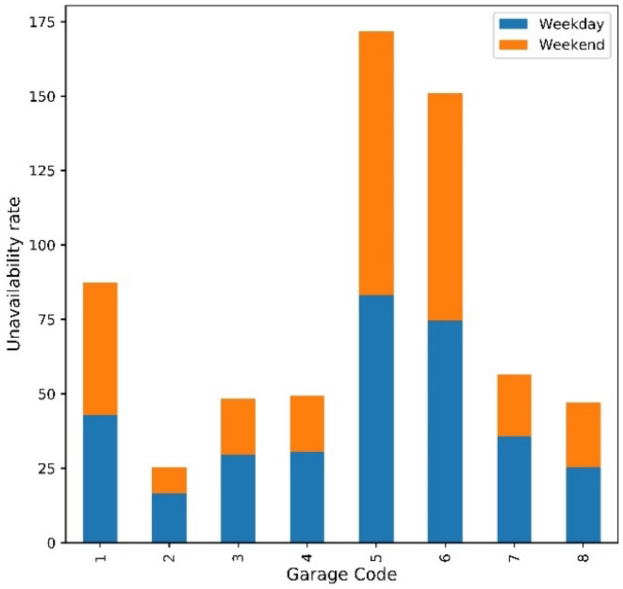

Fig. 9 Availability of car parks on weekends and weekdays

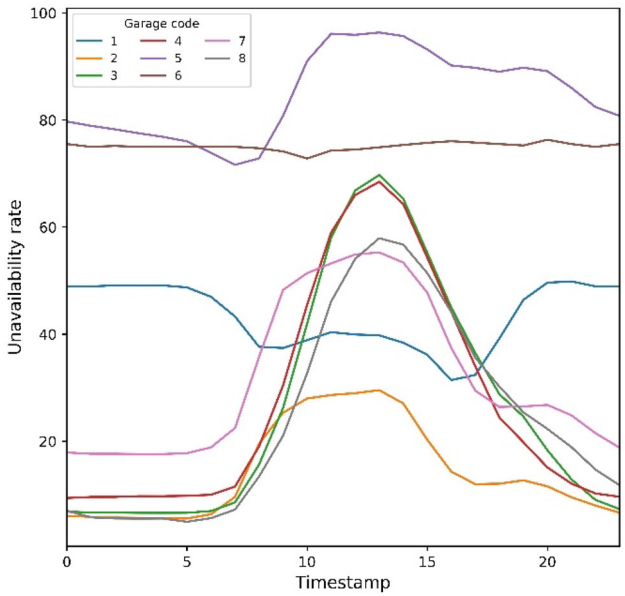

Fig. 10 Availability data after processing

lot occupancy rate by time of day. Occupancy of most parking lots is subject to a major peak at 1 p.m. when demand increases. The parking supply in car park 1 is typically stable with an occupancy rate around 50%, except for a drop to 20% between 5am and 8 pm. The variation in parking availability over the day is practically significant for parking lot 5: parking demand decreases slightly in the morning, increases rapidly at noon and diminishes slowly in the evening. In contrast, the occupancy rate of parking lot 6 is less dispersed, i.e., it is almost constant.

5.2 Experimental results

To evaluate the efficiency of the developed models, the accuracy and precision of their predictions, three error metrics are calculated: root mean square error (RMSE), mean absolute error (MAE) and coefficient of determination (R^2).

RMSE represents the residual variance square root. This latter is calculated by averaging the squared deviations between predicted values ($\widehat{y}$) and observed ones (y) [70]. It evaluates all ratings inexactness, either positive or negative [71]. The lower this metric's values are, the better the prediction model performs.

$$RMSE = \sqrt{\frac{1}{p}\sum_{i=1}^{p}\|y_i - \widehat{y}_i\|}. \tag{8}$$

MAE measures the closeness of the prediction to the eventual outcomes [70]. It denotes the ratio of the error vector $y_i - \widetilde{y}_i$ one norm to the number of samples p (9) [72]. The deviations between the model and the observations increase with the values of this metric.

$$MAE = \frac{1}{p}\sum_{i=1}^{p}|y_i - \widehat{y}_i|. \tag{9}$$

R^2 quantifies the proportion of variance explained by a model [73]. It examines the goodness of the technique's fit by its ability to predict the response variable [74]. This coefficient is between 0 and 1 and increases with the regression adequacy to the model. A good fit is indicated by an R^2 close to 1. Conversely, an R^2 close to 0 reflects a poor fit, but does not imply that no relationship can be established between the variables.

$$R^2 = 1 - \frac{\sum_{i=1}^{p}(y_i - \widehat{y}_i)^2}{\sum_{i=1}^{p}\left(y_i - \overline{y}_i\right)^2} \tag{10}$$

where $\overline{y}_i$ is mean values.

By analyzing the values of these indicators (Table 5), it is clear that MARS is the least performing model with a R^2 less than or equal to 0.061, a MAE greater than or equal to 30.31 and a RMSE larger than or equal to 1203.624 for all four input sets. ERT provides the best results with a R^2 that exceeds 0.98, a MAE that reaches 2.021 and a RMSE that attains 15.756. ANN and SVR come in second rank by a R^2 of 0.55, a MAE of 13.49 and a RMSE of 571.49.

From R^2, ERT based on feature set X0 is the most exact method scoring a coefficient of 0.988, followed by ANN relying on feature set X3 and SVR founded on feature set X2 with performance decreases of 43.927% and 45.242%, respectively. Considering RMSE, ERT using input set X0 performs best and surpasses ANN employing input set X3

Table 5 Metrics values measuring model performance

Regressors	Input set	Error metrics		
		RMSE	MAE	R^2
ANN	X0	870.294	24.632	0.321
	X1	929.604	24.174	0.275
	X2	584.377	17.257	0.544
	X3	571.493	16.489	0.554
SVR	X0	770.569	19.407	0.399
	X1	616.470	18.110	0.519
	X2	587.663	14.488	0.541
	X3	619.345	13.495	0.517
ERT	X0	15.756	2.021	0.988
	X1	22.815	2.323	0.982
	X2	518.579	14.164	0.595
	X3	519.657	14.580	0.594
MARS	X0	1203.624	30.310	0.061
	X1	1206.898	30.419	0.058
	X2	1249.953	31.166	0.025
	X3	1246.830	31.085	0.027

by about 3527% and SVR utilizing input set X2 by around 3629%. In terms of MAE, ERT with explanatory variable set X0 is the most precise algorithm that is 567% better than SVR with explanatory variable set X3 and 715% better than ANN with explanatory variable set X3.

Predictions are more accurate if the model is developed with feature set X0 containing more details apart from SVR which generalizes better on unseen data with feature set X2 and ANN which gets the best performance with feature set X3. For input sets X2 and X3, the three models (ERT, SVR and ANN) give more or less the same accuracy of forecasts with a R^2 that varies from 0.517 to 0.595 (fluctuation band of 27.876%), a MAE that ranges from 13.495 to 17.257 (variation margin of 13.109%) and a RMSE that goes from 518.579 to 619.345 (percentage difference of 19.431%).

Concretely, ERT with all four data sets outperforms the other regressors with any input set. Figure 11 graphically depicts predictions made by ERT versus observed values. The expected curve is a straight line through the origin frame. Using feature sets X0 and X1, ERT yields very interesting outcomes with a R^2 near to 1 (≈ 0.99) and a MAE not more than 2.33%. Its efficiency increases with the accuracy, number and information quality brought by the independent variables, unlike ANN which require as little data as possible to achieve its best performance. SVR is a special case that differs from the two previous ones by necessitating moderately complete information to deliver its maximum performance. The level of detail and accuracy of the information that the independent variables provide does not have a direct effect on models' performance, which depends mainly on their tendency to model one particular distribution rather than another.

The above results suggest that some algorithms perform better than others based on the different metrics values obtained. This means that the algorithm with the best average performance should outperform those with the worst. But to what extent is this difference true and not just the consequence of a statistical fluke? To test the significance of the average score difference, a statistical analysis is carried out, specifically the 5×2 cv paired Student's t test.

This statistical hypothesis test, proposed by Dietterich [97], consists in comparing two models (classifiers or regressors). It relies on splitting the database in 2 (50% of the data for training and 50% of the data for testing) five times. In each of these five iterations, it fits two regressors R1 and R2 to the training set and evaluates their performance on the test one. Then, it alternates the training and test sets such that the training dataset becomes the test dataset and vice versa. Finally, it calculates anew each model's performance, resulting in two performance difference measures:

$$ACC^{(1)} = ACC^{(1)}_{R1} - ACC^{(1)}_{R2} \tag{11}$$

$$ACC^{(2)} = ACC^{(2)}_{R1} - ACC^{(2)}_{R2}. \tag{12}$$

It ends by estimating the mean and variance of the two differences:

$$\overline{ACC} = \frac{ACC^{(1)} + ACC^{(2)}}{2} \tag{13}$$

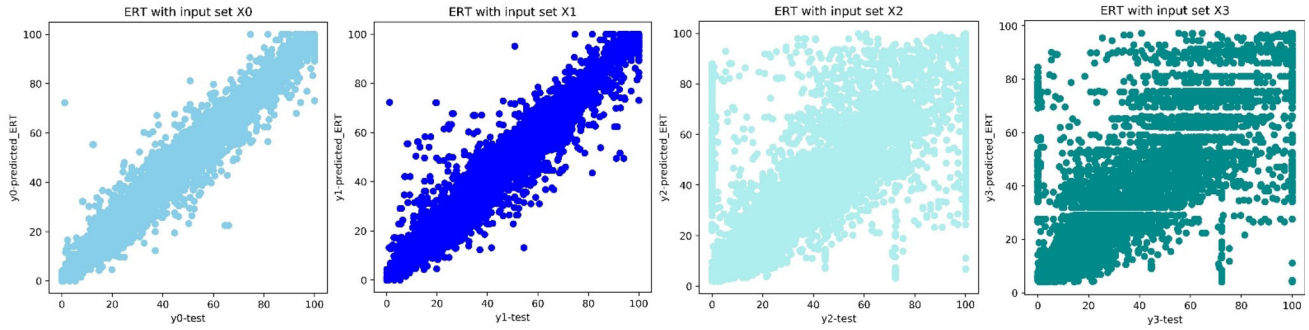

Fig. 11 Comparative plot between ERT predictions for each input set and real values

Table 6 Results of the paired *t* test for X0

	MARS		SVR		ANN		ERT	
	p value	*t* statistic	*p* value	*t* statistic	*p* value	*t* statistic	*p* value	*t* statistic
MARS			0.00000008	30.660	0.08093	2.182	0.0000052	229.853
SVR					0.788	−0.283	0.000003	−91.802
ANN							0.00055	−7.825

Table 7 Results of the paired *t* test for X3

	MARS		SVR		ANN		ERT	
	p value	*t* statistic	*p* value	*t* statistic	*p* value	*t* statistic	*p* value	*t* statistic
MARS			0.00000066	57.066	0.00003	14.246	0.0000097	107.537
SVR					0.804	−0.262	0.00008	−11.683
ANN							0.10955	−1.944

$$\sigma^2 = \left(ACC^{(1)} - \overline{ACC}\right)^2 + \left(ACC^{(2)} - \overline{ACC}\right)^2. \quad (14)$$

The t statistic is computed using the variances of the two differences calculated for the five iterations.

$$t = ACC_1^{(1)} \sqrt{\frac{5}{\sum_{i=1}^{5} \sigma_i^2}} \quad (15)$$

where $ACC_1^{(1)}$ is the $ACC^{(1)}$ obtained from the first iteration.

The t statistic is assumed to approximately follow a t-distribution with 5 degrees of freedom, under the null hypothesis postulating that the two models R1 and R2 have equal performance. Given the t statistic, the p value can be estimated and compared to a previously chosen significance level, in our case $\alpha = 0.05$. If the p value is less than α, the null hypothesis is rejected and thus a significant difference exists between the two models.

Our four algorithms are compared 2 by 2 with two explanatory variable sets X0 and X3. X1 and X2 are not used because they are included in X0. The results are summarized in Tables 6 and 7. The *p* values when comparing MARS with any other algorithm are well below 0.05, which means that any discernible difference between these algorithms is most likely real. It is therefore possible to confirm that MARS is the least efficient model among the four tested. In contrast, ANN and SVR trained with either X0 or X3 result in a p value significantly greater than 0.05. Consequently, it is impossible to reject the null hypothesis. It could just as easily apply SVR or ANN, and the results would be similar. In addition, the difference between mean performance of ERT and remaining algorithms is most likely significant since the p values are less than 0.05. An obvious result is that ERT effectively outperforms the rest of the three models. An exception occurs if ERT is tested against ANN using feature set X3, where the p value informs that for this variable set, the two algorithms perform alike. This finding may be related to the algorithm stochastic nature, assessment methodology or differences in numerical accuracy. In general, the average performances computed with the three error metrics are confirmed by the statistical test executed. This shows that model selection based on mean performance alone may be sufficient in some cases.

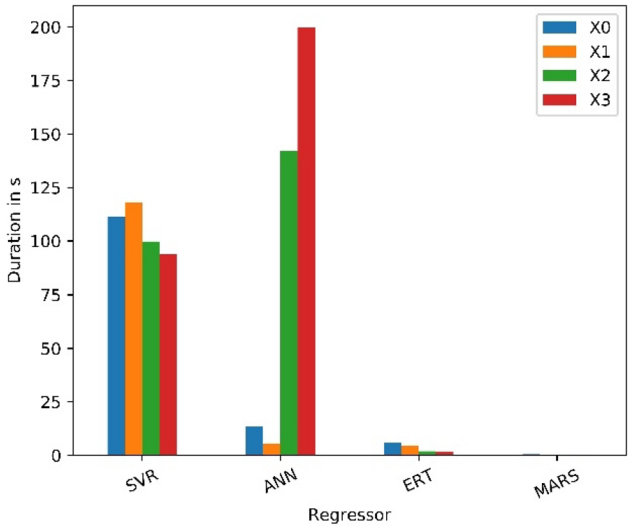

Fig. 12 Training time for each machine learning algorithm

Figure 12 displays the training time of each regressor for each feature set. At first glance, it is remarkable that these times belong to an interval of great length whose upper limit is 199.513 s and the lower limit is 0.449 s. ERT and MARS are characterized by very low computation times that do not exceed 5 s for the first algorithm and 1 s for the second one. Training SVR employing any input set takes 104.901 s on average with a standard deviation of 10.527 s, which is not surprising considering that all four models consistently generate estimates of roughly identical quality. ANN's training durations are highly dispersed, for example using input set X3 it needs about 200 s while with input set X1 it requires a little more than 5 s.

5.3 Cross-validation impact on prediction accuracy

Before analyzing in detail how cross-validation impacts our models' efficiency, it was necessary to start by examining the effect of fold number on their overfitting, consistency and reliability. This procedure was applied to our database with several numbers of folds, namely 3, 5, 7 and 10. Figure 13 presents the RMSE metric derived from the four models with the different cross-validations. For SVR, RMSE decreases by increasing the number of cross-validation folds. Therefore, the best results in terms of prediction accuracy are obtained with the tenfold cross-validation. In contrast, RMSE values related to MARS models increase with the number of folds. Accordingly, the best performance of MARS is achieved by dividing the data into 3 equal folds. ANN is characterized by a non-monotonic RMSE evolution. This indicator reaches its maximum value when the number of folds equals 5 and its minimum value when this number equals 7. As for ERT, RMSE trend depends on the set of explanatory variables used. For X2 and X3, RMSE increases rapidly and becomes constant from 7 folds. RMSE of ERT models based on X0 and X1 first increases, then decreases and finally stabilizes. The predictions made by ERT are more accurate if the models with X0 and X1 are validated by tenfold cross-validation. Yet, those built on X2 and X3 are evaluated by threefold cross-validation. Typically, the worse performance of ERT

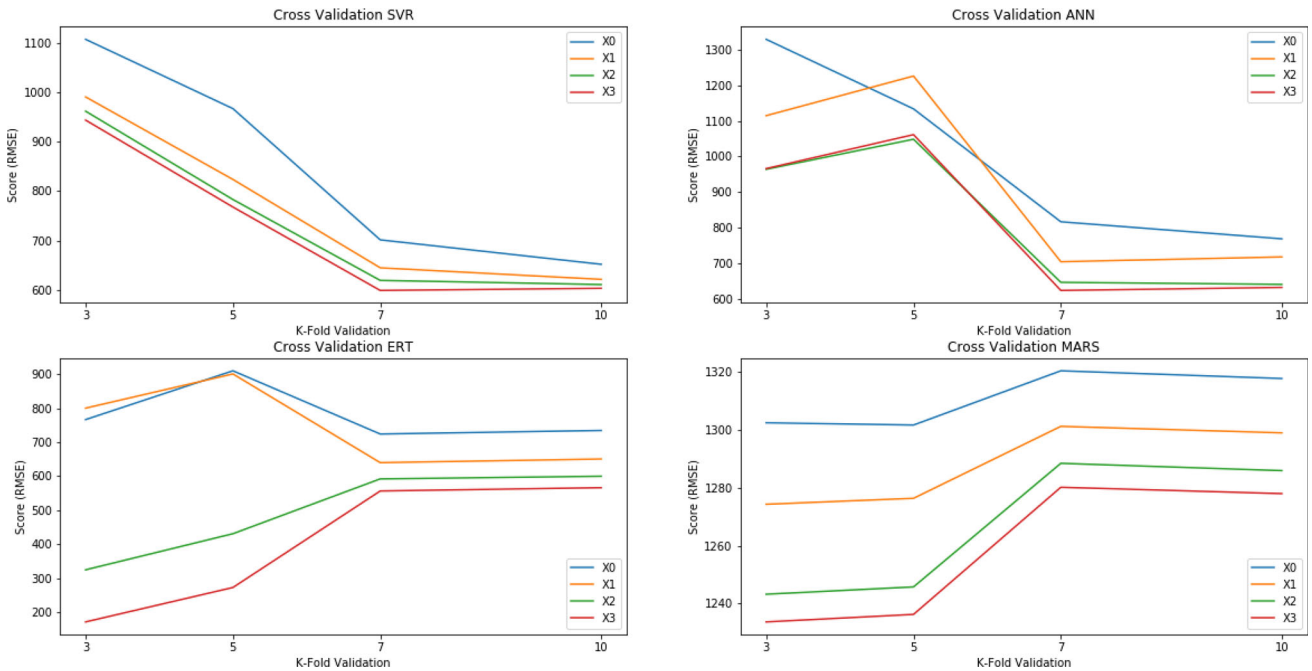

Fig. 13 RMSE results of the different cross-validation techniques

models is much better or at least comparable to the best performance of other algorithms. Roughly speaking, the tenfold cross-validation for most models yielded best performance with respect to the four k-fold cross-validations tested.

Table 8 is mainly intended to compare predictive performance of trained models with and without cross-validation. Its values indicate that the R^2, MAE and RMSE indices tended to improve for all four algorithms as the number of folds in the cross-validation increased. A first observation concerns predictions estimated by MARS, SVR and ANN, which are almost identical for all four feature sets. Thus, the three metrics' values are very close in the four cases. It seems therefore that cross-validation unifies these models' performance for any combination of explanatory variables. By contrast, ERT's effectiveness differs widely from one feature set to another.

Considering the three performance indicators, the top scoring model is X3-based ERT with an R^2 of 0.852, a MAE of 7.64 and an RMSE of 171.84. This finding highlights a considerable improvement in this model's performance, which reaches an average of 56% thanks to the cross-validation. In fact, ERT performance with X2 and X3 evolved positively when they were cross-validated 3 or 5 times. However, they maintained the same accuracy level when evaluated by tenfold or sevenfold cross-validation. Even worse, ERT's efficiency with X0 decreased remarkably, as this drop is at least 100%. ERT using X0 and X1 without cross-validation showed excellent performance but became less than average when the database was divided into any number of equal parts. The most negatively influenced metric is RMSE which rose from an average of 18.5 to an average of 800. Moreover, the cross-validation did not bring anything to MARS efficacy since the metrics values are not really changed. It was unable to improve the prediction accuracy of this machine learning method and therefore improve its ranking.

Similarly, sevenfold or tenfold cross-validation did not impact significantly the results of ANN with X0, X2 and X3. Instead, it minimized the error committed by this predictor based on X1: R^2 increased by 28%, MAE decreased by 23% and RMSE diminished by 24%. In contrast, ANN's prediction precision with any combination of explanatory variables is remarkably reduced due to the threefold or fivefold cross-validation. The scores obtained by SVR with X0 are slightly better thanks to the tenfold and sevenfold cross-validation. Yet, these techniques together with the fivefold cross-validation contributed to keeping the performance of the other models. Further on, threefold cross-validation has an opposite effect on all SVR models, a decrease in efficiency of at least 31% on all error measures is noted.

All these results clearly underline that cross-validation is able to improve as well as to degrade the predictive capacity of machine learning algorithms. Its effect depends mainly on several parameters: underlying model, regularization level, number of nearest neighbors… Problems also arise if the assumptions explicitly made by the cross-validation are violated:

- When the data are dependent, the training and validation samples are no longer necessarily independent, which can induce a rather strong bias for the risk estimation of a learning rule;
- When the time series is non-stationary, simple cross-validation does not guarantee a reliable prediction of the "future" from the "past";
- When estimator selection is performed from an exponential collection [95], the principle of unbiased risk estimation no longer works.

Overall, algorithm training times and efficiency are closely related and vary proportionally, i.e., a model is more accurate if its training time is longer. ERT is superior in terms of accuracy and computing time compared to other machine learning techniques. To understand why it is the best-performing algorithm and how each feature affects its outputs, counterfactual and contrastive explanations are generated by implementing SHAP method.

5.4 Prediction benchmark

Each prediction work mainly targets to obtain values as close as possible to reality. Thus, we compared our best predictions with those presented in [75]. In that paper, smart parking is modeled with multiagent system using long short-term memory neural network. It is evaluated based on the same sequential real data provided online by Aarhus City Council. Its authors have adopted a prediction accuracy metric equivalent to our R^2. After 400 iterations, this indicator converges to its maximum and achieves 0.97%.

ERT with feature set X0 reaches a R^2 of 0.988, implying that our model improves prediction accuracy by about 1.821%. Even better, ERT is a simple algorithm that doesn't require reinforcement by any other method to excel. In contrast, the approach proposed in [75] is considerably complex because it incorporates an advanced deep learning model into an artificial intelligence system. As a result, ERT is faster than the other paper model and needs a very low computation cost.

6 Feature importance

Machine learning models involve numerous complex models known as "black box" ones, for which understanding how they combine explanatory variables to make predictions is exceedingly challenging. Deep learning models such as artificial neural networks and ensemble models like random forests, gradient boosting learners or stacking algorithms

Table 8 K-fold cross-validation results

Regressors	Input set	Tenfold			Sevenfold			Fivefold			Threefold		
		RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2	RMSE	MAE	R^2
ANN	X0	746.06	19.39	0.387	729.86	19.31	0.395	1238.74	29.88	0.154	1279.39	28.33	0.038
	X1	722.13	18.49	0.341	704.28	18.29	0.353	1125.52	29.35	0.180	1098.64	29.56	− 0.001
	X2	631.96	18.34	0.412	645.50	17.93	0.416	1017.32	28.33	0.279	1138.74	28.27	0.123
	X3	613.99	17.57	0.432	617.21	17.71	0.441	993.26	24.57	0.190	998.15	27.17	0.165
SVR	X0	651.71	15.15	0.461	701.27	16.90	0.424	967.41	24.51	0.227	1107.88	25.59	0.119
	X1	621.16	14.04	0.410	644.60	15.56	0.399	823.80	22.12	0.321	991.11	23.73	0.198
	X2	610.67	13.69	0.420	619.07	15.04	0.424	783.09	21.33	0.359	962.19	23.24	0.223
	X3	603.09	13.50	0.421	598.68	14.64	0.438	767.83	21.02	0.374	944.05	22.92	0.239
ERT	X0	922.11	16.92	0.397	766.60	16.48	0.405	922.11	17.30	0.242	766.60	14.58	0.368
	X1	901.18	16.15	0.392	789.43	15.70	0.401	901.18	16.80	0.185	789.43	14.40	0.287
	X2	434.10	15.63	0.446	337.14	15.20	0.452	434.10	11.70	0.640	337.14	9.23	0.725
	X3	268.91	15.26	0.474	171.84	14.81	0.482	268.91	9.85	0.773	171.84	7.64	0.852
MARS	X0	1301.72	31.84	− 0.052	1302.49	31.91	− 0.054	1301.72	32.08	− 0.038	1302.49	32.01	− 0.038
	X1	1276.36	31.62	− 0.067	1274.31	31.68	− 0.068	1276.36	31.40	− 0.037	1274.31	31.29	− 0.035
	X2	1245.72	31.51	− 0.055	1243.18	31.57	− 0.057	1245.72	30.99	− 0.008	1243.18	30.88	− 0.007
	X3	1236.21	31.43	− 0.049	1233.61	31.49	− 0.051	1236.21	30.82	− 0.0005	1233.61	30.69	0.002

constitute examples of highly talented black box models. These methods are known for their remarkably accurate output in a wide variety of fields ranging from urban planning to computer vision. Hence, the greatest task facing researchers is to interpret the way in which predictors influence predictions, notably with conventional statistical methods.

The following paragraphs outline an alternative approach to interpreting black box models, labeled Permutation feature importance [96]. This is a robust tool for detecting features in a data set that have predictive power, no matter which model is chosen. It relies on randomly changing each column's (predictor variable) values, one column at a time. It then evaluates the model. That is, it first calculates the baseline error rate, next computes the same indicator by neutralizing each predictor variable in turn and finally forms the ratio between the two values. These returned scores represent the change in performance of a trained model, after permutation.

This technique refers to a decrease in model score given a single attribute value is randomly shuffled. It consists in indicating to what extent the model depends on the feature by breaking its relationship with the target. It is implemented through the permutation_importance() function of Scikit library which takes as input a fitting model, a data set and a scoring function. In our case, ERT without cross-validation is used as fitting model and the three error metrics as scoring function. Not using cross-validation can be explained by our intention to highlight the algorithm's intrinsic principle without any other technique intervention. The results are plotted in the graphs below.

By inspecting the three graphs, it is easily seen that the three data-generating predictors (Garage code, Month and Hour) have relatively large values for all three scores, meaning that they have significant predictive power in our model. On the other hand, the other predictors show relatively low or zero scores, implying that they are not as relevant to model decision making. Garage code is the explanatory variable that provides the most valuable information for predicting parking occupancy. It helps to increase R^2 value by 1.6, decrease MAE value by 30 and reduce RMSE value by 2000. Month and Hour also appear to be important predictors as they improve R^2 score by more than 0.5, MAE score by at least 14 and RMSE score by no less than 600. The importance of the Day and Day of the Week features is relatively small given that randomly shuffling their values only degrades model performance by 0.2 in R^2, 5 in MAE and 250 in RMSE. Year does not attribute to the model's decision process and therefore has no role in estimating predictions. This little or no impact on ratings prompts consideration of feature selection to remove these irrelevant or redundant predictors in future analyses, thus saving time and resources without sacrificing accuracy (Fig. 14).

Permutation feature importance results can also be visualized by means of a detailed boxplot that presents the average feature importance and the corresponding standard deviations. The graph's X-axis indicates the importance score (RMSE, MAE or R^2) and the Y-axis illustrates as many boxplots as features sorted by importance. The top variable being the most important, and the bottom being the least important. Each boxplot displays the importance values (minimum, maximum, median, first quartile and third quartile), although they are not clearly visible due to small deviations.

The three plots in Fig. 15 assert that Garage code strongly contributed to improving all three scores, an enhancement of 2089 with a standard deviation of 23 for RMSE score, a decrease of 31 with a standard deviation of 0.25 for MAE score and an increase of 1.6 with a standard deviation of 0.02 for R^2 score. Second place belongs to Month feature whose contribution is estimated to be an improvement of 837 with a standard deviation of 16 for RMSE, a diminution of 13 with a standard deviation of 0.18 for MAE and an augmentation of 0.65 for R^2 with a standard deviation of 0.013. For the third influencing parameter, Hour, its contribution is moderate with a decrease in RMSE values by 569 ($\pm$ 8.5), a reduction in MAE values by 12.7 ($\pm$ 0.13) and a small increase in R^2 values by 0.44 ($\pm$ 0.007).

As a conclusion, ERT's outstanding performance based on X0 and X1 is mainly due to modeling parking occupancy by the three most important explanatory variables, namely Garage code, Month and Hour. ERT's prediction accuracy with X0 slightly exceeds that of ERT's predictions with X1 owing to Day of Week's contribution which belongs to X0 and not to X1.

7 Model prediction interpretation

Shapley additive explanations (SHAP) is an approach that unifies methods like LIME [76 77] and DeepLIFT [78 79] under the class of additive feature attribution methods [80 81]. It describes the performance of a machine learning model based on game theory and local explanations [82]. It distributes the total gain or payoff among players, depending on the relative importance of their contributions to the final outcome [83 84].

It approximates an original model f with input variables $x = (x_1, \cdots, x_m)$, where m is the number of independent variables, by an explanation model g with simplified input x' expressed as:

$$f(x) = g\left(x'\right) = \phi_0 + \sum_{i=1}^{p} \phi_i x_i'. \tag{16}$$

ϕ_0 represents the constant value when all inputs are toggled off. Inputs x' and x are related through a mapping function, $x = h_x\left(x'\right)$. Equation 16 admits a single unique

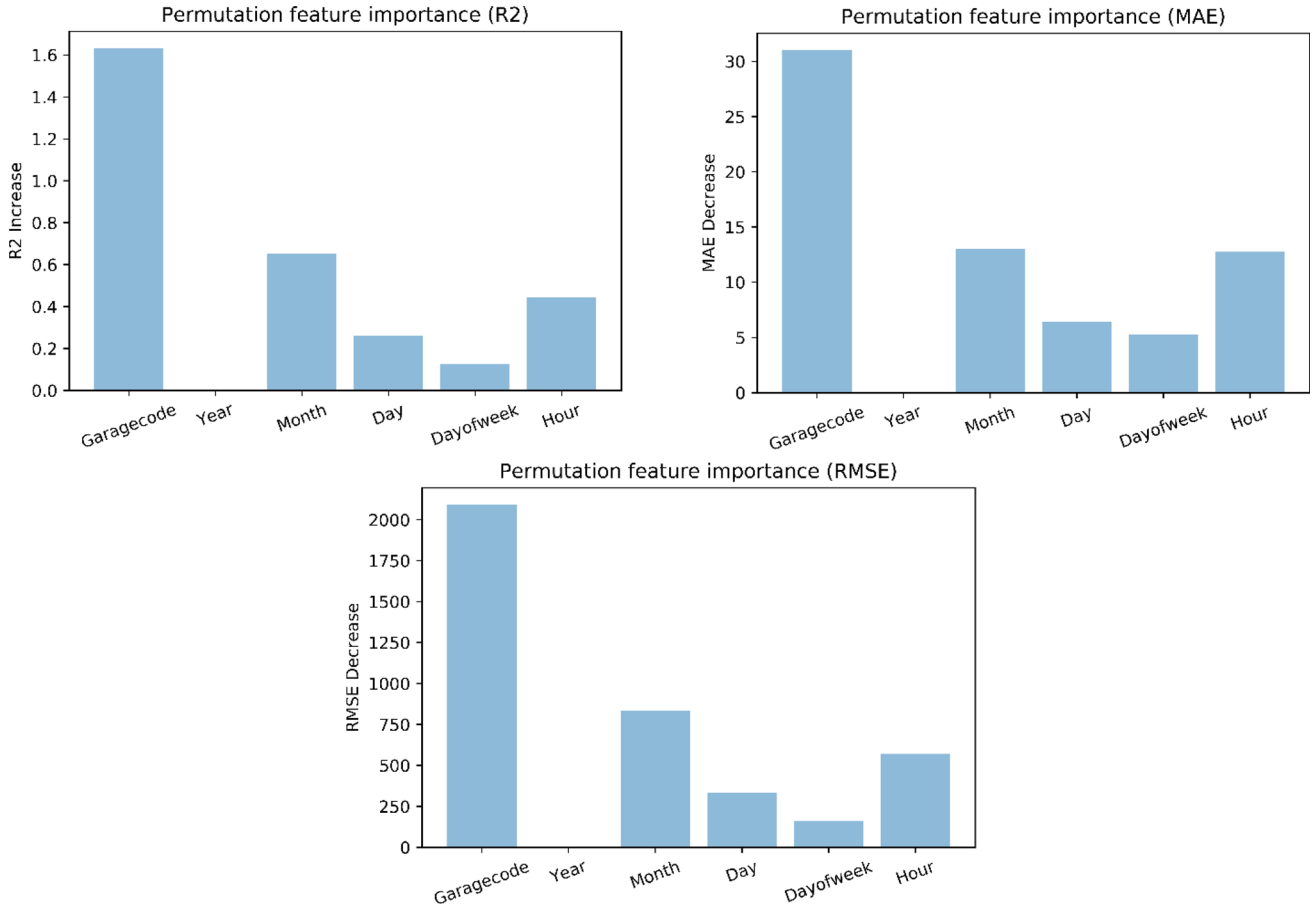

Fig. 14 Permutation feature importance results

solution (Eq. 17) that satisfies three axioms: local accuracy, consistency and missingness [85]. The first property implies that the sum of individual feature attributions must be equal to the original model prediction, i.e., the explanation model should match the original model. The second one states that changing a larger impact feature will not diminish that input's attribution and the last one guarantees that no importance is accorded to the missing features in the original input.

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{\left|z'\right|!\left(p - \left|z'\right| - 1\right)!}{p!}\left[f_x\left(z'\right) - f_x\left(z' \backslash i\right)\right] \tag{17}$$

where $\left|z'\right|$ is the number of nonzero entries in z', ϕ_i is the Shapely values and $f\left(h_x\left(z'\right)\right) \approx g\left(z'\right)$ is ensured by local methods when $x' \approx z'$.

In [85], Lundberg and Lee suggested SHAP values, which is a standardized measure of feature importance, as a solution to Eq. 17, where $f_x\left(z'\right) = f\left(h_x\left(z'\right)\right) = E[f(z)|z_S]$ and S is the set of nonzero indices in z'.

TreeSHAP is a variant of SHAP proposed in [86] and dedicated to tree-based machine learning models. And since it is the case of ERT, we applied it in our study. SHAP values are estimated thanks to the Shap Python library and represented graphically by means of four plots: variable importance, summary, dependency and individual force.

In variable importance plot, features are vertically sorted by their average impact on model output, allowing for a global interpretability and clarifying the whole structure of the model. Based on Figs. 16, 17, 18 and 19, the most important feature for predicting parking availability with an average impact greater than 20% is Garage code followed by Hour in second position and Month in third rank, and this is for the four sets of independent variables. Year is the least powerful indicator with a mean absolute SHAP value of almost zero. Day and Day of week have more or less the same importance, varying the absolute parking occupancy rate on average by 2.5% and slightly exceeding Weekend variable which improves predictions by about 2.2%. Except for Day feature, whose average impact differs from ERT model with X0 to ERT with X1, the other variables retain the same importance.

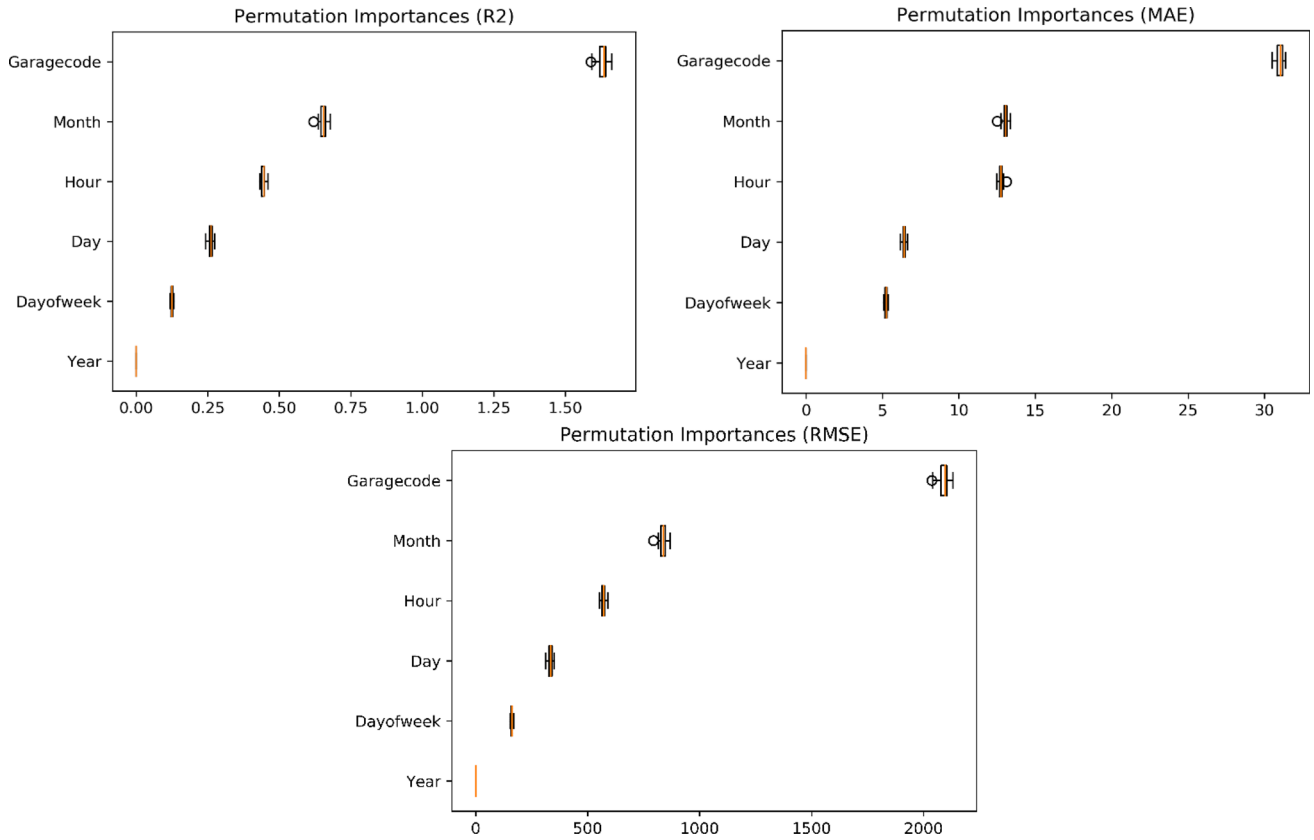

Fig. 15 Mean feature importance and related standard deviations

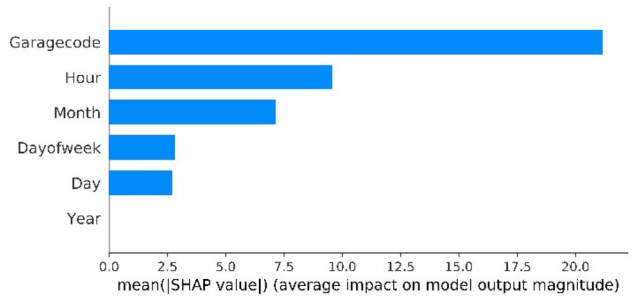

Fig. 16 Variable importance plot of ERT with X0

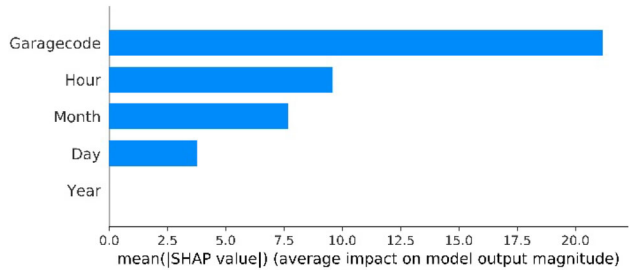

Fig. 17 Variable importance plot of ERT with X1

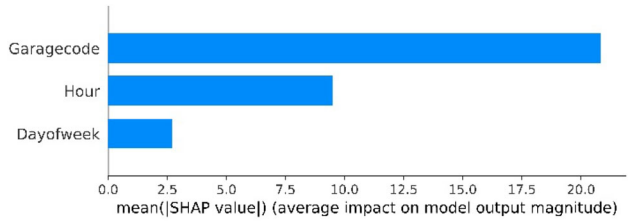

Fig. 18 Variable importance plot of ERT with X2

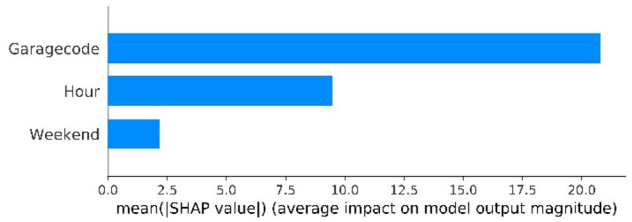

Fig. 19 Variable importance plot of ERT with X3

A summary plot combines feature importance with feature effects. It orders independent variables based on their importance to propel the model's predictive capability [82]. Every dot represents the SHAP value of an input variable and an instance [87]. The color maps the value of the features, from low to high. ERT with X0 and ERT with X1 models ignored Year input, i.e., it has no effect on their outputs. Figures 20 and 21 reveal that the closer is the year-end, the higher is the

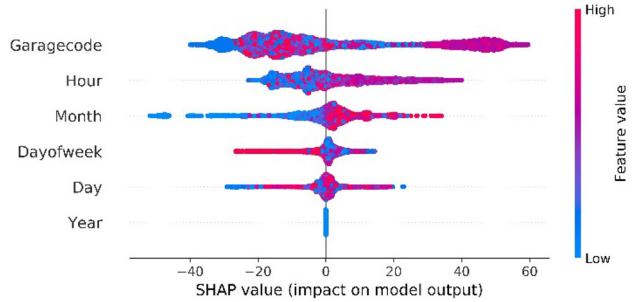

Fig. 20 SHAP summary plot of ERT with X0

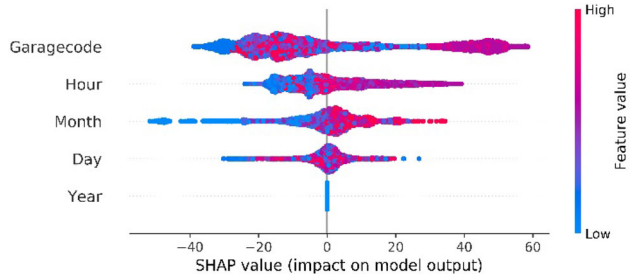

Fig. 21 SHAP summary plot of ERT with X1

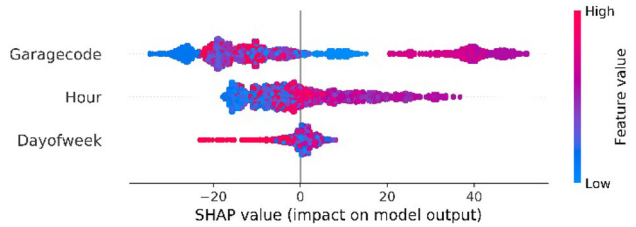

Fig. 22 SHAP summary plot of ERT with X2

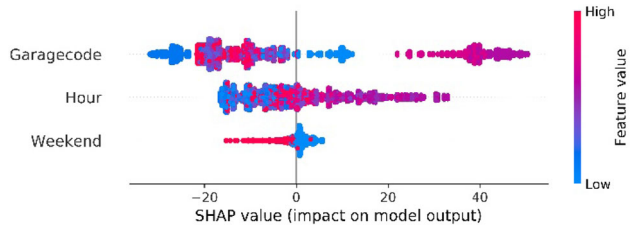

Fig. 23 SHAP summary plot of ERT with X3

SHAP values of these months, and the greater is their positive impact on the predicted parking occupancy rate. Parking lot identifier has no clear behavior regarding its values, high positive SHAP values correspond to parking lots with high identifiers and low negative ones correspond to parking lots with low identifiers but medium values refer to both parking lots with high and low identifiers. The early hours of the day negatively impact the model's output and cause low predictions contrary to the last hours which lead to high forecasts. The last-days of the week are typified by a decrease in the parking occupancy rate, the beginning and the middle of the week are characterized by an increase in this rate (Fig. 22). In other words, parking lots are less busy on weekends than on weekdays as predicted by the model (Fig. 23). The influence of Day variable (days of the month) is not evident enough, the only conclusion to be drawn is that low values (its beginning) are associated with negative impacts on parking availability forecasts.

A dependence plot is a scatter plot that shows the effect of a single feature on the predictions made by a machine learning model. It traces a feature's value relative to its SHAP value across many samples. Each point is a single prediction in the data set. The color corresponds to a second variable that can have an interaction effect with the feature that has been plotted [88]. The following figures display the marginal effect of Garage code feature on the expected outcome of ERT models as well as its interaction with Month, Day of week and Weekend variables. The first attribute is picked since it is the most important in line with the above and we needed to further highlight the spread and variation of its SHAP values. The second features are selected automatically as they interact the most with the chosen variable. The impact of Garage code is highly nonlinear, non-monotonic and very complex for all four curves, demonstrating that examining a single parameter, whether it is positive or negative, is often not conclusive. Its Shap values are dispersed in a significant way and vary from one parking lot to another, i.e., it strongly influences predictions performed for some instances and not for others. In ERT models with X0 and with X1, its impact on the predicted output is positive for car park 5, both positive and negative for parking lots 1 and 6, and negative for the rest. When Garage code is equal to 6, SHAP values for May, June and July range from -20% to 20% and increase up to 55% for August, September, October and November (Figs. 24 and 25). Despite some noise, SHAP values of the parking lot identified by 2 are less than −20% for the last five months and under 0% for the first two ones. Disregarding graphics coloring, the variation amplitude of the SHAP values is clearly lower in ERT models with X2 and X3 than with X0 and X1. Furthermore, their nature has not changed unless for parking lot number 6 where it has become purely positive (Figs. 26 and 27). Globally, there is no explicit relationship between the SHAP values of Garage code and the values of the covariates but their interaction is always important.

A force plot illustrates how the features in the model contribute to pushing its output from the base value (mean prediction) to the actual outcome, thus providing a local interpretability [88]. Features marked in red force the prediction upward, while those in blue force the prediction downward. The graphs above explain the prediction for the first example of the testing dataset. Its base value is equal to 1%, whereas the prediction is equal to 0.65% for ERT models with X0 and X1, since their performance is practically identical, 0.98%

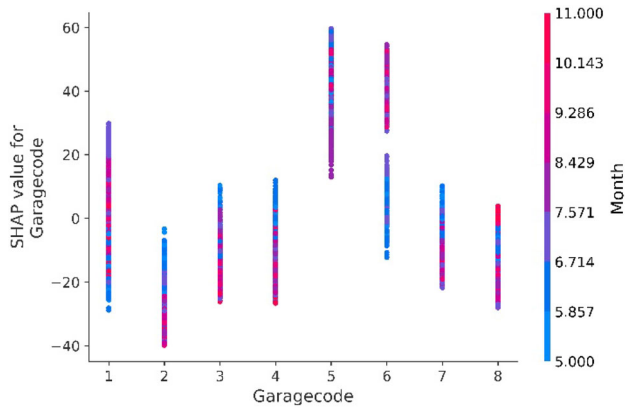

Fig. 24 SHAP dependence plot of ERT with X0

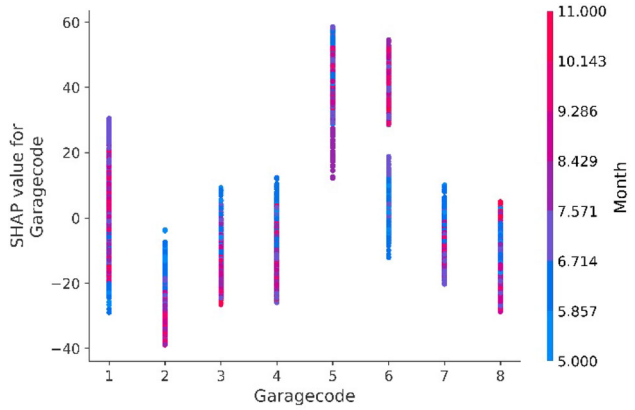

Fig. 25 SHAP dependence plot of ERT with X1

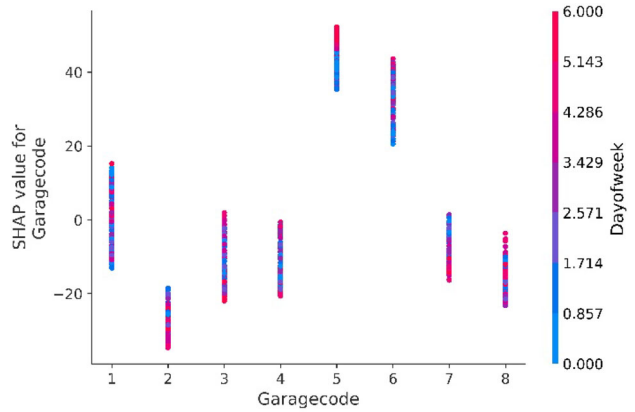

Fig. 26 SHAP dependence plot of ERT with X2

for ERT with X1 and 0.99% for ERT with X0. The observed value for that instance is equal to 0.63%. Garage code and Hour are the features with the biggest impact as the visual size indicates the magnitude of the feature's effect. They resulted in a decrease in predictions and this is also the case for Month and Day variables. Unlike general rule established from Figs. 16 and 17, Day attribute's impact reduced when calculating the ERT model's prediction based on X0 than on

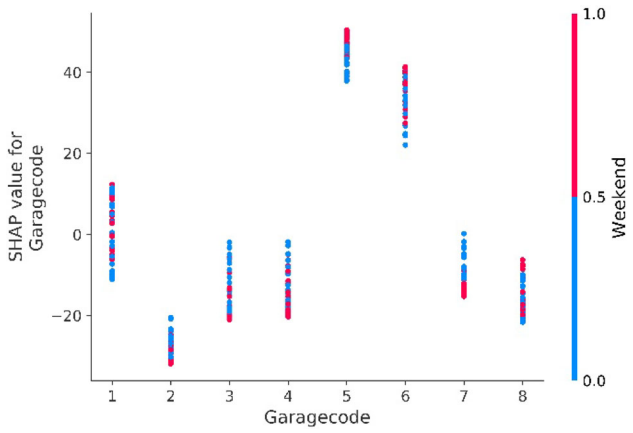

Fig. 27 SHAP dependence plot of ERT with X3

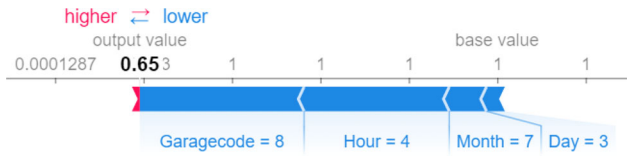

Fig. 28 SHAP force plot of ERT with X0

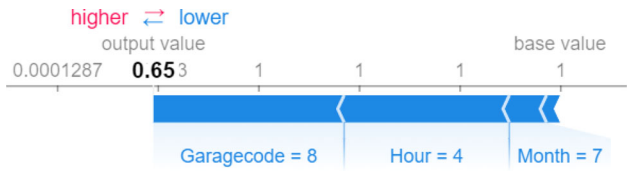

Fig. 29 SHAP force plot of ERT with X1

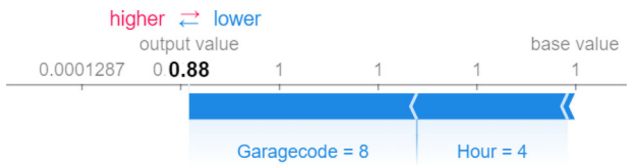

Fig. 30 SHAP force plot of ERT with X2

X1. Day of week weakly affects ERT models with X0 and X2, it drives predictions toward higher values for the first model and lower values for the second one. The positive contribution of Weekend binary feature is very small to zero, which means that the predictions made by ERT model based on X3 are not influenced at all by this variable' values (Figs. 28, 29, 30 and 31).

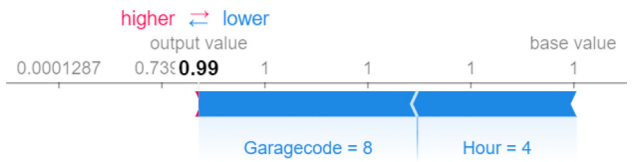

Fig. 31 SHAP force plot of ERT with X3

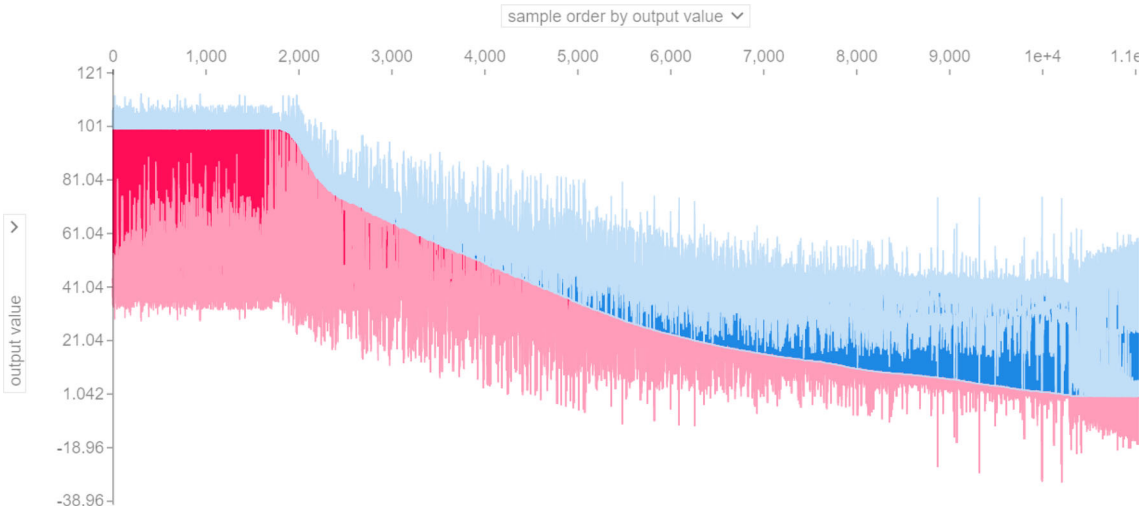

Fig. 32 Global force plot of ERT with X0

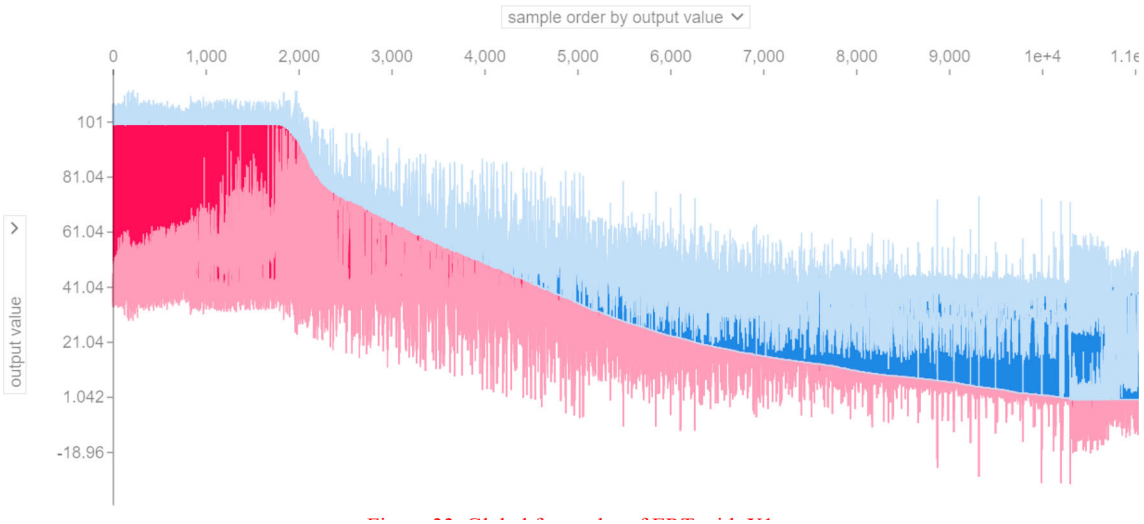

Fig. 33 Global force plot of ERT with X1

Above force plots are for individual observations only, if all observations plots are combined and sorted by output value, an interactive plot is derived, called global force plot. This graph shows SHAP value make-up for the entire dataset. The X-axis of this graph represents all observations in the dataset sorted by similarity, by output value, by original sample ordering or by each feature value. The Y-axis is simply an individual force graph for that observation showing by default its output value. This chart provides a quick summary of record position, output value and relevant parameters that push the prediction up (in red) or down (in blue).

Figures 32, 33, 34 and 35 represent a feature's responsibility for changing model output while sorting the samples by output value. At a high level, the diagram indicates that the model placed a lot of emphasis on Garage code, which is consistent with each parking lot being characterized by its location-dependent attendance level, accounting for wide variation in car park occupancy rates. Hovering over the area highlighted in red, it is apparent that most observations pushing predicted values upward are for records with 5 and 6 as Garage code values. By contrast, data vectors with values below 8 for Hour tend to generally decrease predicted values, which is expected because parking lots are not very busy in night time unless they are residential parking lots. Other variables' values do not follow a given trend, thus not understanding their influence on the models' outputs. Let A, B, C and D be four randomly selected records from our database, with:

$A = (\text{Garagecode} = 7, \text{Month} = 9, \text{Day} = 5)^t$
$B = (\text{Garagecode} = 1, \text{Month} = 7, \text{Day} = 27, \text{Hour} = 14)^t$
$C = (\text{Garagecode} = 1, \text{Day} = 29, \text{Hour} = 15)^t$
$D = (\text{Garagecode} = 5, \text{Day} = 1, \text{Hour} = 0)^t$

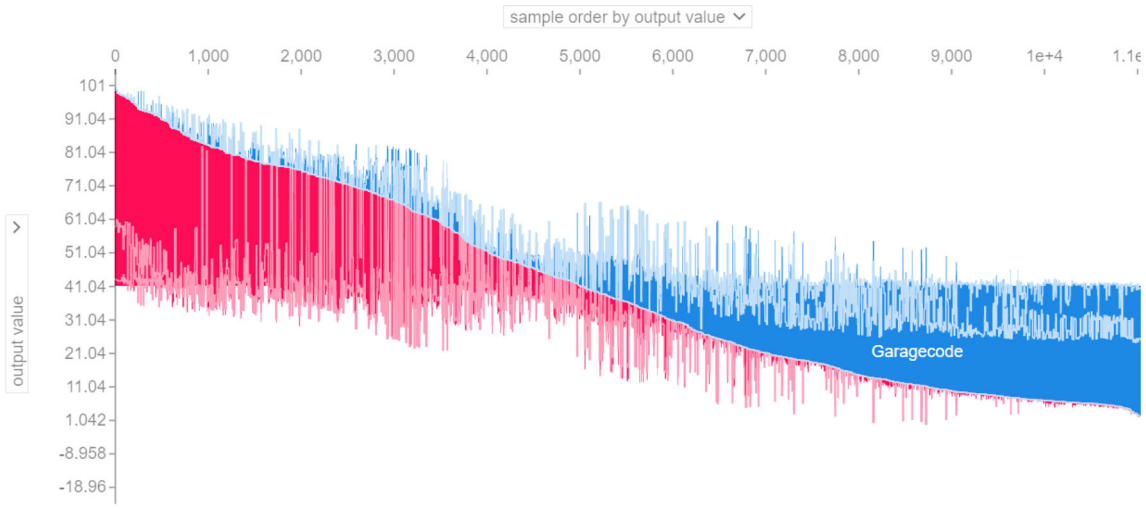

Fig. 34 Global force plot of ERT with X2

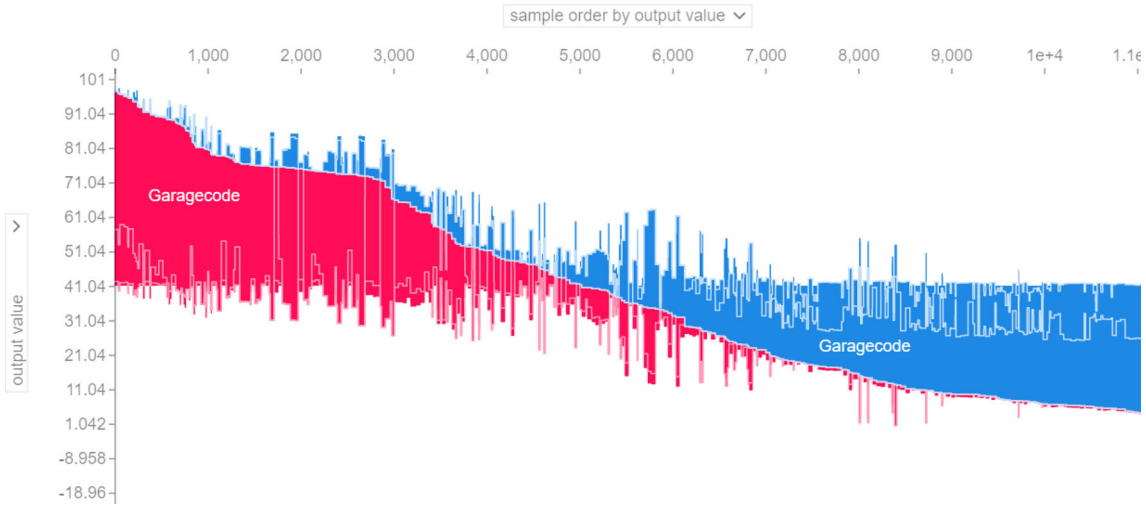

Fig. 35 Global force plot of ERT with X3

In the first two vectors, the variable Day drives predictions upward, while in the last two ones it drives them downward, even though their values are almost successive.

Another interesting plotting function from SHAP library permits an easy and global interpretation of feature contribution in vertically stacked 2D matrices. These are associated with a bar chart on the right displaying the feature importance in descending order and a curve on top depicting the prediction output for all instances. Each 2D matrix is a hot to cold color scheme indicating positive (red pixels) or negative (blue pixels) contribution for each feature.

To note that for explanation purpose, only 1000 data points were captured in the heatmap plot as there is a homogeneity in the overall dataset (Figs. 36, 37, 38 and 39).

The four graphs below emphasize the strong contribution of Garage code and two other features (Hour and Month). Garage code variable improves prediction for more than 200 observations, resulting in a contribution greater than 20% with SHAP values of over 40. In second place comes the contribution (10%–18%) of Hour feature with SHAP values ranging between 20 and 30. A small contribution of Month attribute not exceeding 8% follows in third rank with a SHAP value just below 20. The rest of the features' contribution is not relevant as SHAP values are close to 0.

In addition, above the maps, model output for each instance is plotted. These curves reveal that output value variation is more or less identical for the four feature sets: parking occupancy rates are high for the first few instances, decrease afterward and return to increase slightly again from the 800th instance. The heatmaps also exhibit that high predictions (high values in $f(x)$) are coupled with high SHAP values of the variable Garage code. This again confirms the

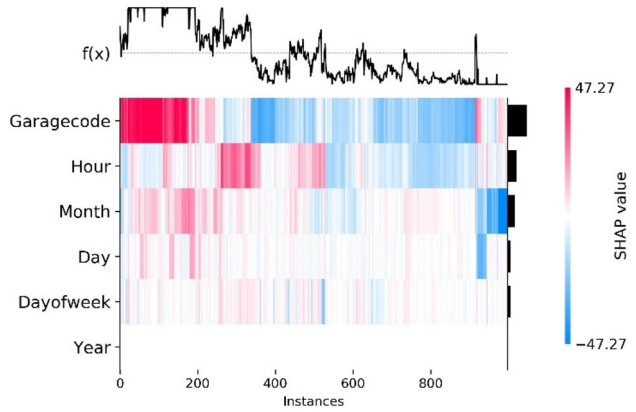

Fig. 36 Heatmap of ERT with X0

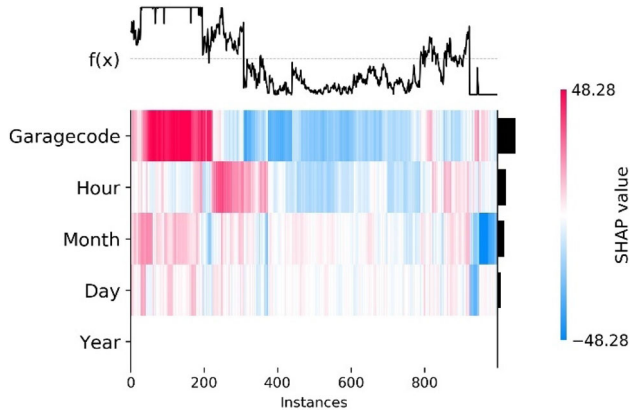

Fig. 37 Heatmap of ERT with X1

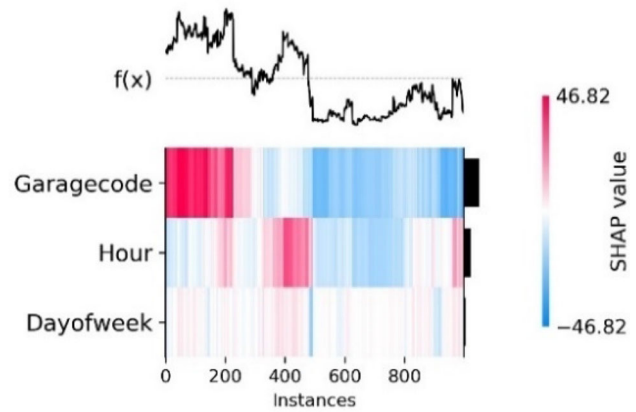

Fig. 38 Heatmap of ERT with X2

strong correlation between this feature and parking occupancy rate.

In short, Garage code has the greatest influence on parking availability prediction and strongly interacts with Month, Day of week and Weekend features. Its SHAP values do not evolve in line with a marked trend, this may be because it is

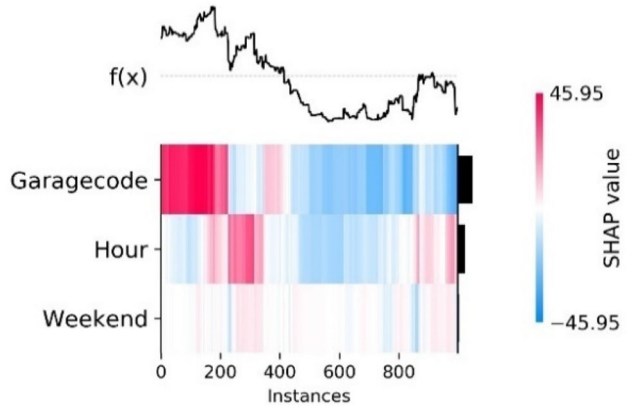

Fig. 39 Heatmap of ERT with X3

a categorical variable and its values do not have a numerical meaning. Thus, this predominant feature is the most correlated variable with the parking occupancy rate, which may explain ERT's performance. Hour and Month notably influence predictions produced by the concerned models and their correlation coefficient is close to 1, improving further their precision.

8 Conclusion

In this paper, we suggested a model-based practical framework for predicting parking occupancy rate to help drivers quickly find where to park. We particularly interested in the way input variables influence the accuracy of predictions. We used, for this purpose, four feature combinations. We compared the ability of four different machine learning algorithms in modeling our target from these features. We got as results that ERT built on X0, the least computationally intensive algorithm compared to ANN and SVR, is the most efficient method. We then tried to understand why it performs best and analyze how each variable contributes to its outputs by applying SHAP values. We discovered, through visualizations, important nonlinear and non-monotonous relationships between independent variables that heavily affect predictions.

As a constraining factor, in order to maximize the exactness of predictions, it would be preferable to shorten prediction time to less than 10 min instead of one hour. Promising improvements may also involve supplementing historical data with real-time information and events calendar. Given that an event in the vicinity will inevitably increase parking demand compared to usual. Having a general idea about parking availability is necessary but not sufficient to eliminate problems associated with searching for vacant spaces. It can be complemented by more precise information such as

the probability of a place be left vacant until the driver arrival and the coordinates or identifiers of available spaces.

Declarations

Conflicts of interest On behalf of all authors, the corresponding author states that they have no affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership or other equity interest; and expert testimony or patent-licensing arrangements) or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject matter or materials discussed in this manuscript.

References

1. Khatoun, R., Zeadally, S.: Smart cities: concepts, architectures, research opportunities. Commun. ACM **59**(8), 46–57 (2016). https://doi.org/10.1145/2858789
2. Giffinger, R., Fertner, C., Kramar, H., Kalasek, R., Pichler-Milanović, N., Meijers, E.: Smart Cities: Ranking of European Medium-Sized Cities. Vienna University of Technology, Centre of Regional Science (SRF) (2007)
3. Harrison, C., Eckman, B., Hamilton, R., Hartswick, P., Kalagnanam, J., Paraszczak, J., Williams, P.: Foundations for Smarter Cities. IBM J. Res. Dev. **54**(4), 1–16 (2010). https://doi.org/10.1147/JRD.2010.2048257
4. Zanella, A., Bui, N., Castellani, A., Vangelista, L., Zorzi, M.: Internet of things for smart cities. IEEE Internet Things J. **1**(1), 22–32 (2014). https://doi.org/10.1109/jiot.2014.2306328
5. Shoup, D.C.: Cruising for parking. Transp. Policy **13**(6), 479–486 (2006). https://doi.org/10.1016/j.tranpol.2006.05.005
6. Giuffrè, T., Siniscalchi, S.M., Tesoriere, G.: "A novel architecture of parking management for smart cities. proc. Soc. behav Sci **53**, 16–28 (2012). https://doi.org/10.1016/j.sbspro.2012.09.856
7. D. Ayala, O. Wolfson, B. Xu, B. DasGupta and J. Lin: "Pricing of parking for congestion reduction," In Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 43–51, doi: https://doi.org/10.1145/2424321.2424328.
8. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Vairo, C.: Car parking occupancy detection using smart camera networks and deep learning. 2016 IEEE Symposium on Comput. Commun. (ISCC) Messina (2016). https://doi.org/10.1109/ISCC.2016.7543901
9. Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., Vairo, C.: Deep learning for decentralized parking lot occupancy detection. Expert Sys. Appl. (2017). https://doi.org/10.1016/j.eswa.2016.10.055
10. S. Yoo, P. Chong, T. Kim, J. Kang, D. Kim, C. Shin, K. Sung and B. Jang, "PGS: Parking Guidance System based on wireless sensor network," 3rd International Symposium on Wireless Pervasive Computing, 2008, pp. 218–222, doi: https://doi.org/10.1109/ISWPC.2008.4556200.
11. Roman, C., Liao, R., Ball, P., Ou, S., de Heaver, M.: Detecting on-street parking spaces in smart cities: performance evaluation of fixed and mobile sensing systems. IEEE Trans. Intell. Transp. Syst. **19**(7), 2234–2245 (2018). https://doi.org/10.1109/TITS.2018.2804169
12. Yan, G., Yang, W., Rawat, D.B., Olariu, S.: SmartParking: a secure and intelligent parking system. IEEE Intell. Transp. Sys. Mag. Spring **3**(1), 18–30 (2011). https://doi.org/10.1109/MITS.2011.940473
13. S. Ahmed, Soaibuzzaman, M. S. Rahman and M. S. Rahaman, "A Blockchain-Based Architecture for Integrated Smart Parking Systems," 2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kyoto, Japan, 2019, pp. 177–182, doi: https://doi.org/10.1109/PERCOMW.2019.8730772.
14. Caicedo, F.: The use of space availability information in '"PARC"' systems to reduce search times in parking facilities. Transp. Res. C Emerg. Technol. **17**, 60–68 (2009). https://doi.org/10.1016/J.TRC.2008.07.001
15. Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., Zeinalipour-Yazti, D.: "Crowdsourcing with Smartphones. In IEEE Internet Comput. **16**(5), 36–44 (2012). https://doi.org/10.1109/MIC.2012.70
16. J. Kopecký and J. Domingue, "ParkJamJAM: Crowdsourcing parking availability information with linked data (DEMO)," Proc. Extended Semantic Web Conf., 2012, pp. 381–386.
17. A. Nandugudi, T. Ki, C. Nuessle and G. Challen: "PocketParker: Pocketsourcing parking lot availability," UbiComp 2014 - Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014, pp. 963–973, doi: https://doi.org/10.1145/2632048.2632098.
18. B. Kifle, J. Villalobos, D. Riley and J. Quevedo-Torrero: "Crowdsourcing automobile parking availability sensing using mobile phones", Proc. Midwest Instruct. Comput. Symp, 2015, pp. 1–7.
19. R. Liao, C. Roman, P. Ball, S. Ou and L. Chen: "Crowdsourcing On-street Parking Space Detection," ArXiv, 2016, abs/1603.00441.
20. W. Viriyasitavat, P. Sangaroonsilp, J. Sumritkij and N. Tarananopas: 2015 Mobile crowdsourcing platform for intelligent car park systems, 2015 International Computer Science and Engineering Conference (ICSEC) Chiang Mai, , doi: https://doi.org/10.1109/ICSEC.2015.7401398.
21. T. Yan, B. Hoh, D. Ganesan, K. Tracton, T. Iwuchukwu, J.-S. Lee, CrowdPark: A Crowdsourcing-based Parking Reservation System for Mobile Phones. University of Massachusetts at Amherst Tech, 2011, Technical Report.
22. Atif, Y., Kharrazi, S., Jianguo, D., Andler, S.F.: Internet of things data analytics for parking availability prediction and guidance. Trans. Emerging Tel. Tech. **31**, e3862 (2020). https://doi.org/10.1002/ett.3862
23. Xiao, J., Lou, Y., Frisby, J.: How likely am I to find parking? – A practical model-based framework for predicting parking availability. Transp. Res. Part B: Methodol. **112**, 19–39 (2018). https://doi.org/10.1016/j.trb.2018.04.001
24. Surafel, T., Di Marzo Serugendo, G.: Cooperative multi-agent system for parking availability prediction based on time varying dynamic markov chains. J. adv. Transp. **2017**, 1760842 (2017). https://doi.org/10.1155/2017/1760842
25. Beheshti, R., Sukthankar, G.: A hybrid modeling approach for parking and traffic prediction in urban simulations. AI & Soc. **30**, 333–344 (2015). https://doi.org/10.1007/s00146-013-0530-7
26. Ji, Y., Tang, D., Blythe, P., Guo, W., Wang, W.: Short-term forecasting of available parking space using wavelet neural network model. IET Intel. Transport Syst. **9**(2), 202–209 (2015). https://doi.org/10.1049/iet-its.2013.0184
27. Hsueh-Chan, L., Chen-Hao, L.: Prediction-based parking allocation framework in urban environments. Int. J. Geogr. Inf. Sci. (2020). https://doi.org/10.1080/13658816.2020.1721503
28. Sergio, D.M., Origlia, A.: Exploiting recurring patterns to improve scalability of parking availability prediction systems. Electronics **9**, 838 (2020)
29. Zhang, W., Liu, H., Liu, Y., Zhou, J., Xiong, H.: Semi-supervised hierarchical recurrent graph neural network for city-wide parking availability prediction. Proc. AAAI Conf. Artif. Intell. **34**, 1186–1193 (2020). https://doi.org/10.1609/aaai.v34i01.5471

30. F. Richter, S. Di Martino and D. C. Mattfeld: "Temporal and Spatial Clustering for a Parking Prediction Service," 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, Limassol, 2014, pp. 278–282, doi: https://doi.org/10.1109/ICTAI.2014.49.
31. F. Bock, S. Di Martino, and A. Origlia: "A 2-Step Approach to Improve Data-driven Parking Availability Predictions," In Proceedings of the 10th ACM SIGSPATIAL Workshop on Computational Transportation Science (IWCTS'17), Association for Computing Machinery, New York, NY, USA, 2017, pp. 13–18, doi: https://doi.org/10.1145/3151547.3151550.
32. Fabusuyi, T., Hampshire, R., Hill, V., Sasanuma, K.: Decision analytics for parking availability in downtown Pittsburgh. INFORMS J. Appl. Anal. **44**(3), 286–299 (2014). https://doi.org/10.1287/inte.2014.0743
33. A. Ionita, A. Pomp, M. Cochez, T. Meisen and S. Decker: "Where to Park? Predicting Free Parking Spots in Unmonitored City Areas," 8th International Conference on Web Intelligence, Mining and Semantics, Novi Sad, Serbia, 2018, pp. 1–12. 10, doi: 1145/3227609.3227648.
34. Y. Rong, Z. Xu, R. Yan and X. Ma, "Du-Parking: Spatio-Temporal Big Data Tells You Realtime Parking Availability," In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 646–654.
35. Y. Zheng, S. Rajasegarar and C. Leckie: "Parking availability prediction for sensor-enabled car parks in smart cities," 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 2015, pp. 1–6, doi: https://doi.org/10.1109/ISSNIP.2015.7106902.
36. Vlahogianni, E.I., Kepaptsoglou, K., Tsetsos, V., Karlaftis, M.G.: A Real-Time parking prediction system for smart cities. J. Intell. Transp. Sys. **20**(2), 192–204 (2016). https://doi.org/10.1080/15472450.2015.1037955
37. J. Li, J. Li and H. Zhang, "Deep Learning Based Parking Prediction on Cloud Platform," 4th International Conference on Big Data Computing and Communications (BIGCOM), Chicago, IL, 2018, pp. 132–137, doi: https://doi.org/10.1109/BIGCOM.2018.00028.
38. Errousso, H., Malhene, N., Benhadou, S., Medromi, H.: Predicting car park availability for a better delivery bay management. Proc. Comput. Sci. **170**, 203–210 (2020). https://doi.org/10.1016/j.procs.2020.03.026
39. D.H. Stolfi, E. Alba and X. Yao, "Predicting Car Park Occupancy Rates in Smart Cities," In: Alba E., Chicano F., Luque G. (eds) Smart Cities. Smart-CT 2017. Lecture Notes in Computer Science, 2017, vol 10268. Springer, Cham.
40. A. Camero, J. Toutouh, D.H. Stolfi and E. Alba, "Evolutionary Deep Learning for Car Park Occupancy Prediction in Smart Cities," In: Battiti R., Brunato M., Kotsireas I., Pardalos P. (eds) Learning and Intelligent Optimization. LION 12 2018. Lecture Notes in Computer Science, 2019, vol 11353. Springer, Cham.
41. Cédric Stéphane, K.T., El Arbi, A.A., Cherif, W., Silkan, H.: Improving parking availability prediction in Smart Cities with IoT and ensemble-based model. J. King Saud Univ. Comput. Inf. Sci. (2020). https://doi.org/10.1016/j.jksuci.2020.01.008
42. A. Chirichigno, S. Vidal, J.A. Diaz-Pace and C. Marcos, "Predicción de Disponibilidad de Estacionamiento en la Vía Pública," El Congreso Nacional de Ingeniería en Informática / Sistemas de Información (CoNaIISI), Sede Mar del Plata, Argentina, 2018.
43. Caicedo, F., Blazquez, C., Miranda, P.: Prediction of parking space availability in real time. Expert Sys. Appl. **39**, 7281–7290 (2012)
44. T. Rajabioun, B. Foster, and P.A. Ioannou, "Intelligent parking assist," in 21st Mediterranean Conference on Control and Automation, Chania, 2013, pp. 1156–1161.
45. Rajabioun, T., Ioannou, P.A.: On-street and off-street parking availability prediction using multivariate spatiotemporal models. IEEE Trans. Intell. Transp. Syst. **16**(5), 2913–2924 (2015). https://doi.org/10.1109/TITS.2015.2428705
46. W. Shao, Y. Zhang, B. Guo, K. Qin, J. Chan and F.D. Salim, "Parking Availability Prediction with Long Short-Term Memory Model," In: Li S. (eds) Green, Pervasive, and Cloud Computing. GPC 2018. Lecture Notes in Computer Science, 2019, vol 11204, Springer, Cham.
47. X. Chen, "Parking occupancy prediction and pattern analysis," Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep. CS229–2014, 2014.
48. Sedgwick, Ph.: Pearson's correlation coefficient. BMJ **345**, e4483–e4483 (2012). https://doi.org/10.1136/bmj.e4483
49. Yeh, C.Y., Huang, C.W., Lee, S.J.: A multiple-kernel support vector regression approach for stock market price forecasting. Expert Syst. Appl. **38**(3), 2177–2186 (2011). https://doi.org/10.1016/j.eswa.2010.08.004
50. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004). https://doi.org/10.1023/B:STCO.0000035301.49549.88
51. Buhmann, M.D.: Radial basis functions. Acta Numer **9**, 1–38 (2000)
52. H. ERROUSSO, J. EL OUADI, S. BENHADOU, et al., "Improving delivery conditions by dynamically managing the urban parking system: Parking availability prediction," In: 13th International Colloquium of Logistics and Supply Chain Management (LOGISTIQUA), IEEE, 2020, pp. 1–6.
53. W. Mckinney, et al., "Data structures for statistical computing in python," In : Proceedings of the 9th Python in Science Conference, 2010, pp. 51–56.
54. Hill, T., Marquez, L., O'Connor, M., Remus, W.: Artificial neural network models for forecasting and decision making. Int. J. Forecast. **10**(1), 5–15 (1994). https://doi.org/10.1016/0169-2070(94)90045-0
55. Qeethara, A.-S.: Artificial neural networks in medical diagnosis. Int. J. Comput. Sci. Issues **8**(2), 150–154 (2011)
56. Park, D.C., El-Sharkawi, M.A., Marks, R.J., Atlas, L.E., Damborg, M.J.: Electric load forecasting using an artificial neural network. IEEE Trans. Power Syst. **6**(2), 442–449 (1991). https://doi.org/10.1109/59.76685
57. Hsu, K., Gupta, H.V., Sorooshian, S.: Artificial neural network modeling of the rainfall-runoff process. Water Resour. Res. **31**(10), 2517–2530 (1995). https://doi.org/10.1029/95WR01955
58. SC. Wang, "Artificial Neural Network," In: Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, 2003, vol 743, Springer, Boston, MA. doi: https://doi.org/10.1007/978-1-4615-0377-4_5.
59. Freidman, J.: Multivariate adaptive regression splines. Ann. Stat. **19**(1), 1–141 (1991)
60. Chou, S.-M., Lee, T.-S., Shao, Y.E., Chen, I.-F.: Mining the breast cancer pattern using artificial neural networks and multivariate adaptive regression splines. Expert Syst. Appl. **27**(1), 133–142 (2004). https://doi.org/10.1016/j.eswa.2003.12.013
61. Zhou, Y., Leung, H.: Predicting object-oriented software maintainability using multivariate adaptive regression splines. J. Syst. Softw. **80**(8), 1349–1361 (2007). https://doi.org/10.1016/j.jss.2006.10.049
62. Lee, T., Chen, I.: A two-stage hybrid credit scoring model using artificial neural networks and multivariate adaptive regression splines. Expert Syst. Appl. **28**(4), 743–752 (2005). https://doi.org/10.1016/j.eswa.2004.12.031
63. Lee, T.-S., Chiu, C.-C., Chou, Y.-C., Lu, C.-J.: Mining the customer credit using classification and regression tree and multivariate adaptive regression splines. Comput. Stat. Data Anal. **50**(4), 1113–1130 (2006). https://doi.org/10.1016/j.csda.2004.11.006
64. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach Learn **63**, 3–42 (2006). https://doi.org/10.1007/s10994-006-6226-1

65. Shang, K., Yao, Y., Li, Y., Yang, J., Jia, K., Zhang, X., Chen, X., Bei, X., Guo, X.: Fusion of five satellite-derived products using extremely randomized trees to estimate terrestrial latent heat flux over Europe. Remote sens. **12**(4), 687 (2020). https://doi.org/10.3390/rs12040687
66. Lee, H., Hien, Ph.: Brain tumor segmentation using U-Net based fully convolutional networks and extremely randomized trees. Vietnam J. Sci. Technol. Eng. **60**, 19–25 (2019). https://doi.org/10.31276/VJSTE.60(3).19
67. Galelli, S., Castelletti, A.: Assessing the predictive capability of randomized tree-based ensembles in streamflow modelling. Hydrol. Earth Syst. Sci. **17**, 2669–2684 (2013)
68. M.I. Ali, F. Gao, A. Mileo, "CityBench: A Configurable Benchmark to Evaluate RSP Engines Using Smart City Datasets," In: Arenas M. et al. (eds) The Semantic Web - ISWC 2015. ISWC 2015. Lecture Notes in Computer Science, 2015, vol 9367, Springer, Cham, doi: https://doi.org/10.1007/978-3-319-25010-6_25.
69. Parking Data Stream provided by City of Aarhus in Denmark, available at http://iot.ee.surrey.ac.uk:8080/datasets.html#parking and consulted on August 26, 2020.
70. T. Adetiloye and A. Awasthi, "Predicting short-term congested traffic flow on urban motorway networks," In Handbook of Neural Computation, 2017, pp. 145–165, doi: https://doi.org/10.1016/B978-0-12-811318-9.00008-9.
71. Gunawardana, A., Shani, G.: A survey of accuracy evaluation metrics of recommendation tasks. J. Mach. Learn. Res. **10**, 2935–2962 (2009). https://doi.org/10.1145/1577069.1755883
72. Pal, R.: Validation methodologies. Predict. Modeling of Drug Sens. (2017). https://doi.org/10.1016/B978-0-12-805274-7.00004-X
73. Nakagawa, S., Johnson, P.C.D., Schielzeth, H.: The coefficient of determination R2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. J. Royal Soc. Interface **14**(134), 20170213 (2017). https://doi.org/10.1098/rsif.2017.0213
74. Renaud, O., Victoria-Feser, M.-P.: A robust coefficient of determination for regression. J. Stat. Plan. Inference **140**(7), 1852–1862 (2010)
75. Anagnostopoulos, T., Fedchenkov, P., Tsotsolas, N., Ntalianis, K., Zaslavsky, A., Salmon, I.: Distributed modeling of smart parking system using LSTM with stochastic periodic predictions. Neural Comput. Appl. **32**, 10783–10796 (2019). https://doi.org/10.1007/s00521-019-04613-y
76. G. Visani, E. Bagli, F. Chesani, A. Poluzzi, and D. Capuzzo: "Statistical Stability Indices for LIME: Obtaining Reliable Explanations for Machine Learning Models, " In: arXiv:2001.11757, 2020.
77. M. Ribeiro, M. Singh and C. Guestrin: Why Should I Trust You?": Explaining the Predictions of Any Classifier, In: arXiv:1602.04938v3, 2016, 97–101, doi: https://doi.org/10.18653/v1/N16-3020.
78. A. Shrikumar, P. Greenside and A. Kundaje: Learning Important Features Through Propagating Activation Differences," In: arXiv:1704.02685, 2017.
79. A. Shrikumar, P. Greenside, A. Shcherbina and A. Kundaje: Not Just a Black Box: Learning Important Features Through Propagating Activation Differences," In: arXiv:1605.01713, 2016.
80. Mangalathu, S., Hwang, S.-H., Jeon, J.-S.: Failure mode and effects analysis of RC members based on machine-learning-based SHapley additive exPlanations (SHAP) approach. Eng. Struct. **219**, 110927 (2020). https://doi.org/10.1016/j.engstruct.2020.110927
81. I. Giurgiu and A. Schumann: 2019 Additive Explanations for Anomalies Detected from Multivariate Temporal Data," Proceedings of the 28th ACM International Conference on Information and Knowledge Management - CIKM '19, doi:https://doi.org/10.1145/3357384.3358121.
82. Parsa, A.B., Movahedi, A., Taghipour, H., Derrible, S., Mohammadian, A.: Toward safer highways, application of XGBoost and SHAP for real-time accident detection and feature analysis. Accid. Anal. Prev. **136**, 105405 (2020). https://doi.org/10.1016/j.aap.2019.105405
83. Rodríguez-Pérez, R., Bajorath, J.: Interpretation of machine learning models using shapley values: application to compound potency and multi-target activity predictions. J Comput Aided Mol Des **34**, 1013–1026 (2020). https://doi.org/10.1007/s10822-020-00314-0
84. Rodríguez-Pérez, R., Bajorath, J.: Interpretation of compound activity predictions from complex machine learning models using local approximations and shapley values. J. Med. Chem. **63**(16), 8761–8777 (2019). https://doi.org/10.1021/acs.jmedchem.9b01101
85. S. Lundberg and S. Lee, "A Unified Approach to Interpreting Model Predictions," In: arXiv:1705.07874, 2017.
86. S. M. Lundberg, G. G. Erion and S.-I. Lee, "Consistent individualized feature attribution for tree ensembles," In arXiv:1802.03888, 2018.
87. K. El Mokhtari, B.P. Higdon and A. Başar, "Interpreting financial time series with SHAP values," In Proceedings of the 29th Annual International Conference on Computer Science and Software Engineering (CASCON '19), IBM Corp., USA, 2019, pp.166–172.
88. I. Murugesan, K. Murugesan, L. Balasubramanian and M. Arumugam, "Interpretation of Artificial Intelligence Algorithms in the Prediction of Sepsis," 2019 Computing in Cardiology (CinC), Singapore, Singapore, 2019, pp. Page 1-Page 4, doi: https://doi.org/10.23919/CinC49843.2019.9005667.
89. E. De Banville, Les systèmes de transport intelligent: un enjeu stratégique Mondial, 1999.
90. E. Winter: The shapley value. Handbook of game theory with economic applications, 2002, vol. 3, p. 2025-2054.
91. Pedregosa, F., Varoquaux, G., Gramfort, A., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
92. L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, ... and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," In arXiv:1309.0238, 2013.
93. Virtanen, P., Gommers, R., Oliphant, T.E., et al.: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nat. Methods **17**(3), 261–272 (2020)
94. Kuhn, M., Johnson, K., et al.: Applied predictive modeling. Springer, New York (2013)
95. S. Arlot, Fondamentaux de l'apprentissage statistique, 2017.
96. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
97. Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. **10**, 1895–1923 (1998)