Intelligent Automation & Soft Computing
DOI: 10.32604/iasc.2023.035605
Article

Tech Science Press

IOT Based Smart Parking System Using Ensemble Learning

Walaa H. Elashmawi[1,3], Ahmad Akram[2], Mohammed Yasser[2], Menna Hisham[2], Manar Mohammed[2], Noha Ihab[2] and Ahmed Ali[4,5,*]

[1]Department of Computer Science, Faculty of Computers and Informatics, Suez Canal University, 4.5 Km the Ring Road, Ismailia, Ismailia, 41522, Egypt
[2]Department of Computer Engineering and Software Systems, Faculty of Engineering, Ain Shams University, Al-Abbaseya, 4543070, Egypt
[3]Department of Computer Science, Faculty of Computer Science, Misr International University, 28 KM Cairo–Ismailia Road, Cairo, 44971, Egypt
[4]College of Computer Engineering and Sciences, Prince Sattam Bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia
[5]Higher Future Institute for Specialized Technological Studies, Cairo, 3044, Egypt
*Corresponding Author: Ahmed Ali. Email: a.abdalrahman@psau.edu.sa
Received: 27 August 2022; Accepted: 13 December 2022

Abstract: Parking space is usually very limited in major cities, especially Cairo, leading to traffic congestion, air pollution, and driver frustration. Existing car parking systems tend to tackle parking issues in a non-digitized manner. These systems require the drivers to search for an empty parking space with no guarantee of finding any wasting time, resources, and causing unnecessary congestion. To address these issues, this paper proposes a digitized parking system with a proof-of-concept implementation that combines multiple technological concepts into one solution with the advantages of using IoT for real-time tracking of parking availability. User authentication and automated payments are handled using a quick response (QR) code on entry and exit. Some experiments were done on real data collected for six different locations in Cairo via a live popular times library. Several machine learning models were investigated in order to estimate the occupancy rate of certain places. Moreover, a clear analysis of the differences in performance is illustrated with the final model deployed being XGboost. It has achieved the most efficient results with a R^2 score of 85.7%.

Keywords: IoT; XGBoost; linear regression; random forest; ensemble learning; isolation forest

1 Introduction

With Cairo's current population approaching 22 million, the parking space problem is growing. It has been assessed that finding a free parking spot in busy areas in Cairo could take more than 20 min on average [1]. The impact of this problem is significant on traffic congestion and air pollution. Vehicles cruising for free parking spaces are estimated to cause a significant proportion of daily traffic congestion in an urban downtown area (between 9 and 56 percent according to [2]), as well as a proportion of CO_2 emissions. Furthermore, driver frustration is a major consequence of such a problem, with drivers

often resorting to the use of the parking spots reserved for people with special needs, such as the disabled, or just double parking on the main streets, preventing drivers from leaving their parking spaces, stalling the flow of traffic, and narrowing the streets. As a result, there are fewer parking spaces available, which is one of the most common issues affecting traffic flow on the roads. Finding parking space has become one of the most difficult challenges to effectively utilizing parking space.

The aforementioned issues are resolved by implementing a smart parking system that enables easy access to available parking spaces while also assisting in reducing traffic congestion. This can be done by incorporating new technologies to provide an excellent solution, saving significant time and resources for drivers, especially in heavily frequented areas.

Over the past several years, many high-tech solutions have been created to make it easier for the driver to locate open parking spaces efficiently and quickly [3]. This leads to the new capital to be developed with the strategic vision of a smart city, integrating smart infrastructure to provide many services to citizens, dodging the problems already present in the old capital; one of these problems is parking [4]. The automation of the user experience in parking can be achieved through the implementation of a sophisticated algorithm to keep the driver adequately informed about the availability of parking spaces and allow him to reserve a parking space via a mobile app prior to reaching the parking destination. The goal of this study is to ease the parking experience for the driver. This can be achieved by the use of various technologies such as the Internet of Things (IoT), Quick Response codes (QR), and Machine Learning (ML) algorithms. The concept of IoT first started as devices that connect and exchange data with other devices and systems over the Internet or other communications networks [5]. Moreover, the flexibility and scalability of cloud computing allow developers to create and host their applications on it. The cloud acts as a perfect partner for IoT as it acts as a platform where all the sensor data can be stored and accessed from remote locations [5].

The (QR) codes are described as a gateway to the IoT. It can be used by users on mobile phones themselves to scan anything or to purchase any product [6]. QR codes are two-dimensional matrix barcodes that have to be scanned, they are a combination of white and black square patterns and have an encoding and functioning region. Recently, QR codes have been widely applied in various domains, such as social media, learning, marketing, and more. It can be applied in smart parking, where an automatic parking system was suggested by the authors of [7]. The use of QR code technology will make it more convenient for users overall.

Nowadays, the use of machine learning (ML) techniques in a variety of real-world application areas, including cybersecurity systems, smart cities, healthcare, e-commerce, agriculture, and many more, has gained a lot of attention. In particular, it can be applied to large datasets to extract relevant information for the prediction of parking occupancy. The performance and results of machine learning algorithms depend on data and model. Given the variety of ML methods, one technical problem is to identify the most suitable model for the problem and the dataset, as the performance of each model varies from problem to problem and dataset to another.

Integration of two or more technologies leads to better management of smart parking systems. In [8], IoT and ML are combined into a single system to improve parking system problems.

To gain the advantages of the above technologies, this study is directed towards integrating different technologies that can be used in parking solutions, of which proposing an IoT-based smart parking system is a main contribution. It is based on an algorithm for automated parking systems and their implementation into a software package that is connected to hardware devices for this purpose with respect to some features as follows.:

- QR Code is generated on driver's device when user books a parking slot. At the entrance gate, the reader directly scans the QR code and verifies the details, authenticates the user, and opens the

gates right away. Moreover, it saves the timestamp of the user's entrance to be used later when leaving the parking to calculate parking fees automatically.

- Mobile app is used by user and admin to collect information about the occupancy state of parking spaces, inform the drivers to the nearest vacant parking spot, and handle payments of parking service. The entering into or leaving the parking slot is controlled by an application.
- Machine Learning (ML) model. In this paper, the proposed model uses parking events and availability data collected from some shopping malls, the data is analyzed, resulting in the selection of some key parameters to form three feature sets to use as the input for the parking availability prediction models that are derived. The main goal of this model is to predict the availability of parking spaces in advance and help drivers be informed of available spaces which will result in controlling traffic congestion.

Accordingly, this paper presents a design and development of a prototype system that consists of a backend system and a mobile client app. The system can provide information about parking availability based on real-time sensor data through a user-friendly mobile app interface that may direct users in a quick and effective manner to complete an online parking reservation for a certain amount of time.

The rest of the paper is organized as follows: Sections two and three describe the related work and the architecture of the proposed system, respectively. Section four discusses the experimental results and performance evaluation. Section five concludes the paper, and at last, section six suggests some future directions towards improving the smart parking system.

2 Related Work

Several review articles that explore and discuss the technical aspects and features of smart parking solutions have been published. Moreover, some commercial applications for smart parking systems already exist and have been used for years.

The authors in [9] have implemented two android applications (i.e., one for the admin side and another for the user side). The admin application is responsible for creating a parking garage entity. Furthermore, it generates a QR code that encodes the garage's data. While the user application allows pre and on-the-spot booking, this process can be done as follows: (1) the user scans the location QR code upon arrival; (2) the app displays the user's phone number and vehicle number for verification; (3) a list of current available spots is displayed; (4) the user selects a slot name; and (5) the slot is reserved. Their model is based on the use of IR sensors, servo motors, and LEDs interfaced with the Raspberry Pi. Although this system relies on on-the-spot booking, it doesn't automate all the parking processes, such as automatic payment or real-time availability of parking lots.

While the authors in [5] proposed an online parking spot management system on campus with two main features (i.e., monitoring and booking capabilities) through accessing a website. However, this system uses an ultrasonic sensor to detect objects instead of infrared sensors (IR). It has some limitations. The first one is a 400-meter radius around the parking lots, which is the only area where the booking option is available. The second limitation is the delay time between sending and receiving the data through the sequential process of the system. It also lacks the automation of the opening-closing gate mechanism, in which these limitations are tackled and solved in the proposed model. Another parking system is proposed in [10,11], which is applied to the university level.

A parking-spot selection algorithm is proposed in [12] to minimize the arrival time. It is based on a search graph which is initiated upon the entry of a vehicle into the parking lot. Based on their experiments, the arrival time decreased by around 20%–60%. The parking availability prediction was implemented using a Markov model. However, the data obtained, and the final accuracy of the model were not tested. In the proposed model, the data is tested on various machine learning algorithms to test

the accuracy of the prediction service. In [13], a non-homogeneous Markov chain model was used to predict the parking space availability. It is reasonable to state that the model accurately captures how the parking lot is used with the highest MAPE (mean absolute percentage error) of 17.3%, despite the fact that the model data is only based on the week's traffic survey.

Other papers focused on determining the occupancy rate of a location with the use of various machine learning algorithms for prediction. In [10], the authors tested their system by using three feature sets on three machine learning approaches (i.e., support vector regressor (SVR), regression trees, and a neural network (NN)). The feature sets were as follows: (1) the hour and day of the week; (2) previous occupancy rate observations; and (3) a combination of feature sets (1) and (2). It was applied to two datasets collected for parking locations in two cities, San Francisco, and Melbourne. Their experimental results were acquired, and the regression tree model has proven the most efficient, especially with the third feature set on the San Francisco dataset, 98.6% mean r^2. The authors in [14] suggested a parking segment prediction where different ML and neural network algorithms (i.e., K-Nearest Neighbors (KNN), Decision Tree (DT), and Random Forest (RF)) are applied to measure the parking availability in San Francisco. According to the proposed system in this paper, the experiments are done on six locations in the city which are the densest and popular, and the occupancy rate prediction is evaluated on the three popular ML models, which are linear regression, random forest, and XGBoost.

In [15], the authors proposed a car parking prediction system based on Deep Extreme Machine Learning. They put their model to the test using a car parking dataset from [16]. While the authors of [17] combined IoT with a predictive model based on ensemble methods to improve the prediction of a parking spot using the Birmingham parking data set.

Moreover, it has been observed that multiple mobile applications are up and running in developed countries. Two of these applications are SmartPark [18] and EasyPark [19]. Both apps use Automatic License Plate Recognition (ALPR) technology on the gates, GPS navigation, allow cashless payment, and display parking history. SmartPark offers real-time parking availability tracking while EasyPark offers occupancy rate prediction. Another mobile application that allows users to easily book a parking space in advance is proposed in [20] by monitoring online whether the space is available or not and working on an application for booking a parking spot with the online payment method.

Although there are several strategies and approaches available to address the parking system issue, the existing smart parking systems still face numerous deployment, connection, and architectural limitations. As a result, this study proposes an IoT-based smart parking system that uses ensemble learning to predict occupancy rate.

3 Materials and Methods

The proposed IoT-based smart parking system maintains and monitors parking lots to improve the ease and comfort of vehicle parking while reducing traffic and travel time. The system overview of the proposed model is shown in Fig. 1. It is composed of four subsystems: hardware devices; gate authentication; machine learning; and reservation. The hardware devices reside on-site in each parking lot. They collect data that corresponds to changes in the environment. This data is then communicated to Firebase in real-time. On the mobile client side, the real-time data on parking availability is fetched from Firebase. As for the gate authentication, the system consists of a QR generator that resides in the mobile application and a QR scanner application that exists at the entry and exit points of the parking lot. The QR scanner is responsible for saving the entry and exit timestamps to the database, and fee calculation is handled accordingly. Regarding Machine Learning, multiple algorithms are studied, of which one is chosen and built into a pipeline using Google Cloud functions and deployed using Google Vertex AI. Finally, the

reservation begins on the mobile application. The application then writes the reservation data to Firestore, and a cloud function is triggered to handle no-show reservations.

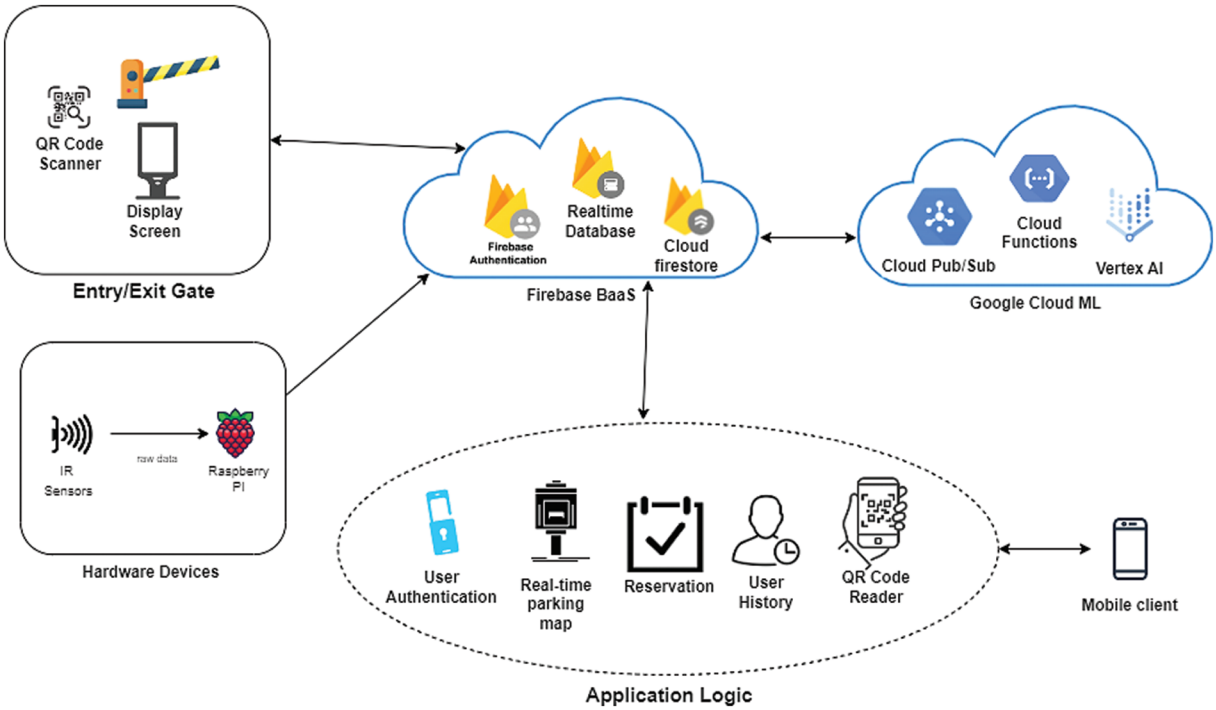


Figure 1: System Overview

Fig. 1 illustrates the proposed system of the IoT-based smart parking system.

The proposed IoT smart parking system's overall flow may be summed up as follows:

- The user starts by reserving a spot using the mobile application.
- The mobile application allows reserving a spot for one hour in advance.
- The machine learning model predicts the availability of the garage in the next hour.
- The user confirms the reservation, and a parking spot will be reserved for the next hour.
- Upon arriving at the entry gates, the user generates a QR code from the mobile application.
- The user then scans the QR code at the gate's portal.
- The user's entry timestamp will be stored in the database.
- The portal will then display the map of the garage, and the gates will then be opened.
- The same steps are repeated at the exit gates.
- When exiting, the user's exit timestamp will be stored in the database.
- The cloud functions will then be triggered, and the fees will be calculated using the entry and exit timestamps.
- The fees will automatically be deducted from the user's credit card.
- The transaction will be saved in the database and the user can view the history from the mobile application.

The following is a comprehensive overview of each of the subsystems.

3.1 Hardware Devices

The proposed hardware subsystem consists of a mobile device, proximity sensors, a barrier gate system, and a Raspberry Pi as described below.

3.1.1 Mobile Device

Users will have to have a mobile device and install the application (which will be available for both Android and IOS) to be able to use the system. Having a mobile device is crucial to reserving a spot, entering, and exiting the gates, and automatically performing a successful payment [21].

3.1.2 Proximity Sensors

Proximity sensors will be embedded under each parking spot in the garage to give the user a real-time view of the garage's occupancy in the application. IR sensors were used in the created physical scale model as they are small in size and easier to connect, but when it comes to real-life application, IR sensors are restricted as they are affected by sunlight, dust, fog, and rain, which may be present, especially in an open-air garage. So, when it comes to real-life applications ultra-sonic sensors will be used, which are cheap, easy to implement and are not affected by weather conditions [22].

3.1.3 Barrier Gate System

The gate system consists of an embedded camera and a screen. The user will use the camera to scan the QR code generated in the mobile application to be able to enter. On successful scanning, the user will be confirmed with a voice confirmation, and then the screen will display the map view of the garage with the available parking spots and the gate system will be opened to the user.

3.1.4 Raspberry PI

The Raspberry PI 4 model B was used to check all the connected sensors and identify the occupied and free spots. This collected data will be sent to the database servers to be stored and further processed. The Raspberry Pi also provides a range of serial communication protocols such as UART, I2C, and SPI. These protocols are used to interface and communicate with hundreds of peripherals/devices using only two, three, or four channels of a serial port, which will be very handy in the proposed system due to having tens of parking spots in the garage that need to be connected to the network [23].

3.2 Gate Authentication

QR codes are used to render information, encode it using a cryptographic algorithm, facilitating user access with their own smart phones, tablets, and other mobile devices to the references hidden behind the code [6]. Given the popularity enjoyed by these codes in Egypt; the proposed system will utilize QR codes to facilitate creating an easy, secure, and reliable method of authentication for the average Egyptian user. The system needs to detect each user's entry and exit times into and out of the parking lot [24]. As shown in Fig. 2, the system is divided into two parts, one being the QR code generator, which will be located in the mobile app of the user, and the other being the QR reader, which will be implemented on the gates of the parking lot. Each of the QR Generator QR Scanner is further explained below.

3.2.1 QR Generator

The QR generator part is present in the mobile application. It was developed using Flutter as the rest of the application. The user can generate a QR code when reaching the gates to either enter or exit. The QR code encapsulates and ciphers the user's ID and the date, which will be used as an entry or exit timestamp later by the scanner. The QR code is also generated periodically every five seconds to ensure security.

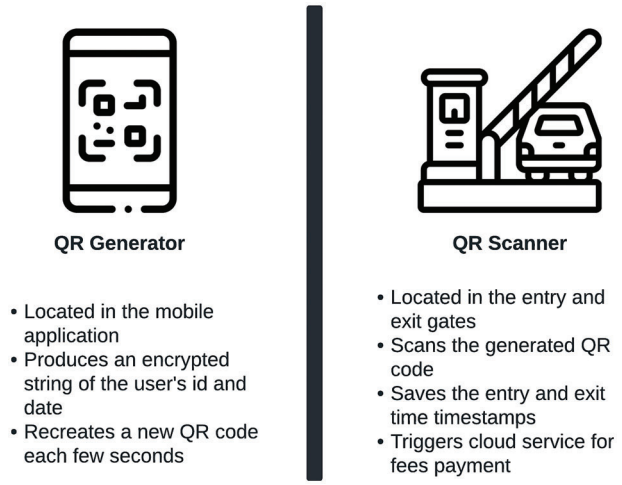


Figure 2: QR Generator *vs*. QR scanner comparison

3.2.2 QR Scanner

The QR generator part will be implemented on the entry and exit gates of the parking lot. A camera will be used to read the generated QR code. The application will then check the validity of this code and decrypt it. The code will then be split into two parts, the first being the user's id and the second being the timestamp. Then this data will be sent over to the database system in a new collection along with the garage name and the state of the car (where the entering state will be ongoing, and the exiting state will be exit). These states will allow the system to match the exit timestamp with only the ongoing record.

The first part (the user id) will be used to identify which user to access in the database system. And the second part (QR generation date) will be saved as entry/exit timestamps. This method allows sharing the account with multiple cars unless they are parking in the same garage, in which case the entry will be rejected to avoid data duplication. The system also supports text-to-speech feature which confirms the user when entering a valid QR code and warns the user when the same QR code is used more than once.

After exiting and having the entry and exit timestamps saved in the database, the cloud service will be triggered automatically, and the parking fee P_T will be calculated as per Eq. (1):

$$P_T = k + r_H * H \tag{1}$$

where k is the amount of money, r_H is the hourly rate, H is the number of hours calculated according to Eq. (2):

$$H = \left(T_{exit} - T_{entry}\right)/3600 \tag{2}$$

where T_{exit} is the Time of exit expressed in seconds, T_{entry} is the Time of entry expressed in seconds.

The fee will be saved in the database, deducted from the saved credit card, and then displayed in the user history tab in the mobile application.

3.3 Machine Learning

Nowadays, machine learning can play a vital role in various domains, especially in parking systems [25]. In the following subsections, a clear view of how machine learning algorithms can add real value to a smart parking system is discussed.

3.3.1 ML Algorithms

The goal of proposed machine learning model is to predict the occupancy percentage of a parking lot using regression, providing a set of input features [26]. During the study of this problem, three main issues were addressed:

1. Finding what features improve the model's performance by experimenting with different feature sets
2. Observing the effect of different methods for dealing with missing data on the model's evaluation metrics
3. Comparing the performance of three different regressor algorithms in predicting the target variable using multiple evaluation metrics

In this sub-section, a description of the three popular algorithms used for predicting parking availability is provided. The algorithms studied are Random Forest [27], XGBoost [28], and Linear Regression.

3.3.2 Random Forest

Random forests [29] are an improved variation of decision trees in which a huge number of de-correlated trees are built and then averaged. In random forests, only a random subset of features is considered for each split of a node, and from this subset, the best split feature is selected. As a result, this tweak in bagging decorrelates the individual trees and reduces variance [30]. The output value from a random forest regressor is described by the following equation:

$$\frac{1}{B}\sum_{b=1}^{B} T(x;\ \Theta_b) \tag{3}$$

where B is the number of trees, $T(x;\ \Theta_b)$ is the output of a tree b, x is the input data point, and Θ_b characterizes the b^{th} random forest tree in terms of split variables, cut-points at each node, and terminal-node values. When choosing the best split feature, the mean squared error is used as the loss function $H(\ )$ and is calculated for each candidate feature as follows:

$$H(Q_m) = \frac{1}{n_m}\sum_{y\in Q_m}(y-\bar{y}_m)^2 \tag{4}$$

In the above equation, Q_m is the data at node m, n_m is the number of samples at node m, y is the actual value, and $\bar{y}_m$ is the predicted value. As shown in Eq. (5), the predicted value of a specific node m is formulated as the mean value of the node.

$$\bar{y}_m = \frac{1}{n_m}\sum_{y\in Q_m} y \tag{5}$$

The chosen splitting feature is the one with the lowest value for the loss function $H(Q_m)$.

3.3.3 XGBoost

XGBoost [31] short for Extreme Gradient Boosting, is an ensemble machine learning algorithm that is based on a gradient-boosting model. The scalability and efficiency of XGBoost are what differentiate it from other tree-based machine learning algorithms. The algorithm was proposed with the goal of achieving higher accuracy and lower processing time. XGBoost is a result of the evolution of tree-based algorithms over the years.

In gradient boosting, the predictions of multiple models are combined to optimize the output of the boosted model instead of optimizing the model parameters themselves. The point of gradient descent is to minimize the objective function defined as follows:

$$obj(\theta) = \sum_i^n l(y_i,\ \hat{y}_i) + \sum_{k=1}^K \Omega(f_k) \tag{6}$$

where l is the loss function between the observed value y_i, and the predicted value $\hat{y}_i$ calculated over n samples. Ω is the regularization term. k is the number of trees [32].

The hyperparameters of XGBoost are divided into General parameters, Booster parameters, and Learning task parameters [31].

3.3.4 Linear Regression

The simple concept of linear regression is an algorithm that fits a line $y = \beta_o + \beta_1 \cdot x$ to the data. Where y is the dependent variable, β_1 is the slope, β_o is the y-intercept, and x is the independent variable. The algorithm quantifies the quality of the fitted line by calculating the SSR (Sum of Squared Residuals) for the given β_o, β_1.

$$SSR = \sum_{i=1}^n (O_i - P_i)^2 \tag{7}$$

n is the number of samples, O_i is the observed value, and P_i is the predicted value.

The line is fitted according to $\min_{\beta_o,\beta_1}$: *SSR*, meaning "minimize SSR over β_o *and* β_1" [33].

This concept is evolved to accommodate a set of features $(x_1, \ldots, x_n)$ for each data point–known as the multiple linear regression phenomena.

$$y = \beta_o + \beta_1 x_1 \ + \ \beta_2 x_2 + \beta_2 x_2 \ + \ldots + \ \beta_n x_n \tag{8}$$

This representation is very versatile and its possible to fit non-linear functions like quadratic equations. However, the model assumes *(1) Linear relationship and independent observations (2) Homoscedasticity-error terms have constant variance (3) Errors are uncorrelated and normally distributed (4) Low multicollinearity* [34].

3.3.5 Isolation Forest

Isolation forest [35] is a tree-based algorithm that detects anomalies/outliers. It builds an ensemble of random trees for a given dataset and returns the anomaly score; the anomalies are points with the shortest average path length.

$$anomaly\ score = 2^{-\frac{E(h(x))}{c(n)}} \tag{9}$$

where $h(x)$ denotes to the no. of edges in a tree for point x, and $c(n)$ is a normalization constant–contamination–that is given as a parameter.

3.3.6 Model Deployment and Pipeline

The model was trained locally using the methods mentioned in the previous section. The model was deployed on Google Cloud's Vertex AI, and the prediction pipeline was done through a cloud function. This section explains the technologies used and shows the prediction pipeline.

3.4 Technologies Used

3.4.1 Vertex AI

Vertex AI [36] brings together the Google Cloud services for building ML under one unified UI and API. Vertex AI allows the training and comparison of various machine learning models. It also allows users to deploy several models to a single endpoint, which can be accessed using the API both externally and internally. The prototype model was deployed on a Vertex AI endpoint.

3.4.2 Google Cloud Scheduler

The Cloud Scheduler is a fully managed cron job scheduler. Cron jobs are scheduled units of work that are sent to targets on some recurring schedule [37]. It allows users to schedule virtually any job and automate almost everything. A cloud scheduler is used in the project to schedule and trigger cloud functions.

3.4.3 Google Cloud Pub/Sub

Cloud Pub/Sub [38] is an asynchronous messages service which helps tool send, receive, and filter events or data streams. As the name suggests, it runs as a publish-subscribe model. A publisher service publishes messages on the pub/sub topic, and subscribers receive those messages. Essentially, cloud Pub/Sub acts as a messaging middleware between services.

3.5 Prediction Pipeline and Dataflow

Fig. 3 shows the prediction pipeline. A scheduler job is scheduled every hour to publish a message to a cloud pub/sub-topic. A cloud function subscribes to the topic and is triggered by it. The cloud function preprocesses the data every hour and formats it, then sends it to the deployed model's endpoint on vertex AI to get the next-hour predictions. Once the cloud function gets the predictions, it writes them to Firebase's real-time database, where the mobile application is connected.

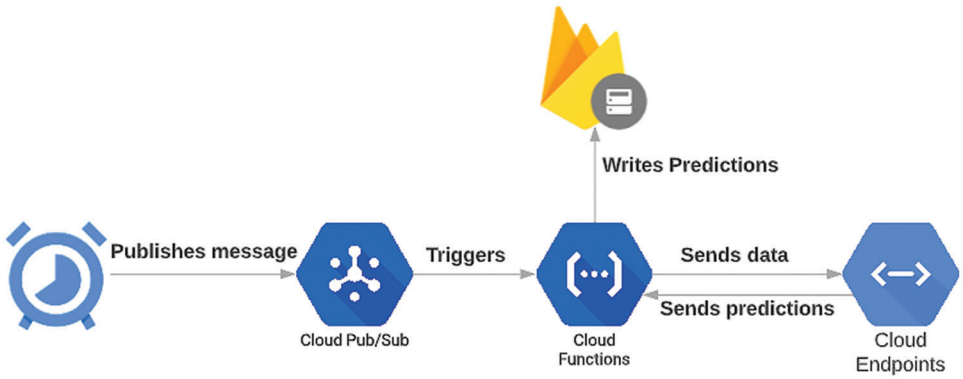


Figure 3: ML Deployment Pipeline

3.6 Reservation

The system allows the user to reserve a parking slot for up to one hour before their arrival, meaning that the user can select their arrival time between 0 and 60 min from the current time. When a user reserves the garage's capacity is decreased by 1. Reservations are automatically cancelled by a cloud function if the user does not arrive within 10 min of their selected arrival time. The system limits the user to one reservation at a time.

Fig. 4 shows the process each time a user reserves. As mentioned above, when a user makes a reservation, the garage's capacity is decremented, and the arrival time the user selects is written into the database. This write-operation triggers a firebase cloud function to schedule an http trigger for 10 min after the arrival time. The http trigger invokes another cloud function that checks if the user has arrived. If the user hasn't arrived, the reservation is cancelled, and the garage's capacity is incremented.

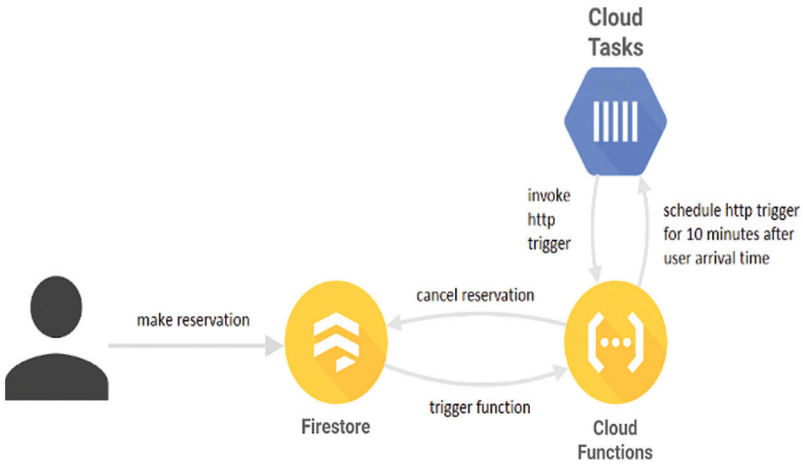


Figure 4: Reservation process

4 Results and Discussions

To validate the system methodology discussed in the previous section, all the components are implemented and evaluated. The results and inner workings of each component in the system are discussed in this section. First introducing the dataset that will be used and explaining how to prepare and specialize it for needed target. in addition to the numerous ML techniques employed for these models' assessment and prediction. The second part is about making the suggested mobile app. The app will have features that any user can use to easily find a free parking spot and reserve it.

4.1 Machine Learning Model

The goal of this sub-section is to predict the occupancy rate of a parking lot on an hourly basis in real time to better utilize parking spaces more efficiently. The output from the model represents the number of available parking spots at a given time. One of the key challenges of the proposed system implementation is the absence of local parking datasets to use as input for model training, validation, and testing. Accordingly, an alternative representation of parking availability in the chosen locations was formulated. The reasoning behind this is that the parking occupancy is directly proportional to the crowdedness of a venue. Six different locations from Cairo's most visited attraction spots are being considered as proof-of-concept locations to demonstrate the parking system. The distribution of locations is shown geographically in Fig. 5.

Figure 5: The distribution of locations

In the sub-sections below, the implementation results of the three defined algorithms are presented and compared based on different feature sets, different methods of dealing with outliers, and different approaches for handling missing values. The final output is a model that can be expressed mathematically as $\hat{Y} = F(X)$ where $\hat{Y}$ is the predicted value for the availability, X is the set of input features, F is the computed function from the model training.

4.1.1 Dataset

The dataset used consists of 8304 records collected for six different locations. The place and type of the chosen locations are shown in Table 1.

Table 1: Dataset location information

Place	Type
City Center Almaza	Shopping mall
Cairo Festival City	Shopping mall
Mall of Arabia	Shopping mall
Seoudi Waterway	Supermarket
Seoudi Maadi	Supermarket
El Serag	Electronics store

Each record consists of a set of input features X where $X_i = \{N, LT, H, dow, R, C, hol, wknd, P\}$, and target feature Y which is the corresponding observed value ranging from 0 to 100 The explanation of each input feature is illustrated in Table 2.

Table 2: Dataset structure information

SYMBOL	TYPE	MEANING
N	String	The name of the location
LT	String	Location type
H	Integer	The hour in 24 h format
dow	String	The day of the week
R	Boolean	Whether the data point was collected in the month of Ramadan
C	Boolean	Whether the place is closed at that point in time
hol	Boolean	Whether it was an official holiday in Egypt
wknd	Boolean	If that day was a weekend (Friday or Saturday) in Egypt
P	Integer	The occupancy percentage of the place during the previous hour

4.1.2 Data Collection

The dataset on which the model was tested is collected using Live Popular Times–an open-source python library that retrieves live popular times data from Google Maps [39]. It is used to scrap the current popularity of the six commonly visited places in Cairo chosen to be the case study. The current popularity retrieved is expressed as a percentage ranging from 0 to 100 to describe how active the current

location is compared to its maximum activity. The distribution of this value is visualized as a histogram in Fig. 6.

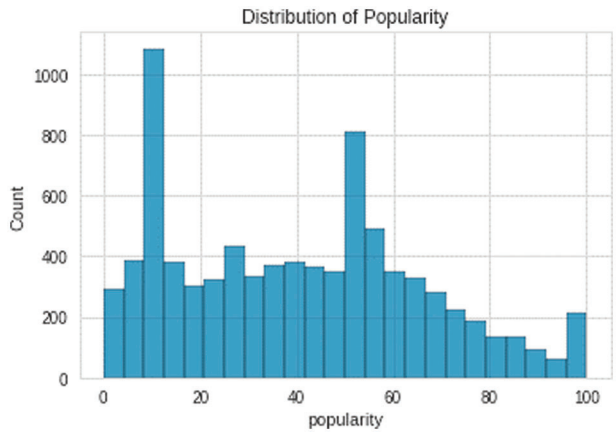


Figure 6: Popularity histogram

To automate the collection of data, a python script was scheduled to run on the Google Cloud over a period between 17 March 2022 and 28 May 2022 at a sampling rate of 1 h time intervals. Every hour, a comma-separated values csv file is created with the current date as the file name. The cloud function then retrieves the current popularity for the six locations by calling the Live Popular Times API and saves the data to the csv with each record containing the name, popularity, time, and day. This csv is then stored in a bucket on Google Cloud Storage. As a result, the data consisted of multiple csv files, one file for every hour of every day that the data was collected. The collected data should undergo some preprocessing before it can be utilized as an input for the machine learning model.

4.1.3 Data Preprocessing

Some preprocessing was performed after retrieving the data. The preprocessing operations included feature engineering, dealing with outliers, and handling missing values. Each operation is discussed in detail below.

Feature Engineering

The features denoted $R, hol, wknd$ were added to the dataset by classifying the dates according to the Egyptian Calendar. In addition, the feature C was created by fetching each place's closing times using the Google Maps API and setting the C value for each record accordingly.

Handling Missing Values

The number of records in the dataset amounted to 1,749 data points for each location, making the total number of records 10494. 3729 of these records had missing Y values. According to Fig. 7, the distribution of popularity for each place was slightly skewed. Therefore, the missing values were imputed according to the median popularity value in accordance to $\{LT, C\}$ features.

However, there were two types of missing values.

1. Non-Random missing values: the Y values when H is between 1 AM and 7 AM
2. Random Y values across all records.

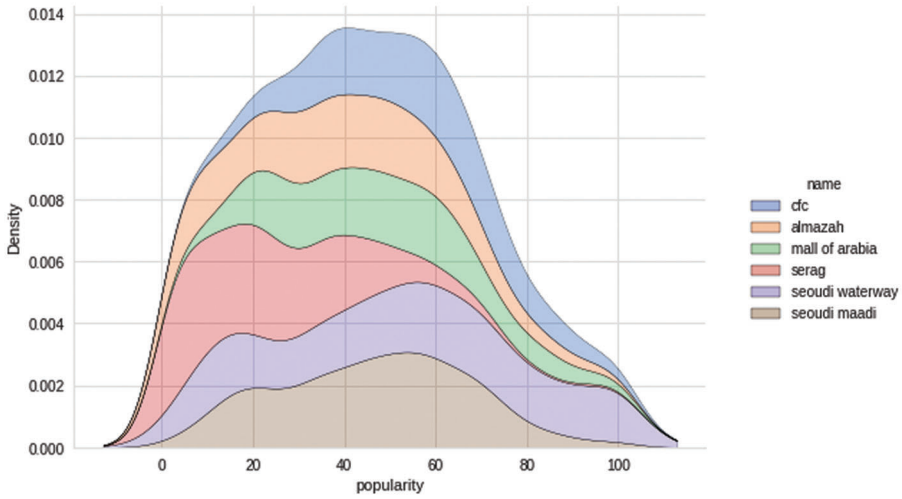


Figure 7: Kernel Density Estimate (KDE) Plot

The median imputation of the non-random missing values caused distortion to the popularity distribution into their equivalent hours. Therefore, all records within that time frame were removed, resulting in a decrease in the record numbers to 8304.

Handling Outliers

Fig. 8 is the box plot of the Serag location as an example, according to the *dow*, it clearly shows the presence of outliers. Two different outlier detection methods were applied separately to the dataset, and their effects on the model's performance were observed. The outliers detected were handled by removing them.

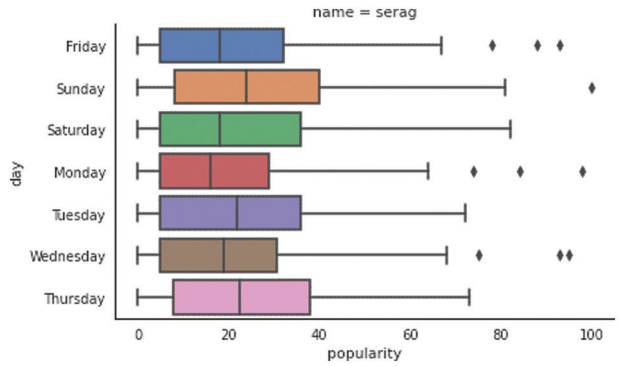


Figure 8: Serag box plot

1. Z-Score: It is a statistical approach that can be used for outlier recognition [40] and can be formulated as in Eq. (10)

$$Z = \frac{x - \mu}{\sigma} \tag{10}$$

where x denotes to the Y value, μ and σ are the mean and standard deviation, respectively, of all records grouped by $\{dow\}$.

2. Isolation Forest (IF) model for anomaly detection [41]. A contamination constant, C, of 0.02 was used.

After evaluation the isolation forest method performed better.

4.1.4 Model Evaluation

In order to evaluate the performance of various ML models, three different evaluation metrics were used [42].:

- Mean Squared Error: it measures the difference between the values that were fitted and the actual data observation and is calculated as in Eq. (11).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{11}$$

where n is the number of data points, Y_i is the observed value for point i, $\hat{Y}_i$ is the predicted value for point i.

- Root mean-squared error: it is the square root of the MSE and is calculated as in Eq. (12).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2} \tag{12}$$

- R Squared: it measures the amount of variation accounted for the fitted model and is calculated as in Eq. (13).

$$R^2 = 1 - \frac{SSR}{SST}, \quad SST = \sum_{i=1}^{n}\left(Y_i - \bar{Y}\right)^2 \tag{13}$$

where SSR is the sum of squared residuals that was computed according to in 7, n is the number of data points, Y_i is the observed value for point i, $\bar{Y}$ is the mean value of a sample.

4.1.5 Model Performance Comparison

In this section, the results of the compared ML algorithms are listed according to two main criteria:

1. The importance of $\{P\}$ feature
2. Different outlier detection methods

As clearly shown in Tables 3 and 4, when testing the models against the importance of the $\{P\}$ feature; all models had more than a 5% increase in their R2 scores, and equivalently, a drop was noticed in the MAE and RMSE values for all the three models. Those results are the outcome of using the full 8304 records dataset–with outliers.

Table 3: With previous popularity

Metric	Linear regression	Random forest	XGBoost
MAE	8.62	6.12	5.94
RMSE	11.88	9.56	9.45
R2	76.78	84.97	85.31
Time	0.04	1.11	4.55

Table 4: Removing outliers using Z-score

Metric	Linear regression	Randomforest	XGBoost
MAE	8.76	6.28	6.1
RMSE	12.28	9.99	9.93
R2	75.73	83.94	84.12
Time	0.02	1.01	4.23

Tables 5 and 6 compare the outlier detection methods stated in 4.1.3.3 Section: Handling Outliers–with the $\{P\}$ feature. The resulting Z-Score dataset had 8289 records while the IF dataset had 8276 records. The IF method outperformed the Z-score by a 2% increase in the linear regression and random forest R2 scores, and a 1.6% R2 score for the XGboost. This result aligns with the statement in [43] stating that IF performs better across a variety of datasets.

Table 5: Without previous popularity feature

Metric	Linear regression	Random forest	XGBoost
MAE	12.17	7.86	7.64
RMSE	15.59	11.84	11.43
R2	60.0	76.95	78.52
Time	0.07	1.08	4.99

Table 6: Removing outliers using IF

Metric	Linear regression	Random forest	XGBoost
MAE	8.76	6.24	6.02
RMSE	12.06	9.73	9.51
R2	77.03	85.04	85.7
Time	0.01	1.01	4.25

According to the above results, the XGBoost model has achieved the optimal results through the different testing criteria. Although the time complexity for XGboost was higher, the proposed system isn't time-critical, so time can be sacrificed for better validation.

4.1.6 Hyperparameters for the Evaluated Models

Tables 7 and 8 show the utilized hyperparameter values for random-forest and XGBoost, respectively, to improve the performance of the model.

The XGBoost hyperparameters stated in Tables 7 and 8 resulted from performing hyperparameter tuning using the grid-search model. As the grid-search is beyond the scope of this paper, a detailed explanation can be found in [44].

Table 7: Random forest hyperparameters

Hyperparameters	Values
max_depth	40
max_features	'sqrt'
min_samples_split	10
n_estimators	150
oob_score	True

Table 8: XGBoost hyperparameters

Hyperparameters	Values
colsample_bytree	0.3
learning_rate	0.05
max_depth	6
n_estimators	500
verbosity	0

Fig. 9 illustrates the learning curve across the number of training instances. As shown, the training score decreased with the increase of instances, indicating that the model is not overfitting to the training dataset. While the cross-validation score increases, an assumption can be made that with the increase in the number of records, more accurate scores will be achieved.

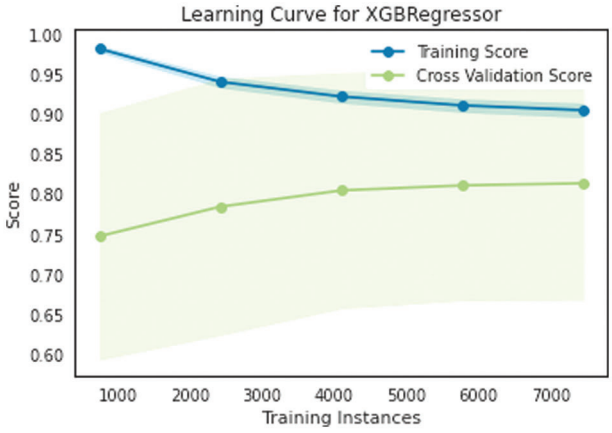
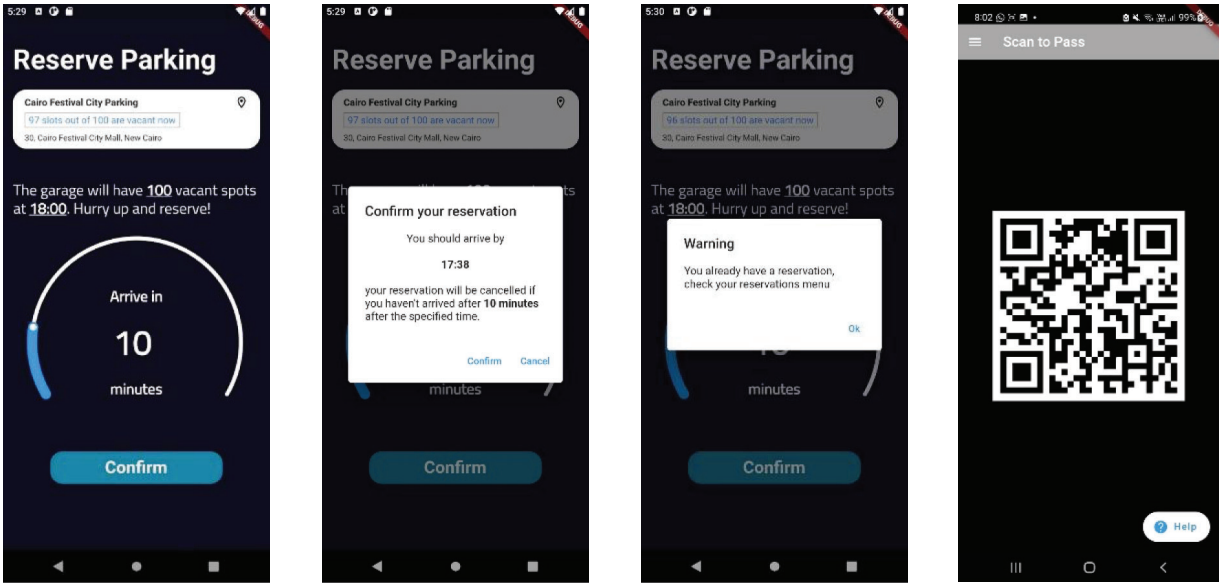


Figure 9: XGB Regressor training curve

4.2 Mobile Application

The relevant screens of the developed mobile application for the proposed IoT smart parking system are shown in Figs. 10a–10d. Below shows the relevant screens from the mobile application developed. The screenshots consist of a sample of the user's reservation and a confirmation about the reservation. Additionally, a QR code screen.



(a) Reservation Screen

(b) Reservation confirmation

(c) Warning notification for cancellation

(d) QR Screen

Figure 10: (a) Reservation screen. (b) Reservation confirmation. (c) Warning notification for cancellation. (d) QR screen.

5 Conclusion

Smart parking facilities have always been at the core of constructing smart cities. The proposed system can easily be implemented in smart cities as they have the proper infrastructure. The system utilizes IoT and machine learning concepts to provide real-time data as well as 1-h ahead predictions about the parking slot's occupancy. A mobile application has been developed to allow users to view the parking status, reserve a parking lot, and authenticate the user with a generated QR on entry and exit of the parking lot. As for the prediction, we evaluated the use of three different feature combinations and investigated the relative strengths of various machine learning algorithms in predicting the occupancy. This system enhances the performance of the parking paradigm in busy cities, thereby conserving resources and enhancing the quality of life for smart city residents.

6 Future Work

This proposed system has the potential for improvement with the addition of ALPR (Automated License Plate Recognition) technology. By integrating ALPR into the barrier gates as an added choice to the QR code, a seamless driver experience can be achieved. Along with providing a computer vision solution to replace the IR sensors, that could be cheaper on a larger scale.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] A. Morsy, *New parking regulations: finding that perfect spot*. Egypt: Al-Ahram Weekly, Ahram Online, 2021. [Online]. Available at: https://english.ahram.org.eg/News/422182.aspx.

[2] Y. Zhu, J. Chen, X. Yan and T. Wang, "Impact of cruising for carking on travel time of traffic flow," *Sustainability*, vol. 12, no. 8, pp. 3079, 2020.

[3] T. Lin, H. Rivano and F. Le Mouël, "A survey of smart parking solutions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3229–3253, 2017.

[4] "The capital Egypt," 2017. [Online]. Available: http://www.acud.eg/.

[5] M. Anshar, R. S. Sadjad, D. M. Hanan, R. Prayudha and M. Abry, "Design and implementation monitoring and booking systems for smart parking at engineering faculty campus," *IOP Conference Series: Materials Science and Engineering*, vol. 875, no. 1, pp. 012036, 2020.

[6] "QR code standardization," [Online]. Available: https://www.qrcode.com/en/about/standards.html.

[7] E. Hoq, S. Paul and M. T. U. R. Erin, "Development of a QR-code based smart car parking system," in *5th Int. Conf. on Advances in Electrical Engineering (ICAEE)*, Dhaka, Bangladesh, pp. 275–279, 2019.

[8] B. B. P. Sahoo, Shahjad and P. S. Tanwar, "Real-time smart parking: Challenges and solution using machine learning and IoT," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 7, no. 2, pp. 451–458, 2021.

[9] Y. Patel, P. Gandhi, S. Shah, S. Soman and A. Desai, "QR code scanner enabled smart car parking system using raspberry Pi with android app access," in *Emerging Technology Trends in Electronics, Communication and Networking. ET2ECN*, Surat, India, pp. 3–18, 2020.

[10] Y. Zheng, S. Rajasegarar and C. Leckie, "Parking availability prediction for sensor-enabled car parks in smart cities," in *IEEE Tenth Int. Conf. on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, Singapore, pp. 1–6, 2015.

[11] S. Alkhuraiji, "Design and implementation of an android smart parking mobile Application," *TEM Journal*, vol. 9, no. 4, pp. 1357–1363, 2020.

[12] M. Balfaqih, W. A. J. Al-Areeqi, M. Khayyat and R. Hassan, "Design and development of smart parking system based on fog computing and internet of things," *Electronics*, vol. 24, no. 10, pp. 3184, 2021.

[13] H. Brozova and M. Ruzicka, "The prediction of parking space availability," *Transport*, vol. 35, no. 5, pp. 462–473, 2020.

[14] I. Dia, E. Ahvar and G. M. Lee, "Performance evaluation of machine learning and neural network-based algorithms for predicting segment availability in AIoT-based smart parking," *Network*, vol. 2, no. 2, pp. 225–238, 2022.

[15] S. Y. Siddiqui, M. A. Khan, S. Abbas and F. Khan, "Smart occupancy detection for road traffic parking using deep extreme learning machine," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. pp 727–pp 733, 2022.

[16] L. Baroffio, L. Bondi, M. Cesana, A. E. Redondi and M. Tagliasacchi, "A visual sensor network for parking lot occupancy detection in smart cities," in *IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy, pp. 745–750, 2015.

[17] S. C. K. Tekouabou, E. A. Alaoui, W. Cherif and H. Silkan, "Improving parking availability prediction in smart cities with IoT and ensemble-based model," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 3, pp. 687–697, 2022.

[18] "Smart parking mobile app," Smart Parking, [Online]. Available: https://www.smartparking.com/smartpark-system/smart-app.

[19] "Australia's most used parking app," Easypark, [Online]. Available: https://easypark.com.au/en_au.

[20] A. Bajpai, "IoT based smart parking system using mobile application," *International Journal for Research in Applied Science and Engineering Technology*, vol. 8, no. 11, pp. 524–530, 2020.

[21] A. Kabir, A. M. Mizan and P. Saha, "An IoT based intelligent parking system for the unutilized parking area with real-time monitoring using mobile and web application," in *Int. Conf. on Intelligent Technologies (CONIT)*, Hubli, India, pp. 1–7, 2021.

[22] Y. Agarwal, P. Ratnani, U. Shah and P. Jain, "IoT based smart parking system," in *5th Int. Conf. on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, pp. 464–470, 2021.

[23] C. W. Wei, W. A. Jabbar, N. Azmi and N. Haironnazli, "An IoT raspberry Pi-based parking management system for smart campus," *Internet of Things*, vol. 14, no. 4, pp. 100387, 2021.

[24] T. Yu and C. Thong, "An enhanced QR code-based smart parking system for mobile environment," in *Proceedings of Int. Conf. on Computing and Communication Networks*, Singapore, Springer, pp. 59–75, 2022.

[25] J. Barker and S. Rehman, "Investigating the use of machine learning for smart parking applications," in *11th Int. Conf. on Knowledge and Systems Engineering (KSE)*, Da Nang, Vietnam, pp. 1–5, 2019.

[26] D. Vuk and D. Androcec, "Application of machine learning methods on IoT parking sensors' data," in *Proc. of Sixth Int. Congress on Information and Communication Technology*, Singapore, Springer, pp. 157–164, 2022.

[27] G. Jelen, V. Podobnik and J. Babic, "Contextual prediction of parking spot availability: A step towards sustainable parking," *Journal of Cleaner Production*, vol. 312, no. 5, pp. 127684, 2021.

[28] X. Han, "Shared parking choice behavior based on machine learning algorithm," *2nd Int. Conf. on Internet of Things and Smart City (IoTSC), SPIE*, vol. 12249, no. 1, pp. 371–375, 2022.

[29] T. Hastie, R. Tibshirani and J. Friedman, "Boosting and additive trees," in *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. New York, USA: Springer, pp. 299–345, 2009.

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

[31] T. Chen, "XGBoost: A scalable tree boosting system," in *Proc. of the 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, California, San Francisco, USA, pp. 785–794, 2016.

[32] "XGBoostXGBoost," 2021. [Online]. Available: https://www.geeksforgeeks.org/xgboost/.

[33] Mit, "Linear regression," in *Statistics for Research Projects*. 2020. [Online]. Available at: https://www.mit.edu/~6.s085/notes/lecture3.pdf.

[34] J. Stramer, "*The StatQuest illustrated guide to machine learning*," pp. 75–83, 2022, StatQuest Publications.

[35] F. T. Liu, K. M. Ting and Z.-H. Zhou, "Isolation forest," in *Eighth IEEE Int. Conf. on Data Mining*, Pisa, Italy, pp. 413–422, 2008.

[36] "Vertex AI," [Online]. Available: https://cloud.google.com/vertex-ai.

[37] "Create and configure cron jobs," [Online]. Available: https://cloud.google.com/scheduler/docs/creating.

[38] "Pub/Sub," [Online]. Available: https://cloud.google.com/pubsub.

[39] "LivePopularTimes," [Online]. Available: https://github.com/GrocerCheck/LivePopularTimes.

[40] P. V. Anusha, C. Anuradha, P. S. C. Murty and C. S. Kiran, "Detecting outliers in high dimensional data sets using Z-score methodology," *International Journal of Innovative Technology and Exploring Engineering*, vol. 9, no. 1, pp. 48–53, 2019.

[41] F. T. Liu, K. M. Ting and Z. H. Zhou, "Isolation forest," in *2008 Eighth IEEE Int. Conf. on Data Mining*, Pisa, Italy, pp. 413–422, 2008.

[42] H. Pham, "A new criterion for model selection," *Mathematics*, vol. 7, no. 12, pp. 1215, 2019.

[43] A. Young, "Towards data science," 2020. [Online]. Available: https://towardsdatascience.com/isolation-forest-is-the-best-anomaly-detection-algorithm-for-big-data-right-now-e1a18ec0f94f.

[44] Á.B. Jiménez, J. R. Dorronsoro and J. L. Lázaro, "Finding optimal model parameters by discrete grid search," in *Innovations in Hybrid Intelligent Systems*. Berlin, Heidelberg: Springer, pp. 120–127, 2007.