Received: 24 December 2022 | Revised: 22 February 2023 | Accepted: 30 March 2023
DOI: 10.4218/etrij.2022-0454

ORIGINAL ARTICLE

ETRI Journal WILEY

# Optimal dwelling time prediction for package tour using *K*-nearest neighbor classification algorithm

**Aria Bisma Wahyutama** | **Mintae Hwang**

Department of Information and Communication Engineering, Changwon National University, Changwon, Republic of Korea

**Correspondence**
Mintae Hwang, Department of Information and Communication Engineering, Changwon National University, Changwon, Republic of Korea.
Email: mthwang@cwnu.ac.kr

**Funding information**
Gyeongnam SW Convergence Cluster 2.0

**Abstract**
We introduce a machine learning-based web application to help travel agents plan a package tour schedule. *K*-nearest neighbor (*K*NN) classification predicts the optimal tourists' dwelling time based on a variety of information to automatically generate a convenient tour schedule. A database collected in collaboration with an established travel agency is fed into the *K*NN algorithm implemented in the Python language, and the predicted dwelling times are sent to the web application via a RESTful application programming interface provided by the Flask framework. The web application displays a page in which the agents can configure the initial data and predict the optimal dwelling time and automatically update the tour schedule. After conducting a performance evaluation by simulating a scenario on a computer running the Windows operating system, the average response time was 1.762 s, and the prediction consistency was 100% over 100 iterations.

**KEYWORDS**
dwelling time prediction, *K*-nearest neighbor classification, machine learning, optimized schedule, web application

## 1 | INTRODUCTION

Under the ongoing coronavirus disease (COVID-19) pandemic, the number of travelers has decreased sharply worldwide [1]. Although many residents strongly support local tourism, they are afraid of contracting COVID-19, which could cause conflict [2]. Nevertheless, tourism will inevitably recover and even strengthen compared with the prepandemic era [2, 3]. Therefore, new and creative methods should be devised to prepare for tourism recovery after the pandemic with people traveling freely and frequently. Implementing information and communication technologies such as machine learning (ML) to determine the optimal dwelling time (i.e., how long tourists should stay at tourist spots) can help travel agents manage package touring. Many travel agents offer package tours, both domestically and internationally, with multiple advantages. However, a common problem with package tours is that they may not provide the appropriate dwelling time, especially when prepared by less experienced travel agents. If the dwelling time is excessively long, the tourists feel bored, while if it is very short, the tourists feel underwhelmed because they cannot enjoy every spot thoroughly. A tourist's dwelling time affects the overall package tour quality, even creating a lousy impression on the tourist if it is inadequate. ML technology may help travel agents prevent this problem by predicting optimal dwelling times.

In global statistics, over 2.4 billion international tourists traveled in 2019, making tourism a representative income source for any country [4]. The United Nations World Tourism Organization has confirmed that cultural tourism is essential, with 89% of domestic tourism administrators using cultural tourism as a part of their policies. Furthermore, over 39% of international tourists travel abroad to enjoy cultural tourism [5]. Hence, many travel agents worldwide are interested in managing personalized package tours.

In this study, we developed an ML model to predict the tourists' optimal dwelling times in spots by implementing the *K*-nearest neighbor (*K*NN) algorithm on a web application. The model uses tourists' personal information such as age, health condition, group type, and tour experience, along with weather conditions and selected tourist spots as features or parameters. Travel agents can then generate predictions using the web application that automatically creates an optimized tour schedule as a table. Once the full schedule has been determined, travel agents can share it with their clients via a mobile application. The predictions made by the ML model can assist travel agents in deciding the dwelling time, which in turn can increase the quality of package tours.

For the proposed system, we first collect raw tourist data in a cloud database (DB) provided by Google Firebase. A web application then obtains the data from the cloud DB, formats the data in the desired format, and feeds them into an ML-based *K*NN algorithm to predict the dwelling time via a RESTful (Rest) application programming interface (API). The Rest API is developed using the PHP + Laravel and Python + Flask frameworks. The system then uploads the prediction results and new schedules to the cloud DB for display on the web application to the travel agents and on the mobile application to clients. *K*NN classification predicts the optimal dwelling time based on nine parameters: age, temperature, humidity, sex, health conditions, tour experience, weather conditions, group type, and tourist spot. Based on these parameters, the prediction belongs to one of eight classes (time in minutes): 60, 90, 120, 150, 180, 210, 240, and 270.

The remainder of this paper is organized as follows. Section 2 discusses related work, research results, and their contributions and limitations. Section 3 describes the system design and architecture, including the algorithm flowchart, architecture, requirements, and initial user interface design. Section 4 describes the implementation of the Rest API as well as web and mobile applications. Section 5 presents the performance evaluation of the system by measuring the response time, ML model performance, and consistency. Finally, Section 6 presents conclusions and directions of future work.

## 2 | RELATED WORK

Tourists or visitors are individuals who voluntarily leave familiar environments where they usually reside to visit another place. Tourists typically engage in specific activities regardless of their distance from familiar environments. Tourism is a temporary, short-term movement that occurs for less than 1 year to visit a particular destination and perform certain activities, including leisure, day visits, and excursions [6]. On the other hand, researchers have shown that ML can solve many real-world problems, such as predicting traffic for intelligent transportation and managing multiple essential aspects of agriculture [7, 8]. Furthermore, ML optimization has attracted the attention of many researchers [9]. Because of its versatility, we aim to apply ML to tourism.

Researchers often classify ML into supervised, semisupervised, unsupervised, and reinforcement learning approaches and consider learning models such as classification, dimensionality reduction, regression, and clustering.

We use a supervised ML method because the DB for training contains several features related to tourists' personal information and preferred dwelling times. Because the dwelling time is categorized into several time frames, we adopt a classification algorithm as the learning model. Table 1 summarizes various related studied on ML applied to tourism [10–16].

The related work reveals challenges and problems. For instance, 5G technology is considered in [10], but it is unevenly available globally. Underdeveloped countries face challenges to implement 5G technology. Consequently, the solution in Peng and others [10] is limited to developed countries. In Srisawatsakul and Boontarig [11], a recommender based on Instagram photographs can only be employed by Instagram users, primarily teenagers, and young adults. Consequently, older adults are excluded because most of them are not "technology friendly" and obviously do not have Instagram accounts. In Nagar and others [12], the scope is limited to those traveling to seek medical attention, making the proposal unsuitable for regular tourists. In Afsahhosseini and Al-Mulla [13], ML is applied to each individual to obtain a personalized result instead of a group schedule, as is usually the case for package tours. Overall, previous studies have some research gaps that we aim to address. For example, in earlier studies [14, 15], the use of robots can be expensive and difficult to implement, particularly in underdeveloped countries. In Parvez [16], ML technology is mainly intended to serve automation systems, especially for hotel customer services. Thus, the establishment must prepare the proper infrastructure, and specialized staff must maintain the system. If not carefully observed, additional costs may be incurred.

TABLE 1 Summary of related work on ML applied to tourism.

| Study | Contribution |
|---|---|
| A human-guided machine learning approach for 5G smart tourism IoT [10] | An ML model is used to help tourists select a particular destination by using the Internet of Things and human-guided ML classification based on tourist behaviors |
| Tourism recommender system using machine learning based on user's public Instagram photos [11] | A Google Cloud Vision API and recommendation algorithm are used to generate a recommendation of tourist attractions from social media. Photos available on social media (Instagram) are collected and mined and fed into the API to extract terms and features from the images. Then, the result is fed to the content-based recommender |
| A review on machine learning applications in medical tourism [12] | ML technology is used to help tourists that travel abroad to seek medical help by recommending a hospital or activity that would help treat a disease |
| Machine learning in tourism [13] | ML embedded into three tourism phases: pretrip, trip, and posttrip. In the pretrip phase, ML predicts tourist demand, while ML provides a recommendation of tourist attractions during a trip. In the posttrip phase, ML analyzes reviews of the tour experience to extract helpful information |
| Robots in tourism: A research agenda for tourism economics [14] | The study considers tourism supply, tourism demand, and destination management. It critically assesses state-of-the-art research on the economics of service robots in the tourism industry |
| Machine learning of robots in tourism and hospitality: interactive technology acceptance model (iTAM)–cutting edge [15] | This study analyzes how consumers perceive advanced artificial intelligence robots in the hospitality and tourism industries. |
| Use of machine learning technology for tourist and organizational services: high-tech innovation in the hospitality industry [16] | This study pinpoints the present and upcoming changes brought on by ML systems in the hospitality sector to make the holiday experience more enjoyable and simplify processes |

Abbreviations: API, application programming interface; IoT, Internet of Things; ML, machine learning.

The novelty of our study is the combination of ML to predict the optimal dwelling times when visiting certain tourist spots, instead of simply developing a system to recommend tourist spots. The proposed system does not require any additional hardware or specialized software and can be used for any tourism purpose, either for individuals or groups. Thus, the system can be used anywhere. The ML model is implemented on a web application that merges the prediction results into a tour schedule controlled by travel agents. Therefore, the system can help travel agents improve their service quality and provide package tours that everyone can enjoy. This study extends our previous work [17–19], in which we developed a package tour management system using geofences and digital game-based learning. In this study, we added dwelling time prediction as a main feature of the package tour management system.

## 3 | DESIGN OF PROPOSED SYSTEM

In this section, we discuss the system architecture, including the dataflow from each component, algorithms of the web application and Rest API, requirements for developing the system for mobile applications, Rest API, and initial user interface design.

### 3.1 | System architecture

The proposed system uses a web application to obtain tourist information from a cloud DB. After receiving tourists' information, the web application calls the Rest API to pass the tourists' information to the ML model and predict dwelling times, which are then uploaded to the cloud DB. After uploading, the web application displays the dwelling times along with an updated schedule showing the prediction results. Figure 1 shows the system architecture. The system is used only after training the ML model, as described in Section 4.

A cloud DB is selected because we adopted it in previous research [17–19], rendering the creation of a new DB to predict the dwelling times inconvenient. Additionally, the cloud DB offers native and seamless communication for sending and retrieving data from multiple platforms, such as web, Android, and iOS applications. Therefore,

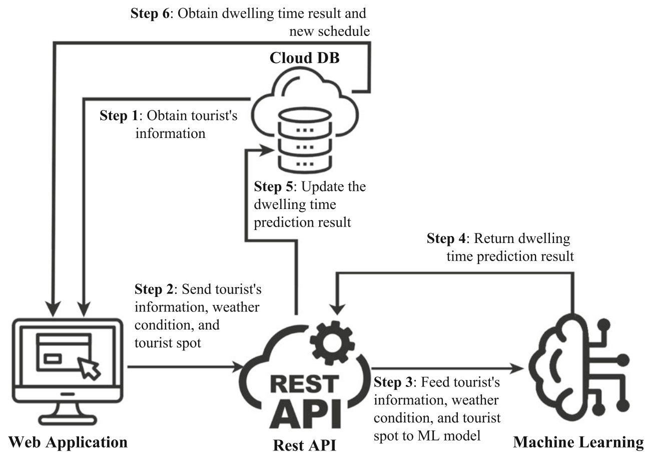

Step 6: Obtain dwelling time result and new schedule
Cloud DB
Step 1: Obtain tourist's information
Step 5: Update the dwelling time prediction result
Step 4: Return dwelling time prediction result
Step 2: Send tourist's information, weather condition, and tourist spot
REST API
Step 3: Feed tourist's information, weather condition, and tourist spot to ML model
Web Application
Rest API
Machine Learning

FIGURE 1 System architecture with required components and their relations.

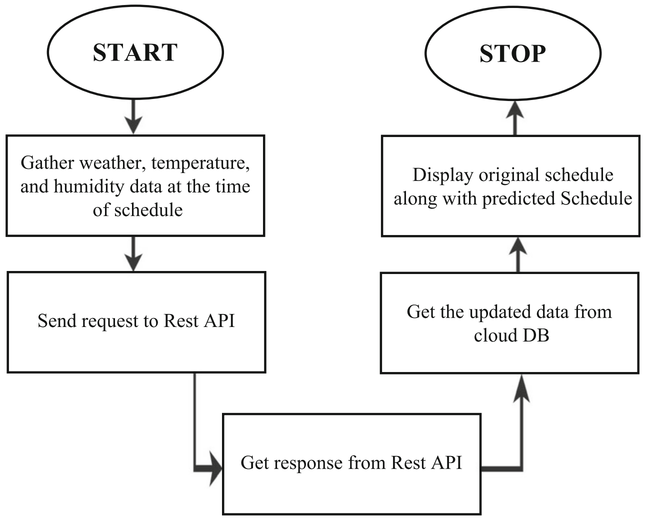

START
Gather weather, temperature, and humidity data at the time of schedule
Send request to Rest API
Get response from Rest API
Get the updated data from cloud DB
Display original schedule along with predicted Schedule
STOP

FIGURE 2 Flowchart of web application.

the system can be expanded to another platform without requiring additional configurations. As web applications and ML models cannot communicate directly, an interface is required to handle the communication. We use a Rest API that also acts as a pivot between the web application and ML model. We detail the latest API developments below.

## 3.2 | System flowchart

Here, we describe the flowchart to determine the input, process, and output of the application before it is encoded using a programming language. The flowchart explanation is intended to support troubleshooting and error handling.

### 3.2.1 | Web application

Travel agents can use the web application to create a schedule based on predicted dwelling time. For the prediction, the travel agent first inputs and configures the tourist spot conditions, including the weather state, temperature, and humidity at the time of the visit. Then, the web application sends a request to the Rest API. After receiving a response, the predicted schedule appears on the screen. Figure 2 shows the flowchart of the web application.

### 3.2.2 | Rest API

The execution of the Rest API is slightly more complicated than that of the web application because it is the backbone of the entire system that communicates with the server and handles all the data and information. When the web application initiates a Rest API request, it triggers the PHP Rest API to collect tourist data from the cloud DB and convert them into an array. The PHP Rest API encodes the array in JavaScript Object Notation (JSON) before executing a POST request to another API written using Python and containing the ML model. When sending a request, the Python Rest API loads the trained ML model, converts the JSON data into a DataFrame format, and feeds the data to the ML model to predict the dwelling times. Finally, the predictions are returned to the PHP Rest API for uploading to the cloud DB, and the tour schedule is updated.

The two Rest APIs in the system serve different purposes. The PHP Rest API is responsible for obtaining the stored data from the cloud DB, processing it into the JSON format, executing the POST request to the Python Rest API, and uploading the dwelling time predictions to the cloud DB. The Python Rest API loads the trained ML model and predicts the dwelling times. Therefore, the PHP Rest API connects directly with the web application and Python Rest API, whereas the Python Rest API connects only with the PHP Rest API. Figure 3 presents a flowchart of the Rest API.

The ML model requires a standalone Python Rest API because the system web application uses the PHP programming language with the Laravel framework and thus cannot natively communicate with the ML model. Therefore, an additional REST API that uses the Flask framework is required for communication.

## 3.3 | System requirements

The main components of the proposed system are the web application and Rest API, whose requirements are

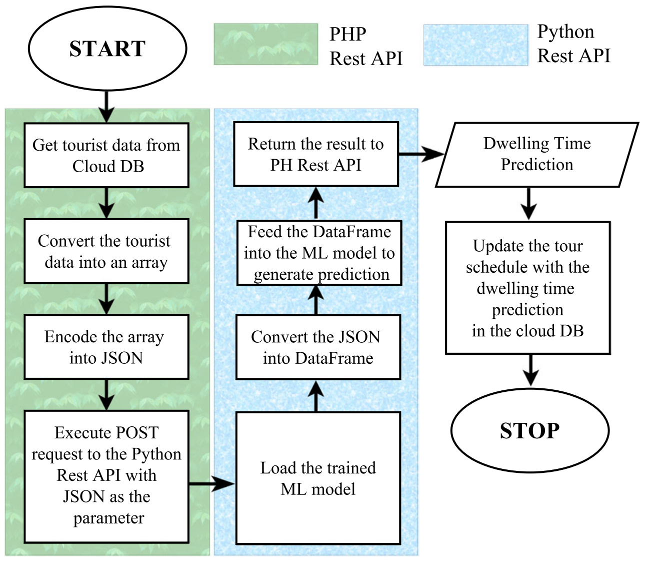

START
PHP Rest API
Python Rest API
Get tourist data from Cloud DB
Convert the tourist data into an array
Encode the array into JSON
Execute POST request to the Python Rest API with JSON as the parameter
Load the trained ML model
Convert the JSON into DataFrame
Feed the DataFrame into the ML model to generate prediction
Return the result to PH Rest API
Dwelling Time Prediction
Update the tour schedule with the dwelling time prediction in the cloud DB
STOP

FIGURE 3 Flowchart of Rest application programming interface (API).

discussed in this section. Figures 2 and 3 can be analyzed to determine the requirements for each component.

The primary requirements of the web application are obtaining tourist spot conditions, requesting the Rest API, obtaining predictions, and presenting the prediction results along with a new schedule.

For the Rest API, the primary requirements are obtaining tourist information from the cloud DB, converting the data into the correct form (JSON or DataFrame), executing the ML model to predict the dwelling times, and delivering the results to the cloud DB.

Tables 2 and 3 list the requirements for the web application and Rest API, respectively.

## 3.4 | User interface design

We propose an initial design for the web application user interface. The main user interface consists of pages for tourist spot configuration and schedule display. On the tourist spot configuration page shown in Figure 4, a travel agent can configure the weather, temperature, and humidity for a tour schedule. A button to predict the dwelling times is located below the configuration box. Once the dwelling times are predicted, the schedule display page appears, showing the original and predicted schedules in separate tables. The travel agent can configure the weather information of each tourist spot for each day according to the schedule. Figure 5 shows the schedule display page containing the original schedule (top), which is editable or can be removed, and predicted schedule (bottom) obtained from the ML model.

TABLE 2 Requirements for web application.

| Requirement | Description |
|---|---|
| Get tourist spots condition configuration | The web application can retrieve conditions such as weather, temperature, and humidity |
| Hit Rest API URL | The web application can hit the Rest API URL with the intended method while sending all necessary parameters |
| Get dwelling time prediction | The web application can get the dwelling time prediction from the cloud DB |
| Create new schedule with predicted dwelling times | The web application can create new schedules with the dwelling time prediction, including starting and stopping times |

Abbreviations: API, application programming interface; DB, database.

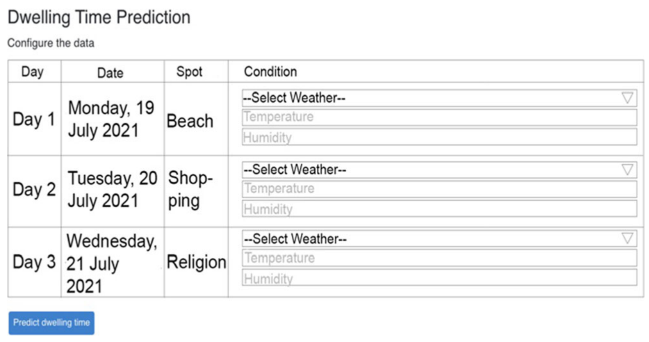
TABLE 3 Requirements for Rest API.

| Requirement | Description |
|---|---|
| Get tourists' information | The PHP Rest API can obtain the tourists' information from the cloud DB |
| Represent tourist's information in JSON format | The obtained tourists' information is represented in the JSON format for the Python Rest API |
| Execute POST request to Python Rest API | The PHP Rest API can execute a POST request to the Python Rest API and pass the JSON format data as parameter |
| Convert JSON to DataFrame | The Python Rest API converts the retrieved JSON format data into DataFrame to be more suitable for the ML model |
| Predict dwelling times | The Python Rest API can predict the dwelling times after feeding the DataFrame data to the trained ML model |
| Return prediction result | The Python Rest API can return the dwelling time predictions to the PHP Rest API |
| Upload prediction result to cloud DB | The PHP Rest API uploads the predictions to cloud DB for loading on the web application |

Abbreviations: API, application programming interface; DB, database; JSON, JavaScript Object Notation.

## 4 | SYSTEM IMPLEMENTATION

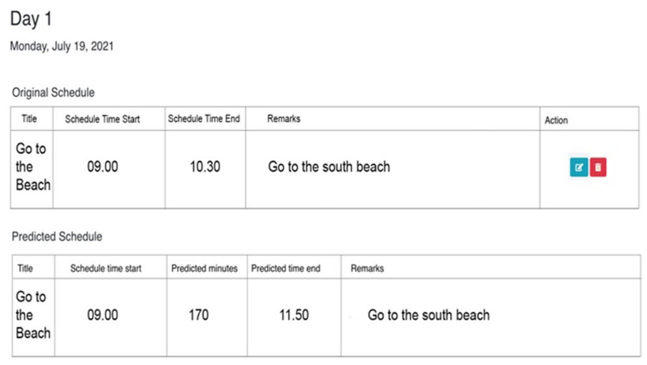
The system implementation involves the ML model, mobile application, and Rest API. In addition, the


Dwelling Time Prediction

Configure the data

| Day | Date | Spot | Condition |
|---|---|---|---|
| Day 1 | Monday, 19 July 2021 | Beach | --Select Weather-- / Temperature / Humidity |
| Day 2 | Tuesday, 20 July 2021 | Shop-ping | --Select Weather-- / Temperature / Humidity |
| Day 3 | Wednesday, 21 July 2021 | Religion | --Select Weather-- / Temperature / Humidity |

FIGURE 4 User interface for tourist spot configuration.


Day 1

Monday, July 19, 2021

Original Schedule

| Title | Schedule Time Start | Schedule Time End | Remarks | Action |
|---|---|---|---|---|
| Go to the Beach | 09.00 | 10.30 | Go to the south beach | |

Predicted Schedule

| Title | Schedule time start | Predicted minutes | Predicted time end | Remarks |
|---|---|---|---|---|
| Go to the Beach | 09.00 | 170 | 11.50 | Go to the south beach |

FIGURE 5 User interface for schedule display.

environments for their building, development, and implementation are presented.

## 4.1 | Implementation environment

We divided the implementation environment of the system into categories of computer, ML, other API, and web applications. The computer ran Windows 11 version 21H2 build 22000.588 and was equipped with an Intel i7-10 700 processor and 32 GB of memory. For developing the ML model, we used Anaconda version 2.1.4 with Python version 3.8.9. The built-in Python library provided by Anaconda was used as is. We also used Anaconda as the Python-integrated development environment with Jupyter Notebook version 6.4.5 to develop the ML model.

We coded the PHP Rest API using the PHP programming language and Laravel framework, while the Python Rest API was coded using Python, the same programming language used for developing the ML model and a lightweight framework called Flask (version 2.1). We coded the entire system using Visual Studio Code version 1.68.1. In addition, the Postman tool version 9.22.2 was used to verify the Rest API results.

The web application was developed using the PHP programming language version 8.1.7 in the Visual Studio Code editor like the Rest API. In addition, XAMPP version 8.1.6, was used to deploy a local server to run the Rest API. Moreover, the web application was developed with the Laravel framework version 9, which contains multiple features to accelerate development, such as fast routing and powerful dependencies.

## 4.2 | DB and ML model

Determining the features for prediction before developing an ML model is crucial. Based on [20, 21], we determined nine features and one label for the DB. In addition, the DB had two data types: integer (age, temperature, humidity, and dwelling time) and string (sex, health condition, weather condition, group type, tourist spot, and tour experience). The integer features were filled with any number, whereas the string features could only take fixed values to simplify ML model training. Specifically, sex had values male or female, and health condition had values healthy, minor complication, and major complication. In addition, weather had values sunny, cloudy, rainy, and snowy. The group type had fixed values family, worker, male friends, female friends, and student. Tourist spots had 11 fixed values: education, water park, culture, sports, religion, amusement park, historical site, shopping mall, tracking, beach, and museum. Finally, tour experience had a value of yes or no.

A minor health complication referred to conditions that affected the tourist's activity but were not considered life-threatening, such as influenza, cold, or light fever. A major health complication was considered as life-threatening or as a condition that can suddenly relapse, such as coronary artery disease, disability, or stroke. Table 4 lists the DB features and their descriptions.

We collected the DB in collaboration with Dot of Corner, a well-established Indonesian travel agency and consultancy company that has been in business for over 15 years. Data were collected from December 2021 to May 2022, resulting in 30 000 datapoints containing multiple tourist scenarios for two- and four-season countries. The data were carefully collected by analyzing previous package tour schedules and conducting in-depth meetings with the Dot of Corner's staff and various tourist representatives. Therefore, we did not require to perform data cleaning when preparing the data for the ML model because it was implicit throughout data collection.

As mentioned above, we used classification because the predicted dwelling time required was a specific number (e.g., 110, 105, and 215 min) instead of arbitrary numbers (e.g., 246.45 and 157.27 min), which can be obtained

TABLE 4 Characteristics of DB for ML model.

| DB characteristic | Description |
|---|---|
| Age | Different age groups have different preferences of dwelling time at a tourist spot |
| Sex | Male and female persons may have different preferences of dwelling time |
| Health condition | Many tourist spots are not appropriate or suitable depending on the health condition, thus affecting the dwelling time |
| Weather condition | The weather, such as sunny, cloudy, or rainy, can affect the dwelling time |
| Group type tourist spot | Group types include friends, family, and workers. Travel agents fill the tourist spot information. |
| Tour experience | This characteristic indicates whether a tourist has visited a spot |
| Temperature | The temperature during the visit can determine the dwelling time |
| Humidity | The humidity during the visit can determine the dwelling time |
| Dwelling time | The optimal dwelling time is predicted by the ML model based on available information |

Abbreviations: DB, database; ML, machine learning.

from regression. Additionally, when a travel agent prepares a package tour schedule, a detailed dwelling time with resolution of minutes is challenging to determine, especially when combined with other nonvisiting schedules (e.g., lunchtime and sleeping time). Furthermore, package tours do not usually involve strict time-sensitive activities that require exact dwelling times. Thus, the classification algorithm only classifies the outputs into one of several fixed classes. The dwelling times in the DB were divided into eight classes/timeframes (in minutes): 60, 90, 120, 150, 180, 210, 240, and 270. We excluded dwelling times shorter than 60 min and longer than 270 min because they are outside tourists' usual stays in a spot, given the various activities usually included in package tours. The predicted dwelling times corresponded to each tourist. For example, five predictions were made for five tourists. Therefore, the final dwelling times for creating the general tour schedule were the average of the predictions across tourists.

Because the classification algorithm cannot process strings, all the features that use string data types were encoded using a Python library called Ordinal Encoder, which assigns a number to each unique string. We used ordinal encoding instead of the common one-hot encoding because the latter represents categorical variables as a binary vector designated to a new column in the DB. Consequently, the number of columns increases, thereby reducing the performance of the ML model. In contrast, the ordinal encoder converts each unique string into a number in its original column. For example, entry sunny in the weather condition is converted into 0, cloudy into 1, rainy into 2, and snowy into 3. We applied the ordinal encoder to each string feature.

Several classification algorithms were compared in Wahyutama and Hwang [22] to determine the best one for our system. The *K*NN algorithm provided the best performance and consistency in multiple comparison scenarios, making it suitable for our system.

### 4.3 | Rest API implementation

Implementing the PHP Rest API was straightforward, because the web application was written in PHP. After obtaining all tourist data from the cloud DB as an array, function Client() in the Laravel framework called the Python Rest API. Then, we defined the Python Rest API URL (Uniform Resource Locator) as the POST method and added tourist data from the cloud DB as the parameter. After execution, the system retrieved data from the Python Rest API to update the cloud DB. We chose the POST method to send data because many parameters were required for prediction, rendering the GET method inconvenient. The source code for the PHP Rest API can be found in the GitHub repository, available at https://github.com/abismaw/DwellingTimePrediction (DwellingTimeController.php).

The Python Rest API was developed in Python and used the Flask framework. Flask is a lightweight Web Server Gateway Interface developed for Python, and it has an extension called Flask-RESTful that offers additional features and functions for developers to create a Rest API.

To use Flask, we created and activated a virtual environment that ran Python 3 (as recommended by Flask) in terminals by executing command "source/bin/activate." After creating the virtual environment, we installed the Flask library using PIP (Peripheral Interchange Program) command "pip install flask." Once installation was complete, the flask was imported into a Python file. Other libraries required for the Rest API to work correctly included Pandas to create DataFrame, Pickle to import the trained ML model, and Ordinal Encoder to encode the string features.

When the Python Rest API was triggered by the PHP Rest API, it retrieved the JSON request and appended the data to the DataFrame provided by the Panda library. After converting every JSON request into a DataFrame, Ordinal Encoder encoded the string features. DataFrame, which contained tourist information and weather conditions, was fed into the trained ML model for prediction. After making various predictions, the average dwelling time was determined and rounded off to the nearest five or zero. Finally, the average results and response status were returned to the web application to be displayed on the webpage. Typical Rest API results are shown in Figure 6. Tables 5 and 6 list the pseudocodes of the PHP and Python Rest APIs, respectively. The source code for the Python Rest API can be found in the GitHub repository, available at https://github.com/abismaw/DwellingTimePrediction (app.py).

TABLE 6 PHP Rest API pseudocode.

```
1. Get tourists' raw data from cloud DB
2. Remove existing null array
3. Get age, health, sex, group type, tour experience, tourist
   spot, weather condition, temperature, and humidity
   information per tourist
4. Push selected data into a new array
5. Call Python Rest API URL link using function Client()
6. Put a new array containing selected data as parameter
7. Get response from the Python Rest API
8. Save response to cloud DB
9. Update tour schedule in cloud DB
```

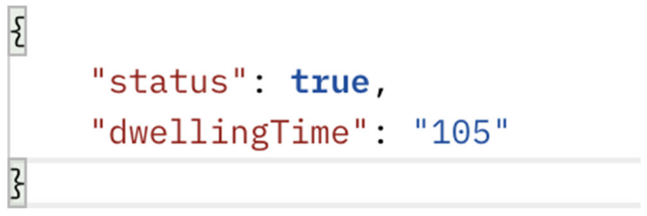
Abbreviations: API, application programming interface; DB, database.

## 4.4 | Web application implementation

The web application was intended to be controlled by travel agents for scheduling package tours. It was written in the PHP programming language with the Laravel framework to speed up development. Users could only predict the dwelling times if they appropriately gathered and stored the information of participating tourists in the cloud DB. A travel agent could access the menu to generate predictions from the main menu, side navigation bar, or top navigation bar showing the choices of an active package tour when clicked.

After selecting the correct package tour, the travel agent saw a tourist spot configuration containing two tables. The first table (top) showed a list of tourist spots and day number, date, and parameter configurations.

The second table (bottom) contained tourist information such as age, experience, health condition, group type, and sex for travel agents to validate the information. Below the first table, a button for predicting the dwelling time based on the given information was available. Figure 7 shows a screenshot of the tourist spot configuration page. After the travel agent clicked the "Predict Dwelling Time" button, two things occurred. First, the results were uploaded to the cloud DB and redirected to the schedule display page, where they could choose the package tour day to see or modify. Once selected, two tables appeared, one containing the original schedule at the top and a new schedule with the predicted dwelling time at the bottom. The original schedule table contained the scheduled activity name, beginning and end, remarks, and a button for editing or deletion. The new schedule table included similar information showing the predicted dwelling time. The prediction was automatically added to the start time, resulting in a predicted finish time, thus creating an optimized schedule for the package tour to visit a certain tourist spot. Figure 8 shows a screenshot of a scheduled page. Additionally, the bottom table notified the travel agents of the remaining predictions of dwelling times.



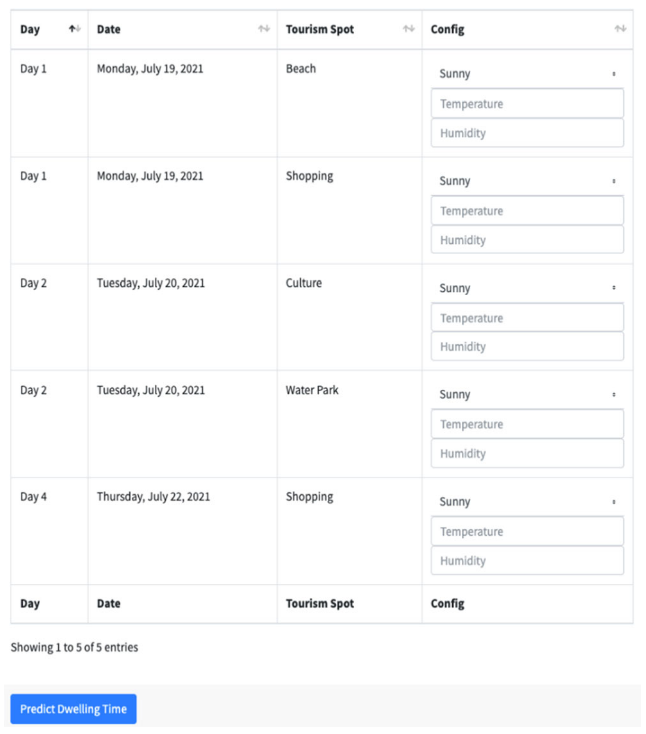
```
{
    "status": true,
    "dwellingTime": "105"
}
```

FIGURE 6 Typical Rest application programming interface (API) response.

TABLE 5 Python Rest API pseudocode.

```
1. Initialize Flask framework
2. Load ML model
3. Set array "content" to tourist's data from PHP Rest API
4. Initialize DataFrame
5. For x = count (content)
     • age = x['age']
     • sex = x['sex']
     • ... (for every required tourist's information entry)
6. End For
7. Initialize Ordinal Encoder
8. Encode sex, health, group type, tour experience, tourist spot,
   and weather condition numerically
9. Predict dwelling time using loaded ML model
10. Calculate average prediction
11. Return average prediction
```

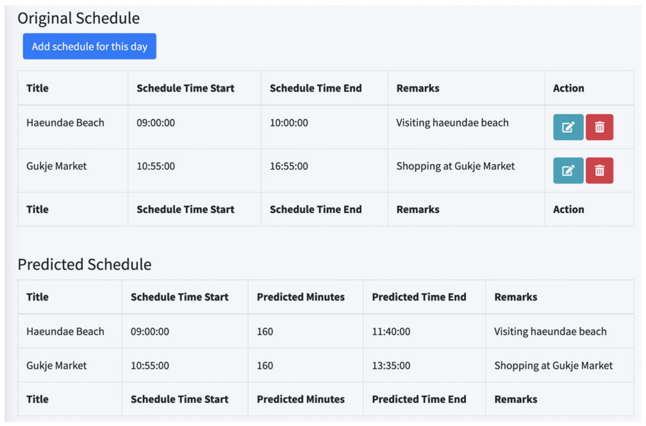
Abbreviations: API, application programming interface; ML, machine learning.



| Day | Date | Tourism Spot | Config |
|---|---|---|---|
| Day 1 | Monday, July 19, 2021 | Beach | Sunny / Temperature / Humidity |
| Day 1 | Monday, July 19, 2021 | Shopping | Sunny / Temperature / Humidity |
| Day 2 | Tuesday, July 20, 2021 | Culture | Sunny / Temperature / Humidity |
| Day 2 | Tuesday, July 20, 2021 | Water Park | Sunny / Temperature / Humidity |
| Day 4 | Thursday, July 22, 2021 | Shopping | Sunny / Temperature / Humidity |
| Day | Date | Tourism Spot | Config |

Showing 1 to 5 of 5 entries

FIGURE 7 Screenshot of configuration page for dwelling time prediction.



Original Schedule

| Title | Schedule Time Start | Schedule Time End | Remarks | Action |
|---|---|---|---|---|
| Haeundae Beach | 09:00:00 | 10:00:00 | Visiting haeundae beach | |
| Gukje Market | 10:55:00 | 16:55:00 | Shopping at Gukje Market | |
| Title | Schedule Time Start | Schedule Time End | Remarks | Action |

Predicted Schedule

| Title | Schedule Time Start | Predicted Minutes | Predicted Time End | Remarks |
|---|---|---|---|---|
| Haeundae Beach | 09:00:00 | 160 | 11:40:00 | Visiting haeundae beach |
| Gukje Market | 10:55:00 | 160 | 13:35:00 | Shopping at Gukje Market |
| Title | Schedule Time Start | Predicted Minutes | Predicted Time End | Remarks |

FIGURE 8 Screenshot of schedule display page after predicting dwelling times.

## 5 | PERFORMANCE EVALUATION

Two performance evaluations were conducted using the proposed system. In the first evaluation, we measured the response time, which is the time elapsed between clicking the "Predict Dwelling Time" button on the tourist spot configuration page and the prediction appearing on the schedule display page. To obtain the response time, we calculated the corresponding difference in Epoch Time (UNIX Timestamps). The measurements were repeated 100 times to obtain the average time required as the final response time. Figure 9 shows the response time per measurement.

The response time was evaluated using Google Chrome version 100.0.4896.60 through an Internet connection provided by KT via a LAN (local area network) cable with download and upload speeds of 824 and 540 Mbps, respectively. We conducted a performance evaluation in a good environment and hosted the Rest API on a local server provided by XAMPP.

After 100 iterations, the average response time for generating the dwelling time prediction was 1761.7 ms, with the fastest time being 1688 ms and the slowest one being 2202 ms. Furthermore, when feeding the same input features 100 times, the ML model predicted the same dwelling time at every iteration.

In the second evaluation, we measured the accuracy, confusion matrix, classification measures, *F*-score, and coefficients of determination ($R^2$) of the ML model using the built-in functions from Python. Table 7 lists the performance indicators of the ML model.

## 6 | DISCUSSION AND LIMITATIONS

After implementing the web application and Rest APIs, the dwelling time prediction was considered successful. The model produced highly accurate predictions in multiple scenarios regardless of the number of tourists. As shown in Table 7, the model generated high accuracy and $R^2$ values, indicating accurate predictions. The confusion matrix shows that the model predicted each class almost correctly with few false predictions (less than 100 per class). Furthermore, the precision, recall, and *F*1-score ranged from 0.82 to 0.98. Finally, the *F*-test revealed four less relevant features (sex, group type, tourist spot, and weather) and five highly relevant features (age, health, tour experience, temperature, and humidity). Additionally, Figure 10 shows the *F*-test results plotted on a bar graph.

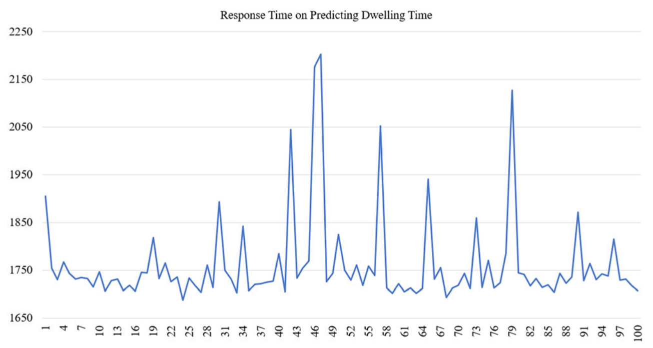


FIGURE 9 Response time for predicting dwelling time.

TABLE 7 ML model performance measure result (accuracy = 0.9075).

| **(A) Confusion matrix** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Class label** | | | | | | | |
| 60 | 90 | 120 | 150 | 180 | 210 | 240 | 270 |
| 849 | 11 | 1 | 2 | 1 | 0 | 0 | 0 |
| 15 | 642 | 12 | 3 | 0 | 0 | 0 | 0 |
| 0 | 11 | 682 | 21 | 5 | 1 | 0 | 0 |
| 0 | 6 | 33 | 945 | 66 | 6 | 4 | 0 |
| 0 | 0 | 5 | 98 | 926 | 35 | 9 | 0 |
| 0 | 0 | 0 | 9 | 70 | 607 | 49 | 1 |
| 0 | 0 | 0 | 0 | 4 | 36 | 525 | 15 |
| 0 | 0 | 0 | 0 | 0 | 7 | 19 | 269 |

| **(B) Classification performance** | | | | | |
|---|---|---|---|---|---|
| **Class label** | **Precision** | **Recall** | ***F*1-score** | **Support** | **(C) *F*-test** |
| 60 | 0.98 | 0.98 | 0.98 | 864 | Feature 0: 475.462219 |
| 90 | 0.96 | 0.96 | 0.96 | 672 | Feature 1: 33.636046 |
| 120 | 0.93 | 0.95 | 0.94 | 720 | Feature 2: 259.073428 |
| 150 | 0.88 | 0.89 | 0.88 | 1060 | Feature 3: 18.162922 |
| 180 | 0.86 | 0.86 | 0.86 | 1073 | Feature 4: 378.652529 |
| 210 | 0.88 | 0.82 | 0.85 | 736 | Feature 5: 38.023789 |
| 240 | 0.87 | 0.91 | 0.89 | 580 | Feature 6: 85.724528 |
| 270 | 0.94 | 0.91 | 0.93 | 295 | Feature 7: 1234.549124 |
| **Accuracy** | | | 0.91 | 6000 | Feature 8: 719.961219 |
| **Macro average** | 0.91 | 0.91 | 0.91 | 6000 | **(D) $R^2$** |
| **Weighted average** | 0.91 | 0.91 | 0.91 | 6000 | 0.9682804992470979 |

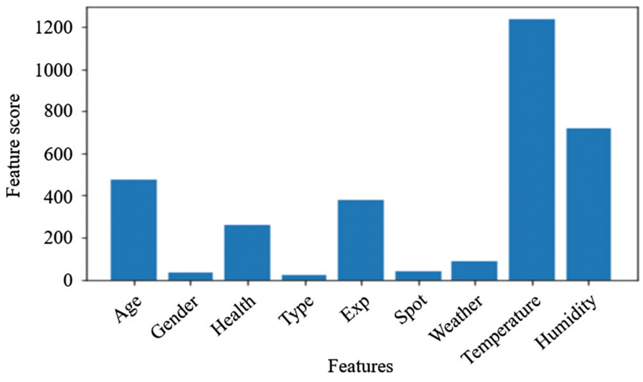


FIGURE 10 Bar graph of *F*-test scores.

Although the ML model and entire system performed satisfactorily, several limitations remain to be addressed. Tour guides or travel agents manually included weather conditions, temperature, and humidity. Furthermore, the dwell time was limited to eight classes. Therefore, the system predicted activities requiring less than 60 min and more than 270 min as 60 min and 270 min, respectively. Another limitation is that the prediction features were only available for the web applications, even if an updated schedule with the predicted dwelling time could be seen in the mobile application for tourists. Therefore, if a tour guide required the generation of new predictions in the middle of a tour, this could only be performed via the web application.

## 7 | CONCLUSION AND FUTURE WORK

We introduce an ML-based system for tourist dwelling time prediction to manage package tours implemented in a web application. The prediction helps a travel agent to find the optimal dwelling time for the tourists to stay at a particular spot to avoid boredom owing to a long stay or underwhelming owing to a brief stay.

A classification problem was formulated because the dwelling time was estimated as a specific timeslot instead of a continuous time value. After comparing several classification algorithms, we found that the *K*NN classification algorithm yielded the best results when a tailored DB was used. After hyperparameter tuning, we trained an ML model and exported it as a file for a Rest API. We implemented two Rest APIs, a PHP Rest API connected to the web application and a Python Rest API managing the data in the cloud DB to convert the data into a suitable format. The Python Rest API was connected to the PHP Rest API and ML model, which predicted the dwelling time and returned the data to the PHP Rest API.

To predict the dwelling time in the web application, a travel agent must configure tourist spot conditions, such as weather state, temperature, and humidity, on the corresponding page and then click the button for prediction. When clicked, the Rest API obtained the tourist information, combined and formatted all the necessary data, fed the data to the trained ML model for prediction, and uploaded the prediction results to the cloud DB. After generating and uploading the predictions, the web application updated the tour schedule based on the new data.

We evaluated the system performance by measuring 100 times the response time after clicking on the generate prediction button until the results were displayed on the web application. On average, prediction and display required 1.762 s. Furthermore, we observed consistent results from the ML model over 100 measurements.

In future work, a mobile application with the same functionalities will be developed while allowing tour guides to predict new dwelling times during tours. Furthermore, instead of manually inputting the weather conditions, temperature, and humidity, a public weather forecast API will be integrated to fully automate the collection of these data. To further enhance the performance of the ML model, the application will be distributed to several travel agents to obtain feedback from tourists, especially regarding dwelling time predictions, after fully developing the package tour management application.

## CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

## ORCID

*Aria Bisma Wahyutama* https://orcid.org/0000-0003-0327-0543

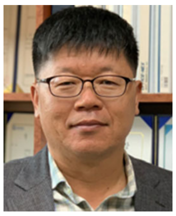
## REFERENCES

1. Y. Kristiana, R. Pramono, and R. Brian, *Adaptation strategy of tourism industry stakeholders during the COVID-19 pandemic: a case study in Indonesia*, J. Asian Financ. Econ. Bus. **8** (2021), 213–223.
2. H. Kamata, *Tourist destination residents' attitudes towards tourism during and after the COVID-19 pandemic*, Curr. Issue Tour. **25** (2022), 134–149.
3. H. Zhang, H. Song, L. Wen, and C. Liu, *Forecasting tourism recovery amid COVID-19*, Ann. Tour. Res. **87** (2021), 103149.
4. The World Bank, International tourism, number of arrivals, 2020. https://data.worldbank.org/indicator/ST.INT.ARVL [last accessed December 2022].
5. G. Richards, *Cultural tourism: a review of recent research and trends*, J. Hosp. Tour. Manag. **36** (2018), 12–21.
6. M. A. Camilleri, *The tourism industry: an overview*, In *Travel marketing, tourism economics and the airline product*, Cham, Switzerland, Springer, 2018, 3–27.
7. G. Meena, D. Sharma, and M. Mahrishi, *Traffic prediction for intelligent transportation system using machine learning*, (Proc. 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things, Jaipur, India), 2020, pp. 145–148.
8. S. Sun, Z. Cao, H. Zhu, and J. Zhao, *A survey of optimization methods from a machine learning perspective*, IEEE Trans. Cybern. **50** (2020), 3668–3681.
9. K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, *Machine learning in agriculture: a review*, Sensors **18** (2018), 2674.
10. R. Peng, Y. Lou, M. Kadoch, and M. Cheriet, *A human-guided machine learning approach for 5G smart tourism IOT*, Electronics **9** (2020), 947.
11. C. Srisawatsakul, and W. Boontarig, *Tourism recommender system using machine learning based on user's public Instagram photos*, (Proc. 5th International Conference on Information Technology Chonburi, Thailand), 2020, pp. 276–281.
12. R. Nagar, Y. Singh, V. Jaglan, and Meenakshi, *A review on machine learning applications in medical tourism*, (Fourth International Conference on Computational Intelligence and Communication Technologies, Sonepat, India), 2021, pp. 208–215.
13. F. Afsahhosseini, and Y. Al-Mulla, *Machine learning in tourism*, (Proc. 3rd International Conference on Machine Learning and Machine Intelligence, Hangzhou, China), 2020, pp. 53–57.
14. S. Ivanov and C. Webster, *Robots in tourism: a research agenda for tourism economics*, J. Tour. Econ. **26** (2020), 1065–1085.
15. H. Go, M. Kang, and S. C. Suh, *Machine learning of robots in tourism and hospitality: interactive technology acceptance model (iTAM)–cutting edge*, J. Tour. Rev. **75** (2020), 625–636.
16. M. O. Parvez, *Use of machine learning technology for tourist and organizational services: high-tech innovation in the hospitality industry*, J. Tour. Futures **7** (2021), 240–244.
17. A. B. Wahyutama and M. Hwang, *Implementation and performance evaluation of package tour management application using geofence technology*, J. Korea Inst. Inf. Commun. Eng. **26** (2022), 85–93.
18. A. B. Wahyutama and M. Hwang, *Design and implementation of digital game-based contents management system for package tour application*, J. Korea Inst. Inf. Commun. Eng. **26** (2022), 872–880.
19. A. B. Wahyutama and M. Hwang, *Implementation of digital game-based learning feature for package tour management application*, J. Korea Inst. Inf. Commun. Eng. **26** (2022), 1004–1012.
20. K. S. Dahiya and D. K. Batra, *Tourist decision making: exploring the destination choice criteria*, Asian J. Manag. Res. **7** (2016), 140–153.

21. A. B. Wahyutama, and M. Hwang, *Design of optimal dwelling time scheduling for package tour by machine learning-based prediction model*, (Proc. International Conference on Future Information & Communication Engineering, Jeju Island, Korea), 2022, pp. 13–17.
22. A. B. Wahyutama and M. Hwang, *Comparison of machine learning algorithms to predict optimal dwelling time for package tour*, Electron. Lett. **58** (2022), 902–904.

## AUTHOR BIOGRAPHIES

**Aria Bisma Wahyutama** received his BE degree in Informatics Engineering from the Department of Informatics Engineering, Pasundan University, Bandung, Indonesia, in 2020. He then received his MSE degree in Information and Communication Engineering from the Department of Information and Communication Engineering, Changwon National University, Changwon, Republic of Korea, in 2022 and is continuing his PhD studies at the same institution. His research interests are web and mobile programming, database design, digital game-based learning, IoT applications, and related topics.



**Mintae Hwang** received his BS, MS, and PhD degrees in Computer Engineering from the Department of Computer Engineering, Pusan National University, Pusan, Republic of Korea in 1990, 1992, and 1996, respectively. From 1996 to 1999, he worked as a senior research member of Protocol Engineering Center, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. Since 1999, he has been a professor at the Department of Information and Communication Engineering, Changwon National University, Changwon, Republic of Korea. His research interests are communication protocols, database design, IoT applications, machine learning, smart cities, and related topics.

**How to cite this article:** A. B. Wahyutama and M. Hwang, *Optimal dwelling time prediction for package tour using* K-*nearest neighbor classification algorithm*, ETRI Journal **46** (2024), 473–484. DOI 10.4218/etrij.2022-0454