The Design of GBDT-based Anomaly Detection for O-RAN Near-RT RIC

Chih-Cheng Tseng[1*], Hsin-Cheng Wu[1], Shao-Yu Lien[2], Jiun-Chi You[1], Chang-Yi Liu[1], and Rui-En You[1]
[1] Department of Electrical Engineering, National Ilan University, Yilan City, Yilan County, Taiwan
[2] College of Artificial Intelligence, National Yang Ming Chiao Tung University, Guiren District, Tainan City, Taiwan
[*] Corresponding author: tsengcc@niu.edu.tw

***Abstract*—With the increasing demand for the 5th Generation (5G) Base Stations (BSs), the global telecommunications industry is promoting technologies with open interfaces and open software and hardware designs, driving the 5G BSs to be cost-effectively developed. While such Open Radio Access Network (O-RAN) based 5G BSs are widely installed, User Equipment (UE) handovers become frequent and important. To instantly and accurately detect anomalous UE, a UE whose performance is below threshold, that needs to be handed over, Anomaly Detection (AD) becomes important. This paper applies the Gradient Boosting Decision Tree (GBDT) algorithm to detect anomalous UE. Based on the O-RAN architecture, the developed GBDT is implemented as an AD xApp. Specifically, using the RSRP and the available number of Physical Resource Blocks (PRBs) extracted from the performance metrics reported from the developed RAN emulator, this paper compares the performance of the developed GBDT-based AD xApp with the Isoforest-based AD xApp released by O-RAN Software Community (SC). Simulation results show the developed GBDT-based AD xApp demonstrates superior AD performance in terms of prediction accuracy and average prediction time. Specifically, when the number of UEs is 40, the prediction accuracy is improved by 102.2% and the average prediction time is reduced by 75.67%.**

***Keywords*—O-RAN, Near-RT RIC, anomaly detection, GBDT, Isolation forest**

I. Introduction

Since February 2018, the Open Radio Access Network (O-RAN) Alliance was initially launched by five operators, AT&T (United States), China Mobile (China), Deutsche Telekom (Germany), NTT DOCOMO (Japan), and Orange (France), at the Mobile World Congress (MWC) in Barcelona, Spain. Subsequently, in addition to the five operators mentioned above, CTOs and representatives from another seven operators, Bharti Airtel (India), China Telecom (China), KT Corporation (South Korea), Singtel (Singapore), SK Telecom (South Korea), Telstra (Australia), and Vodafone (United Kingdom), attended a meeting and signed documents during the MWC in Shanghai in June 2018, officially announcing the establishment of the O-RAN Alliance. The vision of the O-RAN Alliance is to create an *open interface*, *open source codes*, and *intelligent* wireless mobile network that is highly flexible and cost-effective. Specifically, it aims to achieve a hardware and software open, standardized, and intelligent RAN architecture that can be easily, flexibly, and cost-effectively developed and deployed.

With the popularization of 5G technologies and applications, the number of deployed 5G Base Stations (BSs) will increase significantly, leading User Equipment (UE) to more frequent handover between BSs. However, existing handover mechanisms may not accurately and timely make handover decisions for anomalous UEs, resulting in deteriorated BS performance and degraded user experience. Hence, it is important to develop an Anomaly Detection (AD) scheme to pre-identify UEs that may cause performance degradation of BSs due to improper handovers. Once the anomalous UE IDs can be predicted accurately and the list of anomalous UE IDs can be timely considered as one of the important criteria when making handover decisions, the overall system performance and the user experience can be improved.

In the past, technologies of 2G/3G BSs were relatively simple and typically integrated within a single architecture. However, starting from the 4G, the complexity of technology increased dramatically. Consequently, the architecture of BS (or eNB) was divided into the front-end Remote Radio Heads (RRH) and back-end Baseband Unit (BBU). In 5G, the BS (or gNB) is further divided into a three-tier architecture consisting of the Radio Unit (RU), Distributed Unit (DU), and Central Unit (CU). Under the concept of openness, an open architecture means that the hardware and software, interface, as well as the procedures and message formats exchanged between the RU, DU, and CU, need to be fully standardized and open. As a result, the O-RAN Alliance has standardized the three-tier architecture as O-CU, O-DU, and O-RU [1]. According to the logical architecture defined by the O-RAN Alliance as shown in Figure 1 [2], the management side on top of this architecture is the Service Management and Orchestration (SMO) framework, which includes the functions of Non-Real Time RAN Intelligent Controller (Non-RT-RIC). The Near-Real Time RAN Intelligent Controller (Near-RT RIC), O-CU-CP (O-RAN Central Unit-Control Plane), O-CU-UP (O-RAN Central Unit-User Plane), O-DU, and O-RU are included in the radio side. The O-CU is responsible for the Packet Data Convergence Protocol (PDCP) layer and typically runs on x86 computers/servers. The O-DU handles all baseband processing, scheduling, Radio Link Control (RLC), Medium

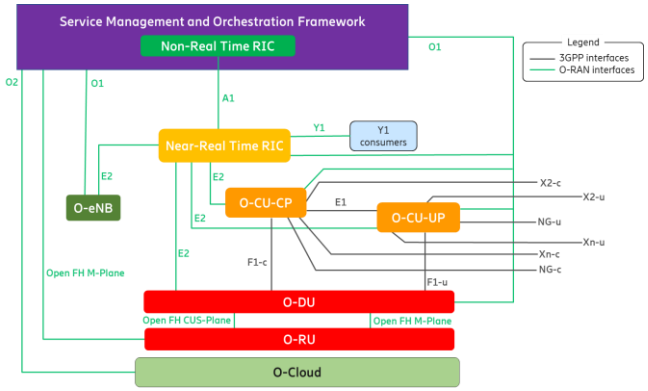

Figure 1 The logical architecture of O-RAN [2].

Access Control (MAC), and the higher Physical (PHY) layer, with virtualization requiring hardware acceleration such as FPGA or GPU. The O-RU is responsible for the lower PHY layer and radio frequency processing components, including the analog components of transmitters and receivers [3]. Finally, at the lowest layer, the O-Cloud (O-RAN Cloud) consists of physical infrastructure nodes, software components, and appropriate management and coordination functions that meet O-RAN standards, forming a cloud computing platform.

II. The Considered Environment

One of the most critical problems when employing AI/ML algorithms to build models is to obtain the required training data sets. In general, the real data generated from mobile networks is stored in the data center of the operators and is confidential to the public. Hence, it is difficult to obtain real data to train models. To solve this problem, a RAN emulator is developed that fully complies with the 3GPP standards. With this RAN emulator, users can configure environment-related parameters (e.g., the size of the considered system environment, the number of BSs, the number of UEs,…, etc.), cell-centric (e.g., location, carrier frequency, bandwidth,…, etc.), and UE-centric (e.g., location, mobility model, required traffic load,…, etc.).

As shown in Figure 2, the entire system environment considered in this paper is a rectangular area of 8,000 m × 6,000 m. In this rectangular area, 18 BSs are deployed and numbered from 1 to 18. The coverage area of each BS is assumed as a circle with a radius of 1,000 m. Hence, BS #1 is deployed at (1000, 1000), BS #4 is deployed at (2000, 2000), while BS #18 is deployed at (7000, 5000). Additionally, this rectangular area is further divided into a business zone and a residential zone. The upper-left and lower-right corners of the two zones are (800, 800), (3200, 5200) and (4800, 800), (7200, 5200), respectively. Consequently, each of the zones is a rectangular area of 2,400 m × 4,400 m. The locations of each UE in each of the two zones are uniformly and randomly selected from each zone. For each UE, the straight line between the two randomly selected locations in each zone is regarded as the commuting route of the UE. Based on a 24-hour day, the RAN emulator assigns UEs according to a three-shift concept: 20% of the UEs for the night shift (00:00 to 09:00), 20% for the evening shift (16:00 to 01:00), and 60% for the day shift (08:00 to 17:00). Initially, UEs are randomly placed in their respective zones based on the percentages assigned to shifts. Then, they will be assigned numbers starting from 1 according to the above percentages. For example, if there are 10 UEs, two UEs are assigned to the night shift and would be numbered 1 to 2. Two UEs are assigned to the evening shift and would be numbered 3 to 4. Finally, the rest of the five UEs are assigned to the day shift and would be numbered from 5 to 10.

To simulate a 24-hour day in the real world, each simulation run in the RAN emulator spans 24 hours, starting at midnight (i.e., 00:00). However, to speed up the simulation process, one second in the real world is scaled to one minute in the RAN emulator. The time in the RAN emulator is slotted and each time slot is one minute. The performance metrics generated in the RAN emulator are collected and reported to the Near-RT RIC every minute. Based on this, the moving speed of a UE in the RAN emulator is set to 100 m/min, which is equal to 6 km/hr in the real world. Besides, based on the

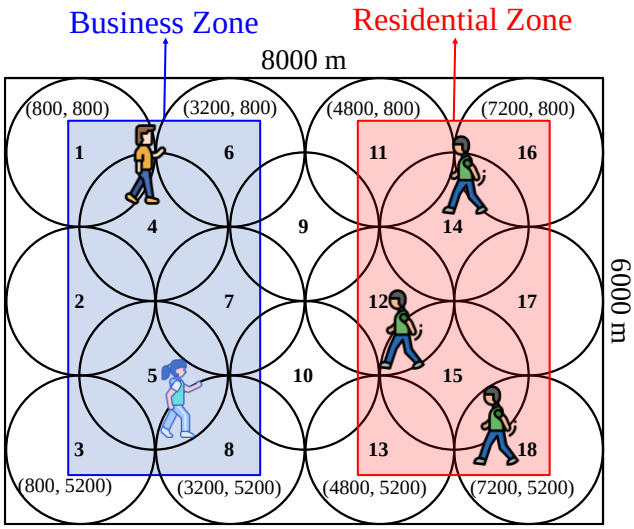

Figure 2 The considered environment.

location at 00:00, each UE is initially connected to the closest BS. To make sure to arrive at the location in the business zone before the beginning of the work shift, a UE starts following the commuting route to move from the location in the residential zone directly to the location in the business zone one hour before the start of the work shift. On the contrary, UE follows the same commuting route to move from the location in the business zone directly back to the location in the residential zone after the work shift ends. It's important to note that once a UE has arrived at the location in the residential or business zone, it will no longer move. Additionally, the downlink (DL) throughput demand of each UE varies depending on whether it is on duty or off duty. While UE is on duty, the DL throughput demand is assumed 18.75 MBytes/min. On the contrary, the DL throughput demand of an off-duty UE is assumed 63.9 MBytes/min.

When generating the training data set, a total of 86,400 records were collected for 60 UEs over a 24-hour day without making any handovers. A small part of the generated training data set is shown in Table I. Each data record (i.e., the data in a row) is labeled as normal '0' or anomalous '1' in the "Anomaly" column on the right-most column of Table I according to the AD criteria described below. The labels are then used for the data fitting while training the model for AD.

The three considered features indicated in red in Table I and the corresponding pre-defined thresholds used as the criterion to label the 'Anomaly' field in the training data set

Table I. Part of The Training Data Set

ETime	UEID	Serving_Cell_ID	Serving_Cell_RSSI	Serving_Cell_SINR	UE_Throughput_Bytes_DL	Anomaly
68	15	6	-117	-6	63900000	1
69	15	6	-118	-7	63900000	1
70	15	6	-120	-9	63900000	1
70	22	4	-118	-7	63900000	1
71	15	6	-121	-10	63900000	1
71	22	4	-119	-8	63900000	1
71	23	5	-118	-7	63900000	1
72	15	6	-121	-10	63900000	1
72	22	4	-120	-9	63900000	1
72	23	5	-119	-8	63900000	1
72	14	2	-118	-7	63900000	1
72	24	4	-117	-6	63900000	1
73	15	6	-122	-11	63900000	1
73	22	4	-121	-10	63900000	1
73	23	5	-120	-9	63900000	1
73	14	2	-119	-8	63900000	1
73	24	4	-118	-7	63900000	1
74	15	6	-123	-12	63900000	1
74	22	4	-122	-11	63900000	1

are described as follows. Please note that the term "Cell" in the data reported by the RAN emulator is a synonym of the term "BS".

A. *Serving_Cell_RSSI*

Based on the definition, the Serving_Cell_RSSI of a UE refers to the strength of a signal sent from the serving cell of the UE and is perceived by the UE. The farther the UE is from its serving cell, the lower the Serving_Cell_RSSI value will be perceived by the UE. Taking Figure 2 as an example, consider a UE located at the center of BS #1 is served by BS #1. In the developed RAN emulator, as the UE moves toward the center of BS #4, the UE is considered anomalous when the perceived Serving_Cell_RSSI value from the serving cell BS #1 falls below the pre-defined threshold value –117 dBm.

B. *Serving_Cell_SNR*

Serving_Cell_SNR of a UE is defined as the ratio between the signal strength from the serving cell and the noise perceived by a UE. Refer to Figure 2, as a UE moves away from the serving cell, the Serving_Cell_SNR value decreases accordingly. In the developed RAN emulator, if a UE served by BS #1 moves toward BS #4, the UE is considered anomalous when the perceived Serving_Cell_SNR from BS #1 drops below the pre-defined threshold –11 dB.

C. *UE_Throughput_Bytes_DL*

The contents of the DL traffic generated at the on-duty and off-duty UEs are assumed majorly business-related and entertainment-related, respectively. Hence, the criterion defined in this paper for an on-duty UE to be labeled as anomalous is when the demanded DL throughput, i.e., 18.75 MBytes/min, cannot be satisfied. However, an off-duty UE will be labeled as anomalous when 35% of its demanded DL throughput, i.e., 63.9 MBytes/min × 0.35 = 22.365 MBytes/min, cannot be met.

III. The Design of Anomaly Detection

A. *Isolation Forest (Isoforest)*

The Isoforest [4] is an unsupervised ML algorithm used by the O-RAN Software Community (SC) to develop the AD xApp [5]. Conceptually, anomalous data in a data set usually have a characteristic that many of their features differ significantly from the majority of the data. To isolate these anomalous data, Isoforest first constructs a forest with t randomly generated binary trees. Then, the Isoforest identifies data as anomalous simply based on the constructed forest. When constructing a binary tree, a subset of data with a size of φ is randomly sampled from the original data set. Among the pre-defined features of data, Isoforest first randomly selects one. Then, a value p between the maximum and minimum values of the selected feature is randomly chosen and used to split the sampled data. In particular, data whose value of the selected feature is greater than p is classified into one group. On the contrary, those less than p are classified into another. By iteratively repeating the random selections of a feature and a split value, a binary tree is built. The number of edges from a data point located in the tree to the root of the tree is defined as the path length between them. Based on the operations of the Isoforest, anomalous data is prone to be isolated earlier. Hence, anomalous data generally has a shorter path length. By averaging or aggregating the results from multiple binary trees, Isoforest can ultimately identify anomalies. While identifying anomalies, the parameter *contamination* is used by the Isoforest as the threshold to evaluate if data is anomalous. The contamination is defined as the expected ratio of the number of anomalies to the total amount of data in the data set. Hence, the value of contamination ranges from 0 to 1. In our study, to find the best contamination value fit the considered environment, 80% of the data in the data set is used to train the model, while the remaining 20% is used to test the trained model. In our study, the contamination value is 0.33 and is obtained by the Grid Search (GridSearchCV) provided by Sklearn in Python.

B. *Gradient Boosting Decision Tree (GBDT)*

Gradient Boosting Decision Tree (GBDT) [6] is a supervised ML algorithm composed of multiple decision trees. In the GBDT, the predicted results of all the trees are accumulated to form the final decision. In general, GBDT works by initially using the average of the target values of the entire training data as an initial prediction for training. With the boosting technique, GBDT builds a new weak model in each iteration to fit the residuals from the previous model. A residual refers to the difference between an actual target value and the corresponding predicted value of the current model. By fitting the residuals, the new model corrects the errors resulting from the previous model. Consequently, the overall accuracy is gradually improved. During the training process, each new weak model is added to the ensemble model with a certain weight. These weights are determined through the gradient descent optimization method with the aim of minimizing the overall loss function of the model. After all iterations are completed, GBDT combines the predictions of each weak model with the respective weight to produce the final prediction output. Specifically, the voting method is typically used to produce the final prediction result in classification tasks, while the results are weighted and summed in regression tasks [6].

Figure 3 provides a simple example of how GBDT works. Suppose there are four individuals A, B, C, and D colored white, red, green, and blue, and are with ages 14, 18, 36, and 40, respectively. First, based on the first condition, i.e., has or has no credit card, the four individuals are divided into two

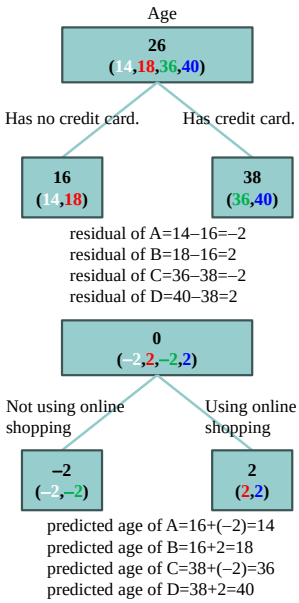

Figure 3 Example of GBDT

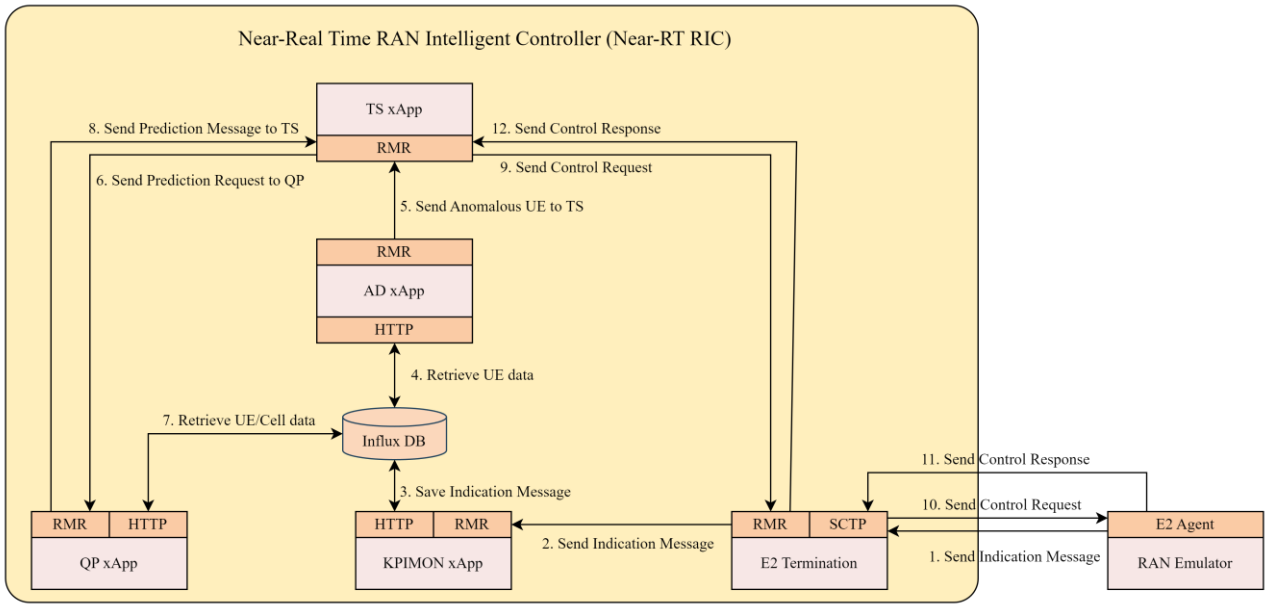

Figure 4 The simulated system architecture.

groups. The average ages of the two groups, i.e., 16 and 38, are calculated and are used as the predicted values. The residual between the predicted value and the actual value is then calculated. Then, the first decision tree is completed. Using the four calculated residuals, i.e., –2, 2, –2, 2, the four individuals are regrouped again based on the second condition, i.e., using or not using online shopping. The average ages of the two new groups, i.e., –2 and 2, are calculated and used as the predicted values. Then, the second decision tree is formed. Now, since the residuals for the four individuals are all zeros, no more decision tree forming is needed. The results from both trees are combined to obtain the final prediction. Since the residuals of the second decision tree are all zero, as shown at the bottom of Figure 3, the built GBDT model can perfectly predict the ages of individuals A, B, C, and D.

The following are the required parameters when using the GBDT algorithm:

- learning_rate: The weighting value of each tree.
- max_depth: The maximum depth of each tree.
- n_estimators: The total number of iterations, i.e., the number of decision trees.

The parameter values used to build the GBDT model are listed in TABLE II and are obtained by using Grid Search (GridSearchCV) provided by Sklearn in Python. The search range for n_estimators is 50 to 150 with increments of 50, the range of max_depth is 2 to 5 with increments of 1, and the learning_rate ranges from 0.005 to 0.025 with increments of 0.005.

IV. SIMULATED SYSTEM ARCHITECTURE

To evaluate the performance of the developed AD, as demonstrated in Figure 4, the simulated system architecture is the integration of Near-RT RIC and the developed RAN emulator. To simplify our discussions, only components and xApps that are highly related to our work are depicted in the Near-RT RIC. As defined by the O-RAN Alliance, messages exchanged among components, xApps, or between component and xApp are through RIC Message Router (RMR). On the top of the RAN emulator in Figure 4, an E2 Agent is exclusively developed by the Institute for Information Industry (III) to send/receive the messages to/from the E2 Termination in the Near-RT RIC through the E2 interface in terms of the Stream Control Transmission Protocol (SCTP). Figure 4 also shows how the Near-RT RIC intelligently steers traffic with the help of the Key Performance Indicator Monitor (KPIMON), AD, Quality of Experience (QoE) Prediction (QP), and Traffic Steering (TS) xApps.

TABLE II. PARAMETER VALUES OF GBDT

Parameter	Value
learning_rate	0.01
max_depth	3
n_estimators	100
loss function	least square
residual	(actual value) – (predicted value)

A. *Procedure of The Simulated System Architecture*

The detailed operations of the procedure with respect to the simulated system architecture depicted in Figure 4, are described below. Please note that the processes in each component and xApp to verify the format of the received messages are mandatory but are not presented below for simplicity.

1. The RAN emulator periodically, specifically every second in the real world, sends an Indication Message in which the performance metrics of the BSs and UEs in the considered environment depicted in Figure 2 are included.
2. Whenever E2 Termination identifies an Indication Message is correctly received, it forwards the Indication Message to KPIMON xApp.
3. KPIMON xApp decomposes the contents of the received Indication Message into cell-centric and UE-centric data and then stores them in the Influx DB.

4. Every second, AD xApp retrieves the latest UE-centric data from the Influx DB and feeds them into the trained AI/ML model to infer the potential anomalous UE ID.
5. The IDs of the inferred potential anomalous UEs are submitted to TS xApp.
6. After verifying the received anomalous UE IDs, TS xApp sends a list of anomalous UE IDs to QP xApp requesting QP xApp to predict the QoE of each of the UE whose ID is in the list.
7. To proceed with the QoE prediction, QP xApp fetches the cell-centric data and UE-centric data that are related to each UE whose ID is in the list from the Influx DB.
8. By feeding the fetched cell-centric and UE-centric data to the trained AI/ML model, QP xApp infers the QoE of each UE whose ID is in the list. The predicted results are then sent back to TS xApp.
9. Based on the received QoE prediction results, TS xApp makes the handover decisions according to the designed criterion and sends a Control Request to E2 Termination in which the target BS numbers of those UE IDs that need to perform handover are included.
10. E2 Termination identifies the Control Request and forwards it to the RAN emulator.
11. Based on the contents of the received Control Request, the RAN emulator proceeds with the necessary handovers. After that, a Control Response that contains the results of the handover is sent to E2 Termination.
12. E2 Termination identifies the Control Response and forwards it to the TS xApp.

B. Simulation Results

In the simulations, based on the system architecture shown in Figure 4, the RAN emulator emulates the considered environment illustrated in Figure 2 for 24 hours. As mentioned in Section II, the time scale between the RAN emulator and the real world is 60:1. Hence, 1,440 Indication Messages will be generated by the RAN emulator within the emulated 24 hours. Every second in the real world, a generated Indication Message will be sent to the Near-RT RIC for detecting the anomalies. TABLE III and TABLE IV show the performance results obtained by the AD xApps implemented based on the Isoforest and GBDT algorithms, respectively. Here, the prediction error is defined as the events that a normal UE is predicted as anomalous (or false negative) or an anomalous UE is predicted as normal (or false positive). The increase in the number of false negatives results in the increase of the QoE prediction load of QP xApp. The increase in the number of false positives results in the deterioration of the system's performance.

By comparing the results presented in TABLE III and TABLE IV, it can be observed that the GBDT-based AD xApp developed in this paper outperforms the Isoforest-based AD xApp released by the O-RAN SC in terms of prediction accuracy and prediction time. Specifically, when the number of UEs is 40, the prediction accuracy is improved by 102.2% and the average prediction time is reduced by 75.67%. From

TABLE III. RESULTS OF ISOFOREST

Result	Isoforest	
Number of UEs	20	40
Number of prediction errors	10689	29146
Prediction Accuracy (%)	62.8854	49.3993
Average prediction time (ms)	47.2470	55.9485

TABLE IV. RESULTS OF GBDT

Result	GBDT	
Number of UEs	20	40
Number of prediction errors	43	66
Prediction Accuracy (%)	99.8506	99.8854
Average prediction time (ms)	13.4110	13.6200

TABLE III and TABLE IV, as the number of UE increases, different from the deterioration of the prediction accuracy and the average prediction time obtained from the Isoforest-based AD xApp, it can be found that both the prediction accuracy and the average prediction time of the GBDT-based AD xApp are less sensitive to the increase in the number of UEs.

V. CONCLUSIONS

With the innovative and intelligent components included in the O-RAN architecture, this paper focuses on designing and deploying an AD xApp within the Near-RT RIC to detect anomalous UE ID based on the UE-centric data reported from the RAN emulator. Under the considered system architecture, simulation results show the developed GBDT-based AD xApp outperforms the Isoforest-based AD xApp released by O-RAN SC. The developed GBDT-based AD xApp is not only capable of accurately predicting anomalous UE ID and reducing the average prediction time but also the resulting accuracy and average prediction time are not sensitive to the increase in the number of UEs. With the great performance improvement of AD xApp, the handover decisions made by TS xApp improve the overall system performance accordingly. Consequently, user experiences are improved accordingly.

REFERENCES

[1] M. Dryjański, "Introduction to O-RAN," Mar. 10, 2021. [Online]. Available: https://rimedolabs.com/blog/introduction-to-o-ran/

[2] O-RAN Alliance., "O-RAN Architecture Description," O-RAN.WG1.OAD-R003-v12.00, June, 2024.

[3] Chih-Lin I, "Embracing Open Ecosystem," May 11, 2022. [Online]. Available: https://opennetworking.org/wp-content/uploads/2020/09/O-RAN-Embracing-Open-Ecosystem-09082020m.pdf

[4] F. Tony Liu, K. M. Ting, and Z. H. Zhou, "Isolation Forest," The 8th IEEE International Conference on Data Mining (ICDM 2008), Pisa Italy, December 15-19, 2008.

[5] O-RAN SC, [Online] Available: https://github.com/o-ran-sc/ric-app-ad

[6] J. H. Friedman, "Greedy Function approximation: A Gradient Boosting Machine," Annals of Statistics, vol. 29, no, 5, pp. 1189-1232, October 2001.

[7] C. Maklin, "Isolation Forest," Medium, July 15, 2022. [Online]. Available: https://medium.com/@corymaklin/isolation-forest-799fceacdda4

[8] O-RAN SC J Release Documentation Home. [Online]. Available: https://docs.o-ran-sc.org/en/latest/

[9] Shruti Dash. "Gradient Boosting – A Concise Introduction from Scratch." Available: https://ithelp.ithome.com.tw/articles/102280