Internet of Things 18 (2022) 100397

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Internet of Things-enabled Passive Contact Tracing in Smart Cities

Zeinab Akhavan[a,1,*], Mona Esmaeili[a,1], Dimitrios Sikeridis[b], Michael Devetsikiotis[b]

[a] Department of Computer Science, University of New Mexico, Albuquerque, NM 87131, United States
[b] Department of Electrical & Computer Engineering, University of New Mexico, Albuquerque, NM 87131, United States

ARTICLE INFO

Article history:
Received 31 December 2020
Revised 22 March 2021
Accepted 24 March 2021
Available online 31 March 2021

Keywords:
Bluetooth Low Energy
Contact Tracing
Internet of Things
Infection Risk Classification

ABSTRACT

Contact tracing has been proven an essential practice during pandemic outbreaks and is a critical non-pharmaceutical intervention to reduce mortality rates. While traditional contact tracing approaches are gradually being replaced by peer-to-peer smartphone-based systems, the new applications tend to ignore the Internet-of-Things (IoT) ecosystem that is steadily growing in smart city environments. This work presents a contact tracing framework that logs smart space users' co-existence using IoT devices as reference anchors. The design is non-intrusive as it relies on passive wireless interactions between each user's carried equipment (e.g., smartphone, wearable, proximity card) with an IoT device by utilizing received signal strength indicators (RSSI). The proposed framework can log the identities for the interacting pair, their estimated distance, and the overlapping time duration. Also, we propose a machine learning-based infection risk classification method to characterize each interaction that relies on RSSI-based attributes and contact details. Finally, the proposed contact tracing framework's performance is evaluated through a real-world case study of actual wireless interactions between users and IoT devices through Bluetooth Low Energy advertising. The results demonstrate the system's capability to accurately capture contact between mobile users and assess their infection risk provided adequate model training over time.

1. Introduction

Recently, smartphone-based contact tracing has emerged as a practical way to trace someones social exposure with many public-safety-related applications including infectious disease tracking [1]. Such systems can monitor peer-to-peer interactions between citizens and retroactively provide alerts to users that were in contact with someone that has been diagnosed or tested positive for an infectious disease. A case in point for the importance of such systems is the recent COVID-19 pandemic where many governments have been looking for effective ways to relax restrictions, resume industry operations and bring back daily routines without risking dangerous outbreaks [2]. Therefore, effective contact tracing can be added to the relatively short list of outbreak-preventive measures that also includes regular hand washing, face covering, and temperature checks.

* Corresponding author.
E-mail addresses: zakhavan@unm.edu (Z. Akhavan), mesmaeili@unm.edu (M. Esmaeili).
[1] Both authors contributed equally to this work. The names are listed alphabetically.

https://doi.org/10.1016/j.iot.2021.100397
2542-6605/

Traditionally, contact tracing is a manual process that requires the collaboration of multiple authorized entities and personnel resulting in a time-consuming operation [3]. Recently, multiple parties have been developing contact tracing applications that rely on peer-to-peer (P2P) architectures that track interactions between individuals through their smartphones or other smart wearables (e.g., smartwatches [4]). The majority of those applications detect either the location or the proximity of two users, and rely mostly on well-known positioning tools that include GPS [5] and more prominently the Bluetooth Low Energy protocol [4,6–9]. Specifically, the list of contact tracing apps that use BLE as their main technology include the BlueTrace protocol -the basis behind Singapore's TraceTogether app [10]-, Apple/Google's decentralized CT protocol [9], the Pan-European Privacy-Preserving Proximity Tracing (PEEP-PT) [6], Privacy-Preserving Automated Contact Tracing (PACT) [11] among others. The logic behind such applications is simple: User devices' transmit and detect BLE advertisement packets, and utilize them to exchange the users' IDs and calculate their proximity. These encounters are logged either on the device or on a remote central authority to be used in case one of the two parties tests positive (in the case of infectious diseases) or is picked out for any reason. Then, using history logs other contacts of that person will be identified and alerted either through the device itself or by other means.

While the aforementioned approaches are promising in terms of performance, the overall designs tend to ignore the impending fact that a massive Internet of Things' infrastructure is actually embedded within urban areas and smart cities. Such Internet of Things (IoT) infrastructure supports city-wide communications [12–14] and consists of computationally powerful devices including Electric Vehicles, smart locks, cameras, drones, various sensors, smart furniture, and WiFi hotspots [15–18]. Utilizing such networks of interconnected and computationally powerful devices can lead to the development of alternative contact tracing applications or (most probably) reinforce the operation of existing P2P contact tracing and significantly improve their performance and accuracy.

In this work, we specifically address this gap by presenting an IoT-assisted correlative contact tracing framework that relies on passive interactions between mobile users and static IoT devices that are already part of a wider smart city ecosystem. Specifically, we:

- Design a distributed framework that enables IoT devices to harvest passive advertising packets and act as contact tracing anchors to detect and log interactions between two smart space occupants.
- Deploy a centralized solution able to combine information from multiple IoT devices concurrently towards identifying more accurate contact tracing information including duration of interaction, and distance.
- Utilize machine learning methods to accurately classify the infection risk between two users given specific attributes of their interaction.
- Evaluate our design on a real-world case study where 46 participants interact with an IoT infrastructure of 32 devices as part of their daily routine using the Bluetooth Low Energy protocol.

The rest of this paper is organized as follows: Section 2 presents related work while Section 3 presents the overall architecture of the IoT-based contact tracing framework. Section 4 outlines the infection risk classification function along with the utilized machine learning algorithms. Finally, Section 5 presents the systems performance evaluation through a real-world case study, and Section 6 concludes this work.

2. Related Work

Due to the 2019/2020 pandemic [2], a lot of contact tracing systems leveraging BLE technology have been studied, mainly due to its broad availability in wearable devices and its low consumption profile [19–21]. The work in [22] designed an algorithm estimating the distance between users in a Personal Area Network (PAN) through RSSI. Based on the estimation distribution and predefined zones (e.g., safe, moderate, and high-risk range), the system sends the appropriate alert to the users depending on their proximity.

The authors in [8] proposed a BLE based contact tracing method where each user estimates his distance to nearby users using RSS measurements on their smartphones. The proximity measured by the users' smartphones is evaluated based on low and high-risk contact using five different machine learning classifiers. To protect users' sensitive information, the work uses non-connectable BLE transmission to disallow access to other users' smartphones. To harden the privacy, they propose a signature protocol where the BLE packet is encrypted by 31 bytes of information obtained from the nearby devices. This signature will be unique and is unlikely to get duplicated on other occasions. However, this signature requires computation to be generated, preventing scalability, and potentially imposing constraints on network traffic. The work in [4] leverages smartwatches to measure the proximity between users for contact tracing purposes. The proposed application also leverages BLE technology to make contact tracing more convenient for users that do not always carry their smartphones.

The authors in [23] discuss the impact of enhanced contact tracing using sensing technologies. The work proposes an IoT-based contact tracing architecture consisting of three components called user endpoint (UE), facility endpoint (FE), and object endpoint (OE) that measure the infection risk over time. The work in [24] also focuses on an IoT-enabled architecture for contact tracing and focuses on privacy and security issues. Their proposed generic architecture delegates technical contact tracing tasks as well as privacy/security tasks to the deployed IoT devices and health authorities towards avoiding the computational and power limitations of mobile devices carried by users in public places.

Finally, we have recently observed an explosion of the use of blockchain technologies in smart city applications that span across multiple fields from local energy markets [25], and smart grid infrastructure security [26] to e-voting systems

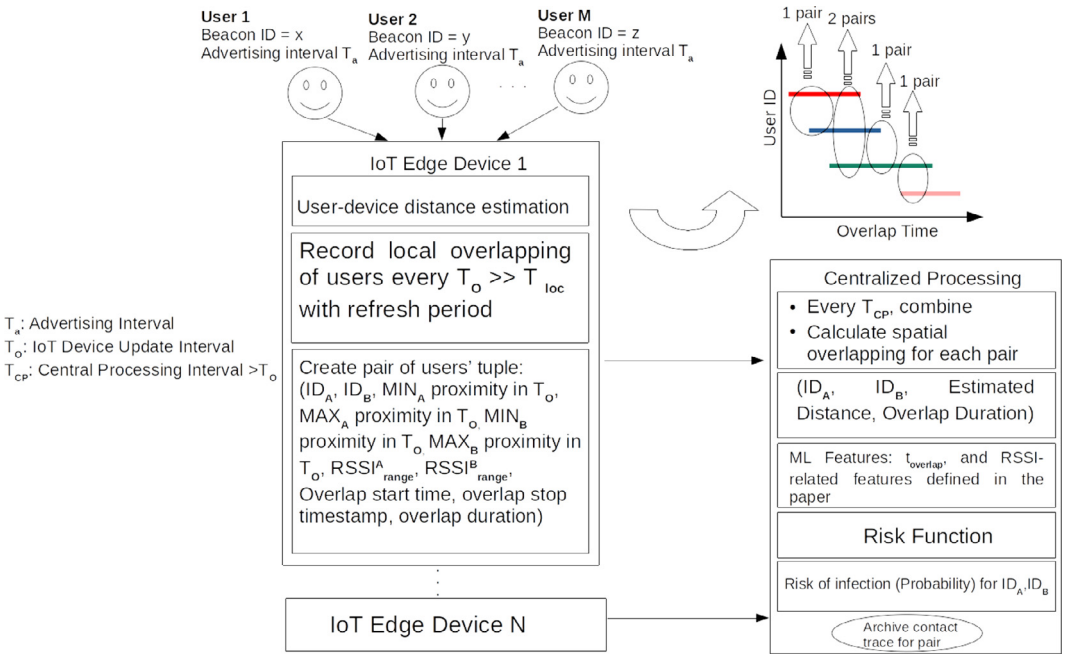

Fig. 1. Framework Architecture.

[27]. Thus, multiple works discuss and investigate the use of blockchain technology for secure data sharing in hospitals [28], and even for COVID-19 detection using CT Imaging [29]. The work in [30] discusses various COVID-19 pandemic challenges and investigates the applicability of blockchain to face them along with the expected performance. Similarly, the authors in [31] investigate practical blockchain applications that attempt to mitigate COVID-19 challenges. The work in [32] presents BeepTrace which is a privacy-preserving blockchain-enabled contact tracing framework. The focus is on reinforcing privacy and security and allows for the decentralization of user identification and location-based information. Finally, the authors in [33] propose a contact tracing framework that uses blockchain technology to decouple any connection between the ownership of on-chain location information and user identity. The proposed framework is evaluated using Bluetooth 5.0 with the focus being on storage/CPU usage, observed delay, energy consumption, and mainly security vulnerabilities.

3. IoT-enabled Contact Tracing Framework

3.1. Architecture

The proposed system utilizes existing IoT devices with multiple wireless interfaces as permanent wireless packet scanners. We will denote as $\mathcal{I}$ the set of $|N|$ IoT devices randomly placed in our smart setting (i.e., following the random placement of IoT devices such as cameras, smart locks, sensors, etc.). We will also denote as $\mathcal{U}$ the set of $|M|$ mobile or static users that are frequent occupants on the smart space and carry their respective personal devices. Our design accounts for any kind of personal smart devices, such as smartphone, smartwatch, or other wearables, as well as any device with connectivity capabilities handed to the users by the facility/city administrators (e.g., smart id cards). Such devices can emit non-intrusive advertising packets by any of their communication interfaces, with the most common being Bluetooth Low Energy (BLE) advertisement packets and WiFi probe request frames. Our design utilizes such passive user-IoT device interactions to calculate the proximity of a user to an IoT device and document their interaction with other users through the common edge device with which they are both associated (common anchor). Each personal device can be anonymously linked to its owner from any wireless interface that might utilize for the contact tracing capability.

Each IoT device reports all the recorded interactions between users to a centralized processing unit (e.g., a local server or cloud environment [34]) that combines information from all sightings and produces a final contact trace between every pair of users. For each pair, our system can calculate the duration of the contact, as well as produce an estimation of the distance between the two users during the contact. Based on the information above, our system outputs an infection risk indicator that will be the outcome of infection probability modeling (performed by a specialist) and machine learning models as described in this work. Fig. 1 shows the overall architecture of the proposed framework, while Table 1 summarizes all the different tuning parameters of the proposed system as found in this section.

Table 1
Key Notations.

Notation	Definition
$\mathcal{I} = \{i_0, \ldots, i_n, \ldots, i_{\lvert N\rvert}\}$	Set of IoT devices
$\mathcal{U} = \{u_0, \ldots, u_m, \ldots, u_{\lvert M\rvert}\}$	Set of users
T_a	User Device Advertising interval
T_o	Update interval of IoT device
$T_{CP} > T_o$	Centralized processing interval
d_{ij}	Estimated distance between user i and user j
t_{ij}	Overlapping time duration for user i and user j

3.2. User-IoT device Distance Estimation using RSSI

Given the constant stream of advertising packets from the users to the IoT devices, our framework's first step is to estimate the user-IoT device distance at any time. This process takes place locally within each IoT device. In what follows, we explain the required data processing steps to estimate the user-device distance.

First, we assume that every personal user device transmits advertising packets every T_a seconds (advertising interval). These packets are picked up by multiple IoT devices, and their Received Signal Strength Indicators (RSSI) are used to calculate the user-IoT device distance locally within each device. To do so, the following process takes place periodically every T_o seconds across all available IoT devices. First, we perform Exponential Smoothing [35] to smooth the RSSI values and remove high bandwidth noise as follows:

$$\hat{rssi}_{t+1|t} = \alpha rssi_t + (1-\alpha)\hat{rssi}_{t|t-1} \tag{1}$$

where $\hat{rssi}_{t+1|t}$ is the estimated RSSI at time $t+1$ is calculated using the weighted average of the most recent $rssi_t$ and the previous estimate $\hat{rssi}_{t|t-1}$. Parameter α controls the weight of the past and recent observations over time (e.g. it associates smaller weights to the older data and greater weights to the recent data).

Next, we apply Kalman filter to provide a better estimation of RSSI values. It consists of two parts: 1) Measurement update (Bayes rule product) and 2) Prediction (total probability convolution). In the update step, two Gaussian distributions used are called prior and measurement. The prior distribution has a mean of m_1 and a variance of ${v_1}^2$, and the measurement distribution has a mean of m_2 and a variance of ${v_2}^2$. We multiply two prior and measurement distributions of RSSI values and obtain the new RSSI measurement using m' and ${v^2}'$. The update equations are as follows:

$$\begin{aligned} m' &= \frac{{v_2}^2 m_1 + {v_1}^2 m_2}{{v_2}^2 + {v_1}^2} \\ {v^2}' &= \frac{1}{\frac{1}{{v_2}^2} + \frac{1}{{v_1}^2}} \end{aligned} \tag{2}$$

In the prediction step, we add up the old mean m_1 and the motion x, and for the variance, we add up the old variance ${v_1}^2$ and ${v_2}^2$. The prediction equations are as follows: x is denoted as motion or estimated RSSI (e.g., if the user moves over to another place with a different received RSSI).

$$\begin{aligned} m' &\leftarrow m_1 + x \\ {v^2}' &\leftarrow {v_1}^2 + {v_2}^2 \end{aligned} \tag{3}$$

Fig. 2 is a visualization of above equations and illustrates the RSSI estimation using two Gaussian distributions. Thus, in this step, we forecast the users' RSSI to the nearby devices.

Finally, the Kalman filter outputs the RSSI estimate values for the specific user. Each RSSI estimation is converted to distance. To do so, we use the following path loss model [37,38] to calculate the corresponding distance value d:

$$p_d = p_{d0} - 10\ \beta\ \log(\frac{d}{d_0}) \tag{4}$$

Where p_d is the observed RSSI value (dBm) d meters away from the IoT device and p_{d0} is the RSSI value (dBm) observed at a reference distance d_0.

3.3. IoT Device Scanning

Every IoT device gathers distance readings from each user in its proximity within the T_o interval. Next, each IoT device locally calculates the overlap time between each pair of users and logs the minimum and maximum distance observed for each of them during T_o. This information is sent at the end of every T_o period to the centralized unit (see Fig. 1). Specifically, each IoT device reports:

- ID_A: ID of the pair's first user
- ID_B: ID of the pair's second user
- $t_{overlap}$: Duration of the pair's encounter

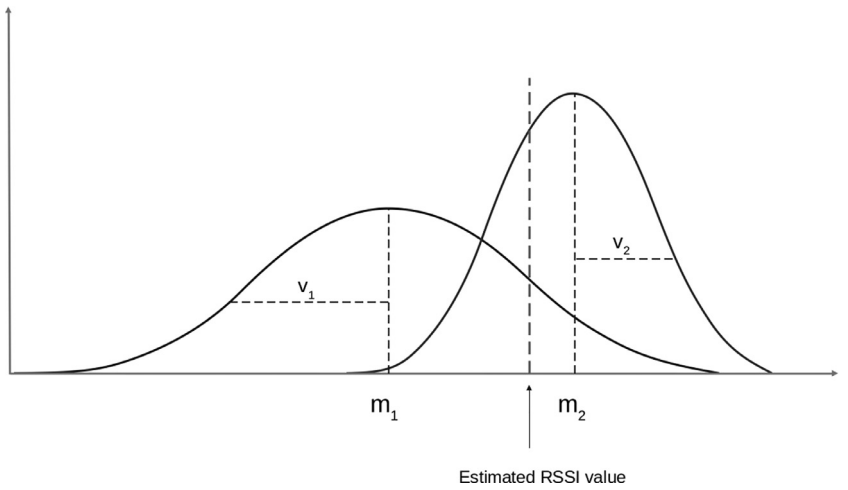

Fig. 2. RSSI Estimation Using Kalman Filter [36].

- min_A: Minimum distance between user A and IoT device within T_o
- max_A: Maximum distance between user A and IoT device within T_o
- min_B: Minimum distance between user B and IoT device within T_o
- max_B: Maximum distance between user B and IoT device within T_o
- $RSSI^A_{range}$: Range of RSSI readings for user A
- $RSSI^B_{range}$: Range of RSSI readings for user B

3.4. Centralized Processing

3.4.1. Combining observations from multiple IoT devices for the same pair

Each user pair's information (overlap time, and distances to the device) may have been collected by multiple nearby IoT devices. Thus, each device will generate alternative values for the same user pair describing the same time interval (T_o). This information is sent to a centralized unit where our framework combines the given distributed information to produce a holistic view for each pair. This process in the central unit happens also during specific time intervals T_{CP} where $T_{CP} > T_o$.

Assuming that one or multiple devices have reported that a specific pair of users A and B are in close proximity, the centralized unit logs the IDs of the interacting users, and their combined duration of interaction:

$$t^{AB}_{overlap} = max({}^1t^{AB}_{overlap}\ {}^2t^{AB}_{overlap}, \ldots,\ {}^nt^{AB}_{overlap}),\ \ with\ n <= |N| \tag{5}$$

In addition, for each user m, we calculate the average RSSI range from all IoT devices:

$$\overline{RSSI^m_{range}} = \frac{\sum_{i=1}^{n} {}^iRSSI^m_{range}}{n},\ \ with\ n <= |N| \tag{6}$$

Regarding the estimated distance between users, we first estimate the distance of each user m from an IoT device n as $r^m_n = \frac{max_m + min_m}{2}$. Next, we identify three possible scenarios depending on the number of IoT devices that intercepted the pair during the T_{CP} interval:

1) A single IoT device n has identified the AB user pair: In this case we estimate their distance d_{AB} as:

$$d_{AB} = |r^A_n - r^B_n| \tag{7}$$

2) Two IoT devices n_1 and n_2 have identified the AB user pair: In this case we rely on the IoT device that was closer to the interaction (e.g., n_1 in case $min(min^{n_1}_A, \min^{n_2}_A, \min^{n_1}_B, \min^{n_2}_A) = \min^{n_1}_A$ *or* $min^{n_1}_B$) to estimate their distance d_{AB} as in Eq. 7.
3) Three or more IoT devices have identified the AB user pair: In this case, we rely on the known locations of each IoT device (x_n, y_n) and calculate the exact location of each user m (x_m, y_m) using trilateration [39,40]. If more than three IoT devices have intercepted the pair, we use the measurements from the three devices that present the lower RSSI readings. Fig. 3 shows the trilateration process. A user's smart device has been identified by three IoT devices. The distance between the smart device and each IoT device represents the radius of a circle showing all possible locations of the mobile smart device. The user m (x_m, y_m) coordinates are obtained using equations for the three circles in the two-dimensional Cartesian coordinate system and by solving the following equations:

$$\begin{aligned}(-2x_1 + 2x_2)x + (-2y_1 + 2y_2)y &= {r_1}^2 - {r_2}^2 - {x_1}^2 + {x_2}^2 - {y_1}^2 + {y_2}^2\\ (-2x_2 + 2x_3)x + (-2y_2 + 2y_3)y &= {r_2}^2 - {r_3}^2 - {x_2}^2 + {x_3}^2 - {y_2}^2 + {y_3}^2\end{aligned} \tag{8}$$

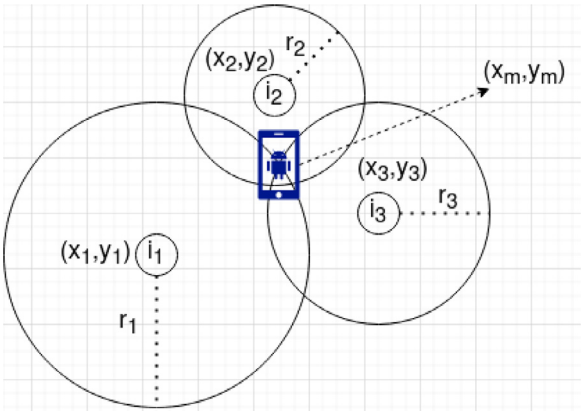

Fig. 3. Trilateration.

Next, given the estimated locations of the two users A (x_A, y_A) and B (x_B, y_B) their distance is given by:

$$d_{AB} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2} \tag{9}$$

Finally, the framework outputs the combined interaction information for each pair of users A, and B within the interval T_{CP}:

- ID_A: ID of the pair's first user
- ID_B: ID of the pair's second user
- $t^{AB}_{overlap}$: Overlapping time of pair
- d_{AB}: The estimated distance between the two users
- $\overline{RSSI^A_{range}}$: Average RSSI range for user A
- $\overline{RSSI^B_{range}}$: Average RSSI range for user B

4. Infection Risk Assessment using Machine Learning

4.1. Risk Classification

Next, our framework estimates the infection risk following the interaction of two users. To do so, we consider the RSSI interactions between the user pair and the IoT devices, and the users' estimated overlapping time to identify this interaction between users as low, medium, or high risk. We adopt a similar risk classification logic, as presented in [8]. We categorize the risk prediction function as a hypothesis test. We suppose x as a feature vector and R as a risk prediction function $R : (x) \rightarrow \{1, 2, 3\}$, where 1 shows low, 2 medium, and 3 high risk. Then we have the following three hypotheses, as shown below:

$$\begin{aligned} H_0 &= R(x) = 1 \\ H_1 &= R(x) = 2 \\ H_2 &= R(x) = 3 \end{aligned} \tag{10}$$

where H_0 represents a low risk, H_1 the average risk, and H_2 the high risk. Under this scenario, a false positive is an error in the classification in which a hypothesis incorrectly shows the interaction as low risk. However, this classification is high-risk, while in a false negative error, the hypothesis incorrectly shows the high-risk user to be low-risk.

The exact function that accurately translates the interaction between two parties to infection risk is a variable that relates to the exact nature of the infection, and specific analysis made by the field specialists[41–43]. Therefore, the proposed framework can be retrofit to any past or future pandemic cases. We will also utilize a supervised machine-learning approach that requires a set of labeled interactions before being able to classify the interaction risk successfully. This labeling can be initially made by the specific infection risk function. However, after this system is available online and more and more interaction cases are identified, they can be labeled accurately by their outcome. For instance, if a user has been identified as infectious, his contacts will be identified. Then, suspicious interactions will be labeled as high risk.

Since, neither a specific risk function or after-the-fact data are currently available to us, we define an alternative labeling procedure, to showcase the applicability of this method. Our manual label classification considers the interaction distance and time overlap duration between any two individuals. We consider specific conditions on d_{ij} and t_{ij} for calculating the labels as shown below and in Fig. 4:

- High-risk = 2, { $d_{ij} < 2$ m and $t_{ij} \geq 60$ s } or { 2 m $\leq d_{ij} \leq 6$ m and $t_{ij} \geq 3600$ s } or { 6 m $\leq d_{ij} \leq 10$ m and $t_{ij} \geq 3600$ s }

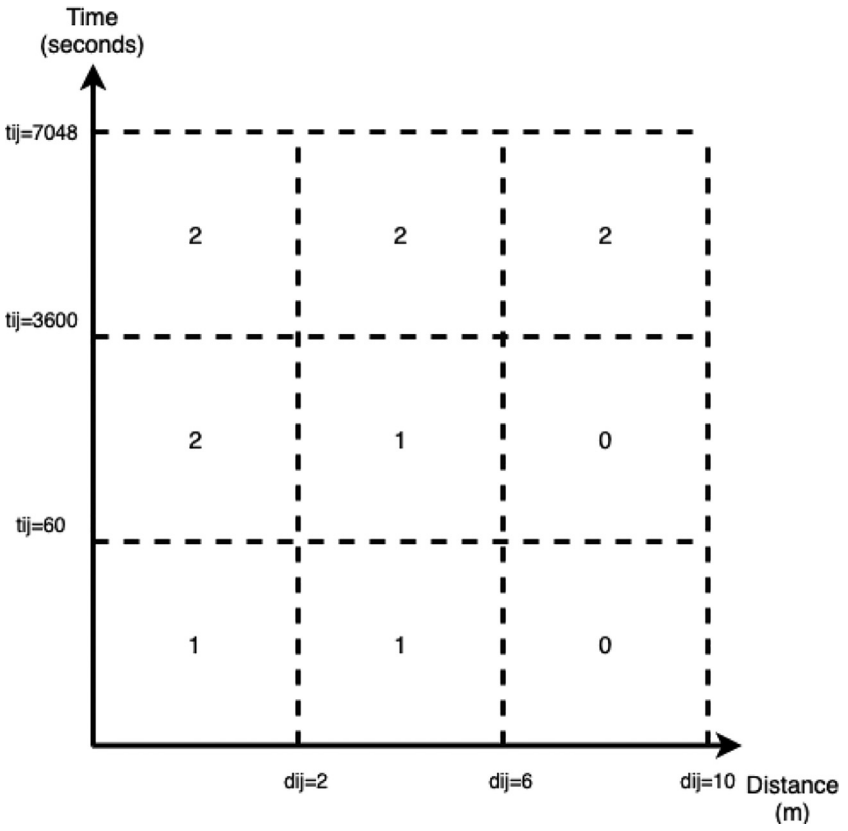

Fig. 4. Label-risk function.

- Medium-risk = 1, { $d_{ij} \leq 2$ m and $t_{ij} < 60$ s } or { 2 m $\leq d_{ij} \leq$6 m and $t_{ij} < 3600$ s }
- Low-risk = 0, { 6 m $\leq d_{ij} \leq$ 10 m and $t_{ij} < 3600$ s}

4.2. Classification Algorithms

Our machine-learning approach utilizes five input features to classify a user pair interaction at a specific risk level. These features are the minimum and maximum RSSI values observed by each user, their estimated position user, the average RSSI observed for each user, and their respective average RSSI ranges. These features provide information on the environmental noise conditions that accompanied each interaction. In addition, we compared different classification methods. We applied multi-class classification (One-vs-all) in all models where we have three classes of labels equaling high risk, medium risk, and low risk. In the following, we will briefly describe these classification models:

1. Decision Tree(DR): Decision Tree is a supervised and nonlinear machine-learning model. The DR uses a training set and splits the feature input into smaller subsets and at the same time creates a related tree. The Gini impurity measurement is used in DR to decide the optimal split for the nodes. The formula for calculating the Gini impurity of a feature is as follows:

$$G = \sum_{i} P(i) * (1 - P(i)) \tag{11}$$

   where P(i) is the likelihood of a certain classification i, for the training data set [44].
2. Extreme Gradient Boosting (XGBoost): XGboost is an excellent technique to improve the performance of the model by reducing overfitting [45]. This model is high-speed and scalable. The objective function of XGBoost is illustrated as below:

$$\begin{aligned} N(\theta) &= L(\theta) + \Omega(\theta) \\ \Omega(\theta) &= \gamma T + \tfrac{1}{2}\lambda \|w\|^2 \end{aligned} \tag{12}$$

   where γ controls the learning rate and set to a value greater than 0 and less than 1, w is the weight of the leaves, and $\Omega(\theta)$ is a parameter which controls the regularization term. T is the number of leaves in a tree and γT prevents over-fitting. $L(\theta)$ calculates the loss function and fits the data for our model. The equation for calculating $L(\theta)$ is illustrated

Table 2
Case-Study Parameters.

Parameter	Value
Number of IoT devices (N)	32
Number of Beacons/Participants (M)	46
Advertising interval (T_a)	1 s
Update interval of IoT device (T_o)	10, 15, 20 s
Centralized processing interval (T_{CP})	40 s, 1 min, 2 min
Smoothing parameter (α)	0.8
Measurement uncertainty (v_2)	4
Motion uncertainty (constant)	2
Initial estimate (m_1)	0
Initial uncertainty (v_1)	10,000

as follows:

$$L(\theta) \approx \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) + \Omega(\theta). \tag{13}$$

where l is loss function, f_t is the t^{th} tree output. g_i and h_i are the first and second derivatives of the loss function as given below:

$$\begin{aligned} g_i &= \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \\ h_i &= \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)}) \end{aligned} \tag{14}$$

The Eq. (14) shows y_i and $\hat{y}_i$ as target label and predicted label accordingly. From Eq. (12) we find the best splitting point for each tree and minimize the objective function and the goal is to create the tree and minimize the target label.

3. Bootstrap aggregating (Bagging): Bagging builds the learning algorithm on a different subset from training data. The algorithm collects the groups of data and trains each bag separately. The new predictions are built by averaging the predictions together from the base learners. This is shown in Eq. (15) where $\hat{g}_{bag}$ is the bagged prediction, x is the record for which we need to produce a prediction, and $\hat{g}_1(x) + \hat{g}_2(x) + \ldots + \hat{g}_h(x)$ are the predictions from the particular base learners.

$$\hat{g}_{bag} = \hat{g}_1(x) + \hat{g}_2(x) + \ldots + \hat{g}_h(x) \tag{15}$$

The aggregation method reduces the variance of the individual base learner.

4. K-nearest neighbors (KNN): KNN is a non-parametric supervised learning method based on distance rule-based techniques. KNN tries to classify a data point to a given category with the training set [46]. KNN uses k samples of group data and finds the distances between a query and all the data's training information. By choosing a specific number for K, the algorithm uses a majority vote for finding the most frequent label. Therefore, K is the tuning parameter that affects the performance of KNN.

5. Experimental Evaluation

The proposed framework is evaluated using the Bluetooth Low Energy protocol as the main means of passive packet advertising, and our real-world trial-generated dataset of user-IoT device interactions, the BLEBeacon dataset [47,48].

5.1. Case Study

Our case study environment is comprised of multiple BLE advertising devices (held by 46 smart space occupants) and a sensing infrastructure of IoT-like devices [48], which in this case are 32 Raspberry Pis (RPi). Note that the locations of the IoT devices were dictated by the need for outlets, and therefore are distributed in a non-uniform fashion in the physical space. This setting emulates a real-world installation of multi-purpose IoT devices very closely. The trial participants (college students and faculty following their usual routines in the indoor environment of a university building) carry BLE beacons (Gimbal Series 10) that produce advertising packets with a transmission rate of 1 Hz ($T_a = 1$ *second*). The generated packets are collected by the RPis, thus emulating interactions between mobile users following their routine (for one month) and IoT devices. Fig. 5 illustrates the RPis' exact locations and Table 2 shows the exact parameter values that were used to evaluate our framework in this real-world scenario.

5.2. User-Device Distance Estimation Accuracy

First, we evaluate the framework's capability to estimate the distance between the user and the IoT device using BLE advertising packets. To do so, we have conducted the following experiment. A user equipped with a BLE beacon was kept static at different distances from an IoT device with the same hardware specifications used in the real-world case study.

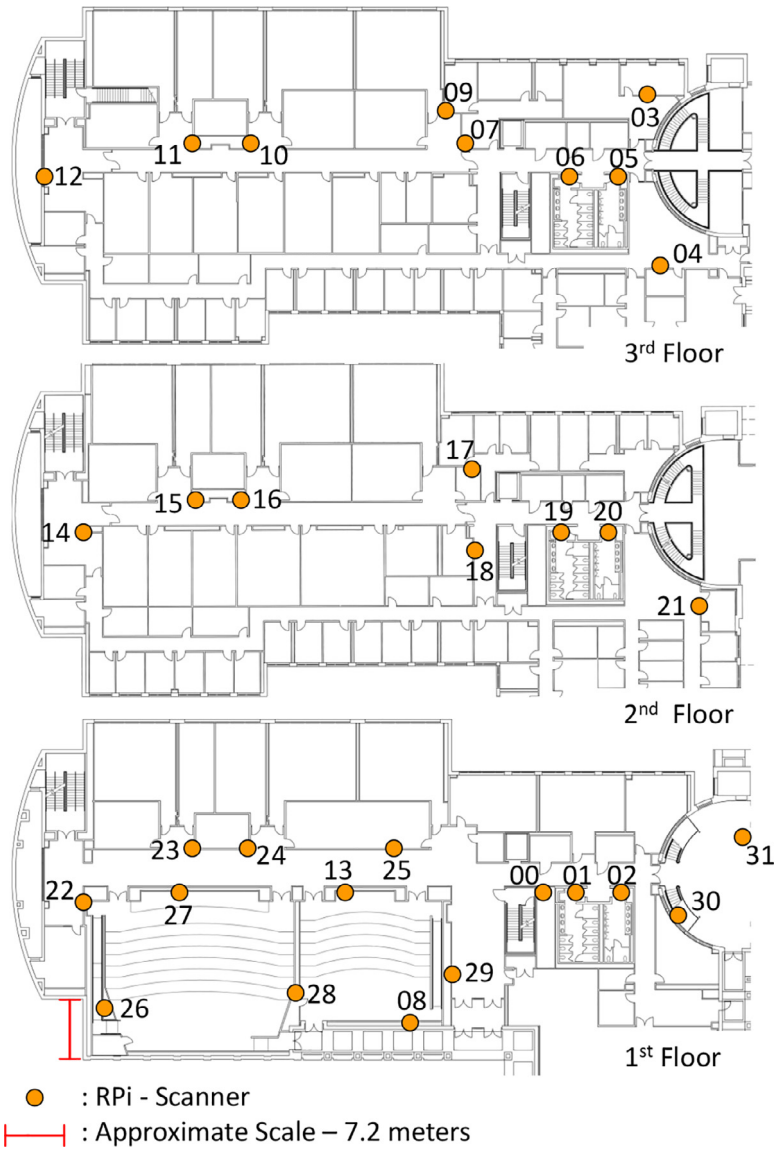

Fig. 5. Location of IoT Devices - Smart Space Topology [48].

Table 3
Relative distance error vs. real distance.

	Relative Error (%)				
Real Distance (m)	8.22	9.14	9.14	11.27	11.58
Exponential with Kalman	22%	16%	31%	23%	48%
Simple Moving Average	25%	16%	29%	26%	47%
Simple Exponential Filter	67%	36%	212%	49%	138%
Raw Data	121%	86%	105%	49%	88%

The device collected packets for approximately 15 minute periods for each distance and logged the RSSI values from each packet. From then on, we evaluated different approaches for filtering the RSSI values for noise reduction. Specifically, we tested the proposed filtering (exponential filter and Kalman) against more lightweight methods, namely simple exponential filtering, moving average filtering [49], and using the raw RSSI values for the distance estimation with Eq. 4 (p_{d0} = -64.63 for d_0 1m). Regarding the exponential filter we tested different values for α (see Eq. (1)) such as 0.2, 0.4, 0.6, 0.8 and lowest average absolute error was achieved with $\alpha = 0.8$.

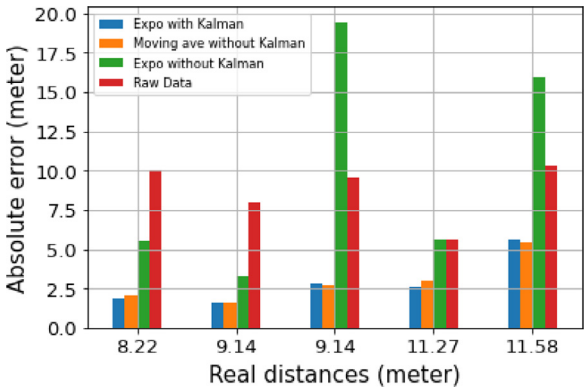

Fig. 6. Average absolute distance error vs. real distance.

Fig. 6 and Table 3 show the average absolute and relative distance estimation error (compared to the real measured distance) for different user-IoT device distances, respectively. Results show that the combined approach followed in our framework that utilizes both exponential and Kalman filtering vastly outperforms the alternative. Specifically, the distance error of the filtered RSSI is reduced by approximately 6–8 m in comparison to the raw RSSI approach.

5.3. System Parameter Evaluation and Overall Accuracy

Next, we evaluate the operation of our system using the real-time-generated data from the BLEBeacon dataset [38,47]. Every entry corresponds to a single interaction between a user and an IoT device (in the form of a BLE advertising packet – see [38,50]). Our analysis considers how our framework would operate during a single day (09/22/2016) and a single week (09/22/2016-09/29/2016).

First, we consider a single day of the experiment [50] and evaluate tuning parameters of our system, namely the update interval of each IoT device (T_o), and the centralized processing server interval (T_{CP}), as seen in Table 2. As mentioned earlier, the centralized processing unit outputs contact tracing results every T_{CP} seconds. We further merge the contact information for those users who have appeared in multiple consecutive (with the less or equal 1-second lag) batches and calculate their total overlapping time duration and average estimated distance. Fig. 7 shows how adjusting the (T_o), and (T_{CP}) values affects the overall accuracy of our framework. Results show a similar number of occurrences regardless of the distributed systems' update/processing periods, attesting to our framework's capability to detect mobile users' contact. Therefore, the system operator will be able to adjust the T_o and T_{CP} parameters in order to optimize network traffic and computational efficiency for both the IoT devices and the centralized unit without losing the ability to detect user contacts (time duration and distance) accurately.

Next, we extend the period under observation to one week. We use an IoT device update period of $T_o = 15$ seconds and central unit update interval of $T_{CP} = 1$ minute and apply the framework to a week-long portion of the BLEBeacn dataset. Fig. 8 shows the overall number of contacts against the observed distance and their overlapping time. The overlapping time results show a strong power law distribution behavior, which follows previous results on the statistical behavior of dwell time for human mobility patterns [51].

5.4. Evaluation of Risk Classification

The interactions (contacts of user pairs and their data e.g., RSSI ranges and user estimated location) that were generated by the week-long study were used to evaluate our machine-learning algorithms for the infection risk classification part of our framework. We split the collected interaction data into a training (80%) and a test (20%) set. We first fit the parameters and weights of the hypothesis function into the training dataset and assess the learned model by using the test data that predicts user-risk. The labeling for this specific evaluation along with the utilized features is explained in Section 4. For all our testing we utilize the Python Scikit-Learn library.

We evaluate four machine-learning classifiers: Bootstrap aggregating (bagging), Decision Tree (DR), Extreme Gradient Boosting (XGBoost), and K-nearest neighbors (KNN). Our study uses KNN as a baseline. We draw the confusion matrix to show each method's performance in Fig. 9. We observe that XGBoost wrongly categorizes high-risk as low-risk 5.1% of the time and as average-risk 29.1% of the time. However, it correctly categorizes high-risk cases at a 65.7% rate. On the other hand, bagging wrongly categorizes high-risk as low-risk 12.6% of the time and as average-risk 28.6% of the time. The correct high-risk characterization is at 58.9%. The decision tree and KNN approaches perform significantly worse than the ones mentioned above. It is very important for the specific scenario of contact tracing that people who are at high risk are not classified as low risk. Therefore, in that aspect, XGboost outperforms all other models.

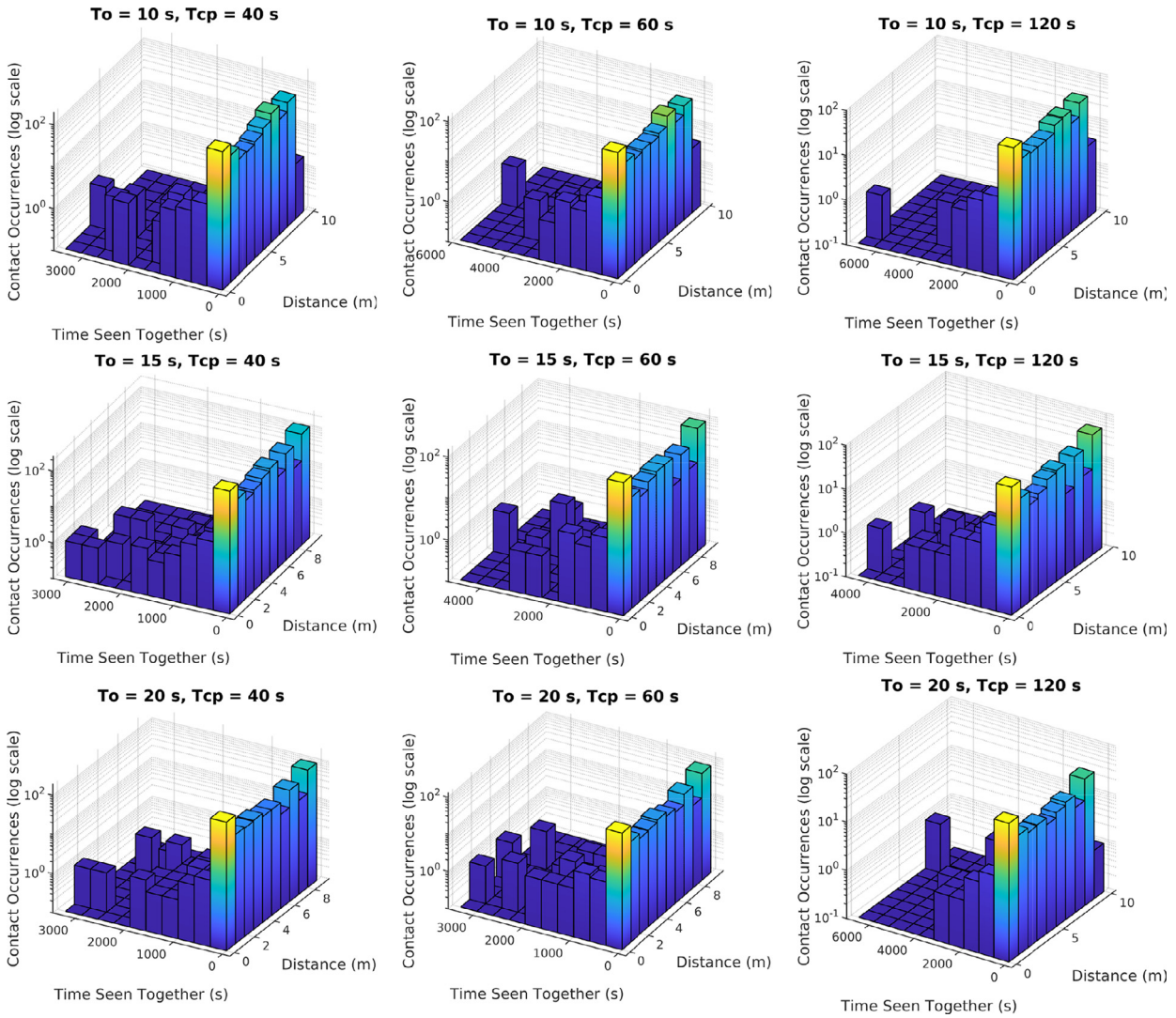

Fig. 7. Contact occurrences vs. overlapping time and distance during a single day.

Table 4
Algorithm Performance.

Classifier	Performance			
	Accuracy	Precision	Recall	F1-score
XGBoost	0.85 (0.83,0.87) 95% CI	0.81 (0.79,0.83) 95% CI	0.87 (0.85,0.89) 95% CI	0.87 (0.86,0.89) 95% CI
Bagging	0.81 (0.79,0.84) 95% CI	0.79 (0.77,0.81) 95% CI	0.84 (0.82,0.86) 95% CI	0.86 (0.84,0.88) 95% CI
Decision Tree	0.77 (0.75,0.79) 95% CI	0.75 (0.73,0.78) 95% CI	0.83 (0.81,0.86) 95% CI	0.83 (0.81,0.85) 95% CI
KNN	0.50 (0.47,0.52) 95% CI	0.45 (0.43,0.48) 95% CI	0.50 (0.47, 0.52) 95% CI	0.53 (0.50,0.56) 95% CI

To calculate the confidence intervals for our models, we iterated the process 100 times and at each iteration randomly divided the data into 80% for training and 20% for the test. From Table 4, we observed that XGboost has 81% precision and 87% recall, while Bagging presents 79% precision and 84% recall. On the other hand, the DT presents only a 75% precision and 83% recall. In contrast, the worst-performing KNN has only 45% precision and 50% recall, which makes the approach inappropriate for this application. High precision and recall are translated to low false positive rates and low false negative rates. Therefore, XGboost has a better performance compared to other models.

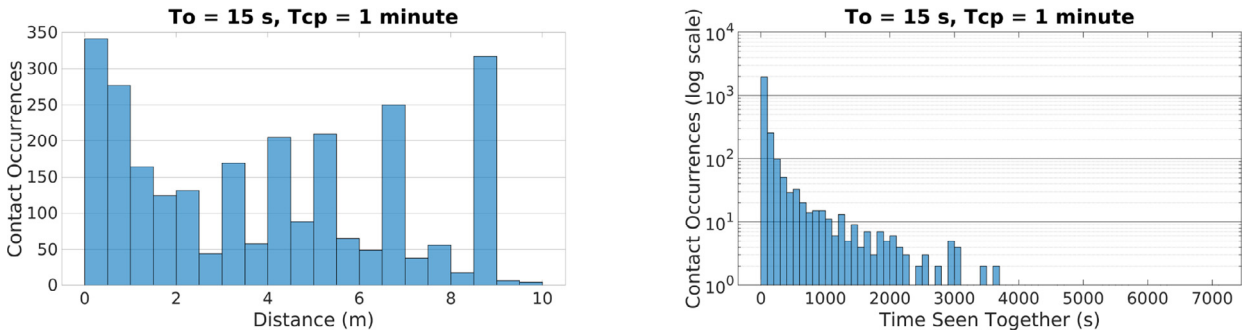

Fig. 8. Overall number of weekly contacts vs (Left) distance, and (Right) overlapping time.

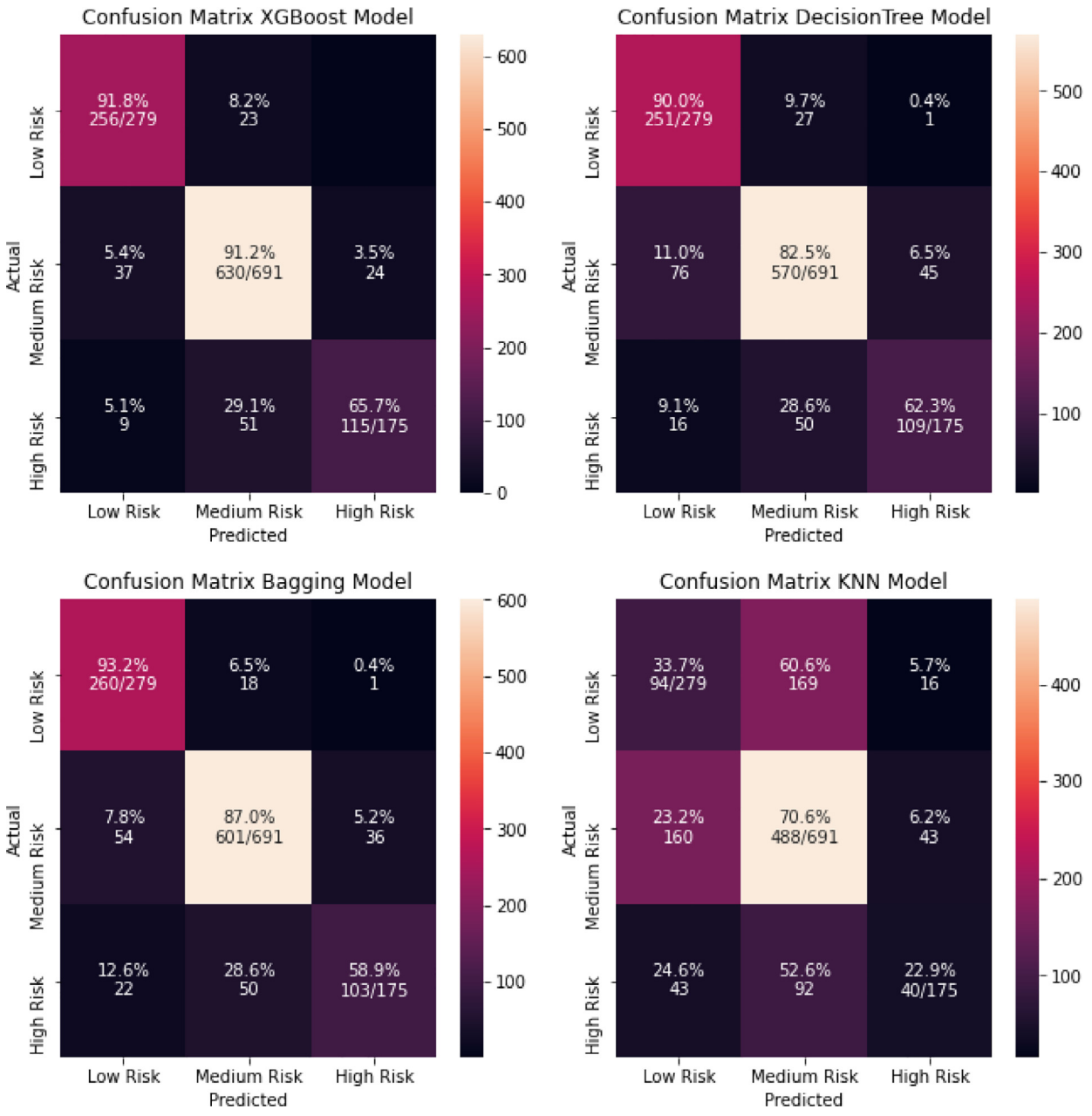

Fig. 9. The classification performance of different models through a Confusion-Matrix.

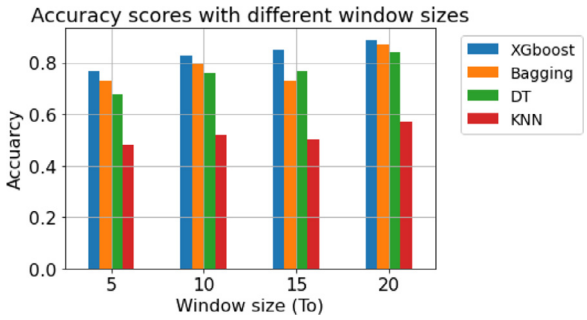

Fig. 10. Accuracy vs. window size.

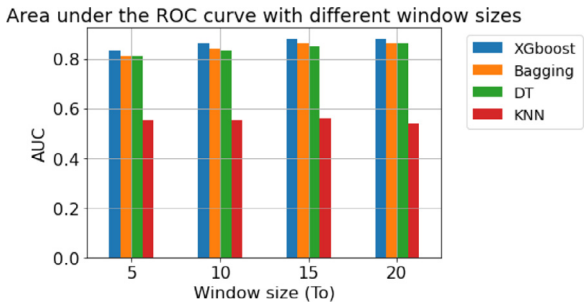

Fig. 11. Accuracy vs. AUC.

5.4.1. Impact of Window Size

Finally, we examined the impact of the T_o window size on accuracy and area under the ROC curve (AUC). The outcome is illustrated in Fig. 10. We can observe that DT has profited more by the window increase, and achieved a higher performance increase than other approaches (DT accuracy was raised from 68% to 84%). Additionally, as seen in Fig. 11 the AUC score of XGboost is higher than any other method, while the AUC score of KNN presented no significant change. In total, we observed an increase in the accuracy of all models as the T_o interval grew. This behavior is attested to the fact that a larger window leads to larger local samples of RSSI measurements, and therefore a better risk estimation for the whole system. All related data and code is open-sourced and available in [52].

6. Conclusion

Fast and accurate contact tracing systems can assist in preventing the spread of highly infectious diseases like Covid-19. This requires a cohesive and complex system. We propose a passive contact tracing framework that relies on passive interactions between users and an IoT infrastructure. The framework relies on proximity sensing while performing risk infection classification using well-known machine-learning techniques. We have also performed a case study of the system using a real-world dataset of BLE-based interactions. Our experimental results suggest that our design can substantially help monitor the contact occurrences between people in smart cities constantly and accurately both as a standalone system or as a complementary system for other contact tracing applications.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] H. Chen, B. Yang, H. Pei, J. Liu, Next generation technology for epidemic prevention and control: data-driven contact tracking, IEEE Access 7 (2018) 2633–2642.
[2] N. Ferguson, D. Laydon, G. Nedjati-Gilani, N. Imai, K. Ainslie, M. Baguelin, S. Bhatia, A. Boonyasiri, Z. Cucunubá, G. Cuomo-Dannenburg, et al., Report 9: impact of non-pharmaceutical interventions (NPIS) to reduce covid19 mortality and healthcare demand, Imper. Coll. Lond. 10 (2020) 77482.
[3] K.T. Eames, M.J. Keeling, Contact tracing and disease control, Proc. R. Soc. Lond. Ser. B 270 (1533) (2003) 2565–2571.

[4] P.C. Ng, P. Spachos, S. Gregori, K. Plataniotis, Epidemic exposure notification with smartwatch: aproximity-based privacy-preserving approach, arXiv:2007.04399(2020).
[5] Hamagen – Israeli health ministry, 2020, (https://govextra.gov.il/ministry-of-health/hamagen-app/download-en/).
[6] P.-P. Team, Pan-european privacy-preserving proximity tracing, 2020.
[7] W. Dong, T. Guan, B. Lepri, C. Qiao, Pocketcare: tracking the flu with mobile phones using partial observations of proximity and symptoms, in: Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, 3, 2019, pp. 1–23.
[8] P.C. Ng, P. Spachos, K. Plataniotis, Covid-19 and your smartphone: ble-based smart contact tracing, arXiv:2005.13754(2020).
[9] K. Michael, R. Abbas, Behind covid-19 contact trace apps: the google–apple partnership, IEEE Consum. Electron. Mag. 9 (5) (2020) 71–76.
[10] J. Bay, J. Kek, A. Tan, C.S. Hau, L. Yongquan, J. Tan, T.A. Quy, Bluetrace: a Privacy-Preserving Protocol for Community-Driven Contact Tracing Across Borders, Tech. Rep, Government Technology Agency-Singapore, 2020.
[11] R.L. Rivest, D. Weitzner, L. Ivers, I. Soibelman, M. Zissman, Pact: private automated contact tracing, 2020.
[12] S.P. Mohanty, U. Choppali, E. Kougianos, Everything you wanted to know about smart cities: the internet of things is the backbone, IEEE Consum. Electron. Mag. 5 (3) (2016) 60–70.
[13] P. Sotres, J.R. Santana, L. Sánchez, J. Lanza, L. Munoz, Practical lessons from the deployment and management of a smart city internet-of-things infrastructure: the smartsantander testbed case, IEEE Access 5 (2017) 14309–14322.
[14] D. Sikeridis, B.P. Rimal, I. Papapanagiotou, M. Devetsikiotis, Unsupervised crowd-assisted learning enabling location-aware facilities, IEEE Internet Things J. 5 (6) (2018) 4699–4713.
[15] F. Sivrikaya, N. Ben-Sassi, X.-T. Dang, O.C. Görür, C. Kuster, Internet of smart city objects: a distributed framework for service discovery and composition, IEEE Access 7 (2019) 14434–14454.
[16] O. Krejcar, P. Maresova, A. Selamat, F.J. Melero, S. Barakovic, J.B. Husic, E. Herrera-Viedma, R. Frischer, K. Kuca, Smart furniture as a component of a smart city definition based on key technologies specification, IEEE Access 7 (2019) 94822–94839.
[17] S.H. Alsamhi, O. Ma, M.S. Ansari, F.A. Almalki, Survey on collaborative smart drones and internet of things for improving smartness of smart cities, IEEE Access 7 (2019) 128125–128152.
[18] D. Jin, C. Hannon, Z. Li, P. Cortes, S. Ramaraju, P. Burgess, N. Buch, M. Shahidehpour, Smart street lighting system: a platform for innovative smart city applications and a new frontier for cyber-security, Electr. J. 29 (10) (2016) 28–35.
[19] N. Ahmed, R.A. Michelin, W. Xue, S. Ruj, R. Malaney, S.S. Kanhere, A. Seneviratne, W. Hu, H. Janicke, S.K. Jha, A survey of covid-19 contact tracing apps, IEEE Access 8 (2020) 134577–134601.
[20] Y.J. Park, Y.J. Choe, O. Park, S.Y. Park, Y.-M. Kim, J. Kim, S. Kweon, Y. Woo, J. Gwack, S.S. Kim, et al., Contact tracing during coronavirus disease outbreak, south korea, 2020, Emerging Infect. Dis. 26 (10) (2020) 2465–2468.
[21] I. Braithwaite, T. Callender, M. Bullock, R.W. Aldridge, Automated and partly automated contact tracing: a systematic review to inform the control of covid-19, Lancet Digit. Health (2020).
[22] M.S. Munir, D. Kim, S. Abedin, C. Hong, A risk-sensitive social distance recommendation system via bluetooth towards the covid-19 private safety, Korean Computer Congress (KCC), 2020.
[23] P. Hu, IoT-based contact tracing systems for infectious diseases: architecture and analysis, arXiv:2009.01902(2020).
[24] P. Tedeschi, S. Bakiras, R. Di Pietro, Iotrace: a flexible, efficient, and privacy-preserving iot-enabled architecture for contact tracing, arXiv:2007.11928(2020).
[25] K. Christidis, D. Sikeridis, Y. Wang, M. Devetsikiotis, A framework for designing and evaluating realistic blockchain-based local energy markets, Appl Energy 281 (2021) 115963, doi:10.1016/j.apenergy.2020.115963.
[26] D. Sikeridis, A. Bidram, M. Devetsikiotis, M.J. Reno, A blockchain-based mechanism for secure data exchange in smart grid protection systems, in: 2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC), IEEE, 2020, pp. 1–6.
[27] Y. Abuidris, R. Kumar, W. Wenyong, A survey of blockchain based on e-voting systems, in: Proceedings of the 2019 2nd International Conference on Blockchain Technology and Applications, in: ICBTA 2019, Association for Computing Machinery, New York, NY, USA, 2019, pp. 99–104, doi:10.1145/3376044.3376060.
[28] R. Kumar, W. Wang, J. Kumar, T. Yang, A. Khan, W. Ali, I. Ali, An integration of blockchain and ai for secure data sharing and detection of ct images for the hospitals, Comput. Med. Imaging Graph. 87 (2021) 101812, doi:10.1016/j.compmedimag.2020.101812.
[29] R. Kumar, A.A. Khan, S. Zhang, W. Wang, Y. Abuidris, W. Amin, J. Kumar, Blockchain-federated-learning and deep learning models for covid-19 detection using ct imaging, arXiv:2007.06537(2020).
[30] A. Kalla, T. Hewa, R.A. Mishra, M. Ylianttila, M. Liyanage, The role of blockchain to fight against covid-19, IEEE Eng. Manag. Rev. 48 (3) (2020) 85–96.
[31] A.A. Abd-alrazaq, M. Alajlani, D. Alhuwail, A. Erbad, A. Giannicchi, Z. Shah, M. Hamdi, M. Househ, Blockchain technologies to mitigate covid-19 challenges: a scoping review, Comput. Methods Programs Biomed. Update (2020) 100001, doi:10.1016/j.cmpbup.2020.100001.
[32] H. Xu, L. Zhang, O. Onireti, Y. Fang, W.J. Buchanan, M.A. Imran, Beeptrace: blockchain-enabled privacy-preserving contact tracing for covid-19 pandemic and beyond, IEEE Internet Things J. (2020), doi:10.1109/JIOT.2020.3025953. 1–1.
[33] W. Lv, S. Wu, C. Jiang, Y. Cui, X. Qiu, Y. Zhang, Decentralized blockchain for privacy-preserving large-scale contact tracing, arXiv:2007.00894(2020).
[34] D. Sikeridis, I. Papapanagiotou, B.P. Rimal, M. Devetsikiotis, A comparative taxonomy and survey of public cloud infrastructure vendors, arXiv:1710.01476(2017).
[35] Simple exponential smoothing, accessed 29 December 2020. https://otexts.com/fpp2/ses.html.
[36] G. Nishad, Kalman Filters: A step by step implementation guide in python, accessed 29 December 2020. https://towardsdatascience.com/kalman-filters-a-step-by-step-implementation-guide-in-python-91e7e123b968.
[37] P. Kumar, L. Reddy, S. Varma, Distance measurement and error estimation scheme for RSSI based localization in wireless sensor networks, in: 2009 Fifth International Conference on Wireless Communication and Sensor Networks (WCSN), IEEE, 2009, pp. 1–4.
[38] D. Sikeridis, M. Devetsikiotis, I. Papapanagiotou, Occupant tracking in smart facilities: An experimental study, in: 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), IEEE, 2017, pp. 818–822.
[39] Cell Phone Trilateration Algorithm, accessed 29 December 2020. https://www.101computing.net/cell-phone-trilateration-algorithm/.
[40] S. Sadowski, P. Spachos, Rssi-based indoor localization with the internet of things, IEEE Access 6 (2018) 30149–30161.
[41] A.R. Akhmetzhanov, K. Mizumoto, S.-m. Jung, N.M. Linton, R. Omori, H. Nishiura, Epidemiological characteristics of novel coronavirus infection: a statistical analysis of publicly available case data, medRxiv (2020).
[42] G.D. Barmparis, G. Tsironis, Estimating the infection horizon of covid-19 in eight countries with a data-driven approach, Chaos Solitons Fract. (2020) 109842.
[43] H. Yun, Z. Sun, J. Wu, A. Tang, M. Hu, Z. Xiang, Laboratory data analysis of novel coronavirus (covid-19) screening in 2510 patients, Clin. Chim. Acta (2020).
[44] L. Jiang, B. Zhang, Q. Ni, X. Sun, P. Dong, Prediction of SNP sequences via GINI impurity based gradient boosting method, IEEE Access 7 (2019) 12647–12657.
[45] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd ACM SIGKDDInternational Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[46] K.Q. Weinberger, J. Blitzer, L. Saul, Distance metric learning for large margin nearest neighbor classification, Adv. Neural Inf. Process. Syst. 18 (2005) 1473–1480.
[47] D. Sikeridis, I. Papapanagiotou, M. Devetsikiotis, Crawdad dataset unm/blebeacon (v. 2019-03-12), 2019.
[48] D. Sikeridis, I. Papapanagiotou, M. Devetsikiotis, Blebeacon: a real-subject trial dataset from mobile bluetooth low energy beacons, arXiv:1802.08782(2018).

[49] D. Ordóñez-Camacho, E. Cabrera-Goyes, An adaptive-bounds band-pass moving-average filter to increase precision on distance estimation from bluetooth RSSI, in: International Conference on Information Theoretic Security, Springer, 2018, pp. 823–832.
[50] M. Inaya, M. Meli, D. Sikeridis, M. Devetsikiotis, A real-subject evaluation trial for location-aware smart buildings, in: 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), IEEE, 2017, pp. 301–306.
[51] X. Zhou, Z. Zhao, R. Li, Y. Zhou, J. Palicot, H. Zhang, Human mobility patterns in cellular networks, IEEE Commun. Lett. 17 (10) (2013) 1877–1880.
[52] IoT enabled Contact Tracing, accessed 22 March 2021. https://github.com/zakhavan/IoT-enabled_Contact_Tracing.