





Article

Smart Parking: Enhancing Urban Mobility with Fog Computing and Machine Learning-Based Parking Occupancy Prediction

Francisco J. Enríquez ^{1,†} , Jose-Manuel Mejía-Muñoz ^{1,*,†} , Gabriel Bravo ^{1,†}  and Oliverio Cruz-Mejía ^{2,*,†} 

¹ Departamento de Ingeniería Eléctrica, Instituto de Ingeniería y Tecnología, Universidad Autónoma de Ciudad Juárez, Ciudad Juárez 32310, Mexico; fenrique@uacj.mx (F.J.E.); gbravo@uacj.mx (G.B.)

² Departamento de Ingeniería Industrial, FES Aragón, Universidad Nacional Autónoma de México, Mexico 57171, Mexico

* Correspondence: manuel@cognitivemfg.net (J.-M.M.-M.); oliverio.cruz.mejia@comunidad.unam.mx (O.C.-M.)

† These authors contributed equally to this work.

Abstract: Parking occupancy is difficult in most modern cities because of increases in the accessibility and use of motor vehicles, and users generally take several minutes or even hours to find a place to park. In this work, we propose a smart parking prediction model in order to help users locate in advance the availability of parking near the places they plan to visit. For this it is proposed a fog computing architecture that integrates a machine learning algorithm based on AdaBoost to predict parking places hours or days in advance. Additionally, a user interface was developed, which involves the collection of user inputs through a mobile application where the user is prompted to enter the destination location and the prediction time interval. Through extensive experimentation using real-world parking flow data, our proposed algorithm demonstrated an improved level of accuracy compared with alternative prediction methods. Moreover, a simulation was conducted to evaluate the system's latency when using cloud computing versus our hybrid approach combining both fog and cloud computing. The results showed that employing the fog module in conjunction with cloud computing significantly reduced response delay in comparison with using cloud computing alone.

Keywords: prediction; parking occupancy; fog computing



Citation: Enríquez, F.J.; Mejía-Muñoz, J.-M.; Bravo, G.; Cruz-Mejía, O. Smart Parking: Enhancing Urban Mobility with Fog Computing and Machine Learning-Based Parking Occupancy Prediction. *Appl. Syst. Innov.* **2024**, *7*, 52. <https://doi.org/10.3390/asi7030052>

Academic Editor: Friedhelm Schwenker

Received: 26 April 2024

Revised: 7 June 2024

Accepted: 13 June 2024

Published: 17 June 2024



Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

The urban landscape presents a complex and intricate challenge when it comes to finding parking spaces for the growing vehicle population, especially in downtown areas during rush hour. The phenomenon of parking shortages is a multifaceted issue intertwined with urban planning, transportation engineering, and human behavior. The growing density of vehicles in urban areas has caused a spatial mismatch between the demand for parking and the available supply, which makes the search for parking spaces a difficult task when resorting to certain areas of the population. This mismatch has given rise to a large number of scientific studies aimed at understanding the spatial, temporal, and behavioral dynamics of the use of parking spaces, as well as the search for a solution to quickly find available parking locations in real time.

Because of the widespread availability of cars, traffic congestion caused by the increasing volume of traffic is a global issue [1,2]. Moreover, finding parking spaces for an ever-growing number of vehicles exacerbates this issue. Drivers seek to find available spots close to their destinations to minimize walking distances, but this search for parking can cause traffic congestion and decrease vehicle speed; some studies have shown that almost 30% of urban traffic is due to drivers searching for parking spots [3]. The issue of traffic congestion and limited parking spaces is becoming a worldwide problem. Despite the restrictions on CO₂ emissions, the number of vehicles on the roads is predicted to rise rather than fall, with conventional cars being replaced by electric cars [4], implying that unless steps are taken to address these issues they will only worsen with time. Therefore,

allocating parking spaces is a critical concern for large- and medium-sized cities where high traffic and a growing demand for parking availability are necessary for daily life. Finding parking spaces is not only time-consuming but also stressful for drivers, especially in busy urban areas [5]. The issue of parking allocation has been a topic of active research, with several proposals emerging in various areas such as optimizing parking allocation, intelligent parking, and automatic detection of vacant spaces. However, one area of opportunity that has gained significant attention is the prediction of parking spaces, which is an essential aspect of an all-encompassing solution to the parking allocation problem. Accurately predicting the availability of parking spaces can significantly improve the efficiency of parking space utilization, reduce traffic congestion, and save time for drivers. This approach requires the use of advanced technologies such as machine learning and artificial intelligence to analyze real-time data on parking lot occupancy and traffic patterns. Several innovative solutions have been proposed to predict parking space availability, such as using sensors, cameras, and mobile applications. These technologies provide real-time information to drivers on parking availability and guide them to the nearest available parking spot. Moreover, the data gathered from these systems can be analyzed to generate insights into parking usage patterns, which can be used to optimize parking lot allocation and enhance parking space utilization.

For instance, in ref. [6], the authors used a graph-convolutional neural network followed by a long short-term memory layer to extract features of traffic flow data and predict parking occupancy. Their work demonstrates better predictions for business areas than recreational locations. Another study [7] predicts the intent and trajectory of vehicles using a transformer model combined with convolutional neural networks trained with video data of persons driving in parking lots and traffic scenarios. In ref. [8] is proposed a deep learning architecture based on combining GRU with a graph convolutional network to extract both temporal and spatial correlation information. The GRU detects temporal information, while the graph convolutional network is incorporated into the GRU cell to extract spatial correlations. The research by ref. [3] involves a comparison of various prediction methods for parking spaces, including deep learning, standard machine learning, and classical methods. The methods they examined include long short-term memory from deep learning, seasonal autoregressive integrated moving average as a classical method, and the ensemble-based decision trees as a general method of deep learning. The study found that the ensemble method and the long short-term memory generally performed better in producing predictions with lower errors compared with the classic autoregressive-based method. The study conducted by ref. [9] compared the predictive performance of various models including linear regression, support vector machine, neural networks, and autoregressive integrated moving average. It was found that the support vector machine model offered the most stable and accurate predictions among the models reviewed.

Predicting parking space availability is a vital area of research that can significantly improve the efficiency of parking space utilization and reduce traffic congestion. As technology advances, more innovative solutions are likely to emerge that will transform the way we manage parking allocation and enhance the overall driving experience.

The studies reviewed above demonstrate that deep learning techniques, particularly the transformer, are commonly employed. However, these techniques require substantial computational power, and therefore cloud computing may be necessary. Nonetheless, the high computational cost and the increasing number of prediction requests may result in latency problems, even with the use of cloud computing.

In this work, we propose a fog computing-based system that predicts parking time for each vehicle and parking lot availability based on parking history records. The system incorporates a machine learning algorithm integrated into the cloud module, which reduces response time and provides data analytics on parking statistics. By utilizing this system, cities can optimize their parking spaces, reduce traffic congestion, and enhance the efficiency of daily life.

This paper presents several contributions, including the development of a fog computing platform to minimize response delays in the system. Additionally, infrastructure based on the AdaBoost algorithm, which is of low computational cost, is proposed for integration to a fog node. This model is considered as a generalization of the autoregressive model. Another contribution is the creation of a mobile application that acts as the user interface for the system. Finally, various machine learning techniques are compared to determine the most suitable method for parking space prediction. Through these contributions, the paper provides a comprehensive framework that enhances the efficiency and accuracy of parking space prediction.

The remainder of this paper is structured in the following manner. Section 1 provides an introduction to the theory of fog computing and the AdaBoost algorithm. Section 2 presents the proposed fog computing system, which includes the fog node and the user interface, implemented as a mobile application. Section 3 describes the methodology used for experimentation, while Section 4 presents the results obtained from the prediction and latency calculations. Finally, Section 5 summarizes the conclusions drawn from the research.

2. Theory

In this section, we will provide definitions for key concepts related to fog computing, as well as an overview of the machine learning (ML) algorithm known as AdaBoost.

2.1. Fog Computing

Fog computing was proposed by Cisco to solve latency problems in cloud-based computing systems. A cloud computing setup involves transferring large amounts of data from the client application to where the cloud server is located, which can lead to overheads in terms of performance, time, cost, and power consumption [10]. This can be critical for applications that need to run in real time and that require a minimal of latency and maximum throughput, such as surveillance, medical, and industrial systems, among others. In addition, most of the current Internet of Things (IoT) infrastructures do not have a homogeneous platform, which causes problems in the integration between the different heterogeneous hardware and software services for their communication with the cloud. The purpose of fog computing is to alleviate the problems that arise in applications where the processing of large amounts of constantly generated data that have to be processed with a minimum of latency is required. Fog computing addresses the above challenges through placing the fog nodes at the edge of the network, and this proximity to the device layer is in turn helpful in the following points [11]:

- A reduction in latency, since the data generated by devices such as sensors and actuators can be served by the fog nodes in the local area network, which significantly reduces the movement of data over the Internet and provides fast and high-quality localized services.
- Increased interoperability, which helps integrate vertically fragmented IoT platforms. Fog nodes achieve this by supporting communication across a diverse range of protocols [12].
- Optimized bandwidth allocation. The data transfer rate or capacity of the network is augmented by the help of the fog layer, because part of the processing is carried out in the fog nodes and only more specific information is transferred to the cloud.

2.2. Parking Prediction and AI

With the continuous acceleration of urbanization, the parking problem is becoming increasingly serious. Effective management of parking resources has become an urgent issue, and one solution is to enhance management and planning through parking prediction. Recent studies on parking space availability leverage deep learning techniques. For example, the work in ref. [13] employs a short-term demand prediction algorithm based on a convolutional neural network (CNN) and long short-term memory (LSTM) neural network. Similarly, ref. [14] uses LSTM alone for the prediction task. In contrast,

ref. [15] incorporates a multi-head attention mechanism into an LSTM network, finding an increase in accuracy compared with using LSTM alone. The study in ref. [16] utilizes a transformer-based scheme for spatial-temporal prediction. While these methods harness the power of deep learning to make predictions, they typically require large amounts of data to achieve good generalization. This requirement can hinder their performance when adapting to specific parking lots with limited data availability. In this work, we evaluate machine learning techniques that are lightweight and generally adapt better than deep learning techniques when there are few data. Additionally, they are easier to implement in embedded systems, as they do not require extra processing power, such as GPUs, to achieve good performance.

2.3. AdaBoost

A boosting algorithm enhances a weak learning algorithm in the sense of PAC (probably approximately correct) [17] to achieve high accuracy. AdaBoost is a boosting algorithm that creates a new classifier by sequentially training multiple weak classifiers. The goal is to have each subsequent classifier improve the classification accuracy of the previous one. Finally, the algorithm combines the outputs of all the classifiers using a weighted majority vote. In the case of binary classification, the training algorithm for AdaBoost is illustrated in Algorithm 1 [18,19].

Algorithm 1 AdaBoost Algorithm

- 1: Given $D = \{(x_i, y_i) : i = 1, \dots, m\}$
 - 2: Initialize weights $w_i = \frac{1}{m}, i = 1, 2, \dots, m$
 - 3: **for** $t = 1$ to T **do**
 - 4: Use weak hypothesis $h_t(x)$ providing it with the distribution w over D
 - 5: Compute the weighted error $\epsilon_t = \sum_{i=1}^m w_i^{(t)} \mathbf{1}_{\{h_t(x_i) \neq y_i\}}$ of $h_t(x)$
 - 6: Compute $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon_t}{\epsilon_t}$
 - 7: Update weights $w_i^{(t+1)} = \frac{w_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, where $Z_t = \sum_i w_i^{(t)} \exp(-\alpha_t y_i h_t(x_i))$ is a normalization factor
 - 8: **end for**
 - 9: **return** $H(x) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(x)\right)$
-

In Algorithm 1, D is the dataset, y_i is the corresponding label vector with $y_i \in \{-1, +1\}$, and T is the number of weak classifiers or base learners to combine in the ensemble. The weights, w_i , are initialized as equal to $\frac{1}{m}$, where m is the size of the dataset. At each iteration, t , a weak classifier, $h_t(x)$, is selected that minimizes the weighted error on the dataset. The weighted error, ϵ_t , is then used to compute the weight, α_t , of the weak classifier in the final ensemble. The weights of the data points, w_i , are updated according to the misclassification of the weak classifier.

3. Methods

This section outlines the architecture of a proposed fog system designed to manage and predict parking space availability. The architecture consists of three layers, as shown in Figure 1. The first layer is the IoT layer, which includes the parking lots themselves that provide real-time updates on their space availability. Users in their vehicles or on their mobile devices can receive predictions of the number of available parking spaces that will be available at a specific time, helping them to plan their parking ahead of time. The second layer is the fog layer, which receives information requests from the users and the parking data. This layer incorporates a predictor of the number of parking spaces that will be available in the future, which utilizes a ML algorithm based on AdaBoost. This algorithm helps to accurately predict the number of available parking spaces, based on historical data and real-time updates. Finally, the cloud layer receives data from the parking lots, stores it, and generates statistics on their usage. Additionally, based on the performance of

the predictor, this layer is responsible for training or retraining the algorithm to improve its accuracy over time. The main types of fog agents considered are fog gateways, fog node/server, and fog storage [20]. With respect to the supported communication protocols between the fog layer and the IoT layer, the main protocols are TCP/IP and 5G or 6G, as the primary interaction with users will be through mobile applications. However, for communication between the parking lot and the fog node, the protocols must be compatible with the sensor network in the parking lot. Sensor networks can consist of individual space sensors or cameras covering a large area. The communication protocols can be wireless, such as TCP/IP, Zigbee, or Bluetooth, or they can be wired, using protocols like I2C and MQTT, among others. Based on the investigation in ref. [21], which explores the relationship between the optimal number of fog nodes and the path loss coefficient to account for the impact of shadowing and fading, we can estimate the required number of fog nodes. One of its experiments suggests that, on average, 1.52 fog nodes are needed for 262 end devices, with certain conditions given in ref. [21]. Therefore, for example, for parking lots with 1000 spaces, we recommend using approximately six fog nodes to ensure adequate coverage and performance.

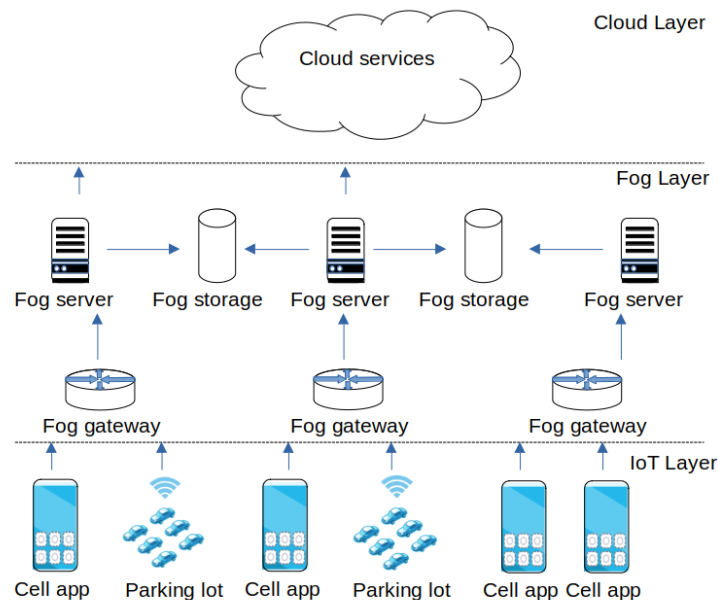


Figure 1. Architecture of the proposed system based on fog computing.

Overall, this proposed fog system for parking space management and prediction has the potential to significantly improve parking space utilization, reduce traffic congestion, and save time for drivers. By utilizing advanced technologies, such as IoT, fog computing, and ML, this system can provide real-time updates and accurate predictions to users, helping to streamline the parking experience and enhance overall driving efficiency.

3.1. Module for Occupancy Prediction

This study proposes a fog module that utilizes a machine learning regressor to predict the availability of parking spaces over time. This fog module has the potential to significantly enhance the efficiency of parking space management in cities, resulting in reduced traffic congestion and improved accessibility for drivers. The fog module consists of two subsystems: the user interface management subsystem and the prediction subsystem. Additionally, it incorporates fog storage to save recent historical data on parking usage for the last seven days. These subsystems are depicted in Figure 2. The user interface management subsystem receives data from the user application. The prediction subsystem receive requests for prediction from the user interface management subsystem and uses recent historical data to return predictions. The prediction subsystem uses the AdaBoost

regressor to forecast parking space availability for a given date. Parking data are also transferred to the cloud for storage and statistical management via the cloud analytics module. This module determines the need to retrain the regressor based on the error rate.

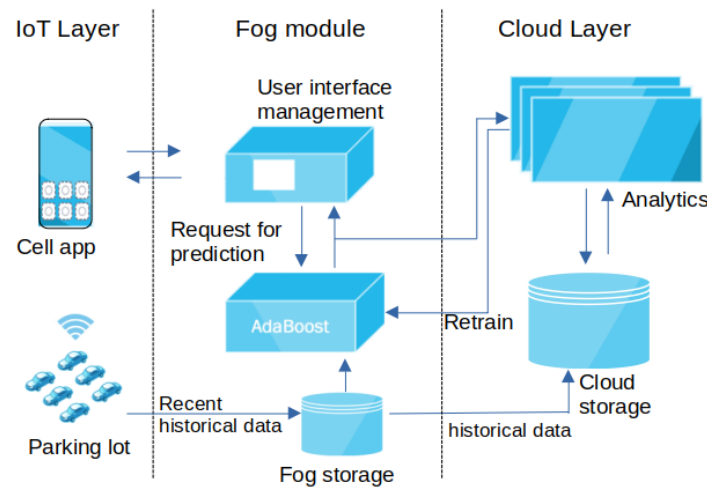


Figure 2. The proposed system: featuring a fog module for processing user requests, highlighting the iterative interactions between subsystems.

In the present investigation, we proposed utilizing past parking occupancy information for prediction, which is represented as a sequence of data over time, $S = \{S_n\}_{n=1}^N$, where S_n is the parking occupation at time slot n . In addition, we incorporate features, f_n , linked to each component of the sequence, S_n , which expands the dataset as $S_e = \{(S_n, f_n)\}_{n=1}^N$. Moreover, we consider the traditional autoregressive model of order I as a base model, which is given by

$$S_{n+1} = \sum_{i=0}^I \beta_{n-i} S_{n-i} + e_n, \tag{1}$$

$$S_{n+1} = \beta_n S_n + \beta_{n-1} S_{n-1} + \dots + \beta_{n-I} S_{n-I} + e_n. \tag{2}$$

From this, we also propose to use the AdaBoost algorithm as a generalization to the classic model, similar to the neural autoregressive model of [22], and we express this as follows,

$$S_{n+1} = \text{AdaBoost}(S_n, S_{n-1}, \dots, S_{n-I}) + e_n \tag{3}$$

where e_n is white noise with zero mean and variance σ_e^2 . In (3), the features are the samples taken backwards according to the order of the model I , f_n is the set with elements $S_n, S_{n-1}, \dots, S_{n-I}$, and $\text{AdaBoost}(\cdot)$ is a function that combines the output of R number of regressors according to the AdaBoost regression algorithm. The formula for calculating the output, S_{n+1} , is as in [18]

$$\text{Adaboost}(f_n) = \inf \left\{ y \in Y : \sum_{t: h_t(f_n) \leq y} \log(1/\beta_t) \geq \frac{1}{2} \sum_t \log(1/\beta_t) \right\}, \tag{4}$$

where $h_t(\cdot)$ are weak regressors, and each β_t depends on the error, ϵ_t , of regressor t , which are calculated in the training phase.

In this work, we also propose to calculate the output of the regressor using the Tukey’s biweight location estimator [23]. The median, on which Equation (4) is based, is a resistant

estimate, but it has only moderate robustness of efficiency. However, the biweight location estimator is resistant with robust efficiency. The formula for the biweight is given by

$$\text{Adaboost}(f_n) = h^* = \frac{\sum_t w_t h_t(f_n)}{\sum_t w_t}, \tag{5}$$

where

$$w_t = \max(1 - (\frac{h_t(f_n) - h^*}{cS})^2, 0)^2 \tag{6}$$

and S is the median of the set $\{|h_t(f_n) - h^*|\}_t$, and c is a parameter of the algorithm. Note that the value h^* of the estimator is computed iteratively.

As far as we are aware, no previous research has proposed the biweight with the AdaBoost regressor, particularly in the context of parking prediction. In Figure 3, it is shown the scheme for training the AdaBoost algorithm, using windows of past samples as input and comparing the output of the regressor with the one-step-ahead sample.

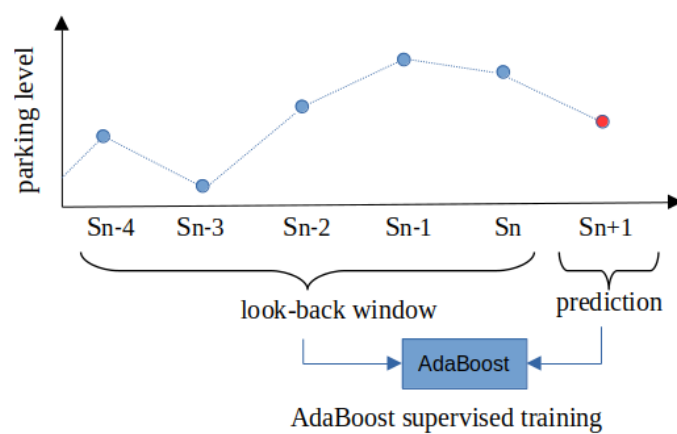


Figure 3. Adaboost training from a window of past samples.

To evaluate the effectiveness of the method, we conducted experiments using a parking space database and predicted availability for the next day. In this case, we chose a seventh-order model because the parking data varied weekly. The Adaboost estimator was configured with $R = 550$ decision tree regressors, a learning rate of 1.1, and a quadratic loss function for each boosting iteration; these settings achieve the most precise forecast of parking space availability over time.

3.2. User Interface

The user interface (Figure 4) consists of receiving data from the user through a mobile application. In this application, user are asked for the location of the destination where they want to go and the time frame of the prediction; for now, only three locations are implemented as Parking one, Parking two, and Parking three (Figure 5a). The application shows the available parking lots of the selected destination and offers, in real time, the amount of space available in each parking lot.

To this end, an Android mobile application was designed to establish communication with the fog node for parking space availability prediction (Figure 4). In addition, the application can save the location of the parked vehicle to facilitate its retrieval. To obtain predictions of parking space availability, users need to select a “Choose place/time” button in the app, to access a list of parking lot options. After selecting the desired parking lot, a second list appears for selecting the time period for which the prediction is requested. Once the user has made these selections, the information is sent to the fog node using Google’s Firebase service [24,25], which generates and sends the prediction back to the application for display.

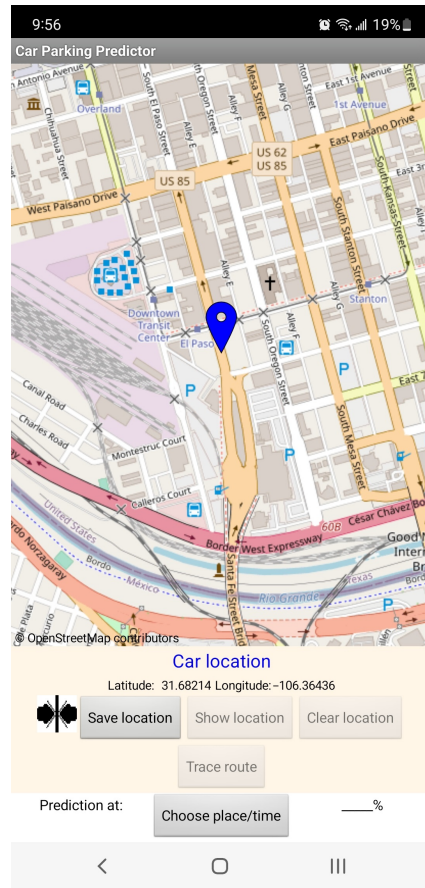


Figure 4. Fog node app user interface.

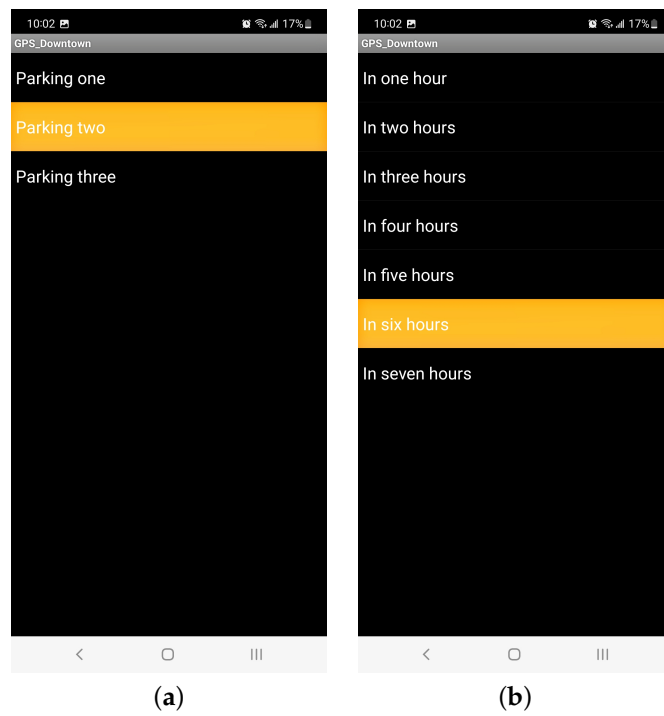


Figure 5. Drop-down lists for parking lot and time frame selection. (a) Parking lot list. (b) Frame time list.

Figure 4 provides an example of the user interface of the application, where the current location of the mobile device is represented by a blue globe on the map. To access the

parking space availability prediction server, the user needs to tap on a “Choose Place/time” button located at the bottom of the application. This action will display two consecutive selection lists. The first list allows the user to choose one of the three available parking lots in the predictor, as shown in Figure 5a. Once the user selects the parking lot of interest, a second list will appear, showing the available time frames, as depicted in Figure 5. After selecting both the parking lot and the desired time frame, the options are sent to the fog node, and the button that was initially labeled as “Choose place/time” displays the predicted parking lot availability percentage on the right side of the button, as illustrated in the example shown in Figure 6. Algorithm 2 shows the procedure executed by the app, while Figure 7 illustrates the sequence diagram depicting a user’s prediction request initiated by clicking the “Choose place/time” button. The diagram shows the flow of information, including parking lot preferences and time inputs sent to Google’s Firebase for generating a prediction, which is then presented on the app’s graphical user interface (GUI).

Algorithm 2 Parking space availability prediction procedure

Input: “Choose place/time” button clicked

Output: Predicted parking space availability is displayed

- 1: **Initialization** Choose parking lot from the displayed menu.
 - 2: Choose time from the displayed menu.
 - 3: Selected location and time are sent to fog Node via Google “Firebase”
 - 4: Fog Node send calculated parking space availability prediction to App via Google’s “Firebase”
 - 5: Parking space availability prediction is displayed in App
-

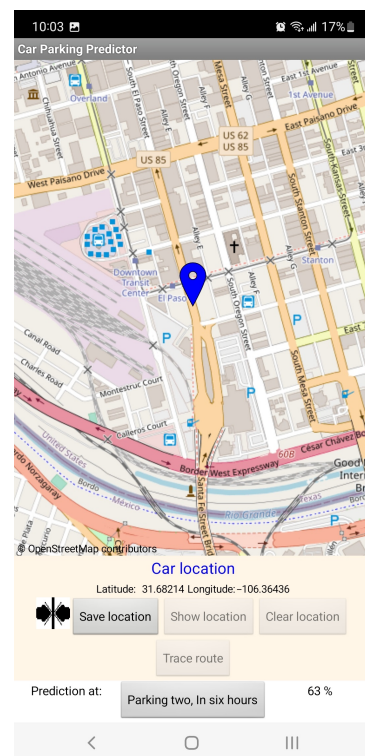


Figure 6. Percentage availability of parking spaces prediction.

An additional feature of the application is to save a location through the “Save location” button (Figure 8a), which can be used to save the location of the parking space where the vehicle has been left; this location will appear with a red balloon on the application map (Figure 8) and can continue to be seen while the mobile device is moving (Figure 9a,b), until the location is deleted by pressing the “Clear location” button (Figure 8c). If the location is

not deleted, it can be shown centered on the map by pressing the “Show location” button (Figure 8b); the map centered on the saved location is shown in Figure 9c. Another very useful functionality of the application is to show the route to reach the saved location; when pressing the “Trace route” button (Figure 8d), on the map, a convenient walking path is displayed from the mobile device’s current location to the saved parking space location where the vehicle was left (Figure 10).

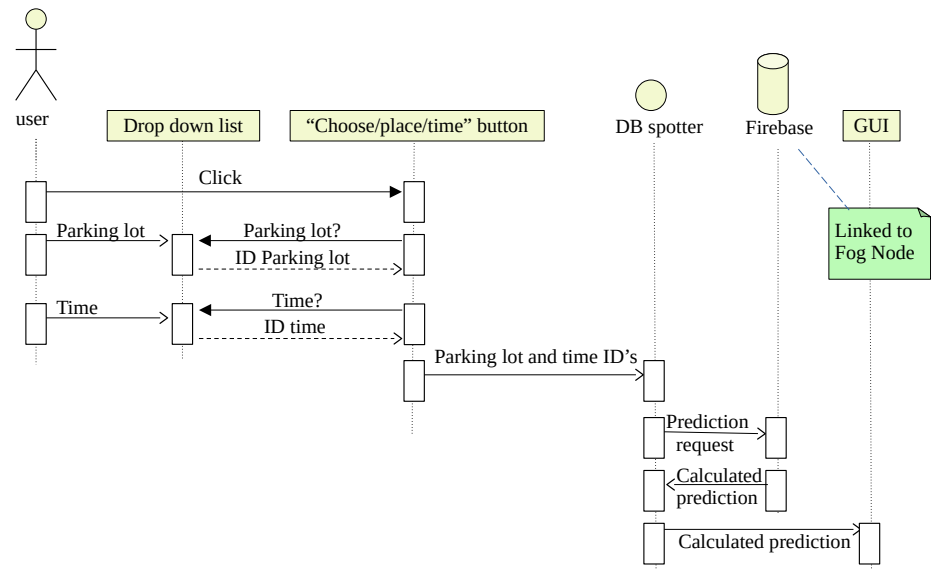


Figure 7. Prediction request sequence diagram.

Finally, the application has a button to align the current location (Figure 11) with which the application constantly centers the map on the current location; when pressing the alignment button, the icon will change (Figure 11a), and the position of the map will remain where the user manually locates it, returning to its original behavior of centering the current location when the alignment button is pressed again and returning to the icon shown in Figure 11b.



Figure 8. Application functionality for saved location; after pressing (a) “Save location” button, a red balloon is placed on the map for the actual position; functionality buttons (b) “Show location”, (c) “Clear location”, and (d) “Trace route” are enabled.

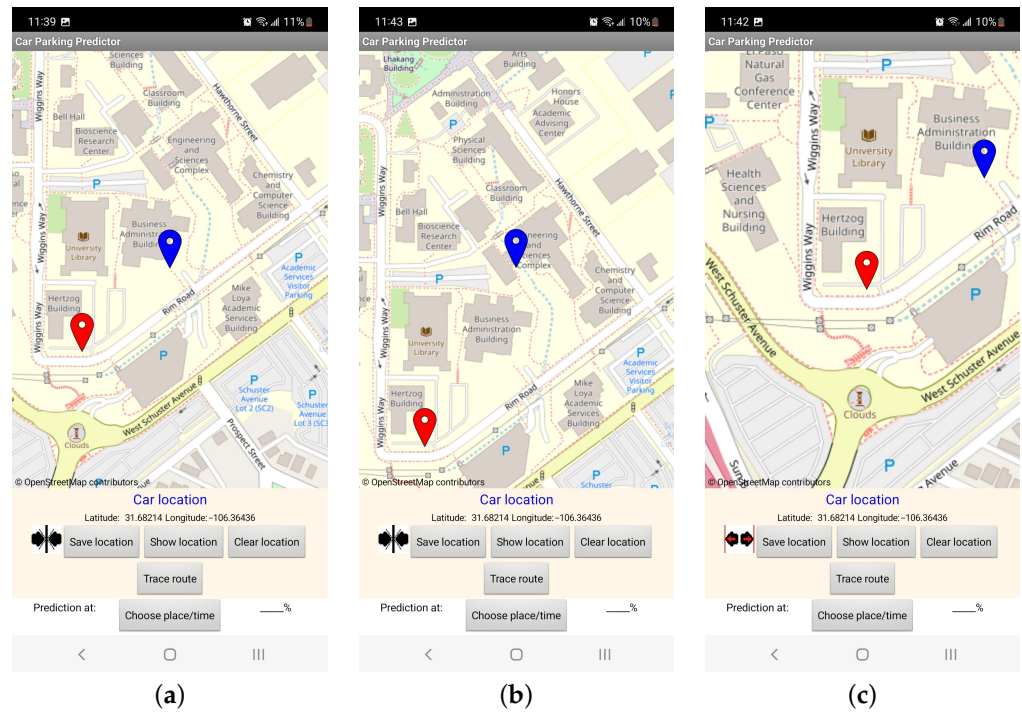


Figure 9. Walking away (blue globe) from the saved location car (red globe), (a) the blue globe is centered on the map and following the pedestrian, (b) the pedestrian moved again, the blue globe is centered on the map for every pedestrian move; (c) the saved location car (red globe) is centered on the map after clicking “Show location” button.

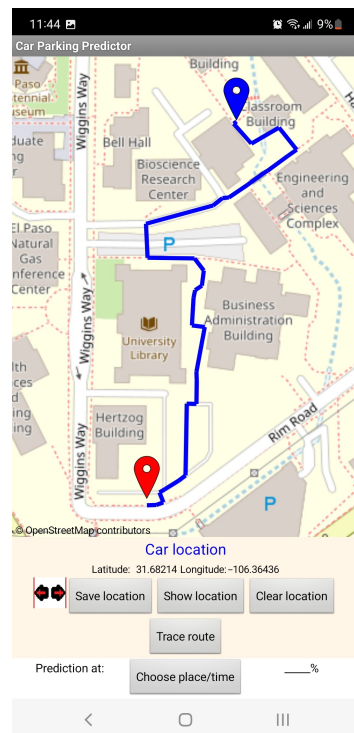


Figure 10. Traced route to reach the saved location.

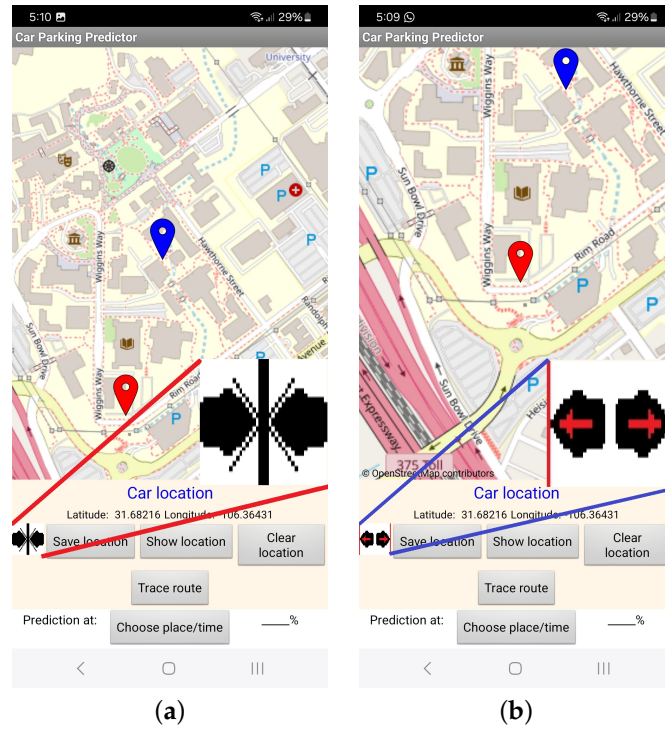


Figure 11. Button to select the current central location option within the map. (a) Button not selected. (b) Button selected.

3.3. Metrics

We used several metrics to evaluate the performance of the predictor. We assume a dataset of N observations, and the metrics are described below. The R^2 score, or coefficient of determination, is the percentage of the variation in the dependent variable that is predictable from the independent variable [26], and can be calculated as

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{7}$$

where \hat{y}_i represents the prediction for data point y_i , \bar{y} denotes the mean of all data points, and the summation extends over N , the total number of observations.

The maximum error (MaxError), is simply the maximum of the residuals, defined as

$$\text{MaxError} = \max_{\hat{y}_i} (|y_i - \hat{y}_i|) \tag{8}$$

The mean absolute error (MAE) measures the errors between the data observed and the prediction

$$\text{MAE} = \frac{1}{N} \sum_i |y_i - \hat{y}_i| \tag{9}$$

The median absolute error (medianAE)

$$\text{medianAE} = \text{median}(|y_i - \hat{y}_i|) \tag{10}$$

The root mean square error

$$\text{RMSE} = \sqrt{\frac{\sum_i (y_i - \hat{y}_i)^2}{N}} \tag{11}$$

4. Results

For this work we used the database of ref. [27] for training and testing of the subsystem. The database reports the capacity of a given parking lot and consists of the following fields: parking lot system, capacity, occupancy, date, and time. The size of the database is 35,717 entries divided in 30 parking lots. The period of data collection for most parking lots is from 4 October 2016 to 19 December 2016; however, some parking lots have a more reduced interval. The total number of data points collected per parking lot is shown in Figure 12a. In the database, parking data were acquired every 30 min; however, we use the data per day, as shown in Figure 12b.

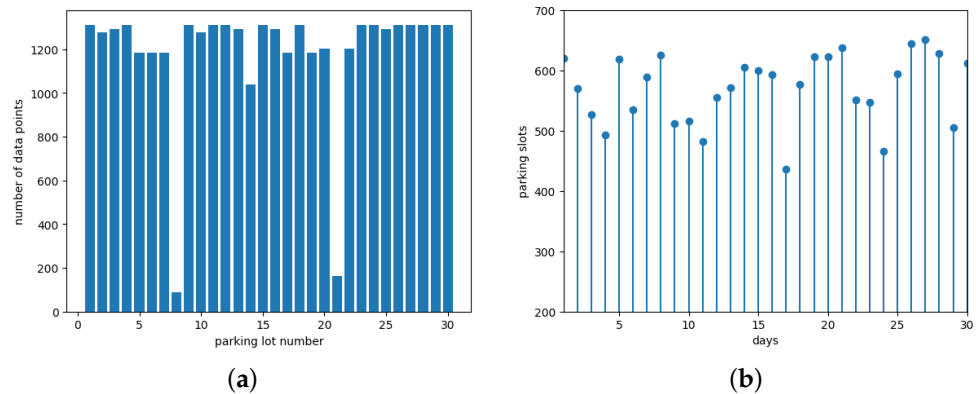


Figure 12. (a) Number of data points per parking lot. (b) A sample of 30 days for a parking lot, showing the parking slots used per day.

For prediction of parking spaces, we used several ML algorithms: AdaBoost, XGB, lasso, random forest, K-neighbors, support vector machine, and kernel ridge regression, and we denote by TAdaBoost the modified AdaBoost algorithm by the use of Equation (5). All algorithms were trained using a split of the data: 80% for training and 20% for testing. We used 315 days of the database, and thus 63 days were used for testing. In Table 1 are shown the values used for the most common parameters in each algorithm. For a complete list of the parameters and the value used, please see scikit-learn library documentation in default values [28].

Table 1. Parameters for the considered methods.

Algorithm	Parameters		
AdaBoost	estimators = 550	loss = quadratic	
XGB	estimators = 500	max depth = 4	
Lasso	alpha = 1.0		
Random Forest	estimators = 100	split quality = squared error	
K-Neighbors	neighbors = 5	distance metric = minkowski	
SVR	kernel = Radial Basis	C = 1.0	
Kernel Ridge	alpha = 1	kernel = linear	
TAdaBoost	estimators = 270	loss = linear	c = 2.1

The evaluation of the algorithms using various metrics is presented in Table 2. According to the results, the TAdaBoost algorithm performs better than all other methods in terms of R2 score, MAE, and RMSE. However, it has a higher MaxError than the other methods. AdaBoost has the best MedianAE, with 10.5, which is 3.099 units lower than the next best method, TAdaBoost. However, TAdaBoost has a consistently low error for most of the predictions.

The curves in Figure 13 display a portion of the occupancy data series obtained from the database; only the best three methods are shown. It is apparent that the TAdaBoost algorithm fits the actual occupancy curve more accurately. The data show a cyclic pattern that deviates around days 25–30, where all methods demonstrate poor performance. Also,

on day 26, most methods predict an increase that the actual data do not show, but the TAdaBoost and the AdaBoost algorithms do not make this error and follow the actual trend.

Table 2. Evaluation metrics.

Algorithm	R ² Score	MaxError	MAE	MedianAE	RMSE
TAdaBoost	0.518	130.914	25.095	11.8	1811.6
AdaBoost	0.505	128.001	28.194	10.5	1996.6
XGB	0.420	123.850	35.307	19.9	2276.7
Lasso	0.409	118.322	37.179	22.2	2168.0
Random Forest	0.397	130.860	37.5	26.100	2201.6
K-Neighbors	0.244	132.400	42.6	31.200	2777.4
SVR	−0.133	167.753	46.090	22.1	4147.7
Kernel Ridge	−7.857	291.064	186.5	175.430	36,449.8

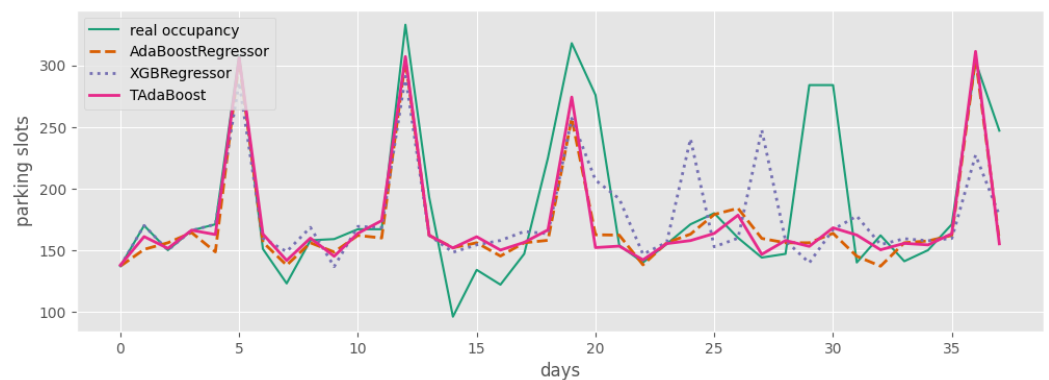


Figure 13. A portion of the occupancy data series obtained from the database and the prediction of the different methods against the real data.

Figure 14, shows boxplots of the four best methods. The errors between prediction and actual measures were used as data. Most methods have a median of zero error; however, the TAdaBoost method presents less variance but presents more outliers than the second best, AdaBoost.

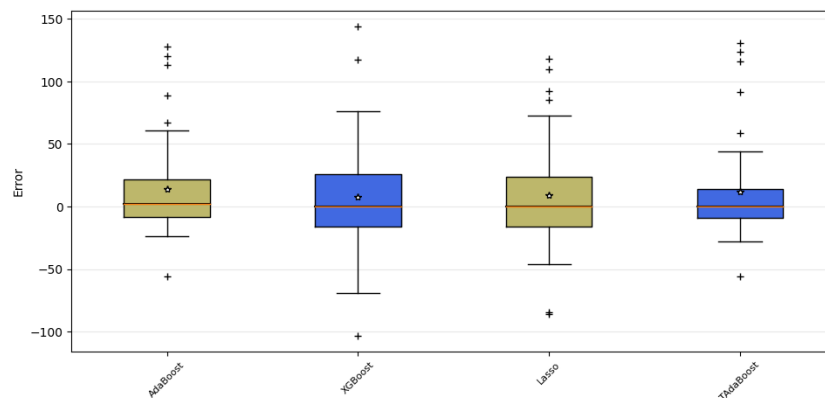


Figure 14. Boxplots for the different methods.

We simulate a network composed of a variable number of cellular phone requests to the server with fog and only cloud. The overall end-to-end delay of the system is approximated as the sum of the nodal processing delay, D_{drop} , queuing delay, D_{queue} , serialization delay, D_{ser} , and propagation delay, D_{prop} [29]. The advancements in hardware and software have decreased the processing and serialization delay to microseconds, and the propagation delay is around 5 microseconds per kilometer. The queuing delay can be optimized using quality of service (QoS) techniques for prioritized data.

In the simulation, the nodes' distance, d_n , was dynamically changed within a 10 km diameter to more realistically simulate mobile devices moving with respect to fixed fog servers. The latency calculations were performed using the following parameter values: $D_{prop} = 100e - 6 * d_n$, $D_{proc} = 1.6E - 3$, and $D_{queue} = 1.5E - 3 + r * 0.70E - 3$, where r is a random variable between 0 and 1; note that D_{ser} was not considered in this work.

Figure 15 displays the delay curve, which shows that fog computing reduces the overall delay. The reduction is not significant with fewer than 100 nodes, and, when there are fewer than 50 mobile phones, the delay is negligible, less than a second. However, as the number of requests increase, the delay with fog computing is almost half of that with cloud-only computing.

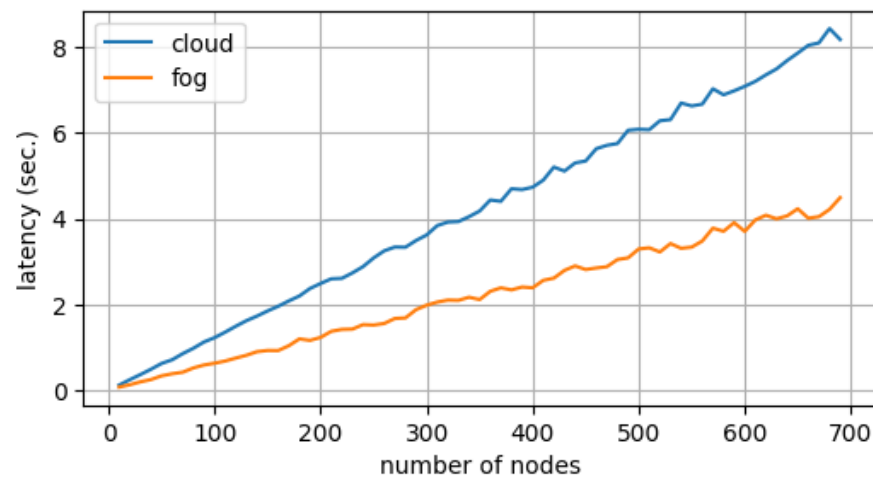


Figure 15. Latency as the number of nodes increase.

Finally, with regard to study design, we assess both the internal validity and external validity of our study. Internal validity, in our context, focuses on two primary threats. Firstly, it pertains to the suitability of the dataset for training the algorithms. We have chosen a database containing 30 parking lots, for providing data for algorithm training. Secondly, it relates to the comparison of selected machine learning algorithms. To address this concern, we have opted for the most successful techniques commonly employed in the literature. External validity, in the context of our study, revolves around result generalization. There is a risk that the machine learning model may not perform effectively when applied to a parking lot not included in the database. However, we have implemented two strategies to mitigate this risk. Firstly, we selected machine learning algorithms known for their ability to recognize significant patterns in data, especially over extended periods, such as in time series analysis. These algorithms typically offer better generalization compared with more traditional methods like simple regression. Consequently, we anticipate that other parking lots will exhibit similar patterns to those on which the algorithm was trained. Secondly, even if generalization to other parking lots proves challenging, our fog nodes incorporate a retraining module. This module allows the algorithm to adapt to specific parking lots in the event of significant error levels, ensuring continued performance optimization.

5. Conclusions

In this study, we proposed a fog computing architecture that employs a ML algorithm based on a modified AdaBoost to forecast parking slot availability in advance. We used Tukey's biweight to calculate the output of the AdaBoost algorithm, denoted in this paper as TAdaBoost. To enable interaction with the fog nodes, a mobile application was developed. We evaluated the performance of the prediction model using a real-world database of parking lot usage. Our findings indicated that the TAdaBoost algorithm outperformed other methods in terms of R2 score, MAE, and RMSE. The simulation tests showed that the response delay was reduced, as the number of nodes in the net increased, when using

the fog modules in conjunction with the cloud, compared with using the cloud only, as shown by the performance curves in Figure 15. Our results indicate that integrating ML algorithms based on fog computing architecture can be an effective approach to solving complex prediction problems in real-world applications. Future work is suggested to focus research on adding more services to the fog and cloud servers, such as the ability to display real-time images of the parking lot occupancy status.

Author Contributions: Conceptualization, G.B. and J.-M.M.-M.; methodology, F.J.E., G.B., J.-M.M.-M. and O.C.-M.; software, G.B. and F.J.E.; formal analysis, O.C.-M. and J.-M.M.-M.; research, G.B. and F.J.E.; resources and data collection, J.-M.M.-M. and G.B.; writing, G.B., O.C.-M. and F.J.E.; validation, G.B., O.C.-M. and F.J.E. All authors read and approved the final manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The supporting data can be provided on request.

Acknowledgments: The authors would like to thank the Autonomous University of Ciudad Juarez (UACJ) for its research facilities. This work was partially supported by UNAM-PAPIIT IN303523. We declare the use of AI, ChatGPT, for grammar and spelling review. The methodology, analysis, originality, validity, and integrity were carried out exclusively by the authors. No AI was used for such a purpose.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ML	machine learning
IoT	the Internet of Things
PAC	probably approximately correct
GUI	graphical user interface
MAE	mean absolute error
medianAE	median absolute error
RMSE	root mean square error

References

1. Ratli, M.; El Cadi, A.A.; Jarboui, B.; Artiba, A. Dynamic assignment problem of parking slots. In Proceedings of the 2019 International Conference on Industrial Engineering and Systems Management (IESM), Shanghai, China, 25–27 September 2019; pp. 1–6.
2. Caliskan, M.; Barthels, A.; Scheuermann, B.; Mauve, M. Predicting parking lot occupancy in vehicular ad hoc networks. In Proceedings of the 2007 IEEE 65th Vehicular Technology Conference-VTC2007-Spring, Dublin, Ireland, 22–25 April 2007; pp. 277–281.
3. Paidi, V. Short-term prediction of parking availability in an open parking lot. *J. Intell. Syst.* **2022**, *31*, 541–554. [[CrossRef](#)]
4. Muratori, M.; Alexander, M.; Arent, D.; Bazilian, M.; Cazzola, P.; Dede, E.M.; Farrell, J.; Gearhart, C.; Greene, D.; Jenn, A.; et al. The rise of electric vehicles—2020 status and future expectations. *Prog. Energy* **2021**, *3*, 022002. [[CrossRef](#)]
5. Mansour, M.B.M.; Said, A.; Ahmed, N.E.; Sallam, S. Autonomous parallel car parking. In Proceedings of the 2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, UK, 27–28 July 2020; pp. 392–397.
6. Yang, S.; Ma, W.; Pi, X.; Qian, S. A deep learning approach to real-time parking occupancy prediction in transportation networks incorporating multiple spatio-temporal data sources. *Transp. Res. Part C Emerg. Technol.* **2019**, *107*, 248–265. [[CrossRef](#)]
7. Shen, X.; Lacayo, M.; Guggilla, N.; Borrelli, F. Parkpredict+: Multimodal intent and motion prediction for vehicles in parking lots with cnn and transformer. In Proceedings of the 2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC), Macau, China, 8–12 October 2022; pp. 3999–4004.
8. Feng, Y.; Xu, Y.; Hu, Q.; Krishnamoorthy, S.; Tang, Z. Predicting vacant parking space availability zone-wisely: A hybrid deep learning approach. *Complex Intell. Syst.* **2022**, *8*, 4145–4161. [[CrossRef](#)]
9. Zhao, Z.; Zhang, Y.; Zhang, Y. A comparative study of parking occupancy prediction methods considering parking type and parking scale. *J. Adv. Transp.* **2020**, *2020*, 1–12. [[CrossRef](#)]

10. Mehdipour, F.; Javadi, B.; Mahanti, A. FOG-Engine: Towards big data analytics in the fog. In Proceedings of the 2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), Auckland, New Zealand, 8–12 August 2016; pp. 640–646.
11. Hu, P.; Dhelim, S.; Ning, H.; Qiu, T. Survey on fog computing: Architecture, key technologies, applications and open issues. *J. Netw. Comput. Appl.* **2017**, *98*, 27–42. [[CrossRef](#)]
12. Karthika, P.; Ganesh Babu, R.; Karthik, P. Fog computing using interoperability and IoT security issues in health care. In *Micro-Electronics and Telecommunication Engineering: Proceedings of the 3rd ICMETE, Ghaziabad, India, 30–31 August 2019* 2019; Springer: Berlin/Heidelberg, Germany, 2020; pp. 97–105.
13. Xu, Z.; Tang, X.; Ma, C.; Zhang, R. Research on parking Space Detection and Prediction Model based on CNN-LSTM. *IEEE Access* **2024**, *12*, 30085–30100. [[CrossRef](#)]
14. Lyu, M.; Ji, Y.; Kuai, C.; Zhang, S. Short-term prediction of on-street parking occupancy using multivariate variable based on deep learning. *J. Traffic Transp. Eng.* **2024**, *11*, 28–40. [[CrossRef](#)]
15. Zhang, F.; Shang, K.; Yan, L.; Nan, H.; Miao, Z. Prediction of Parking Space Availability Using Improved MAT-LSTM Network. *ISPRS Int. J.-Geo-Inf.* **2024**, *13*, 151. [[CrossRef](#)]
16. Huang, Y.; Dong, Y.; Tang, Y.; Li, L. Leverage Multi-source Traffic Demand Data Fusion with Transformer Model for Urban Parking Prediction. *arXiv* **2024**, arXiv:2405.01055.
17. Haussler, D.; Warmuth, M. The probably approximately correct (PAC) and other learning models. In *Foundations of Knowledge Acquisition: Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1993; pp. 291–312.
18. Freund, Y.; Schapire, R.E. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.* **1997**, *55*, 119–139. [[CrossRef](#)]
19. Schapire, R.E. Explaining adaboost. In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–52.
20. Tanwar, S.; Vora, J.; Kaneriya, S.; Tyagi, S.; Kumar, N.; Sharma, V.; You, I. Human arthritis analysis in fog computing environment using Bayesian network classifier and thread protocol. *IEEE Consum. Electron. Mag.* **2019**, *9*, 88–94. [[CrossRef](#)]
21. Balevi, E.; Gitlin, R.D. Optimizing the number of fog nodes for cloud-fog-thing networks. *IEEE Access* **2018**, *6*, 11173–11183. [[CrossRef](#)]
22. Uria, B.; Côté, M.A.; Gregor, K.; Murray, I.; Larochelle, H. Neural autoregressive distribution estimation. *J. Mach. Learn. Res.* **2016**, *17*, 7184–7220.
23. Mosteller, F.; Tukey, J.W. Data analysis and regression. A second course in statistics. In *Addison-Wesley Series in Behavioral Science: Quantitative Methods*; Addison-Wesley Publishing Company: Reading, MA, USA, 1977.
24. Chatterjee, N.; Chakraborty, S.; Decosta, A.; Nath, A. Real-time communication application based on android using Google firebase. *Int. J. Adv. Res. Comput. Sci. Manag. Stud* **2018**, *6*; pp. 74–79.
25. Albertengo, G.; Debele, F.G.; Hassan, W.; Stramandino, D. On the performance of web services, google cloud messaging and firebase cloud messaging. *Digit. Commun. Netw.* **2020**, *6*, 31–37. [[CrossRef](#)]
26. Taylor, R. Interpretation of the correlation coefficient: A basic review. *J. Diagn. Med. Sonogr.* **1990**, *6*, 35–39. [[CrossRef](#)]
27. Stolfi, D.H.; Alba, E.; Yao, X. Predicting car park occupancy rates in smart cities. In Proceedings of the Smart Cities: Second International Conference, Smart-CT 2017, Málaga, Spain, 14–16 June 2017; Proceedings 2; Springer: Berlin/Heidelberg, Germany, 2017; pp. 107–117.
28. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
29. Li, J.; Zhang, T.; Jin, J.; Yang, Y.; Yuan, D.; Gao, L. Latency estimation for fog-based internet of things. In Proceedings of the 2017 27th International Telecommunication Networks and Applications Conference (ITNAC), Melbourne, VIC, Australia, 22–24 November 2017; pp. 1–6.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.