Journal of Computer Science and Cybernetics, V.38, N.4 (2022), 377–403
DOI no 10.15625/1813-9663/38/4/17592

TWO-PHASE COMBINED MODEL TO IMPROVE THE ACCURACY OF INDOOR LOCATION FINGERPRINTING

VAN-BINH NGO[1], VAN-HIEU VU[2], DO-THANH-TUNG HOANG[2]

[1]*FPT University, Ha Noi, Viet Nam*
[2]*Institute of Information Technology, VAST, Ha Noi, Viet Nam*

Abstract. Wi-Fi Fingerprinting based Indoor Positioning System (IPS) aims to help locate and navigate users inside buildings. It has become a popular research topic in recent years with two goals of removing redundant data and increasing positioning accuracy and it has produced acceptable results. In this paper, we propose a feature reduction method and apply a machine learning model that combines two-phases. Specifically, Phase one builds machine learning models according to classification and regression algorithms including K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Random Forest (RF), Regressor additional tree (extraTree), Light Gradient Enhancer (LGBM), Logistic Regression (LR), and Linear Regression (LiR). Phase two uses a regression algorithm that combines the data predicted in Phase one is used for the training data. The proposed technique is tested on UJIIndoorLoc [a] dataset showing a prediction accuracy of 98.73% by building-floor, and an estimated accuracy of 99.62% and 99.52%, respectively, by longitude and latitude. When compared with the results of the models in which we use independent algorithms, and of other researches that have different models using the same algorithms and on the same dataset, most of our results are better.

Keywords. Wi-Fi fingerPrinting; Received signal strength-RSS; Indoor positioning system; Machine learning.

1. INTRODUCTION

Nowadays, outdoor positioning using GPS [1] is a mature field of research with a high level of precision of 1 to 5m. Since GPS operates most efficiently with Line-of-Sight (LoS) as it uses satellite signals for positioning, it is not feasible to use GPS in an indoor environment [2]. IPS has wide application in real life. In addition to navigation in rescue and emergency operations in buildings, IPS help users to locate and navigate their positions in the environments such as museums, hospitals, and mega malls. As a result, IPS has become a broad research area over the past 20 years. Methods for indoor positioning are introduced in [3], in which the method using RSS of radio signals emitted from Access Points (AP) and obtained from Reference Points (RP) is the most used and popular in indoor navigation system. Fingerprinting is the most widely used technique [4, 5] among numerous indoor positioning techniques based

*Corresponding author.
E-mail addresses: binhnv11@fe.edu.vn (V.B. Ngo); vvhieu@ioit.ac.vn (V.H. Vu); tunghdt@ioit.ac.vn (D.T.T. Hoang)

[a] `https://archive.ics.uci.edu/ml/datasets/ujiindoorloc`

on RSS values. The traditional fingerprinting method is divided into two stages as shown in Figure 1. IPS typically includes several APs disseminating Wi-Fi and mobile devices

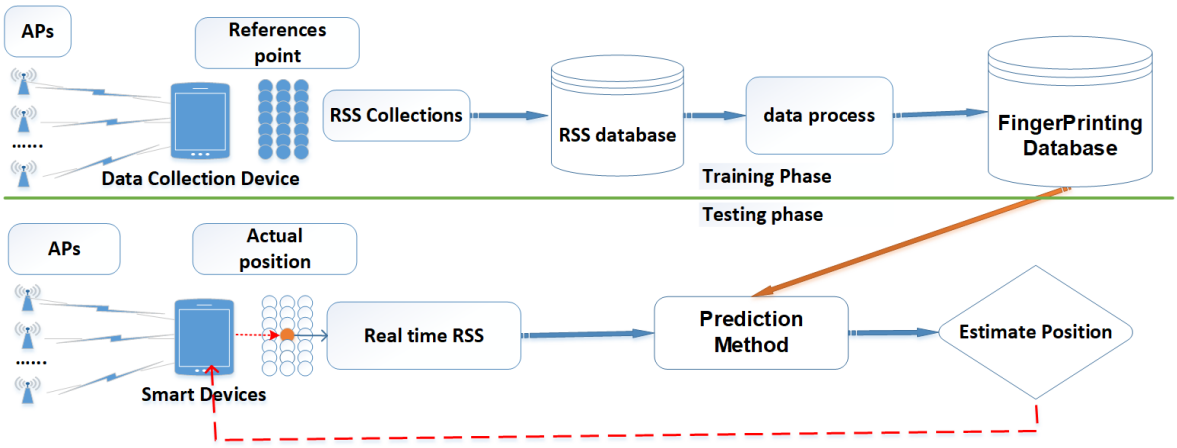


Figure 1: Indoor positioning model by traditional fingerprinting

connecting to Wi-Fi. To predict the user's location via mobile devices, RSS is used to build a fingerprinting database. There are two phases in fingerprinting: The training phase and the testing phase. In the training phase, RSS values of APs from known RPs are collected to build a fingerprinting database. Specifically, for each sampling, if the number of APs is n, the fingerprinting at the i^{th} RP has coordinates (x_i, y_i) defined as Eq.(1)

$$f_i = \left[(x_i, y_i), RSS_1, RSS_2, ..., RSS_n\right]. \tag{1}$$

To increase data quality, at an RP, the number of samples is taken many times, then the fingerPrinting values of the i^{th} RP form a matrix of Eq. (2)

$$F_i = \left[(x_i, y_i), RSS_1^1, ..., RSS_n^1, ..., RSS_1^m, ..., RSS_n^m\right], \tag{2}$$

where n is the number of APs, and m is the number of samples. The fingerprinting database obtained from all k reference locations (RPs) such as Eq.(3). The collected data is then called the training set

$$D_k(F_i) = \{F_{i1}, F_{i2}, ..., F_{ik}\}. \tag{3}$$

In the testing phase, the RSS values are sampled in real time at the unknown location. The role of pattern matching in traditional fingerprinting algorithms is to determine the similarity between training and testing fingerprints. The purpose is to find a pair consisting of testing and training points that are nearest to one another in the fingerprint space, and then use the position information of the training point to approximate (and estimate) the testing point in the location space.

The fingerprinting method faces two main problems. First, since many obstacles (like windows, doors, and walls) exist in indoor space, the radio frequency signal will propagate through different paths, which causes the signal to reach the receiver at different times. A different receiving time causes different phases, which can be superposed and cause signal distortion. This phenomenon above is called the multipath effect. Due to the multipath effect,

RSS quality is degraded, which makes fingerPrinting values become irregular and unstable, and the fingerPrinting database may be large and unreliable [6], leading to an increase in positioning error [7,8]. To address this issue, some filtering and feature reduction methods such as Principal Component Analysis (PCA) [9] and Linear Discriminant Analysis (LDA) [10,11] have been used to improve the quality of the dataset. The second concern is the quality of positioning which requires high accuracy and efficiency. Many machine learning algorithms such as K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), Light Gradient Boosting Machine (LGBM), Logistic Regression (LR), Convolutions Neural Network (CNN), Deep Neural Networks (DNN), and Recurrent Neural Network (RNN) have been tested by many research groups to increase location accuracy with remarkable results. However, until now there is still not an algorithm that would be applicable in different environment settings [12, 13]. Therefore, it is very valuable and feasible to improve the efficiency of the fingerprint database and build new machine learning models to improve location accuracy.

In this research, we propose a feature reduction method and an indoor localization method using a machine learning models that combine the algorithms of KNN [14, 15], SVM [16], RF [17], ExtraTree [18] , LGBM [19], LR [20], and Linear Regression (LiR) [21]. We use the classification problem to predict the floor number. The latitude and longitude values are estimated by the regression problem, the user's location is predicted based on the estimated latitude and longitude values. We test the proposed model on the UJIIndoorLoc [22] dataset which is a public dataset used by many research groups [13]. The main contributions of this research are summarized as follows:

- Proposing a method to reduce features of the classification problem.
- Proposing a two-phase combination model and modelling training algorithm.
- Deploying the combined model into the classification problem to predict the building-floor by LR, KNN, and SVM algorithms.
- Deploying the combined model into the regression problem to estimate the longitude and latitude using extraTree, KNN, FR, LGBM, and LR algorithms.

The remainder of this paper is structured as follows: Section 2. Related works, Section 3. Feature reduction method, proposed model, architecture description, and positioning process based on our proposed model, followed by Section 4 in which we describe experimental results and evaluations. Finally, we summarize the contribution of this research in Section 5.

2. RELATED WORKS

KNN was used very early in traditional machine learning methods applied in Wi-Fi RSS fingerprinting-based IPS. In 2000, the Microsoft Research team developed a positioning system called RADAR [15] using KNN. The result showed that the location estimation model using KNN outperformed fingerprinting algorithms. The average accuracy of this system was roughly 3m with 75% of the localization errors were below 4.7m. This was considered to establish a research base for using KNN algorithms in particular and machine learning algorithms in general. In [23], the authors used KNN combined with the user's travel history.

According to the findings, the new method had a higher positioning efficiency than KNN by 45%. In [24], with the method applying the weighted KNN, the positioning error of the result was between 1.42m and 1.61m, while for the KNN method, it was from 1.78m to 2.18m depending on the used K value. KNN is also used in combined models such as a combined model of DNN and KNN used by the authors in [25]. The results varied depending on the number of K chosen. The average error was from 1.39m to 1.5m. Compared with other machine learning methods such as DT, KNN, DNN, SVM, and RF, this solution produced better results. The SVM classification algorithm-based approach was proposed in [26, 27]. It showed that SVM had a more accurate result than fingerprinting. The accuracy of the study using [26] was reported as 2m in 77% of the testing cases, and [27] was 93.75% in 98.75% of the testing cases. DNN, KNN, and SVM algorithms were used in [28]. KNN produced better results than DNN. MSE for KNN ranged from 3.485m to 5.950m, with an average MSE of 4.163m, while DNN had a corresponding value of 4.169m, 4.163m, and 4.166m. KNN, however, lacked DNN's level of stability. SVM performed the worst with an average MSE of 11.06m. In spaces without walls or obstructions, the authors in [29] used RF with smart watches. As a result, RF accuracy increased to 97.5%, and execution time was significantly improved. The proposed algorithm by the authors in [30], which utilize regional grid division to reduce the maximum error, and adopt adjusted cosine similarity to match the proper grid and fingerprint, resulted in a decrease of the maximum error by 1.15m compared with the original RF. When combined with the extraTree, DL, and RF in [18], the Root Mean Square Error (RMSE) was 8.79m and 8.83m, respectively, for the X and Y axes. In [19], the authors used LGBM in a setting combining Wi-Fi with pictures. The test result showed an accuracy of 90% within 1.53m. It was an increase in accuracy by more than 20% when compared with the fingerprint positioning method, and improvement in performance by more than 15% when compared with DT and RF. Logistic Regression (LR) was employed by the authors in [20]. A 95.83% accuracy was obtained after data optimization, which was an increase of 80% greater than K-Mean. Chenlu Xiang *et al.* used LR combined with data optimization and tested their model in the standard laboratory [31, 32], both resulted in a positioning error of 92cm. Linear Regression was used in [21], in which the authors built an automatic tool to improve the instability of RSS. As a result, the average error was reduced from 8.95m to 4.03m. Liye Zhang *et al.* used LiR in [33], with the maximum error reduced from 10m to 4.5m and the average error reduced from 3.72m to 2.31m.

When examining other researches that used the JIIndoorLoc dataset, we find that the authors of Stanford University [34] tested the KNN, SVM, Gradient Boosting (BG), and DT models to predict the floor number. They created new test datasets using random values from the original dataset with magnitudes of 100, 200, 500, 1000, 2000, and 5000 samples, then used PCA to reduce the features. Their analysis revealed that KNN had good performance for large datasets. Although DT was a quick training method, KNN was more effective. Gradient Boosting had a small cross-validation error for small datasets and was less affected by missing data. SVM had the lowest accuracy and less efficiency. KNN was used by the authors in [35] to perform two tasks: the classification KNN to predict the building location, and the KNN regression to estimate the coordinates. Floors were determined by the RF algorithm. The research team removed the useless data before the experiment. The results were presented for each building, in which the accuracies of the floor were 97.95%, 90.87%, and 95.86%; The Mean Absolute Error (MAE) values for longitude

were 6.05m, 7.1m, and 9.08m; And the MAE values for latitude were 5.08m, 8.26m, and 8.13m for three buildings. The study in [36] was named the Fast-Accurate-Reliable Localization System (AFARLS) by combining COSELM (constrained online sequential extreme machine learning) with KNN. Accordingly, the accuracy by floor predicted by the AFARLS result was 95.41%, while the KNN result was 89.92%. The highest mean of the positioning accuracy of AFARLS was 6.4m. Before applying KNN, SVM, and RF in [37], quantization was used to reduce noise. The results showed that the accuracy by position for KNN, SVM, and RF algorithms were 67.49%, 62.71%, and 68.5%, respectively. The RF algorithm used to predict the floor number had an accuracy of 97%. Liye Zhang *et al.* divided the training dataset into two parts which were 80% for training and 20% for testing as they aimed for room-level positioning [38]. The team proposed a novel feature extraction algorithm named JLGBMLoc (Joint Denoising Auto-encoder (JDAE) with LGBM algorithm). The test results revealed that the proposed method had a room-level positioning accuracy of 96.73%, and a floor-level positioning accuracy of 99.32%.

In general, the accuracy of the location was generally improved by the application of data processing techniques such as weight matching, data normalization, and size reduction in all of the aforementioned methods. Those methods have not, however, been applied effectively yet. This was caused by their autonomous operation, insufficient data used in the training process as well as the incidence of over-fitting. To address those issues, in this paper we introduce a new machine learning model which combines machine learning algorithms in two-phase for the training phase. UJIIndoorLoc dataset is used to test the model.

3. PROPOSED METHOD AND MODELS

3.1. Dataset description

The UJIIndoorLoc dataset covers a 108,703m^2 area that includes 3 buildings, each having 4 or 5 floors [1]. It consists of 21,049 samples, of which 19,938 samples are for training, and 1,111 samples are for testing. Each sample consists of 529 features labeled as 'WAP01', 'WAP02',..., 'WAP520', 'LONGITUDE', 'LATITUDE', 'FLOOR', 'BUILDINGID', 'SPACEID', 'RELATIVEPOSITION', 'USERID', 'PHONEID', 'TIMESTAMP'. The first 520 features consist of different APs with RSS values ranging from -104dB to 0dB. RSS value is set to 100 for each undetectable AP. The remaining 9 features consist of longitude and latitude values for each sample given in meter; Floor number is given as either 0, 1, 2, 3, or 4; BuildingID is given as either 0, 1, or 2; SpaceID corresponding to the type of location where the measurement was collected (e.g., office, classroom); Relative position in relation to the SpaceID (e.g., inside or outside the door); UserID of the user who takes the samples; PhoneID of the phone used to take the samples; and lastly, Timestamp as the time that the samples are taken. The value of the features FLOORID, BUILDINGID, SPACEID, RELATIVEPOSITION, and USERID in the validation data is set to 0. RSS values in UJIIndoorLoc are collected by spaceID, fingerPrinting of $spaceID_i$ is as in Eq. (4)

$$f_i = [RSS_1, RSS_2, .., RSS_{520}, longitude, latitude, floorID, buildingID, spcaeID_i, relativePosition, userID, phoneID, timeStamp]. \quad (4)$$

For each spaceID, the fingerPrinting samples are collected multiple times by different userID and phoneID. And by design, locations in different buildings, different floors have set

the same spaceID value, for example, spaceID=101 has been used for many pairs (floorID, buildingID) like [(2,1); (3,2); (4,2); (1,2),..]. Therefore, the fingerPrinting values of the i^{th} spaceID form a matrix as Eq. (5)

$$\begin{aligned} F_i = [&(RSS_1^1, ..., RSS_{250}^1, longtitude^1, latitude^1, floorID^1, buildingID^1, \\ &spcaeID_i, relativePosition^1, userID^1, phoneID^1, timeStamp^1) \\ &(.....) \\ &(RSS_1^m, ..., RSS_{250}^m, longtitude^m, latitude^m, floorID^m, buildingID^m, \\ &spcaeID_i, relativePosition^m, userID^m, phoneID^m, timeStamp^m)], \end{aligned} \quad (5)$$

where m is the number of samples.

Because the UJIIndoorLoc dataset is very large, besides that spaceID can appear in many different buildings and floors, leading to RSS values of overlapping and overlapping fingerprinting, leading to the time of the classification problem to predict buildings and floors increase. Therefore, it is necessary to have a solution to remove the features to reduce the time of the classification problem.

3.2. Proposed feature reduction method

In the proposed method, we aim to reduce the computation time and increase positioning performance by reducing the dimension of the feature. In UJIIndoorLoc, each building has a set of multiple data lines. Each data line is represented as $[RSS_1, RSS_2, \cdots, RSSI_{520}$, buildingID]. The first 520 features are RSS values measured from 520 APs. The last element is the building number from 0 to 2.

For each building, we calculate the average value of all 520 features by spaceID. Since we have 3 buildings in the dataset, we create a new dataset as shown in Eq.(6), in which the first 520 elements are the expected values of the features and the last is the building number

$$\begin{bmatrix} \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 0 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 1 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 2 \end{bmatrix}. \quad (6)$$

In the classification problem, the more informative the building features are, the better the building classification is. To identify the most informative features of each class, we determine the absolute difference between the pairs 0-1, 0-2, and 1-2 by cross-matching the corresponding pairs in the dataset Eq. (6). This gives us three arrays that represent building pairings, which then are sorted in descending order. Those with low-difference or duplicate features are then removed. As a result, each array with N features has the highest difference. The three arrays are combined to create a feature set that has $3 \times N$ values for the building classification model. The building feature reduction method is illustrated in Algorithm 1.

We do the same with regards to floorID in the given dataset as the data row now reads as $[RSS_1, RSS_2, \cdots, RSSI_{520}$, floor], where the first 520 features are the measured signal levels and the last element is the floor number from 0 to 4. The results such as "floorArray"

Algorithm 1 Building feature reduction algorithm

Input:

$\mathcal{B}_0 \leftarrow$ all lines in the dataset have buildingID=0;
$\mathcal{B}_1 \leftarrow$ all lines in the dataset have buildingID=1;
$\mathcal{B}_2 \leftarrow$ all lines in the dataset have buildingID=2;

Output:

$\mathcal{B}$: The set of features has been reduced

Step 1:

$\mathcal{BA}_0 \leftarrow$ Calculation of mean by SpaceID in class $\mathcal{B}_0$;
$\mathcal{BA}_1 \leftarrow$ Calculation of mean by SpaceID in class $\mathcal{B}_1$;
$\mathcal{BA}_2 \leftarrow$ Calculation of mean by SpaceID in class $\mathcal{B}_2$;

Step 2:

$\mathcal{B}_{01} \leftarrow \mathcal{BA}_0 \cup \mathcal{BA}_1$;
$\mathcal{B}_{02} \leftarrow \mathcal{BA}_0 \cup \mathcal{BA}_2$;
$\mathcal{B}_{12} \leftarrow \mathcal{BA}_1 \cup \mathcal{BA}_2$;

Step 3:

Sort arrays $\mathcal{B}_{01}$, $\mathcal{B}_{02}$ and $\mathcal{B}_{12}$ in descending order

Step 4:

Remove duplicate, low-difference features

Step 5:

$\mathcal{B} \leftarrow \mathcal{B}_{01} \cup \mathcal{B}_{02} \cup \mathcal{B}_{12}$

is defined as Eq. (7)

$$\begin{bmatrix} \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 0 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 1 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 2 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 3 \\ \overline{RSS_1}, & \overline{RSS_2}, & \cdots, & \overline{RSSI_{520}}, & 4 \end{bmatrix} \tag{7}$$

To obtain the most informative features of each floor, dataset Eq.(7) is processed in the same manner as dataset Eq.(6). Following the combination of the pairings 0-1, 0-2, 0-3, 0-4, 1-2, 1-3, 1-4, 2-3, 2-4, and 3-4, the sorting and removing process are executed. Eventually, we receive 10 arrays. Each one has N features with the highest difference.

3.3. Proposed model

The fingerprinting method based on machine learning (ML) consists of two phases, as shown in Figure 2: training and testing. In the training phase, the fingerprint database (or training dataset) is used for ML model. A localization function during training attempts to learn the mapping between real-time RSS value and device/user locations by training dataset. In the testing phase, the ML model uses the trained localization function to predict the real-time locations of devices based on RSS measurements.

In this research, we propose a new training model combining algorithms in two phases. The first phase consists of n different models with their training dataset and generates n corresponding prediction models. The combination model in the second phase continues to

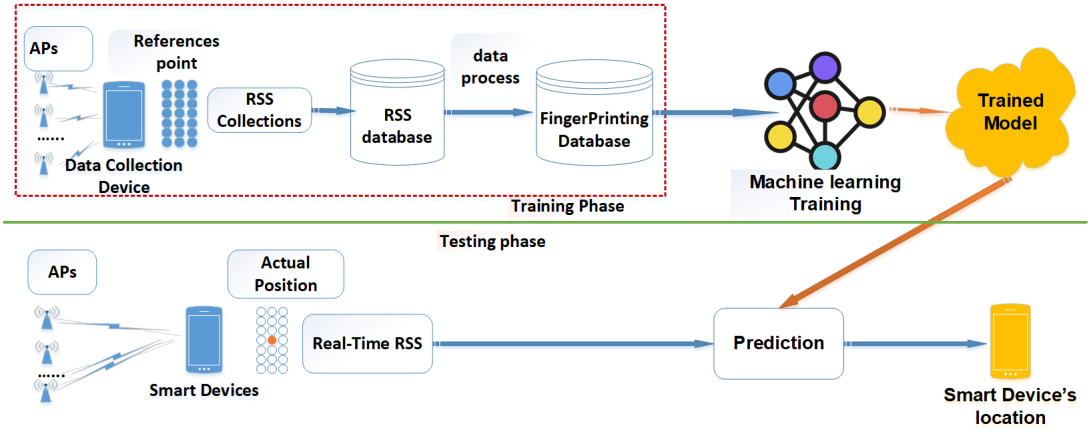


Figure 2: The basic process of ML-based indoor localization using Wi-Fi RSS fingerprints

train based on the results of the first phase and gives the final prediction. Our proposed model is shown in Figure 3.

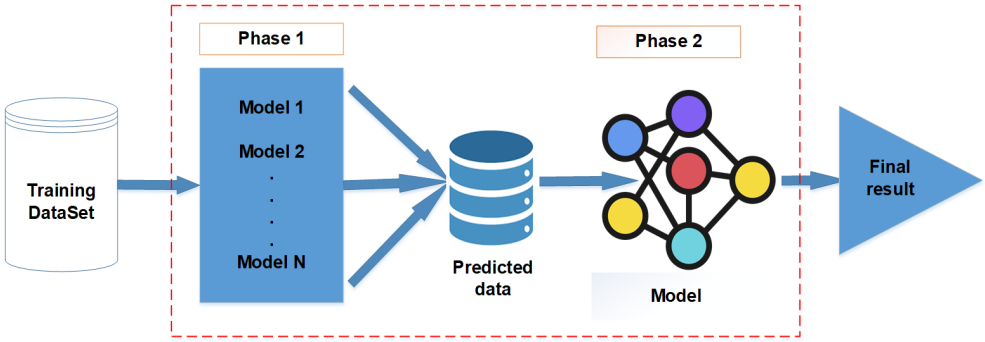


Figure 3: Proposed combined model

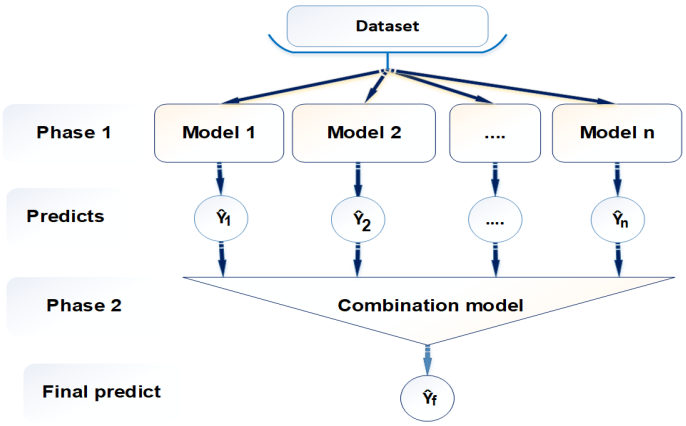


Figure 4: Combination process of the models

The two-phase training model is shown in Figure 4, where $\hat{Y}_1$ $\hat{Y}_2$, ..., and $\hat{Y}_n$ are the

prediction results of the n models in the first phase, $\hat{Y}_f$ is the final result of the second phase. The detailed training of the model is shown in Algorithm 2. Algorithm 2 has computational complexity $\mathcal{O}\left(\|\mathcal{D}_i\| \times m \times n\right)$.

Algorithm 2 Two-phase combined training algorithm

Input: $\mathcal{D} \leftarrow \{x_i, y_i\}_1^m$,$x_i \subset \mathcal{X}$, $y_i \subset \mathcal{Y}$. Where $\mathcal{X}$ is the set of features, $\mathcal{Y}$ is the set of labels, m is the number of rows in the dataset

Output: $\hat{Y}_f$

Step 1:

Initialize $\{M_1, M_2, ..., M_n\}$; ▷ n machine learning algorithms for the first phase

Divide $\mathcal{D}$ into subsets Split the subsets $\{\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_n, \mathcal{D}_{n+1}\}$; ▷ $n+1$ subdataset of $\mathcal{D}$

$\mathcal{D}^{'} \leftarrow \emptyset$; ▷ training dataset of second phase

Step 2: Training using algorithms in the first phase

for $i = 1$ to n **do**

  $\left(X_i^{train}, y_i^{train}, X_i^{test}, y_i^{test}\right) \leftarrow D_i$; ▷ Split training and test datasets

  $Model_i^0 \leftarrow train\left(M_i, \left(X_i^{train}, y_i^{train}\right)\right)$; ▷ Modeling by M_i

  $\hat{Y}_i \leftarrow Model_i^0(X_i^{test})$; ▷ Predicted results by $Model_i^0$

  $\mathcal{D}_i^{'} \leftarrow \left(X_i^{test}, \hat{Y}_i\right)$; ▷ Data is combined for the Second phase

  $\mathcal{D}^{'} \leftarrow \mathcal{D}^{'} \cup \mathcal{D}_i^{'}$;

end for

Step 3: Training using the algorithm in the second phase

Initialize: $M_{Combine}$;

$Model^1 \leftarrow train\left(M_{Combine}, \mathcal{D}'\right)$; ▷ The Second phase is trained the model

$\hat{Y}_f \leftarrow Model^1\left(\mathcal{D}_{n+1}\right)$; ▷ Predicted results by $Model^1$

The algorithms used in the combined models during the first phase are selected from those tested in the independent models. The test results are used for two tasks: selecting the best algorithms for the first phase of the combined model and comparing the results of the independent models with the combined model. The algorithm selection phase is called the "start-up phase". In the second phase, the regression model is chosen in such a way that the prediction results of the separate models in the first phase are combined to get the final prediction.

In experiments, the proposed model is used to solve two problems: predicting floors and estimating longitude and latitude. To solve these problems, we build two combined models: a classification model to predict floors and a regression model to estimate longitude and latitude.

4. EXPERIMENTS, RESULTS, AND EVALUATIONS

4.1. Data preprocessing

The dataset contains many important features, however, as the purpose of the problem is to identify building, floor, and location, other unimportant features such as SpaceID, Relative Position, UserID, PhoneID, and Timestamp will be excluded. The remaining features to

be used include WAP01, WAP02,..., WAP520, LONGITUDE, LATITUDE, FLOOR, and BUILDINGID.

4.2. Evaluation metrics

Classification model. Classifiers attempt to predict the probability of discrete outcomes (in this research it is the floor). Out of many ways to measure classification performance, in our research, we use Accuracy, Precision, Recall, and F1-score metrics.

- Accuracy: Accuracy measures how often the classifier correctly predicts and is defined as

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \tag{8}$$

- Precision: Precision explains how many of the correctly predicted cases actually turned out to be positive and is defined as Eq. (9)

$$Precision = \frac{TP}{TP + FP}. \tag{9}$$

- Recall: Recall explains how many of the actual positive cases we were able to predict correctly with our model and is defined as Eq. (10)

$$ReCall = \frac{TP}{TP + FN}. \tag{10}$$

- F1-Score: The F1 score is a single evaluation metric that aims to account for and optimize both precision and recall. It is defined as the harmonic mean of precision and recall. A model will have a high F1 score if both precision and recall are high. The F1 score is computed as Eq. (11)

$$F_1 = 2\frac{Precision.Recall}{Precision + Recall}, \tag{11}$$

where TP, TN, FP, and FN are collected from the confusion matrix. They are defined as follows:

- True Positive (TP): number of samples in the class "true floor" correctly classified as "true floor".
- True Negative (TN): the number of samples in the class "not true floor" correctly classified as "not true floor".
- False Positive (FP): number of samples in the class "not true floor" incorrectly classified as "true floor".
- False Negative (FN): the number of samples in the class "true floor" incorrectly classified as "not true floor".

In this dataset, the classification problem belongs to the type of multi-class classifier. The macro average metric is used since the macro average is a good measure in case of predicting class well. Macro average calculates metrics for individual classes, then computes their average values regardless of the overall size. We calculate the macro average metric for Precision, Recall, and F1-score. Therefore, in the result section, the Precision, Recall, and F1-score metrics for each class are detailed by class. Then, we display their macro average values alongside the accuracy metric.

Regression model. Evaluating the performance of a regression model requires an approach and metrics different from which are used to evaluate a classification model. The regression model estimates continuous values (in this research they are longitudes and latitudes). Therefore, regression performance metrics quantify how close the model predictions are to actual (true) values. The used regression performance metrics are as follows:

- $R^2 - Score$ evaluates the performance of a regression-based machine learning model defined as Eq. (12)

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \overline{y}_i)^2}, \tag{12}$$

  where y_i is either the actual latitude or longitude value, $\hat{y}_i$ is either the estimated latitude or longitude value, $\overline{y}_i$ is the average value.

- MSE (Mean Squared Error): MSE measures the average of the squared difference between predictions and actual output values and is defined as Eq. (13)

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{N}, \tag{13}$$

  where y_i is either the actual latitude or longitude value, $\hat{y}_i$ is either the estimated latitude or longitude value, and N is the total number of samples.

- MAE (Mean Absolute Error): MAE measures the absolute error between predicted and actual values and is defined as Eq. (14)

$$MAE = \frac{\sum_{i=1}^{n}|y_i - \hat{y}_i|}{N}, \tag{14}$$

  where y_i is either the actual latitude or longitude value, $\hat{y}_i$ is either the estimated latitude or longitude value, and N is the total number of samples.

4.3. Building the classification model used to predict the building-floor

4.3.1. Start-up phase

Hyperparameter optimization is an important process for high performance machine learning models. Optuna [39] is a popular Python library for hyperparameter optimization that supports many optimization algorithms. The models are evaluated based on the following metrics: Accuracy, Balanced Accuracy, ROC AUC, and F1 Score. Accordingly, several classifiers are selected for classification with a building-floor dataset such as LogisticRegression (LR), LinearDiscriminantAnalysis (LDA), KNeighborsClassifier (KNN), Classification

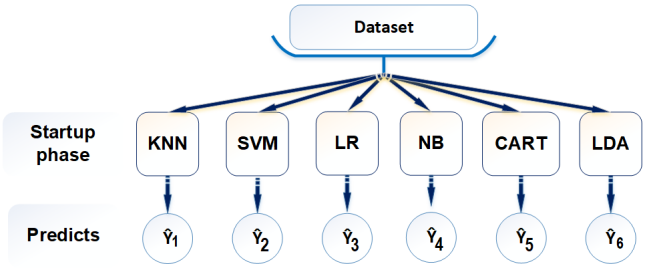


Figure 5: Process of the independent models

and Regression Tree (CART), GaussianNB (NB), and Support vector machine (SVM). The independent models are implemented as shown in Figure 5.

Results and evaluations of the independent models

The classification results of each model according to the precision and recall metrics are shown in Tables 1a and 1b. The F1-score metric can be seen in Table 2, in which 'Bx_y' represents the building x and the floor y. Figures 6a, 6b, and 6c show the comparison charts of Precision, Recall, and F1-Score metrics presented in Table 1 and 2. These graphs demonstrate that the LR, KNN, and SVM algorithms produce superior results to the remaining algorithms.

Table 1: Precision and Recall metrics of the independent models used to predict the building-floor

(a) Precision metric

Floor	LR	LDA	KNN	CART	NB	SVM
B0_0	97.49	94.06	93.81	95.61	56.66	98.51
B0_1	95.93	94.50	96.58	96.60	30.93	98.32
B0_2	94.12	94.12	98.19	94.74	82.22	97.90
B0_3	96.25	96.27	96.32	96.99	80.00	98.11
B1_0	97.06	94.72	97.82	98.89	59.91	98.51
B1_1	97.63	91.37	100.00	95.79	57.40	98.43
B1_2	97.03	92.00	99.63	97.74	90.70	98.18
B1_3	93.00	95.08	94.15	92.46	68.58	95.43
B2_0	98.97	98.97	98.48	99.74	56.80	99.49
B2_1	98.08	94.33	99.51	96.91	86.76	99.28
B2_2	96.88	93.56	98.71	95.48	44.00	99.06
B2_3	97.56	95.53	98.33	97.38	93.88	99.07
B2_4	96.08	92.90	98.73	96.18	20.24	99.35

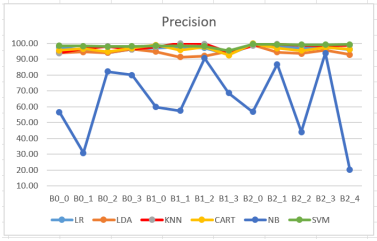
(b) Recall metric

Floor	LR	LDA	KNN	CART	NB	SVM
B0_0	96.04	94.06	97.52	97.03	99.01	98.02
B0_1	94.65	91.97	94.31	94.98	54.52	97.99
B0_2	95.10	95.10	94.76	94.41	12.94	97.90
B0_3	97.72	98.10	99.62	98.10	38.02	98.86
B1_0	98.14	93.31	100.00	99.63	98.88	98.51
B1_1	97.24	91.73	97.24	98.43	50.39	98.43
B1_2	96.67	93.70	98.89	95.93	28.89	99.63
B1_3	96.37	90.16	100.00	95.34	80.31	97.41
B2_0	97.23	96.47	98.24	97.48	100.00	97.48
B2_1	97.37	95.23	97.37	97.37	14.08	98.57
B2_2	97.79	96.21	96.21	93.38	34.70	99.37
B2_3	97.74	96.43	99.62	97.74	8.650	99.81
B2_4	94.84	92.90	100.00	97.42	99.35	98.06

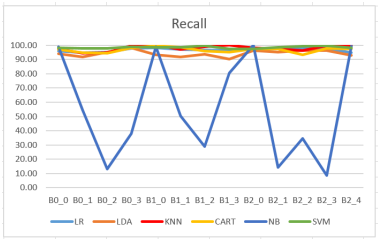
The macro averages and accuracy metrics of the independent models are shown in Table 3. The improved classification performance of the independent models more specifically displayed in Figure 7. In which, Figure 7a is a comparison chart of macro average Precision, macro average Recall, macro average F1-Score, and Accuracy metrics presented in Table 3. Figure 7b is an image from the system showing the comparison of the independent models. They all make it more evident that LR, KNN, and SVM algorithms produce better results than the remaining algorithms. The metrics of the two algorithms SVM and KNN are all greater than 97% and surpass those of the other algorithms. Although the metrics of LR

Table 2: F1-score metric of the independent models used to predict the building-floor

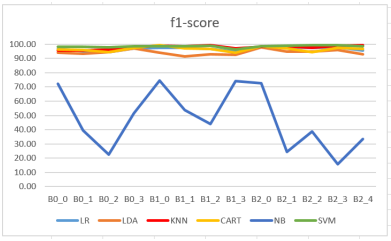
Floor	LR	LDA	KNN	CART	NB	SVM
B0_0	96.76	94.06	95.63	96.31	72.07	98.26
B0_1	95.29	93.22	95.43	95.78	39.47	98.16
B0_2	94.61	94.61	96.44	94.57	22.36	97.90
B0_3	96.98	97.18	97.94	97.54	51.55	98.48
B1_0	97.60	94.01	98.90	99.26	74.61	98.51
B1_1	97.44	91.55	98.60	97.09	53.67	98.43
B1_2	96.85	92.84	99.26	96.82	43.82	98.90
B1_3	94.66	92.55	96.98	93.88	73.99	96.41
B2_0	98.09	97.70	98.36	98.60	72.45	98.47
B2_1	97.72	94.77	98.43	97.14	24.23	98.92
B2_2	97.33	94.87	97.44	94.42	38.80	99.21
B2_3	97.65	95.98	98.97	97.56	15.83	99.44
B2_4	95.45	92.90	99.36	96.79	33.62	98.70



(a) Precision metric comparison



(b) Recall metric comparison



(c) F1-Score metric comparison

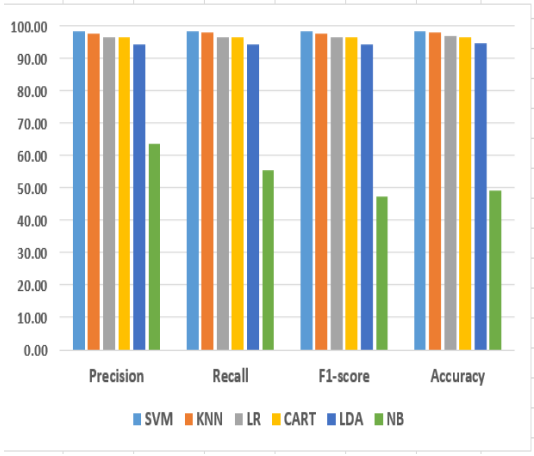
Figure 6: Evaluation metrics comparison of the independent models used to predict the floor

and CART are approximatelly the same, we only select one algorithm. Despite having lower recall metrics than the CART algorithm (96.69% vs 96.71%), the LR algorithm's F1-Score is higher (96.71% vs 96.60%), which proves that the LR model is better. Therefore, we select LR.

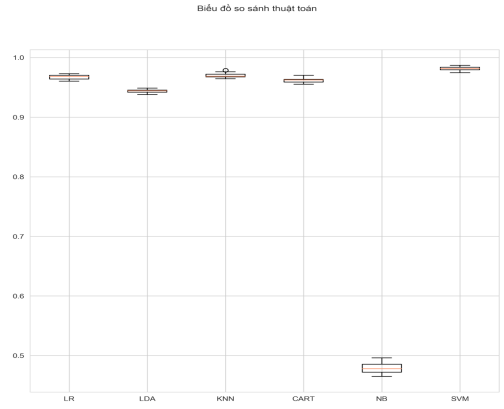
Table 3: Macro averages and accuracy metrics of the independent models used to predict the building-floor

	SVM	KNN	LR	CART	LDA	NB
Macro averages-Precision	**98.43**	**97.71**	**96.62**	96.50	94.42	63.70
Macro averages-Recall	**98.47**	**97.98**	**96.69**	96.71	94.26	55.37
Macro averages-F1 score	**98.45**	**97.83**	**96.65**	96.60	94.33	47.42
Accuracy	**98.57**	**97.93**	**96.86**	96.76	94.66	49.09
Time (s)	7.95	0.04	3.19	0.47	1.21	0.67

Finally, we choose three algorithms with the best results which are LR, KNN, and SVM for the first phase of the combined model. In the second phase, we choose the Logistic Regression algorithm.



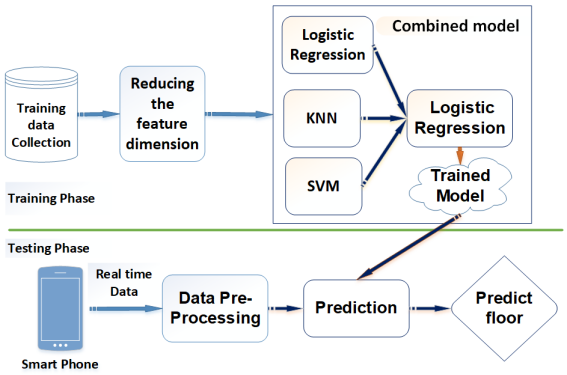
(a) Comparison of the macro averages and Accuracy metrics



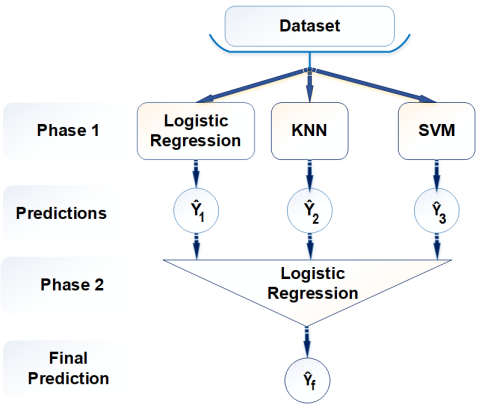
(b) Accuracy comparison of the independent models used to predict the floor

Figure 7: Classification performance comparison of the independent models used to predict the building-floor

4.3.2. Combined model used to predict the building-floor



(a) Structure of the combined model



(b) Combination process of the classification models

Figure 8: The combined model to predict the building-floor

Figure 8 shows our combined model to predict the buildings - floors, in which Figure 8a shows a combined model of classification algorithms used to predict the buildings - floors. First, the feature reduction algorithm is applied. Then, the model is trained by the three algorithms LR, KNN, and SVM in the first phase, and it continues to be trained by the Logistic Regression algorithm in the second phase.

Figure 8b shows in detail the combination process of the algorithms used for floor prediction, where $\hat{Y}_1$ $\hat{Y}_2$, and $\hat{Y}_3$ are the prediction results of the three models in the first phase, $\hat{Y}_f$ is the result of the second phase (final predictions).

4.4. Building the regression model used to estimate latitude

4.4.1. Start-up phase

Similar to the selection of machine learning algorithms to classify building floors, Optuna [39] is also used to select regression models in the prediction of datasets by latitudes. The models are evaluated based on the following metrics: Accuracy, Balanced Accuracy, ROC AUC, and F1 Score, selected some of the best models: Support vector machine, ExtraTreeRegressor (ExtraTree), GradientBoostingRegressor (GB), KNeighborsRegressor (KNN), RandomForestRegressor (RF), and LGBMRegressor (LGBM) to choose the optimal ones for the first phase of the combined model as shown in Figure 9.

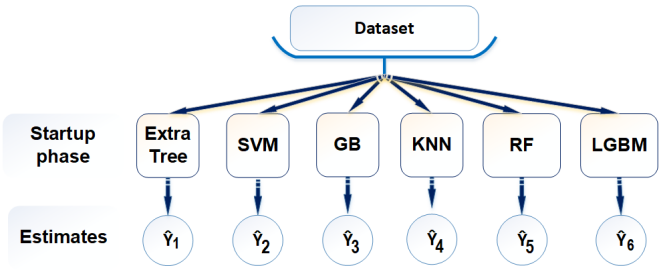


Figure 9: Process of the regression independent models

Results and evaluations of the independent models
The test results are shown in Table 4.

Table 4: Performance of the independent models used to estimate latitude

	SVM Regressor	**ExtraTree Regressor**	GB Regressor	**KNN Regressor**	**RF Regressor**	**LGBM Regressor**
R2-Score(%)	96.1	**98.6**	95.5	**99.3**	**99.4**	**98.8**
MSE(m)	175.2	**54.4**	200.5	**31.03**	**24.8**	**52.2**
MAE(m)	8.32	**2.75**	10.50	**2.55**	**2.18**	**4.61**
Time(s)	66.35	**0.38**	9.5	**0.027**	**37.8**	**0.32**

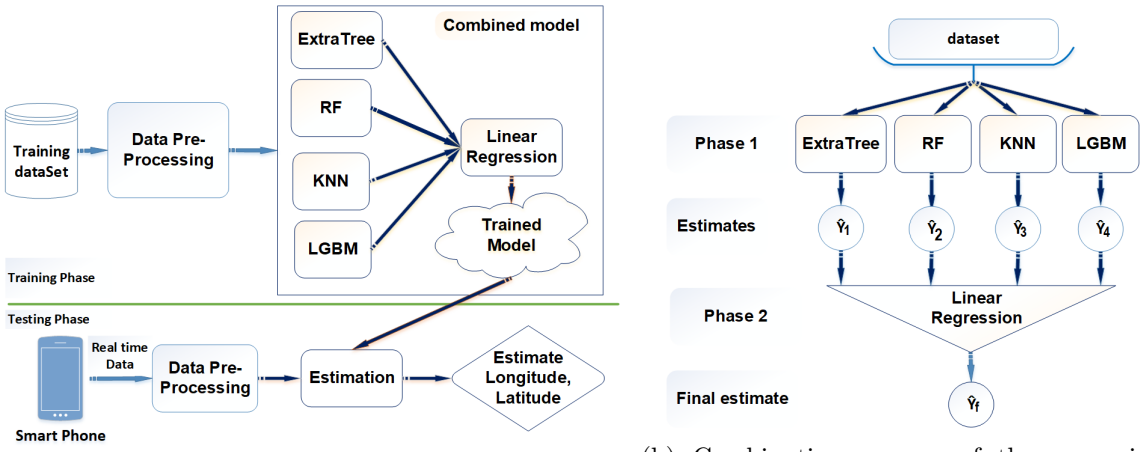
According to this result, the four regression algorithms ExtraTree, KNN, RF, and LGBM, which have R2-Score results of 98.6%, 99.3%, 99.4%, and 98.8%, respectively, are superior to the SVM and the GB algorithms. In addition, they have significantly better MAE and MSE scores. As a result, in the combined models used to estimate longitude and latitude, we choose ExtraTree, KNN, RF, and LGBM algorithms for the first phase and Linear Regression algorithm for the second phase.

4.4.2. Combined model used to estimate latitude

The combined model for latitude estimation is shown in Figure 10. Figure 10a shows the model structure. In the first phase, after preprocessing the training dataset, the model is trained by the ExtraTree, KNN, RF, and LGBM regression algorithms.

The model continues to be trained by the Linear Regression regression algorithm in the second phase according to the process shown in algorithm 2. Figure 10b shows in detail the combination process of the regression algorithm, where $\hat{Y}_1$, $\hat{Y}_2$,$\hat{Y}_3$, and $\hat{Y}_4$ are the training

results (estimation results) of the four models in the first phase, and $\hat{Y}_f$ is the result of the second phase (final estimates).



(a) Structure of the combined model

(b) Combination process of the regression models

Figure 10: The latitude estimation combined model

4.5. Building the regression model used to estimate longitude

4.5.1. Start-up phase

In the same way in Optuna [39], we test the independent algorithms SVM, ExtraTree, GB, KNN, RF, and LGBM regarding the model used to estimate the longitude shown in Figure 9.

Results and evaluations of the independent models

The test results are shown in Table 5. The R2-Score results for all of the algorithms ExtraTree, KNN, RF, and LGBM are greater than 99% while those for the two algorithms SVM and GB are just close to 97%. SVM and GB have very big MSE metrics, compared with the remaining algorithms that have the metrics less than 3 times. Moreover, the MAE metrics of SVM and GB are roughly 3 to 8 times greater than those of other algorithms. Consequently, in the combined model used to estimate longitude we choose ExtraTree, KNN, RF, and LGBM regression algorithms for the first phase, and Linear Regression algorithm for the second phase

Table 5: Performance of the independent models used to estimate longitude

	SVM Regressor	**ExtraTree Regressor**	GB Regressor	**KNN Regressor**	**RF Regressor**	**LGBM Regressor**
R2-Score(%)	96.94	**99.30**	96.7	**99.49**	**99.606**	**99.2**
MSE(m)	477.36	**109.4**	509.3	**79.39**	**61.5**	**112.4**
MAE(m)	13.85	**3.62**	16.02	**3.25**	**2.72**	**5.99**
Time(s)	59.11	**0.35**	9.63	**0.027**	**34.3**	**0.32**

4.5.2. Combined model used to estimate longitude

The algorithms selected at the "startup" phase are the ones used in the regression model to estimate the latitude. Therefore, the combined model using the latitude estimate in Figure 10 is also used to estimate the longitude.

4.6. Results and evaluations of combined models

In this section, we present the experimental results as well as the evaluations of the combined models that we have proposed. These results have been compared with those of independent models.

In the floor prediction model, the training dataset is

$$\mathcal{D}=\{RSS_1, RSS_2, .., RSS_{520}, floor, buildingID\}_1^m,$$

where m is the total number of data lines and where $\mathcal{X}=\{RSS_1, RSS_2, .., RSS_{520}\}_1^m$, $\mathcal{Y}=\{floor, buildingID\}_1^m$. The training data set is randomly divided into four parts X_1, X_2, X_3, and X_4. X_1 to X_3 are used for the three algorithms in the first phase, and X_4 is used for the algorithm in the second phase. The training data used for the longitude and latitude estimation model is defined as Eq. (15)

$$\mathcal{D} = \{RSS_1, RSS_2, .., RSS_{520}, longtitude, latitude\}_1^m, \tag{15}$$

where $\mathcal{X} = \{RSS_1, RSS_2, .., RSS_{520}\}_1^m\}$, and $\mathcal{Y} = \{longtitude, latitude\}_1^m$. Similar to the floor prediction model, the training data of the latitude and longitude estimation model is randomly divided into five parts, four parts for the first phase and the rest for the second phase.

4.6.1. Hyperparameter tuning with grid search for building-floor classifier models and longitude, latitude regression models

When working on datasets and using Machine Learning models, it is difficult to know which set of hyperparameters will yield the best results. To get the best set of hyperparameters, we can use Grid Search [40] then pass all combinations of hyperparameters into the model one by one and check the results. In the end, we get the best result in the model.
Hyperparametric model is a property of the model that is outside the model and its value cannot be estimated from the data. The value of the hyperparameter must be set before the learning process. For example, c in Support Vector Machines, k in k-Nearest Neighbors, number of hidden layers in Neural Networks. In contrast, some parameters are internal features of the model, and their values can be estimated from the data. For example, the beta of a logistic/linear regression or a support vector in a Support Vector Machine. Some important parameters in parameter editing with GridSearchCV are defined as:

- **Estimator**: This is the model we want to use in GridSearchCV.
- **Param_grid**: Dictionary or list of model or function parameters for GridSearchCV to choose the best one.
- **Scoring**: A measure that evaluates the performance of the model to decide the best hyperparameter, if not specified, it will use the estimator score.

- **CV**: An integer value, as it represents the number of splits required for cross-validation. By default, it is set to 5.
- **n_jobs**: Number of jobs to be run in parallel, -1 indicates all processor usage.

The results that GridSearchCV determines can be considered the best based on the parameters set into the "param_grid" to create a combination of parameters that improves the performance of the model. Table 11 and Table 12 in the Appendix are the model and hyperparameter tuning for the building-floor classifiers and the longitude-latitude regressors.

4.6.2. Results and evaluations of the classification model used to predict the building-floor

The results of the combined model are shown in Tables 6 and 7. Table 6 shows the results of Precision, Recall, and F1-score metrics. Table 7 shows the values of the metrics: macro average Precision, macro average Recall, macro average F1-score, and Accuracy. Figure 11 shows the performance comparison of the combined models with the independent models. The results show that the combined model performs better, having better metrics than the independent model of the LR, KNN, and SVM algorithms.

Table 6: Precision, Recall, and F1-score metrics of the combined model used to predict the building-floor

Floor	Precision	Recall	F1-score
B0_0	98.51	98.51	98.51
B0_1	98.65	97.66	98.15
B0_2	97.55	97.55	97.55
B0_3	97.74	98.86	98.30
B1_0	98.53	99.63	99.08
B1_1	98.81	98.43	98.62
B1_2	98.89	98.89	98.89
B1_3	99.46	95.85	97.63
B2_0	98.02	100.00	99.00
B2_1	99.76	98.33	99.04
B2_2	98.73	98.42	98.58
B2_3	99.25	99.81	99.53
B2_4	99.36	100.00	99.68

Table 7: Macro averages and accuracy metrics of the combined model used to predict the building-floor

	Macro avg Precision	Macro avg Recall	Macro avg F1-Score	Accuracy	Time(s)
Combined model	98.71	98.61	98.66	98.73	99.31

Results and evaluation without using the feature reduction method

We also experimented with the floor prediction combined model without using the feature reduction method that we proposed. The results of the model in cases of using and not

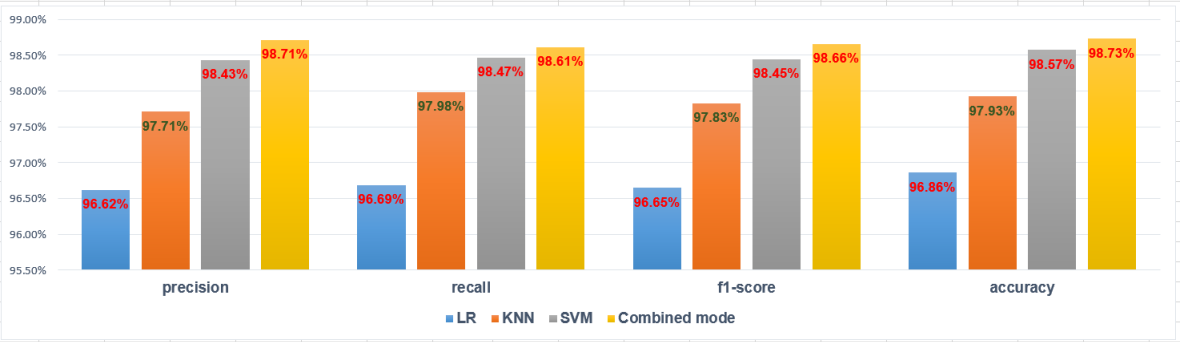


Figure 11: Performance comparison of the combined model with the independent models used to predict the building-floor

using the feature reduction methods have been summarized in Table 8. This shows that when preprocessing the dataset by the method of feature reduction, the accuracy slightly increases and the execution time decreases. Thus, the feature reduction method, which we have proposed, is also beneficial.

Table 8: Combined model in cases of using and not using feature reduction method

Model	precision	recall	f1-score	accuracy	Time(s)
Combined model **with** feature reduction method	98.71%	98.61%	98.66%	98.73%	99.31
Combined model **without** feature reduction method	98.22%	97.13%	97.67%	98.24%	118.56

4.6.3. Results and evaluations of the regression model used to estimate latitude

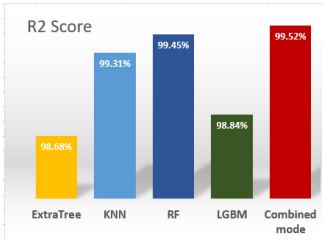
Table 9 shows the performance of the combined model used to estimate latitude. Figure 12 is the performance comparison of the combined model with the independent models. The comparison results show R2 metric of the combined model is higher than the independent models (figure 12a); MSE and MAE metrics (figures 12b,12c) of the combined model is lower than the independent models.
In summary, the combined model has higher performance and more accurate latitude estimates than the independent models.

Table 9: Performance of the combined model used to estimate latitude

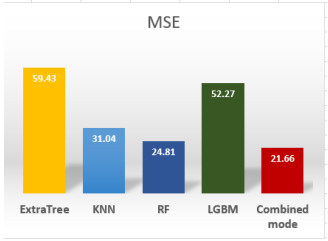
	R2-Score(%)	MSE(m)	MAE(m)	Time(s)
Combined model	99.52	21.66	1.95	170.82

4.6.4. Results and evaluations of the regression model used to estimate longitude

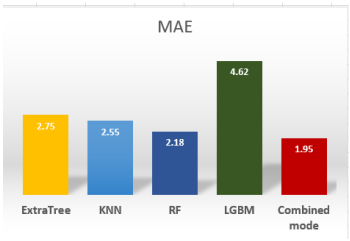
The combined model results used to estimate longitude are shown in Table 10. The combined model has the R2 metric of 99,621%, the MSE metric of 59.32m, and the MAE



(a) Comparison of the R2 metric



(b) Comparison of the MSE metric



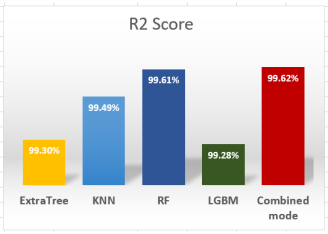
(c) Comparison of the MAE metric

Figure 12: Performance comparison between the combined model and the independent models used to estimate latitude.

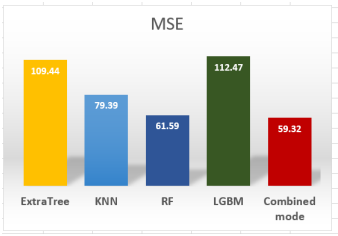
metric of 2.7m, which are better than those of the independent models. The performance comparison of the combined model and the independent models used to estimate longitude are shown in Figure 13. The detailed comparisons are shown in Figures 13a, 13b, and 13c. It is once again confirmed that the combined model performs better than the independent models.

Table 10: Performance of the combined model used to estimate longitude

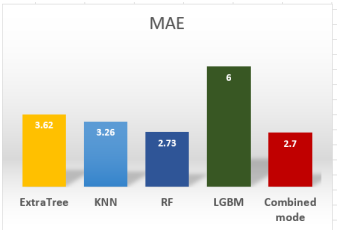
	R2-Score(%)	**MSE(m)**	**MAE(m)**	**Time(s)**
Combined model	99.621	59.32	2.70	165.00



(a) Comparison of the R2 metric



(b) Comparison of the MSE metric



(c) Comparison of the MAE metric

Figure 13: Performance comparison between the combined model and the independent models used to estimate longitude.

In conclusion, across all tests the combined model we propose has high performance and accuracy in floor prediction and location estimation. The test results of the combined model are better than the independent models.

4.7. Evaluation of the positioning accuracy in the testing phase

Figure 14 shows how models performed with localizing targets for PhoneID=14. This image shows the result of location accuracy in terms of longitude and latitude at three buildings. When the user moves, the time is recorded, and the corresponding latitude and longitude values are also recorded, Figure 15 shows the results of the location accuracy in

longitude and latitude according to this movement. The green color represents the estimated location. The orange color represents the actual location. The color positions indicate a close match between the estimated position and the actual position. These images confirm the accuracy of our proposed model.

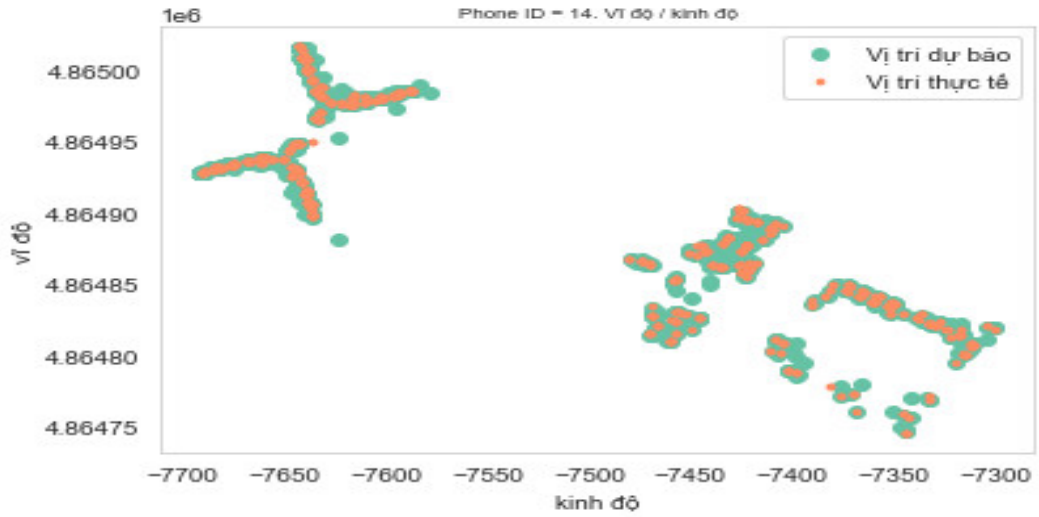


Figure 14: Test results with the phoneID=14

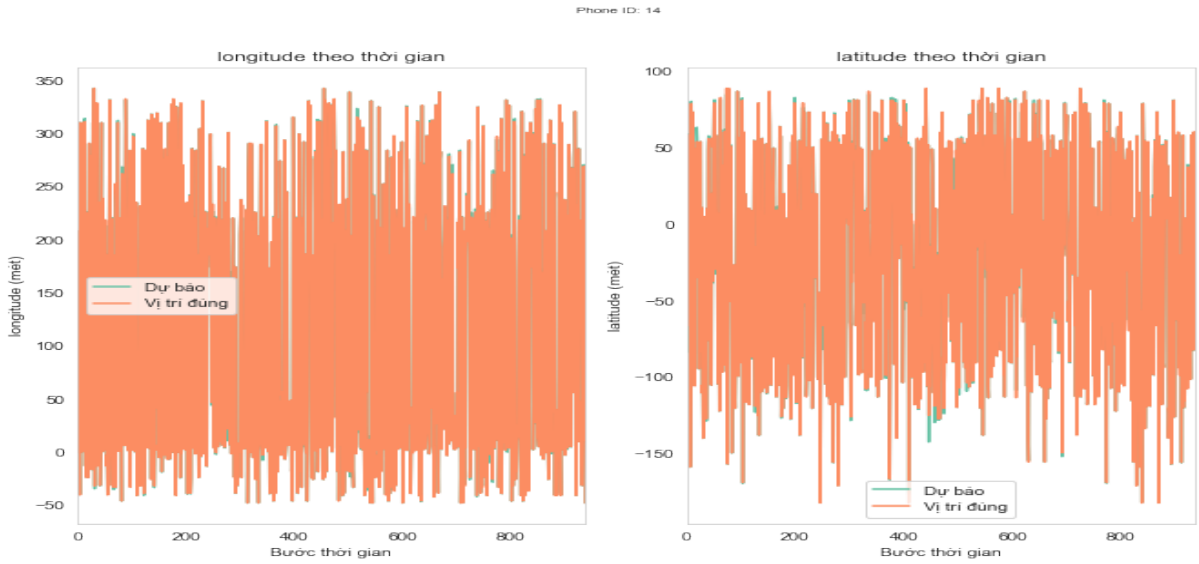


Figure 15: Test results by time with the phoneID=14

4.8. Evaluation of the results between the proposed model and other models on the same UJIIndoorLoc dataset

The algorithms of other studies are numbered according to the reference number for the purpose of comparison.

4.8.1. Evaluation of the floor prediction results

Figure 16 shows the floor prediction results of our proposed model and other studies. Accordingly, our model has superior accuracy than majority of other models. The results of the study in [38] were better than ours. The distinction is that the authors of that study divided the training dataset into two datasets (training and testing) in an 80/20 ratio and their goal was room-level positioning. As a result, the training datasets of our study and theirs include different amounts of samples and distinct purposes, making the conclusions of the comparison analysis just relative.

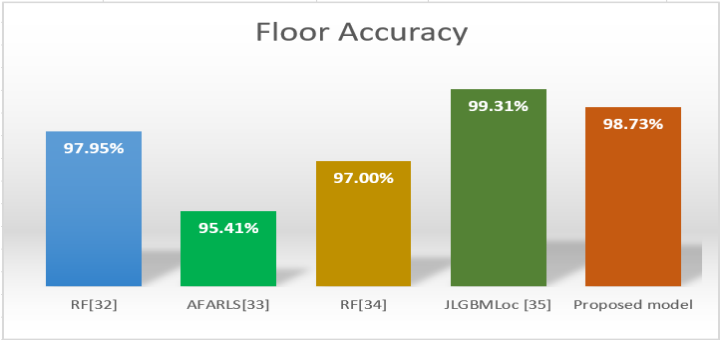


Figure 16: Comparison of the floor prediction results of the proposed model with other models.

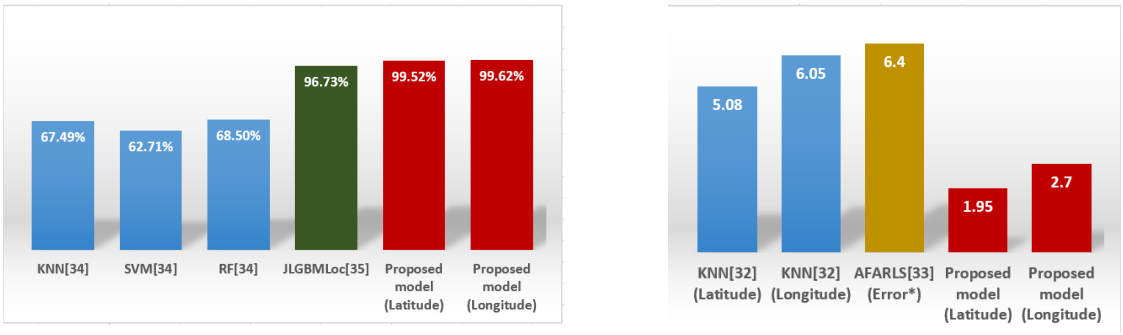
4.8.2. Evaluation of the results of longitude and latitude estimates

In our study, the latitude and longitude of the user's location are estimated. To compare our findings with those of other studies, we thus refer to estimates for latitude and longitude. The positioning performance between our proposed model and other models is shown in Figure 17. In which, Figure 17a shows the accuracy comparison, and Figure 17b shows the MAE comparison between the models. Noted that one of these models is taken from the study [36] in which the authors used the Error* Metric meaning "the highest mean positioning accuracy". The charts in Figure 17 show that our model has better results than other studies. Noticeably, the accuracy by the position of our proposed model is higher than that of the study in [38].

5. CONCLUSION

In this study, we have proposed a machine learning model that combines algorithms in two phases. From this proposed model, we built a floor prediction model as well as longitude and latitude estimation models.

As a result, the performance and accuracy of all three models are higher than those of the independent models. Majority of other models built on the same datasets have lower performance and accuracy when compared with our models. We have also proposed a feature reduction method for the classification problem. The running duration of our models is a drawback because they are a combination of numerous methods. However, since the models are used during the training phase, this is not a major issue. Our models, otherwise, have a



(a) Comparison of the accuracy

(b) Comparison of the MAE

Figure 17: Performance comparison of location positioning between the proposed model and other models.

lot of potential for improvement because all of the algorithms we used are traditional machine learning algorithms, which can be replaced by deep learning algorithms in the future, and to evaluate the model's performance, we will additionally test it on several datasets. At the same time, we also evaluate the feature reduction method for its ability to increase accuracy when using it.

APPENDIX

The hyperparameter tuning using scikit-learn's GridSearchCV [40] runs through all the different parameters fed into the parameter grid and generates the best combination of parameters, based on the selected scoring metric (accuracy, f1, etc.). However, one limitation that GridSearchCV is the best parameter is limited and takes a long time. The "best" parameters that GridSearchCV defines are technically the best that can be generated, but only using those that you have included in your parameter grid.

Example using Support Vector Machine as a Machine Learning model to use GridSearchCV. The first define the parameters of the model passed into GridSearchCV to get the best parameters. So we create a parameter dictionary consisting of 'C' or 'gamma'.

```
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
svm = SVC()
# defining parameter range
parameters = {'C':[0.1,1,10,100,1000],'gamma': [1, 0.1, 0.01,
                0.001, .0001],'kernel':['rbf']}
grid_SVM = GridSearchCV(estimator = svm, param_grid = parameters,
cv = 2, n_jobs=-1)
# fitting the model for grid search
grid_SVM.fit(X_train, y_train_bf)
# print best parameter after tuning
print(grid.best_params_)
```

Table 11: Optimal parameters for building - floor classification using GridSearchCV

Building - Floor Classifier				
Model	**grid_params**	**Estimator Model**	**model tuned-hyperparameters**	**best_params_**
SVM	param_grid = {'C': [0.1, 1, 10, 100, 1000], 'gamma': [1, 0.1, 0.01, 0.001, 0.0001], 'kernel': ['rbf']}	svm = SVC()	grid_svm = GridSearchCV(estimator = svm, param_grid = param_grid, cv = 2,n_jobs = -1)	{ 'C': 100, 'gamma': 0.1, 'kernel': 'rbf'}
KNeighbors Classifier	k_range = list(range(1, 31)) param_grid = dict(n_neighbors=k_range)	knn = KNeighborsClassifier()	grid_knn = GridSearchCV(estimator = knn, param_grid= param_grid,cv = 10, scoring= 'accuracy', return_train_score = False,verbose = 1)	{ 'n_neighbors':1}
Logistic Regression	param_grid = 'C':np.logspace(-3,3,7), 'penalty':['l1', 'l2']}	logreg = LogisticRegression()	grid_logreg = GridSearchCV(estimator= logreg, param_grid = param_grid, cv = 10)	{ 'C': 10.0, 'penalty': 'l2'}

Table 12: Optimal parameters for Longitute - Latitude Regression using GridSearchCV

LONGITUDE				
Model	**grid_params**	**model**	**model & tuned-hyperparameters**	**best_params_**
ExtraTrees Regressor	param_grid={ 'max_features': range(50,401,50) }	extreg= ExtraTrees Regressor()	grid_extree = GridSearchCV(estimator = extreg, param_grid = param_grid, scoring='r2', cv=5)	{'max_features': 100}
KNeighbors Regressor	param_grid= { 'n_neighbors': range(1, 20)}	knnreg = KNeighbors Regressor()	grid_knnreg = GridSearchCV(estimator= knnreg, param_grid = param_grid, scoring= 'neg_mean_squared_error', cv=10)	n_neighbors=4
RandomForest Regressor	param_grid = [{ 'RF__max_depth': [8, 12, 16], 'RF__min_samples_split': [12, 16, 20], 'RF_criterion': 'gini', 'entropy']}]	rfreg = RandomForest Regressor()	grid_rfreg = GridSearchCV(estimator= rfreg, param_grid= param_grid, cv=5, n_jobs=-1, verbose=2)	{'max_depth': 100, 'max_features': 3, 'min_samples_leaf': 1, 'n_estimators': 600}
LGBM Regressor	param_grid = { 'num_leaves': [31, 127], 'feature_fraction': [0.5, 1.0], 'bagging_fraction': [0.75, 0.95], 'reg_alpha': [0.1, 0.5]}	lgb_reg = lgb.LGBM Regressor()	grid_lgbreg = GridSearchCV(estimator=lgb_reg, param_grid=param_grid, cv=10)	{'bagging_fraction': 0.75, 'feature_fraction': 0.5, 'num_leaves': 127, 'reg_alpha': 0.5}

REFERENCES

[1] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*. Springer Science & Business Media, 2012.

[2] C. Nagel, T. Becker, R. Kaden, K.-J. Li, J. Lee, and T. H. Kolbe, "Requirements and space-event modeling for indoor navigation - how to simultaneously address route planning, multiple localization methods, navigation contexts, and different locomotion types," 2010.

[3] S. Chan and G. Sohn, "Indoor localization using wi-fi based fingerprinting and trilateration techniques for lbs applications," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 38, no. 4, p. C26, 2012.

[4] R. K. Harle, "A survey of indoor inertial positioning systems for pedestrians," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 1281–1293, 2013.

[5] C. Basri and A. El Khadimi, "Survey on indoor localization system and recent advances of wifi fingerprinting technique," in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*. IEEE, 2016, pp. 253–259.

[6] E. Laitinen, E. S. Lohan, J. Talvitie, and S. Shrestha, "Access point significance measures in WLAN-based location," in *2012 9th Workshop on Positioning, Navigation and Communication*, 2012, pp. 24–29.

[7] S.-H. Fang, T. Lin, and K.-C. Lee, "A novel algorithm for multipath fingerprinting in indoor wlan environments," *IEEE Transactions on Wireless Communications*, vol. 7, 2008.

[8] K. Kaemarungsi and P. Krishnamurthy, "Properties of indoor received signal strength for wlan location fingerprinting," *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.*, pp. 14–23, 2004.

[9] I. T. Jolliffe and J. Cadima, "Principal component analysis: a review and recent developments," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 374, 2016.

[10] D. Chu, L.-Z. Liao, M. K. Ng, and X. Wang, "Incremental linear discriminant analysis: A fast algorithm and comparisons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, pp. 2716–2735, 2015.

[11] A. B. Adege, H.-P. Lin, G. B. Tarekegn, and S.-S. Jeng, "Applying deep neural network (DNN) for robust indoor localization in multi-building environment," *Applied Sciences*, 2018.

[12] A. Nessa, B. Adhikari, F. Hussain, and X. N. Fernando, "A survey of machine learning for indoor positioning," *IEEE Access*, vol. 8, pp. 214 945–214 965, 2020.

[13] N. Singh, S. Choe, and R. Punmiya, "Machine learning based indoor localization using Wi-Fi RSSI fingerprints: an overview," *IEEE Access*, 2021.

[14] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern classification, 2nd edition," 2000.

[15] P. Bahl and V. N. Padmanabhan, "Radar: An in-building RF-based user location and tracking system," in *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*, vol. 2. Ieee, 2000, pp. 775–784.

[16] C. Sujin and G. Kousalya, *Fingerprint-based support vector machine for indoor positioning system*, 01 2019, vol. 670, pp. 289–298.

[17] D. J. Suroso, A. S. H. Rudianto, M. Arifin, and S. Hawibowo, "Random forest and interpolation techniques for fingerprint-based indoor positioning system in un-ideal environment," *International Journal of Computing and Digital Systems*, vol. 10, pp. 701–713, 2021.

[18] M. Maduranga and R. Abeysekera, "Treeloc: An ensemble learning-based approach for range based indoor localization," *International Journal of Wireless and Microwave Technologies*, 2021.

[19] H. Zhang and Y. Li, "Lightgbm indoor positioning method based on merged Wi-Fi and image fingerprints," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[20] Z. Peng, Y. Xie, D. Wang, and Z. Dong, "One-to-all regularized logistic regression-based classification for Wi-Fi indoor localization," *2016 IEEE 37th Sarnoff Symposium*, pp. 154–159, 2016.

[21] F. Vanheel, J. Verhaevert, E. Laermans, I. Moerman, and P. Demeester, "Automated linear regression tools improve rssi wsn localization in multipath indoor environment," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, pp. 1–27, 2011.

[22] J. Torres-Sospedra, R. Montoliu, A. Martínez-Usó, J. P. Avariento, T. J. Arnau, M. Benedito-Bordonau, and J. Huerta, "Ujiindoorloc: A new multi-building and multi-floor database for wlan fingerprint-based indoor localization problems," in *2014 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2014, pp. 261–270.

[23] M. T. Hoang, Y. Zhu, B. Yuen, T. Reese, X. Dong, T. Lu, R. Westendorp, and M. Xie, "A soft range limited k-nearest neighbors algorithm for indoor localization enhancement," *IEEE Sensors Journal*, vol. 18, no. 24, pp. 10 208–10 216, 2018.

[24] X. Zhu, "Indoor localization based on optimized KNN," *Netw. Commun. Technol.*, vol. 5, pp. 34–39, 2020.

[25] P. Dai, Y. Yang, M. Wang, and R. Yan, "Combination of DNN and improved KNN for indoor location fingerprinting," *Wireless Communications and Mobile Computing*, vol. 2019, 2019.

[26] L. Zhang, Y. Li, Y. Gu, and W. Yang, "An efficient machine learning approach for indoor localization," *China Communications*, vol. 14, no. 11, pp. 141–150, 2017.

[27] Y. Rezgui, L. Pei, X. Chen, F. Wen, and C. Han, "An efficient normalized rank based SVM for room level indoor Wi-Fi localization with diverse devices," *Mobile Information Systems*, vol. 2017, pp. 1–19, 07 2017.

[28] D. Z. Abidin, S. Nurmaini, Erwin, E. Rasywir, and Y. Pratama, "Indoor positioning system in learning approach experiments," *J. Electr. Comput. Eng.*, vol. 2021, pp. 6 592 562:1–6 592 562:16, 2021.

[29] S. Lee, J. Kim, and N. Moon, "Random forest and wifi fingerprint-based indoor location recognition system using smart watch," *Human-centric Computing and Information Sciences*, vol. 9, pp. 1–14, 2019.

[30] J. Gao, X. Li, Y. Ding, Q. Su, and Z. Liu, "Wifi-based indoor positioning by random forest and adjusted cosine similarity," *2020 Chinese Control and Decision Conference (CCDC)*, pp. 1426–1431, 2020.

[31] C. Xiang, Z. Zhang, S. Zhang, S. Xu, S. Cao, and V. K. N. Lau, "Robust sub-meter level indoor localization - a logistic regression approach," *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, 2019.

[32] C. Xiang, S. Zhang, S. Xu, X. Chen, S. Cao, G. C. Alexandropoulos, and V. K. N. Lau, "Robust sub-meter level indoor localization with a single wifi access point—regression versus classification," *IEEE Access*, vol. 7, pp. 146 309–146 321, 2019.

[33] L. Zhang, X. Meng, and C. Fang, "Linear regression algorithm against device diversity for the WLAN indoor localization system," *Wirel. Commun. Mob. Comput.*, vol. 2021, pp. 5 530 396:1–5 530 396:15, 2021.

[34] D. Li, L. Wang, and S. xun Wu, "Indoor positioning system using wifi fingerprint," 2014.

[35] P. R. Shivam Wadhwa and R. Kaushik, "Machine learning based indoor localization using wi-fi fingerprinting," *International Journal of Recent Technology and Engineering*, 2019.

[36] H. Gan, M. H. B. M. Khir, G. Witjaksono Bin Djaswadi, and N. Ramli, "A hybrid model based on constraint oselm, adaptive weighted src and knn for large-scale indoor localization," *IEEE Access*, vol. 7, pp. 6971–6989, 2019.

[37] W. Charoenruengkit, S. Saejun, R. Jongfungfeuang, and K. Multhonggad, "Position quantization approach with multi-class classification for wi-fi indoor positioning system," in *2018 International Conference on Information Technology (InCIT)*, 2018, pp. 1–5.

[38] L. Yin, P. Ma, and Z. Deng, "Jlgbmloc—a novel high-precision indoor localization method based on lightgbm," *Sensors (Basel, Switzerland)*, vol. 21, 2021.

[39] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[40] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Received on October 17, 2022
Accepted on December 26, 2022