Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.033342
Article

Android IoT Lifelog System and Its Application to Motion Inference

Munkhtsetseg[1] and Jeongwook Seo[2,*]

[1]Next Social Platform LLC., Ulaanbaatar, 15150, Mongolia
[2]Department of IT Transmedia Contents, Hanshin University, Osan-si, 18101, Korea
*Corresponding Author: Jeongwook Seo. Email: jwseo@hs.ac.kr
Received: 14 June 2022; Accepted: 24 August 2022

Abstract: In social science, health care, digital therapeutics, etc., smartphone data have played important roles to infer users' daily lives. However, smartphone data collection systems could not be used effectively and widely because they did not exploit any Internet of Things (IoT) standards (e.g., oneM2M) and class labeling methods for machine learning (ML) services. Therefore, in this paper, we propose a novel Android IoT lifelog system complying with oneM2M standards to collect various lifelog data in smartphones and provide two manual and automated class labeling methods for inference of users' daily lives. The proposed system consists of an Android IoT client application, an oneM2M-compliant IoT server, and an ML server whose high-level functional architecture was carefully designed to be open, accessible, and internationally recognized in accordance with the oneM2M standards. In particular, we explain implementation details of activity diagrams for the Android IoT client application, the primary component of the proposed system. Experimental results verified that this application could work with the oneM2M-compliant IoT server normally and provide corresponding class labels properly. As an application of the proposed system, we also propose motion inference based on three multi-class ML classifiers (i.e., k nearest neighbors, Naive Bayes, and support vector machine) which were created by using only motion and location data (i.e., acceleration force, gyroscope rate of rotation, and speed) and motion class labels (i.e., driving, cycling, running, walking, and stilling). When compared with confusion matrices of the ML classifiers, the k nearest neighbors classifier outperformed the other two overall. Furthermore, we evaluated its output quality by analyzing the receiver operating characteristic (ROC) curves with area under the curve (AUC) values. The AUC values of the ROC curves for all motion classes were more than 0.9, and the macro-average and micro-average ROC curves achieved very high AUC values of 0.96 and 0.99, respectively.

Keywords: Android; Internet of Things; lifelog; motion inference; oneM2M

1 Introduction

Smartphones with various sensors have enabled researchers to develop and commercialize many useful systems and applications in different domains such as social science, health care, digital therapeutics, etc. [1–9]. A heart rate monitoring device working on a smartphone application was presented for the elderly and for

patients with heart disease in [1], the structure of social networks constructed with smartphone datasets was analyzed [2], and personal lifelog data generated from smartphones were used for modeling and discovering human behavior for identification purpose [3]. A smartphone based framework called Smart Diary was developed to infer, predict, and summarize people's behavior patterns and life styles, and it could interwork with a wearable galvanic skin response signal monitor [4]. To create practical guidelines on the use of smartphones as a behavioral observation tool in psychological science, highlighted areas of opportunity for psychological research and practical considerations, including ongoing methodological and ethical challenges, for designing smartphone studies were discussed in [5]. Two different smartphone-based monitoring systems (i.e., the Pulso and the Trilogis-Monsenso) were compared to examine their feasibility and usability in patients with bipolar disorder in [6]. As a pilot study, the authors in [7] developed a digital therapeutic software system for hypertension based on a smartphone and a home blood pressure monitoring device. Nowadays, Ribiro et al. in [8] narratively reviewed the existing literature over the past decade which were related to lifelogging, a process analyzing and understanding personal experiences and behaviors by using images, audio, location, physical activity, and physiological signals from smartphones and wearable devices. By narratively reviewing smartphone-sensing literature from the past 5 years in [9], Kulkarni et al. highlighted the predominance of mental health studies and discussed the opportunities of using standardized sensing approaches and machine-learning (ML) advancements. However, the previous studies mentioned above did not exploit any standardized approaches which can conserve resources for developing and deploying smartphone-based applications and also accelerate research on them.

The Internet of Things (IoT) has been a promising technology in many vertical industries such as healthcare, agriculture, smart home, smart factory, smart city, etc. because that enables advanced services by interconnecting physical and virtual things (e.g., sensors, actuators, etc.) and collecting various data from them [10–14]. Yun et al. in [10] presented an oneM2M standards-compliant device software platform for consumer electronics to lead to interoperability across different IoT consumer electronics where oneM2M is a global partnership initiative between the world's standards development organizations (SDOs) to develop standards (i.e., technical specifications) that ensure the most efficient deployment of the IoT and machine-to-machine (M2M) communications. Kim et al. in [11] introduced and tested standardized interworking interfaces and procedures based on oneM2M standards to verify use cases involving multiple IoT platforms. The authors in [12] designed and implemented a smart home scenario by using oneM2M-based IoT platform where the IoT end-user Android application was developed only for monitoring the data from remote sensors and actuators. In [13], the authors emphasized the importance of IoT-based health big-data process technologies to reduce medical data management costs and enhance safety, and the authors in [14] addressed massive and cellular IoT in 5G as new studies for enhancement of the 5G core network. Unfortunately, to the best of our knowledge, we could not find any study published in the literature that proposed smartphone data collection systems based on the global IoT standards, namely oneM2M to collect and utilize users' lifelog data effectively and widely.

Therefore, this paper proposes an Android IoT lifelog system whose high-level functional architecture was designed according to the oneM2M standards to collect lifelog data in smartphones such as motion, location, phone call, short message service (SMS) message, application usage statistics, etc. and provide two manual and automated class labeling methods necessary for machine learning services (e.g., motion inference). In addition, for motion inference as one of the applications of the proposed system, we present three multi-class ML classifiers (i.e., k nearest neighbors, Naive Bayes, and support vector machine) created by using only motion and location data and motion class labels. The rest of the paper is organized as follows. In Section 2, the proposed Android IoT lifelog system with an Android IoT client application, an oneM2M-compliant IoT server, and an ML service is explained in details. In Section 3, we describe implementation and experimental results of the proposed system and then compare performances of three multi-class ML classifiers for motion inference in terms of confusion matrices or receiver operating characteristic (ROC) curves with area under the curve (AUC) values. In Section 4, the concluding remarks are given.

2 Android IoT Lifelog System

The proposed Android IoT lifelog system is illustrated in Fig. 1. It has three main components: An Android IoT client application, an oneM2M-compliant IoT server, and an ML service. First, the Android IoT client application collects all data related to a personal record of one's daily life, referred to as a lifelog, from a smartphone. In detail, the sensor manager offers accelerometer and gyroscope data, and the location manager does longitude, latitude, and speed data. Also, the content provider provides application usage statistics and calendar events, and the broadcast receiver does phone calls and short message service (SMS) events. The open weather application programming interface (API) provides weather and air pollution information. Then, all data are stored in the SQLite database until the synchronization service transmits them to the thing adaptation software (TAS) through socket communication running in the background. The TAS adapts received data to an open IoT client platform called &Cube [12]. The &Cube modeled as an application dedicated node-application entity (ADN-AE) sends requests to an open IoT server platform called Mobius [12] modeled as an infrastructure node-common service entity (IN-CSE) through a reference point M2M communication with application entity (Mca) to ask for creating its registration, access control policy, initial resources such as container, content instance, subscription etc. The Mobius processes the requests and generates the corresponding responses. In addition, it stores all data from the &Cube in the MySQL database.

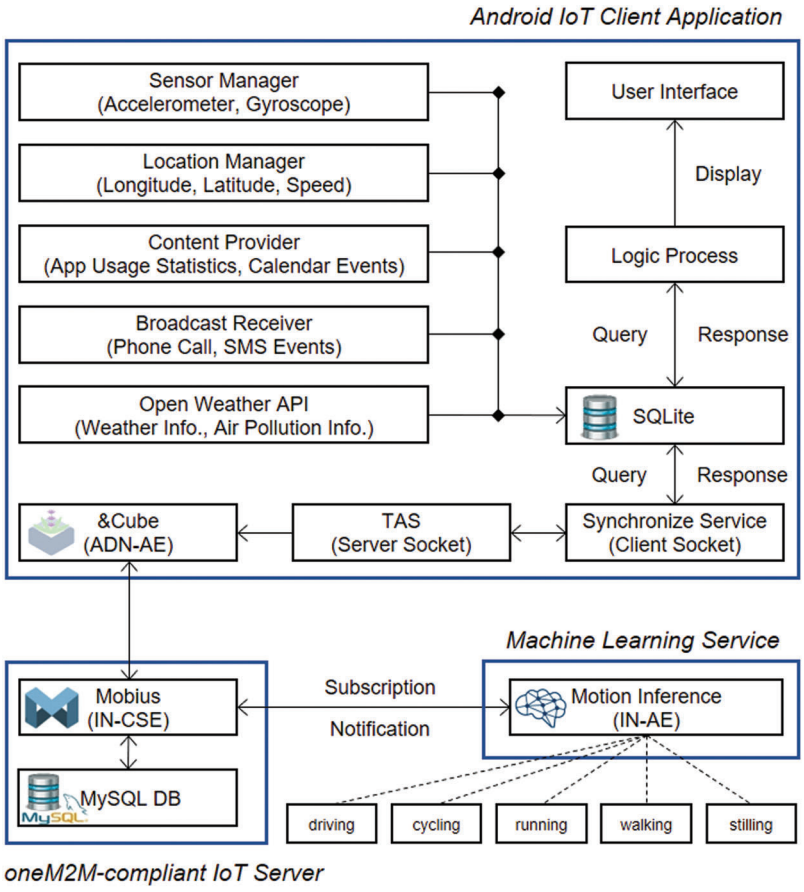

Figure 1: Block diagram of the proposed Android IoT lifelog system with an Android IoT client application, an oneM2M-compliant IoT server, and a machine learning service

In Tab. 1, we describe the data in detail. We define eight types of Android IoT lifelog data: motion, location, phone call, short message service (SMS) message, application usage statistics, calendar event, weather information, and air pollution information. We can use motion data to infer mobility-related user activities such as driving, cycling, running, walking, and stilling. Location data provides hints on visited places and location context and it also can be used to infer mobility-related user activities. Based on phone calls and SMS data, we track user's human relationships and social activities. By analyzing application usage statistics, we guess user's interests and preferences. The events in a calendar tell us important schedules such as appointments, birthdays, and business meetings. They allow us to estimate how busy or free a user will be and predict whom the user will meet and when and where the user will go. Weather and air pollution information reveals the user's environmental conditions. We might find its effects on the patterns of user's movements and daily activities if we analyze it with other data.

Table 1: Description of Android IoT lifelog data in the MySQL database

Data type	Details
Motion	Accelerometer (x, y, z), gyroscope (x, y, z), datetime
Location	Latitude, longitude, speed, datetime
Phone call	Phone number, call start datetime, call end datetime, call time (duration), call type (incoming, outgoing, missed)
SMS message	Address (phone number), message type (receive, send), datetime
App usage statistics	App name, duration, end datetime, start datetime, last datetime
Calendar event	Title, begin datetime, end datetime
Weather information	Weather, humidity, wind speed, temperature, max temperature, min temperature, datetime
Air pollution information	Air quality index, CO (Carbon monoxide), NO (Nitrogen monoxide), NO2 (Nitrogen dioxide), O3 (Ozone), SO2 (Sulphur dioxide), PM2.5 (Fine particles matter), PM10 (Coarse particulate matter), NH3 (Ammonia), datetime

2.1 Android IoT Client Application

This section explains activity diagrams of the proposed Android IoT client application to explain its workflow graphically. In Fig. 2, the activity diagram shows the workflow from login to main activity. If a user runs the application, it first checks permissions such as Internet, access coarse location, access fine location, access background location, read phone state, read SMS, read contacts, read calendar, process outgoing calls, etc. In the login process, a user has to insert a unique identifier (ID) such as his/her phone number which is also used as the ADN-AE name in the registration process of the oneM2M-compliant IoT server. After checking or verifying user registration, Internet connection, and ID, the Android IoT client application starts the main activity.

In Fig. 3, the user interface (UI) of the main activity component has six fragments. During its lifecycle, each fragment defines and manages its own layout, and it can handle its own input events and data described in Tab. 1. In the Looper, the data stored in the SQLite database are added to the ListView and displayed on the smartphone screen through the Adapter.

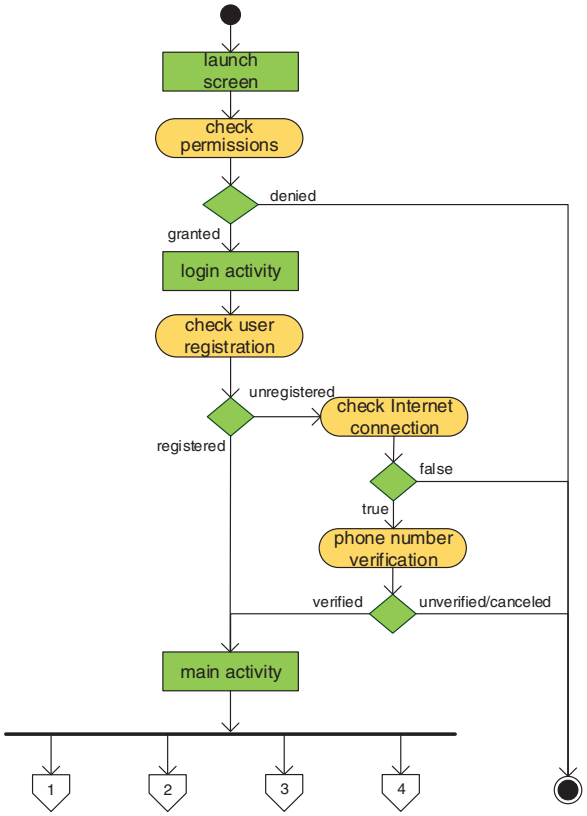

Figure 2: Activity diagram from login to main activity

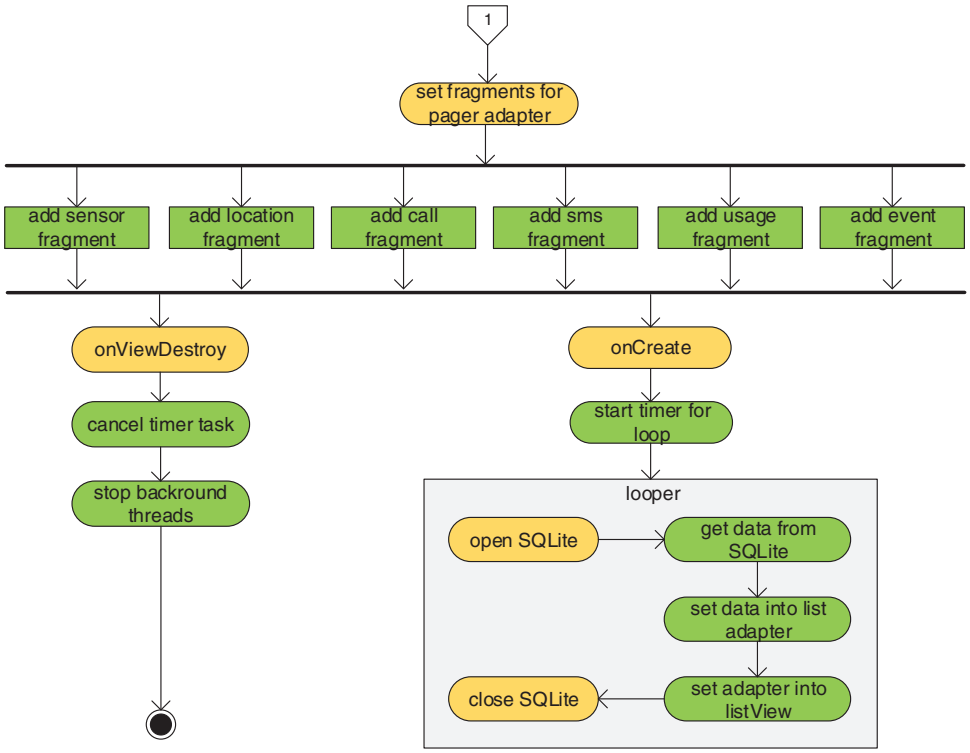

Figure 3: Six fragments of the main activity component to display the data stored in the SQLite database on the smartphone screen

Fig. 4 shows the broadcast receiver component that receives phone call or SMS and stores their data in the SQLite database through the Intent. The service component running in the background is also shown in Fig. 4 where location, sensor, and calendar services are used for repetitively checking for their new data and storing them in the SQLite database.

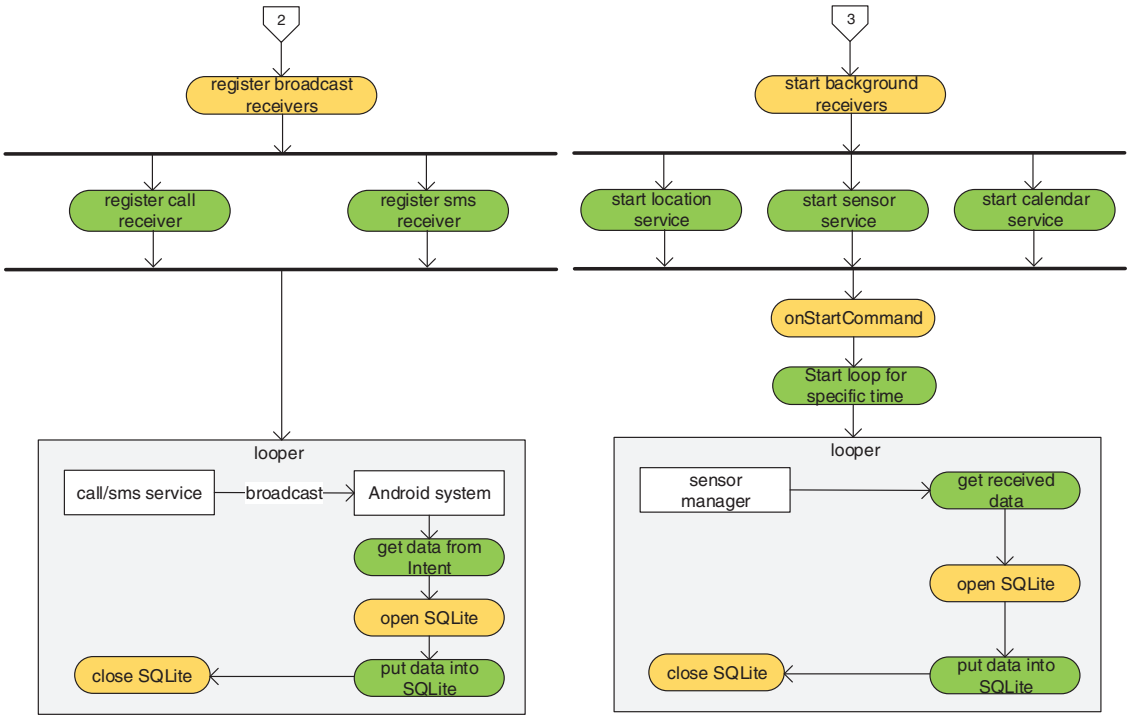

Figure 4: Two broadcast receiver components for phone call and SMS and three background services for location, sensor, and calendar to store each data in the SQLite database

Fig. 5 presents the synchronized service component as a client socket to send the data stored in the SQLite database and the app usage statistics to the Things Adoption Software (TAS) server as a server socket once a day. The stored data and app usage statistics are saved to a temp list which will be changed to the JavaScript Object Notation (JSON) array object when the socket connection is successful. The JSON array object is sent to the TAS server, and then all data are removed from the SQLite database.

2.2 Machine Learning for Motion Inference

Machine learning, referred to as ML, is a branch of artificial intelligence that enables machines to automatically learn and improve from experience without being explicitly programmed, and machine learning algorithms for classification problems have been applied to nearly every industry from healthcare to web services [15–18].

In this section, we present three multi-class ML algorithms used for motion inference. We assign five mobility-related user activities: driving, cycling, running, walking, and stilling as motion class labels, and we only consider motion data (acceleration force along x, y, z axis, gyroscope rate of rotation around x, y, z axis) and location data (speed kilometers/hour) stored in the MySQL database in the Mobius server.

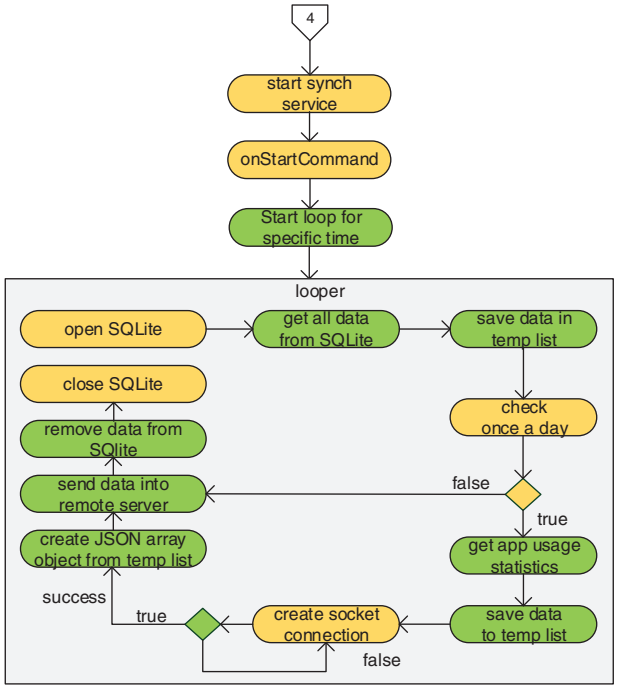

Figure 5: Synchronize service to transmit all data to the TAS server

A training dataset D_N can be represented by

$$D_N = \{(\mathbf{x}_n,\ y_n)\}_{n=1}^{N} \tag{1}$$

where $\mathbf{x}_n = \left[x_{1,n},\ x_{2,n},\ \cdots,\ x_{M,n}\right]^T$ denotes a training feature vector of length M, and $y_n \in \{1,\ 2,\ \cdots,\ C\}$ denotes a corresponding one among C class labels. We apply three ML classifiers created by the training dataset for motion inference.

First, the k nearest neighbors (kNN) classifier is a kind of instance-based classifier that is a simple non-parametric algorithm abstracting no information from the training dataset during the training stage [19,20]. For a new instance or feature vector $\mathbf{x}_{new}$, it is labeled as the same class as that of the majority of its k nearest neighbors, which is shown in Eq. (2). The nearest neighbors are measured by the Euclidean distance function in Eq. (3).

$$\hat{y}_{new} = \underset{c=1,2,\cdots,C}{\arg\max} \sum_{i=1}^{k} I(c,\ y_i) \tag{2}$$

where $I(a,\ b)$ denotes an indicator function that $I(a,\ b) = 1$ if $a = b$ and $I(a,\ b) = 0$ otherwise.

$$d(\mathbf{X}_n,\ \mathbf{X}_{new}) = \|\mathbf{X}_n - \mathbf{X}_{new}\|_2 = \sqrt{\sum_{m=1}^{M} \left(x_{m,n} - x_{m,new}\right)^2} \tag{3}$$

The kNN is suitable for small datasets, and it does not learn anything from the training data. If the dataset is large with high dimensions, the searching of k neighbors can be slow and inefficient. Also, it can be easily affected by noisy data and outliers [19].

Second, the Naive Bayes classifier is a simple probabilistic classifier that depends on Bayesian theorem. Here, we define $P(c)$ as the probability of the occurrence of the class c. The conditional probability of a new instance, called a likelihood function, is given by

$$P(\mathbf{X}_{new}|c) = \prod_{m=1}^{M} P\left(x_{m,new}|c\right) \tag{4}$$

Then, the Naive Bayes classifier can be represented by

$$\hat{y}_{new} = \underset{c=1,2,\cdots,C}{\arg\max}\, P(c) \prod_{m=1}^{M} P\left(x_{m,new}|c\right) \tag{5}$$

The advantages of the Naive Bayes classifier are that it is easy to handle large amount of data, and fast to train and make classifications and it is not sensitive to irrelevant features. However, the Naive Bayes classifier assumes features are independent, and it is loss of accuracy which is its drawbacks [19].

Third, the support vector machine (SVM) method is considered as an optimization algorithm. The basic ideas of SVMs are that it creates optimal hyperplane for linearly separable patterns, and for the patterns that are not linear separable, kernel functions can be used to transform the original data into new space [19,21]. SVMs and kernel methods have a theoretical model that guarantees the performance. In addition, SVMs are not affected by local minimum and the curse of dimensionality. We consider one-versus-all (OvA) multi-class SVM classifier with the radial basis function kernel which can be represented by.

$$\hat{y}_{new} = \underset{c=1,2,\cdots,C}{\arg\max} \sum_{n\in SV} \alpha_{c,n} y_{c,n} K(\mathbf{X}_n,\ \mathbf{X}_{new}) + b_c \tag{6}$$

where SV denotes a set of support vectors, and $\alpha_{c,n}$ denotes a Lagrange multiplier with a constraint $0 \le \alpha_{c,n} \le \delta$ for the class c. Here, δ denotes a hyper-parameter controlling trade-off between slack variable penalty and the margin [19]. Also, $y_{c,n}$ denotes a temporary label for the class c, and b_c denotes a bias term for the class c. The radial basis function kernel is given by

$$K(\mathbf{X}_n,\ \mathbf{X}_{new}) = \exp\left(-\gamma \|\mathbf{X}_n - \mathbf{X}_{new}\|_2^2\right) \tag{7}$$

where γ denotes a hyper-parameter defining how far the influence of a single training sample $\mathbf{X}_n$ reaches as support vectors [19], and $\|\cdot\|_2$ denotes the L2 norm. The OvA is a heuristic method using binary classifiers for multi-class classification by splitting a multi-class dataset into multiple binary classification problems [21]. The SVM classifier has low generalization error and no distribution requirement. Also, it is computationally inexpensive and easy to interpret results. However, its performance is sensitive to two hyper-parameters of δ and γ [19].

3 Experimental Results

The proposed Android IoT lifelog system was developed by using oneM2M-compliant IoT client and server platforms called &Cube and Mobius as mentioned before. In Fig. 6, we presented implementation results of the Android IoT client application working on Android 10. It was developed according to the activity diagrams explained in Section 2. The two images on the top-left corner show the login process where a user enters his/her phone number and the verification code for permission to the Android IoT client application. The phone number are also used as the AND-AE name for the registration process between the Android IoT client application and the oneM2M-compliant IoT server. The other six images present motion data, call log, SMS log, application usage statistics, calendar events, and location data,

respectively. In summary, all data can be viewed on monitoring fragments as well as stored in the SQLite database.

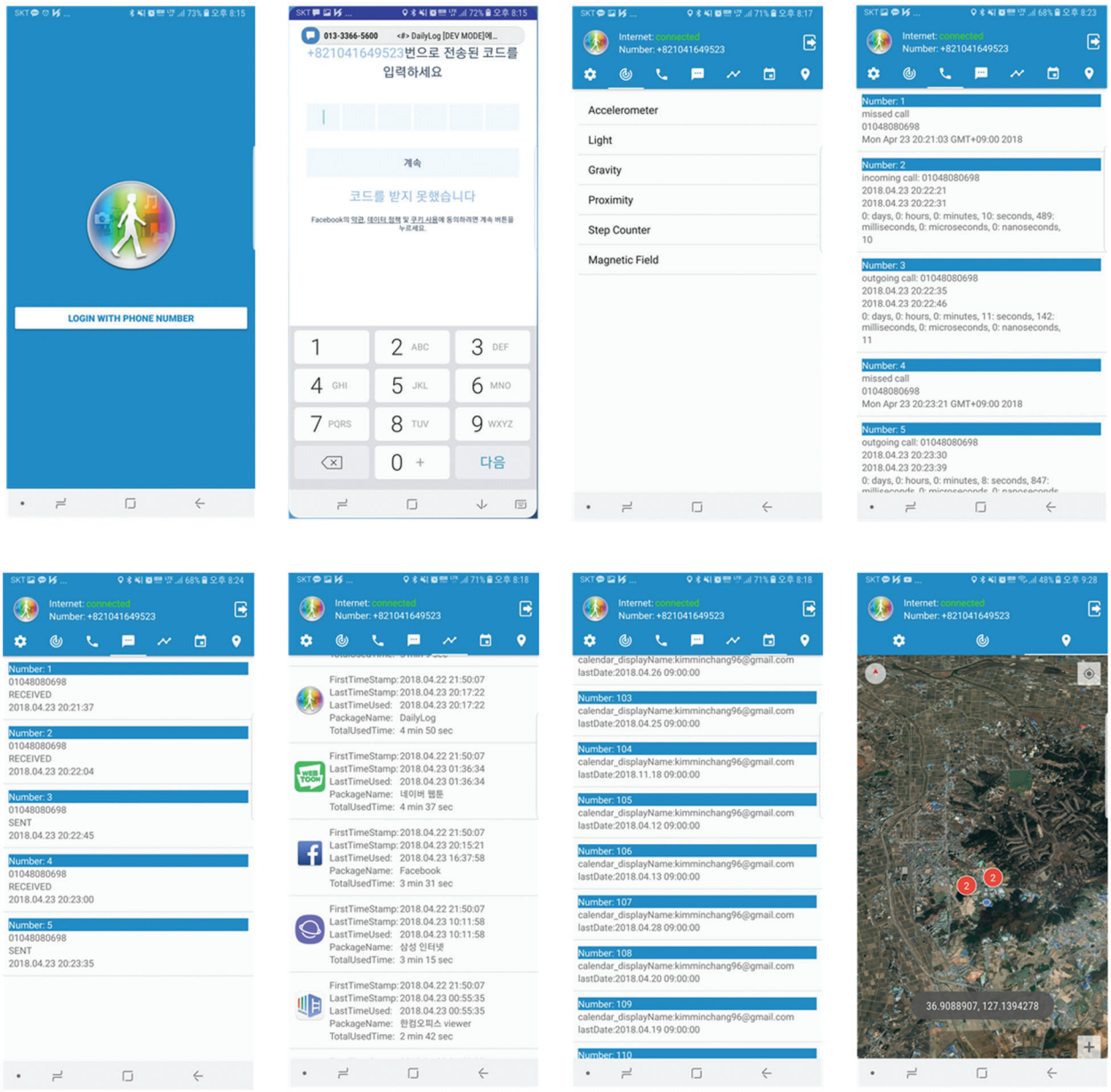

Figure 6: Implementation results of the Android IoT client application working on Android 10

In Fig. 7, we presented a resource tree of the oneM2M-compliant IoT server that displays all data received from the Android IoT client application, called Android IoT lifelog data described in Tab. 1. The resource tree has eight container resources such as *Sensors*, *Gps*, *Call*, *Sms*, *Usage*, *CalEvent*, *Weather*, and *AirPollution* to store corresponding content instance resources including the Android IoT lifelog data. For instance, a content instance resource including motion data such as accelerometer and gyroscope is shown in detail. If you look at the key-value pairs in the JSON format, you can find accelerometer and

gyroscope data. These experimental results confirm that the proposed Android IoT lifelog system is running normally. Moreover, there are two key-value pairs for manual and automated motion class labeling (see the keys of *Activity state button* and *Activity state google*). As mentioned before, the motion class labels such as driving, cycling, running, walking, and stilling will be used to build a dataset for motion inference. The manual motion class labeling denotes that the user should manually push motion class buttons in the Android IoT client application before measuring and storing motion and location data. The automated motion class labeling was implemented by using the Activity Recognition API for Android. We primarily used the manual method and considered the automated one to verify the class labels resulting from the manual one.

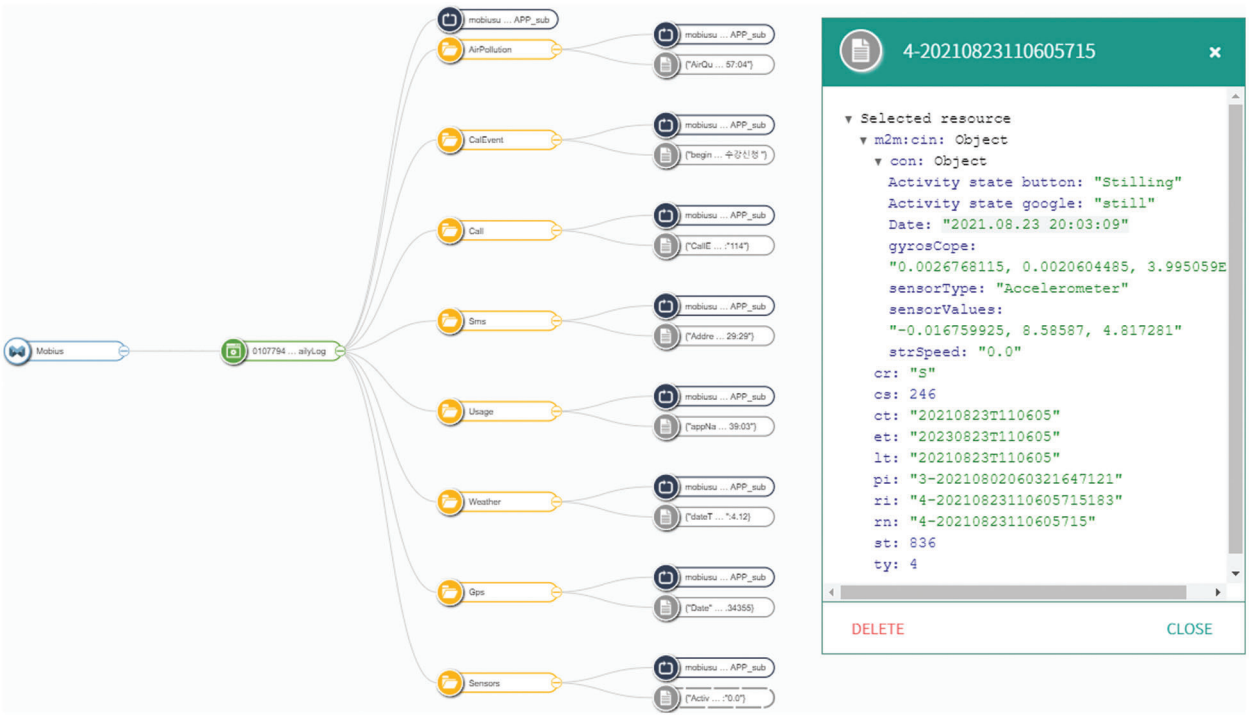

Figure 7: Resource tree of the oneM2M-compliant IoT server for monitoring Android IoT lifelog data

To build a dataset for motion inference, three persons collected the motion and location data for two months through their own Android IoT client applications. In Fig. 8, we visualized some samples in the dataset in a two-dimensional space where acceleration force, gyroscope rate of rotation, and speed concerning the driving class were shown according to arbitrary time indices. By analyzing the curve of speed changes, we can estimate that a car stopped for a while at about 2200 time index. If you look at about 6000 time index, you can find that the car was traveling at about 100 km/h.

We trained and evaluated three multi-class ML classifiers for motion inference in terms of confusion matrices [22,23]. In Tab. 2, we show the confusion matrix of the kNN classifier with a default hyper-parameter, $k = 5$. Actual motion class labels are listed along the y-axis, and predicted motion class labels are listed along the x-axis. The correct classifications are along the diagonal, whereas all the other entries show misclassifications. The rightmost column shows the accuracy for each actual motion class label. In this confusion matrix, it can be seen that 726 samples of the driving class are correctly predicted, and 84 remaining samples are wrongly predicted. The stilling class has the best accuracy of 99.50%, while, in contrast, the running class has the worst accuracy of 71.36%. The confusion matrix of the Naive Bayes

classifier is presented in Tab. 3. It can be seen that the stilling class has the best accuracy of 94.53%, and the walking class does low accuracy of 77.15%. The driving, cycling, and running classes have very low accuracies of less than 65%. Overall, the Naïve Bayes classifier shows poor performance in terms of the accuracy metric. In Tab. 4, the confusion matrix of the SVM classifier with default hyper-parameters, $\delta = 1$ and $\gamma = 0.14$ is presented. It can be seen that the stilling class has the best accuracy of 99.57%, and the running class does the worst accuracy of 55.12%. The cycling class still has low accuracy of 76.79%. When compared with confusion matrices of three multi-class ML classifiers, it can be observed that, in general, the kNN classifier outperforms the other Naive Bayes and the SVM classifiers. The reason is mainly that insufficient datasets were used to create the other two classifiers. On the other hand, the kNN classifier works well for a small dataset. In addition, the scale and distribution of motion and location data for running and walking classes may be indistinguishable.

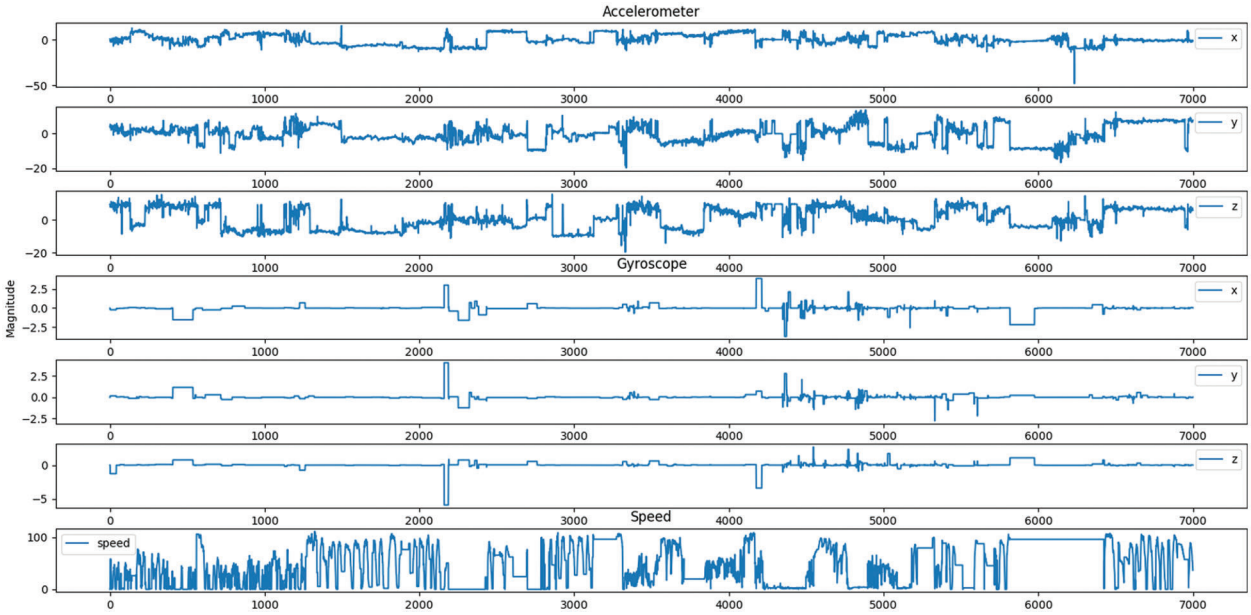

Figure 8: Graphical visualization of motion and location data with respect to the driving class: acceleration force along x, y, z axis, gyroscope rate of rotation around x, y, z axis, and speed

Table 2: Confusion matrix of motion inference using multi-class kNN classifier

		Predicted motion				Accuracy (%)	F1 Score	
		Driving	Cycling	Running	Walking	Stilling			
Actual motion	Driving	726	6	1	32	45	90.98	0.86
	Cycling	21	335	6	34	9	84.19	0.85
	Running	9	26	238	49	10	71.36	0.78
	Walking	63	2	13	2393	147	90.45	0.91
	Stilling	37	1	3	106	33588	99.50	0.99

Table 3: Confusion matrix of motion inference using multi-class Naive Bayes classifier

	Predicted motion					Accuracy (%)	F1 Score	
	Driving	Cycling	Running	Walking	Stilling			
Actual motion	Driving	511	5	5	230	59	63.09	0.70
	Cycling	47	252	18	82	6	62.2	0.71
	Running	24	40	215	40	13	64.75	0.62
	Walking	5	2	103	2020	488	77.15	0.60
	Stilling	65	1	26	1752	31891	94.53	0.96

Table 4: Confusion matrix of motion inference using multi-class SVM classifier

	Predicted motion					Accuracy (%)	F1 Score	
	Driving	Cycling	Running	Walking	Stilling			
Actual motion	Driving	699	3	4	57	47	86.29	0.89
	Cycling	11	311	12	68	3	76.79	0.85
	Running	3	12	183	130	4	55.12	0.68
	Walking	26	0	3	2416	173	92.28	0.89
	Stilling	14	0	4	127	33590	99.57	0.99

In Fig. 9, we evaluate the output quality of the kNN classifier in terms of ROC curves with AUC values [23]. It typically plots a true positive rate on the y-axis and a false positive rate on the x-axis, which means that its top-left corner is the ideal point where the false positive rate is zero, and the true positive rate is one. The ROC curves of all classes (class 0 = driving, class 1 = cycling, class 2 = running, class 3 = walking, and class 4 = stilling) are shown with AUC values. For instance, the ROC curve of class 2 has an AUC value of 0.93. It can be seen that all ROC curves show the AUC value of 0.9. The macro-average ROC curve means the arithmetic average of the ROC curves for all class, and the micro-average ROC curve means a weighted macro-average in which each class contribution to be average is weighted by the relative number of samples available for the class [23]. In other words, the macro-average ROC curve does not take imbalanced datasets into account, but the micro-average ROC curve does [23]. The macro-average ROC curve has the AUC value of 0.96, and the micro-average ROC curve does the AUC value of 0.99.

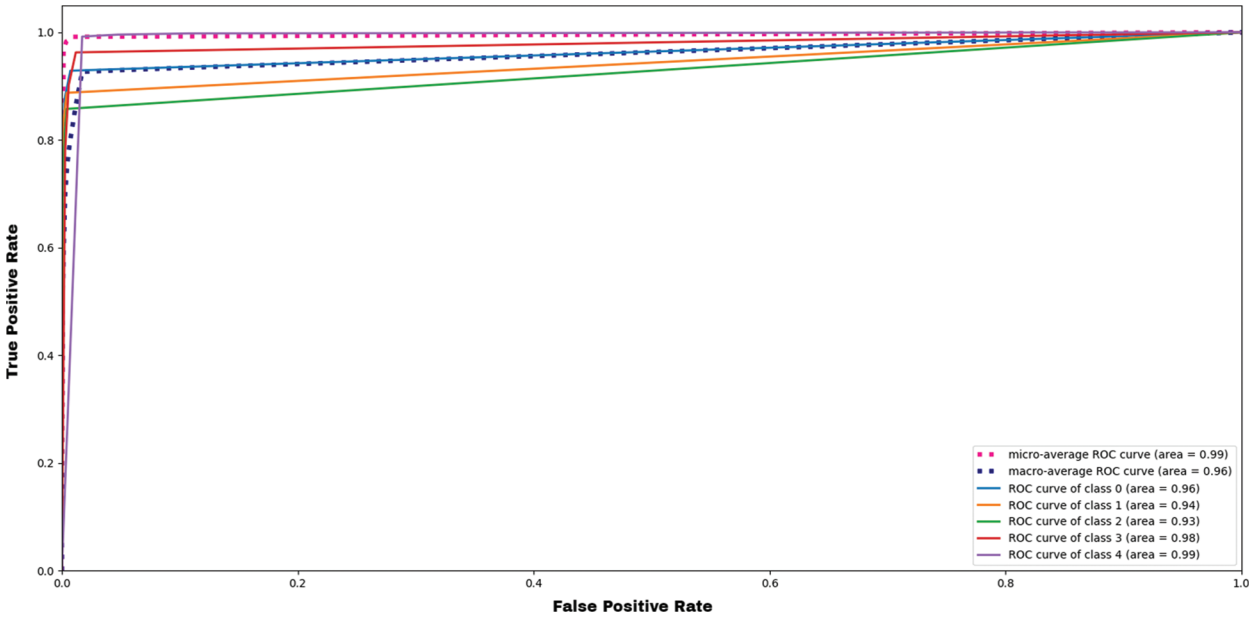

Figure 9: ROC curves of all classes, macro-average and micro-average ROC curves with AUC values to evaluate the output quality of the kNN classifier

4 Conclusions

Smartphone data have played important roles in social science, health care, digital therapeutics, etc. However, existing smartphone data collection systems are not good enough to be effectively and widely used because they do not comply with any IoT standards and not provide any class labeling methods for ML services. In this paper, the Android IoT lifelog system for inferring users' daily lives was proposed to collect various lifelog data in smartphones such as motion, location, phone call, SMS message, application usage statistics, weather information, and air pollution information. The proposed system consisted of an Android IoT client application, an oneM2M-compliant IoT server, and an ML server. It should be noted that its functional architecture (i.e., AND-AE, IN-CSE) was based on oneM2M standards that are open, accessible and internationally recognized. We mainly explained activity diagrams of the Android IoT client application that is the primary component of the proposed system. The advantage of the Android IoT client application is that it works well with the oneM2M-compliant IoT server and that it supports manual and automated class labeling methods for the ML server. Through implementation and experimental results, we showed that the proposed system was running normally. In addition, as an application of the ML server in the proposed system, we developed three multi-class ML classifiers for motion inference where we considered only motion and location data (i.e., acceleration force, gyroscope rate of rotation, and speed) and assumed five motion class labels (i.e., driving, cycling, running, walking, and stilling). Experimental results showed that when compared with confusion matrices of the ML classifiers: a kNN classifier, a Naive Bayes classifier, and a SVM classifier, the kNN classifier was superior to the other two classifiers because small datasets were used for creating them. To evaluate the output quality of the kNN classifier, its ROC curves with AUC values were presented. All ROC curves for all classes achieved the AUC of more than 0.9. The macro-average and micro-average ROC curves showed the AUC values of 0.96 and 0.99, respectively. Accordingly, we found that the kNN classifier, created by motion and location data in the proposed Android IoT lifelog system, can provide very good accuracy.

Funding Statement: This work was supported by Hanshin University Research Grant.

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

References

[1] P. Pierleoni, L. Pernini, A. Belli and L. Palma, "An android-based heart monitoring system for the elderly and for patients with heart disease," *International Journal of Telemedicine and Applications*, vol. 2014, article ID 625156, pp. 1–11, 2014.

[2] V. D. Blondel, A. Decuyper and G. Krings, "A survey of results on mobile phone datasets analysis," *EPJ Data Science*, vol. 4, no. 10, pp. 1–57, 2015.

[3] R. Mafrur, I. G. D. Nugraha and D. Choi, "Modeling and discovering human behavior from smartphone sensing life-log data for identification purpose," *Human-centric Computing and Information Sciences*, vol. 5, no. 31, pp. 1–18, 2015.

[4] J. Liao, Z. Wang, L. Wan, Q. C. Cao and H. Qi, "Smart diary: A smartphone-based framework for sensing, inferring, and logging users' daily life," *IEEE Sensors Journal*, vol. 15, no. 5, pp. 2761–2773, 2015.

[5] G. M. Harari, N. D. Lane, R. Wang, B. S. Crosier, A. T. Campbell *et al.,* "Using smartphones to collect behavior data in psychological science: Opportunities, practical considerations, and challenges," *Perspectives on Psychological Science*, vol. 11, no. 6, pp. 838–854, 2016.

[6] M. Faurholt-Jepsen, E. Torri, J. Cobo, D. Yazdanyar, D. Palao *et al.,* "Smartphone-based self-monitoring in bipolar disorder: Evaluation of usability and feasibility of two systems," *International Journal of Bipolar Disorders*, vol. 7, no. 1, pp. 1–11, 2019.

[7] K. Kario, A. Nomura, A. Kato, N. Harada, T. Tanigawa *et al.,* "Digital therapeutics for essential hypertension using a smartphone application: A randomized, open-label, multicenter pilot study," *Journal of Clinical Hypertension*, vol. 23, no. 5, pp. 923–934, 2021.

[8] R. Ribeiro, A. Trifan and A. J. R. Neves, "Lifelog retrieval from daily digital data: Narrative review," *JMIR Mhealth and Uhealth*, vol. 10, no. 5, pp. 1–20, 2022.

[9] P. Kulkarni, R. Kirkham and R. McNaney, "Opportunities for smartphone sensing in E-health research: A narrative review," *Sensors*, vol. 22, no. 10, pp. 1–21, 2022.

[10] J. Yun, I. Y. Ahn, N. M. Sung and J. Kim, "A device software platform for consumer electronics based on the internet of things," *IEEE Transactions on Consumer Electronics*, vol. 61, no. 4, pp. 564–571, 2015.

[11] J. Kim, J. Yun, S. C. Choi, D. N. Seed, G. Lu *et al.,* "Standard-based IoT platforms interworking: Implementation, experiences, and lessons learned," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 48–54, 2016.

[12] V. Andrianto, J. Lam, R. Widodo, S. -G. Lee, H. -J. Lee *et al.,* "Toward implementation of oneM2M based IoT platform," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 2, pp. 418–425, 2018.

[13] H. Yoo, R. C. Park and K. Chung, "IoT-based health big-data process technologies: A survey," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 3, pp. 974–992, 2021.

[14] S. Husain, A. Kunz and J. Song, "3GPP 5 G core network: An overview and future directions," *Journal of Information and Communication Engineering*, vol. 20, no. 1, pp. 8–15, 2022.

[15] A. A. Farhan, C. Yue, R. Morillo, S. Ware, J. Lu *et al.,* "Behavior vs. introspection: Refining prediction of clinical depression via smartphone sensing data," in *Proc. IEEE WH*, Bethesda, MD, USA, pp. 1–8, 2016.

[16] S. Durga, R. Nag and E. Daniel, "Survey on machine learning and deep learning algorithms used in internet of things (IoT) healthcare," in *Proc. ICCMC*, Erode, Tamil Nadu, India, pp. 1018–1022, 2019.

[17] J. Moon, S. Kim, J. Song and K. Kim, "Study on machine learning techniques for malware classification and detection," *KSII Transactions on Internet and Information Systems*, vol. 15, no. 12, pp. 4308–4325, 2021.

[18] M. Hasnain, I. Ghani, M. F. Pasha and S. R. Jeong, "Machine learning methods for trust-based selection of web services," *KSII Transactions on Internet and Information Systems*, vol. 16, no. 1, pp. 38–59, 2022.

[19] J. Watt, R. Borhani and A. K. Katsaggelos, *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press, Cambridge, UK, 2020.

[20] L. Jiang, Z. Cai, D. Wang and S. Jiang, "Survey of improving K-nearest-neighbor for classification," in *Proc. FSKD*, Haikou, Hainan, China, pp. 679–683, 2007.

[21] C. -W. Hsu and C. -J. Lin, "A comparison of methods for multiclass support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 415–425, 2002.

[22] S. Visa, B. Ramsay, A. Ralescu and E. van der Knaap, "Confusion matrix-based feature selection," in *Proc. MAICS*, Cincinnati, OH, USA, pp. 1–8, 2011.

[23] A. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Sebastopol, CA, USA. 2016.