A Survey on the Impact of Hyperparameters on Random Forest Performance using Multiple Accelerometer Datasets

Hong-Lam Le[*‡§]
VNU Information Technology Institute, Hanoi, VIETNAM
Vinh University of Technology Education, Vinh City, VIETNAM

Thanh-Tuoi Le and Thi-Thu-Hien Vu[§]
Vinh University of Technology Education, Vinh City, VIETNAM

Doan-Hieu Tran[§]
Ho Chi Minh University of Banking, Ho Chi Minh City, VIETNAM

Dinh Van Chau[§]
Electric Power University, Hanoi, VIETNAM

Thi-Thu-Trang Ngo[†§]
Posts and Telecommunications Institute of Technology, Hanoi City, VIETNAM

Abstract

Previous studies have examined the effects of hyperparameters on random forests, but little research has been done in the context of fall detection. To address this gap, our study aimed to examine how hyperparameters influence the performance and training time of a random forest algorithm used in fall detection systems. Our findings highlighted the best range of values for each hyperparameter to achieve high performance. Moreover, we discovered that certain combinations of hyperparameters could either enhance or reduce the random forest's performance compared to the default settings. To conduct these investigations, we performed experiments using two datasets: MobiAct v2.0 and UP-Fall, which were collected from accelerometers in smartphones and wearables. These insights can contribute to the optimization of hyperparameters for more effective fall detection systems.

Key Words: Hyperparameter, random forest, max_depth, num_Tree, num_features, min_samples_leaf, min_samples_split.

* Corresponding author, created the first draft, and created the stable version of this document.
† Corresponding author, created the stable version of this document.
‡ University Information Technology Institute.
§ lehonglam@vnu.edu.vn, fredleektv@gmail.com, thuhienktv@gmail.com, hieutd@hub.edu.vn, chaudv@epu.edu.vn, trangnttl@ptit.edu.vn.

1 Introduction

Fall detection systems play a crucial role in ensuring the well-being and health of older adults and individuals with mobility impairments. These systems are designed to quickly identify falls and notify caregivers or emergency responders, allowing for prompt medical attention and reducing the risk of harm or fatality. However, current fall detection methods have drawbacks, such as low accuracy and high false alarm rates, which affect their reliability. Thus, it is vital to develop and improve fall detection systems to advance human healthcare.

Enhancing the accuracy, sensitivity, specificity, and reliability of fall detection systems is a primary goal in preventing or mitigating the negative consequences of falls. Various machine learning algorithms, such as J48, Logistic Regression (LR), K-Nearest Neighbor (kNN), Support Vector Machine (SVM), Decision Tree (DT), Naïve Bayes (NB), and Random Forest (RF), have been utilized to classify different human activities, including falls. Studies have shown that RF outperforms other algorithms in fall detection using the MobiAct v2.0 and UP-Fall datasets [5, 10-11, 13]. However, the RF algorithm's performance heavily relies on the selection of hyperparameters [7, 9, 14-15], which poses a significant challenge.

The random forest algorithm is a general classification method that uses hyperparameters to adjust or optimize the loss function. These hyperparameters impact both the model's speed and accuracy, making it crucial to find the optimal values [8, 26]. However, the optimal hyperparameter values are not

universal and depend on the specific problem and data. The influence of hyperparameters on random forest performance, particularly for human daily activity data crucial for fall detection systems, has not been extensively studied and requires further research.

This study determined the optimal range for each hyperparameter of the random forest algorithm to achieve high performance. The results also revealed that combining adjustments of min_samples_leaf and min_samples_split hyperparameters could decrease the random forest's performance. The hyperparameters max_depth, NumTree, and max_features had the most significant impact on performance and model-building time. Combining adjustments of NumTree and max_features, or combining max_depth with NumTree and max_features, resulted in the best performance. These findings have practical implications for constructing optimal models for fall detection systems.

The rest of this paper is structured as follows: Section 2 reviews related work and provides an analysis and summary of studies on hyperparameter tuning for random forests. Section 3 describes our research's evaluation methods, including the research process, the dataset used, and the implementation of the random forest algorithm with adjusted hyperparameters. Section 4 presents our experimental results on the effect of hyperparameters on the random forest algorithm''s performance and provides an interpretation of these results. Finally, the conclusions section summarizes our main findings, discusses their implications, suggests directions for future research, highlights study limitations, and proposes areas for further investigation.

2 Related Works

Recently, random forest algorithms have gained widespread use in machine learning systems due to their high accuracy and robustness against interference. Several studies have proposed solutions to enhance the performance of random forests by optimizing their hyperparameters. For instance, Philipp Probst et al. [15] surveyed the impact of hyperparameters on predictive performance in the field of credit risk. The research was divided into two parts. The first section provided an overview of the adjustment strategies. Later, the author team utilized the tuneRanger R package to automate the tuning of random forests with MBO (Model-Based Optimization) to demonstrate the validity of applying one of these strategies. The benchmark research was conducted on datasets from OpenML and downloaded through the OpenML R package. The authors concluded that the mtry parameter had the most impact on performance. Finally, they recommended using a high number of trees and SMBO (Sequential Model-Based Optimization) to simultaneously tune the mtry parameters, sample size, and node size to enhance the performance of the random forest algorithm.

Kavita M Kelkar et al. [9] proposed an effective learning system based on the random forest algorithm to detect learners' emotional states. This research proposed analyzing hyperparameters such as n_estimators, max_depth, min_samples_split, min_samples_leaf, and criterion to improve the system's performance. However, their paper did not provide a method for investigating the influence of hyperparameters. The experimental results showed that the optimal value for the hyperparameter max_leaf_nodes was 0, which gave the best Kappa score. Additionally, max_features did not affect the accuracy or Kappa value, while a max_depth > 10 provided more accurate results. Of the hyperparameters surveyed, n_estimators were the most important, and their value could range from 25 to 30 without affecting accuracy or Kappa values.

The research by Ningyuan Zhu et al. [27] proposed a random forest-based intrusion detection model for application in electric industrial control systems. The authors also introduced an improved mesh search algorithm (IGSA) to optimize the hyperparameters of the random forest model and improve its efficiency and accuracy. The research adjusted the seven hyperparameters of the random forest model in descending order of preference, including max_depth, min_samples_leaf, min_samples_split, criterion, n_estimators, bootstrap, and max_features. The optimization results revealed that the optimal values for the hyperparameters were max_depth = 24, min_samples_leaf = 1, min_samples_split = 2, criterion = gini, n_estimators = 240, bootstrap = default, and max_features = 9. The test results demonstrated that the intrusion detection method based on hyperparameter optimization achieved higher accuracy, resilience, F1 score, and ROC-AUC score than other methods.

Bernard et al. [1] recommend using the value mtry = sqrt(p) as it provides a reasonable error rate. They emphasize that the number of predictors influences the optimal value of mtry. When many predictors are involved, mtry should set small to both the most and least influential variables are a selection for separation. It may increase relevance but potentially reduce performance. The computation time of the model is devoted to the selected separable variables, so the computation time decreases linearly with decreasing mtry values, similar to the research of author Wright et al. [25]. Conversely, setting mtry too high may exclude less influential variables from contributing to the prediction, as more influential variables are preferred for separation, inadvertently "blurring" variables of less influence.

Probst et al. [16] proposed a general structural framework to evaluate the tunability of hyperparameters in algorithms, including Random Forest. They presented this approach through an application to 38 datasets. The results showed that the mtry parameter had the most influence on the AUC index, followed by the sample size, while the node size had less effect. Hyperparameter tuning was a crucial step, as the results showed that the hyperparameter-tuned random forest model had higher prediction accuracy than the default one. Besides, the performance of default hyperparameters was often inconsistent.

In general, authors in each field have their conclusions about how hyperparameters affect the performance of random forests. Increasing the number of trees has improved classification accuracy [17, 19], while some studies suggest that increasing the maximum depth can lead to overfitting. The computation time of the model increases with the number of trees, and with a smaller number of trees, the computation time will be faster.

However, they did recommend using a large enough number of trees to ensure a stable error rate. Some studies indicate that the best performance of random forests is typically achieved with 100 trees [6, 17], while Boehmke and Greenwell [2, Ch. 11] recommend selecting ten times the number of trees as the number of features of the dataset for forest construction. Out-of-bag error curves increase with the number of trees and can be used to test algorithm convergence [17]. Research by P. Contreras et al. [6] indicated that numTree had the most influence on model performance, particularly in the value range of 0 to 100, which is consistent with the research by Probst et al. [16]. Therefore, numTree tuning is a straightforward method to achieve optimal model performance.

Based on a search on Google Scholar, we learned about related work on hyperparameter tuning for random forest algorithms. However, we have not found studies on this topic in the range of human daily activity recognition, especially in fall detection systems. Therefore, we conducted a survey and evaluated the impact of hyperparameters on the performance of the random forest algorithm using accelerometer data for fall detection systems.

3 Evaluation Methods

3.1 Evaluation Process

In this article, the feature extraction and random forest algorithms are built based on the Java language using the Weka library version 3.9.6. Figure 1 illustrates the experimental procedure for evaluating each hyperparameter and combination of hyperparameters in the random forest.

First, two raw data sets, MobiAct v2.0 and UP-Fall, were prepared for the experiment. The training dataset was created by segmenting the raw data into windows of 256 samples with a 50% overlap, applying preprocessing techniques to the segmented data, and extracting 44 features [11]. Next, we split the dataset into a training set (80%) and a test set (20%). We independently selected the test set to represent the overall dataset.

We conducted hyperparameter selection by surveying each hyperparameter individually to assess its influence on the performance of the random forest model. Afterward, we examined combinations of two, three, four, and five hyperparameters to determine the best and worst combinations.

Finally, we analyzed the results to evaluate and draw conclusions regarding the impact of hyperparameters and their combinations on the performance and training time of the random forest model. We utilized the accuracy and F1-score measures to estimate the performance of the random forest. Accuracy and F1-score are widely recognized as performance measures of efficient models by many studies.

Accuracy (Acc) was defined as the ratio of quantity correctly classified instances to the total quantity instances in the dataset. It is a simple and popular metric for evaluating the efficiency of classification algorithms. This measure has the following definition:

$$\text{Accuracy} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{1}$$

where TP, TN, FP, and FN are symbols used in the classification problem and evaluate the accuracy of a classification algorithm, specifically:

TP (True Positive) represents the amount of data that actually belongs to a class and the algorithm correctly predicts it to be that class.

TN (True Negative) represents the amount of data that actually does not belong to a class and the algorithm correctly predicts it to not belong to that class.

FP (False Positive) represents the amount of data that actually does not belong to a class, but the algorithm predicts it does.

FN (False Negative) represents the amount of data that actually belongs to a class, but the algorithm predicts it does not belong to that class.

F1-score is a measure of the efficiency of a classification algorithm. It combines the precision and recall of the data classes in a classification model. Formally, accuracy has the following definition:

$$\text{F1-core} = 2 * \frac{(\text{Precision} * \text{Sensitivity})}{(\text{Precision+Sensitivity})} \tag{2}$$

where Precision is the ratio between the TP and the (TP + FP), and Recall is the ratio between the TP and (TP + FN).

The F1-score ranges from 0 to 1, with a higher value indicating a better classification model. It is used in binary classification problems where the balance between precision and recall is crucial.

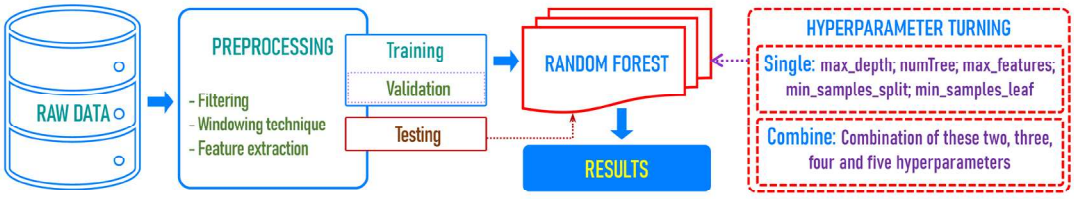

Figure 1: Hyperparameter evaluation process of random forest

3.2 Experimental Dataset

To evaluate the impact of hyperparameters, we conducted experiments using two commonly used datasets in human activity recognition and fall detection, namely MobiAct v2.0 [23] and UP-Fall detection [13]. These datasets are used widely in related research to action recognition and fall detection due to their large size and popularity. What sets these datasets apart is the diversity of the number of actions, volunteers, and collection methods. The MobiAct v2.0 dataset was collected using sensors on smartphones, whereas the UP-Fall dataset utilized wearable sensors for collection.

The MobiAct v2.0 dataset [23] was collected from the accelerometer, gyroscope, and orientation sensor of the Samsung Galaxy S3 smartphone with a sampling frequency of around 85 Hz for all activities. This dataset has 16 actions, of which 12 were daily activities (ADLs) and four fall behaviors, collected from 66 volunteers. The sampling of fall behaviors was done three times for action, with each sample lasting 10 seconds. Standing and walking activities were sampled only once for 5 minutes. Other daily activities have sampling times ranging from six seconds to 30 seconds, depending on the action. The sample rate of fall behaviors accounts for only about 8% of the total data collected from smartphone accelerometers. We only use the data collected from the accelerometer in this dataset for our research.

For the UP-Fall dataset [13], Martínez-Villaseñor and colleagues used five Mbientlab MetaSensor wearable sensors to collect raw data from the 3-axis accelerometer and gyroscope. They also used a NeuroSky MindWave headset to measure brainwave signals in the forehead, installed six infrared sensors as a grid 400mm above the lab floor to measure changes in actions, and positioned two Microsoft LifeCam Cinema cameras 1820 mm above the floor to collect images of the subject from the front and side. All types of sensors work simultaneously to collect data for each type of action. The standardized sampling frequency for all samples is 100 Hz. In this research, we only used data collected from the 3-axis accelerometer of the IMU device placed inside the right pocket of the volunteer. This part of the data includes 11 types of actions collected, of which the number of samples for five falling behaviors accounts for nearly 9% of the dataset.

The owners and authors of both datasets have permitted us to use them in community support and educational studies. These datasets have been reviewed and approved by experts in human research ethics.

3.3 Processing

The preprocessing steps applied to the raw dataset include data cleaning, noise filtering, normalization, and extraction of matching features. The data cleaning step removes missing or corrupted data, normalizes the data to keep it in a consistent range, and extracts feature that identify characteristics suitable for classifying human actions and behaviors.

The dataset used in this research includes 44 features extracted from MobiAct v2.0 and UP-Fall. We used the same preprocessing and feature extraction methods as in previous studies [11], and details of these methods have been presented in previous publications [11].

For experimentation, we segmented the data into sliding window sizes of 256 samples with an overlap rate of 50%, extracting 44 features [11]. However, the MobiAct v2.0 dataset was so large that using all the data for experimentation would have been time-consuming. Therefore, to shorten the experimental time, we used only 1/3 of the sample number of each action in the dataset to build the experiment data. The total sample quantity of the MobiAct v2.0 and UP-Fall datasets were used for experimental 15,776 and 2,025, respectively.

3.4 Random Forest Algorithm

Random Forest (RF) is a supervised learning algorithm [3] that uses multiple classifiers instead of a single one to achieve higher accuracy in predicting future cases. It is an extended version of a decision tree that uses two random steps to generate highly diverse sub-datasets, reducing variance error. Unlike traditional decision trees, each Classification and Regression Tree (CART) [4] in RF can only select a random subset of features, making the trees in the model more diverse. RF is particularly effective in handling datasets with specific issues [24] and can help resolve complex interactions between input features, enabling good over-model matching [22]. Additionally, RF can estimate the importance of each feature in the feature space [12]. Each decision tree in RF does not use all the training data or all the attributes of the data to build the tree. The information from the trees complements each other, leading to a low-bias and low-variance model with good prediction results and fast training time.

3.5 Hyperparameters in Random Forest

The random forest algorithm uses hyperparameters to control the learning process and time, which has a significant impact on its performance. Understanding the function of each hyperparameter is crucial to optimize the algorithm. These hyperparameters include the maximum depth of a tree, the number of trees in the forest, the maximum number of features for node splitting, the minimum number of samples in leaf nodes, the minimum number of samples for node separation, and the data percentage used for tree construction [3, 15]. Additionally, another set of hyperparameters focuses on dividing the nodes within each tree.

Failure to find the optimal hyperparameter value can reduce the performance of the RF algorithm. However, adjusting the hyperparameters for each dataset can improve classification performance or speed up the model's predictive ability [15]. This article investigates identifying the hyperparameter value range that yields the best RF performance and predictability. Based on relevant studies, we conclude that hyperparameters significantly affect the RF classification performance and time in various ways, such as:

(1) max_depth (M_D) is a hyperparameter that determines the

maximum depth at which a tree in a forest can grow. It is an essential hyperparameter that considerably impacts the model's accuracy [15, 21]. Increasing the tree's depth improves the model's accuracy by providing more information and data division. However, setting M_D too high will complicate the tree's structure and result in an overfit to the data. Therefore, selecting an appropriate M_D value is crucial in optimizing the model.

(2) num_tree (N_T) is the number of decision trees (DT) utilized in the forest, and it is correlative with the training dataset's size. The number of trees should be sufficient to stabilize the error rate. The higher the number of trees, the greater the classification accuracy. However, using too many trees increases the computation time. There is no rule for determining the optimal number of trees, but some sources suggest that [2, Ch. 11] the number of trees should be approximately ten times the number of dataset features. The initialization trees' number can be increased or decreased depending on hyperparameters such as max_features and min_samples_leaf.

(3) The max_features (M_F) parameter represents the maximum number of features used when the algorithm searches for node separation. It is a crucial hyperparameter that influences the model's classification performance [15, 21]. By setting M_F to a low value, the correlation between trees decreases, which enhances prediction stability. However, when M_F is too low, the selected features may not be optimal, thus affecting the performance of the forest. Conversely, when M_F is set too high, the trees become similar, which results in overfitting.

The optimal value of M_F depends on the dataset used and should be adjusted through cross-validation. For regression problems, the default value of M_F is p/3, where p is the number of features in the training dataset. For classification problems, M_F can take one of four values: "none", "sqrt", "log2", and "auto", and the default value is usually "sqrt". Let the number of features in the training dataset be p (n_features), if M_F = "none", then m_{try} = p; if M_F = "sqrt", then m_{try} = sqrt(p); if M_F = "log2", then m_{try} = log2(p)+1; and if M_F = "auto", then m_{try} = p/3. The choice of M_F value should balance the stability and accuracy of each tree in the forest.

(4) The hyperparameter min_samples_leaf (m_L) specifies the minimum number of samples required for a node to become a leaf after splitting. Changing its value can affect the depth of the tree, so allowing us to control it. A small value of m_L may lead to a deeper tree, increasing the possibility of overfitting. However, if m_L is set too high, the model may fail to learn from the data.

(5) The hyperparameter min_samples_split (m_S) represents the minimum number of samples necessary to split a node into child nodes. When the number of samples in a node exceeds m_S and is not pure, the splitting process continues until purity is attained or the sample count in the node is less than or equal to m_S. By increasing m_S, the total number of splits decreases, which reduces the number of parameters and can potentially prevent overfitting. However, increasing m_S too much can result in decreased model performance.

4 Results and Discussion

This section presents the results of experimenting with each hyperparameter to evaluate its impact on the performance of the random forest algorithm. Our program was written in Java using the library Weka 3.9.6 and running on a Dell Precision 5510 laptop which is pre-installed with Eclipse software and runs on the Windows 11 64-bit operating system. The basic configuration of this laptop includes an Intel Core i7-6820HQ CPU, 24GB RAM, and NVIDIA M1000M GPU.

To ensure the objectivity and analogy of the data, we clear the cache and restart the computer after each hyperparameter result is collected.

Five hyperparameters that were to have the most influence on classifier performance were selected to testing [15, 2, Ch. 11]. These hyperparameters included: max_depth (M_D), numTree (N_T), max_features (M_F), min_samples_split (m_S), and min_samples_leaf, (m_L). For each hyperparameter, 18 different values were selected for testing within specific ranges, as follows: $M_D \in [0, 100]$, $N_T \in [1, 100]$, $M_F \in [1, 150]$, $m_L \in [0, 150]$, $m_S \in [0.001, 150]$. We then tested combinations of two, three, four, and five hyperparameters to determine the best and worst combinations. The survey results for each hyperparameter are as follows:

4.1 Effect of Max_Depth (M_D)

This hyperparameter reflects the maximum depth a tree in the forest can grow. Conceivably, the nodes are expanding until all leaves are pure or the leaves have a sample number less than min_samples_split. The default value of this parameter in Weka is 0 [18], and in Scikit-learn, it is "none" [20]. Figure 2 shows the experimental results.

The graph in Figure 2 shows that when the tree depth is too small, the model's performance is low because the input data does not provide enough information to train the model. Although the Accuracy measure received classification results, the F1-score and MCC measures did not produce results for tree depths $M_D < 7$ for the MobiAct v2.0 dataset and $M_D < 3$ for the UP-Fall dataset, as some actions occur quickly with a small number of data samples, resulting in insufficient information for evaluation. However, as the tree depth increases, the model's performance rapidly improves, along with an increase in the time taken to build the model. Both datasets had similar results, and when the tree depth increases to a certain threshold ($M_D > 11$ for the MobiAct v2.0 dataset and $M_D > 7$ for the UP-Fall dataset), both the performance and training time of the model reached near saturation.

4.2 Effect of numTree (N_T)

This hyperparameter represents the number of decision trees (DT) used in the forest. In toolkits such as Weka and Scikit-learn, the default number of trees is 100. In this experimental part, we evaluate the effect of the number of trees in the forest

on the performance and computation time of the model. Figure 3 is the synthetic results from the experimental evaluation of the influence of numTree.

The results presented in Figure 3 demonstrate that the model's performance improves with an increase in the number of trees. To a certain threshold, adding more trees does not further enhance the model's performance. The performance of RF is better when the number of selected trees exceeds the default value. However, choosing too many trees increases the model's complexity and computational time. Therefore, increasing the number of trees to improve the model's performance is not an optimal solution.

4.3 Effect of Max_Features (M_F):

In this experiment, we evaluated the influence of the max_features hyperparameter on the performance of the random forest model. This hyperparameter controls the maximum number of features the algorithm uses when searching for node splits. In Weka, the default value for this hyperparameter is 0, while in Scikit-learn, it is 1.0. The test results for two datasets, MobiAct v2.0, and UP-Fall, are displayed in Figure 4.

The synthesized results in Figure 5 indicate that the RF achieves nearly optimal performance with the default value. Gradually increasing the value of m_L causes the model's performance and the computation time to decrease, although the reduction is not significant.

4.4 Effect of Min_Samples_Leaf (m_L)

This hyperparameter controls the minimum number of samples required for a node to be considered a leaf after a split. It is set to 1 by default in Weka and Scikit-learn. Figure 5 shows the experimental investigation of the influence of this hyperparameter.

The synthesized results in Figure 5 indicate that the RF achieves nearly optimal performance with the default value. Gradually increasing the value of m_L causes the model's performance and the computation time to decrease, although the reduction is not significant.

4.5 Effect of Min_Samples_Split (m_S)

This hyperparameter controls the minimum number of samples to split into child nodes. If the number of samples in a node $>$ m_S, then splitting continues until £ m_S. The default values for this hyperparameter in Weka and Scikit-learn are 1e-3 and 2, respectively. Figure 6 displays the results of an investigation into the influence of this hyperparameter.

Similar, to the m_L hyperparameter, the performance of RF gradually decreases as the minimum number of samples to split into child nodes increases. However, adjusting this hyperparameter does not significantly affect the computation time.

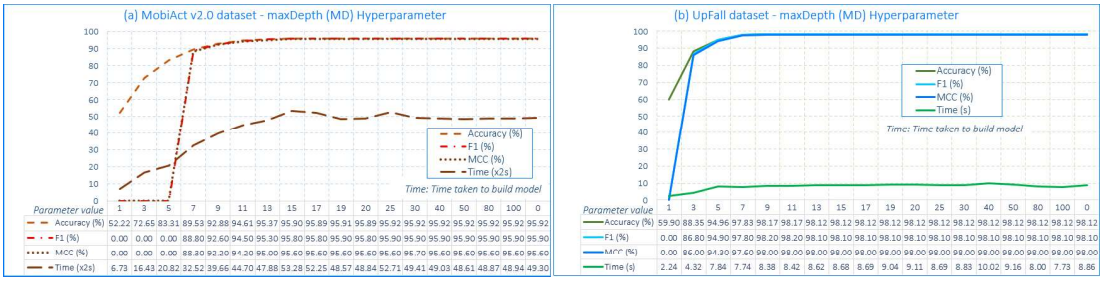

Figure 2: As the tree depth (M_D) increased, the RF classification efficiency and model building time also increased on both (a) MobiAct v2.0 and (b) UP-Fall datasets

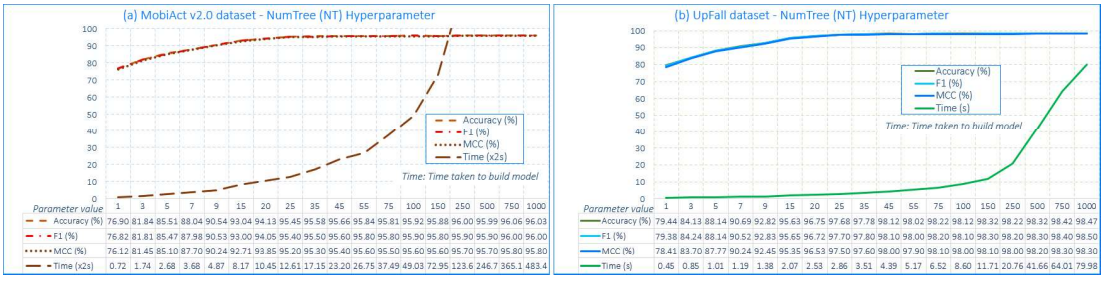

Figure 3: The accuracy in classification increases as the number of trees increases and the computation time also increases very quickly

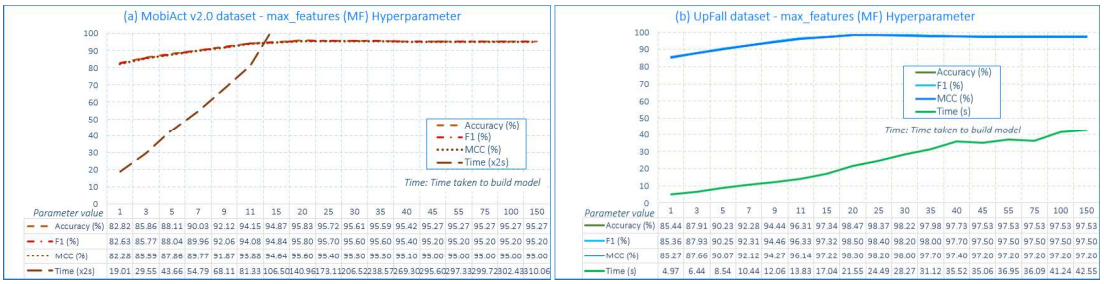

Figure 4: The performance of the model does not change much when adjusting the value of the max_features hyperparameter (MF), but the computation time increases very quickly with increasing MF

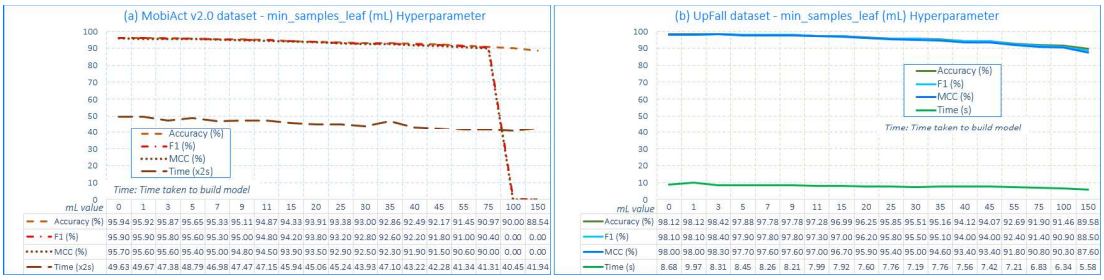

Figure 5: The model's performance decreases as the minimum number of samples in a leaf node increases

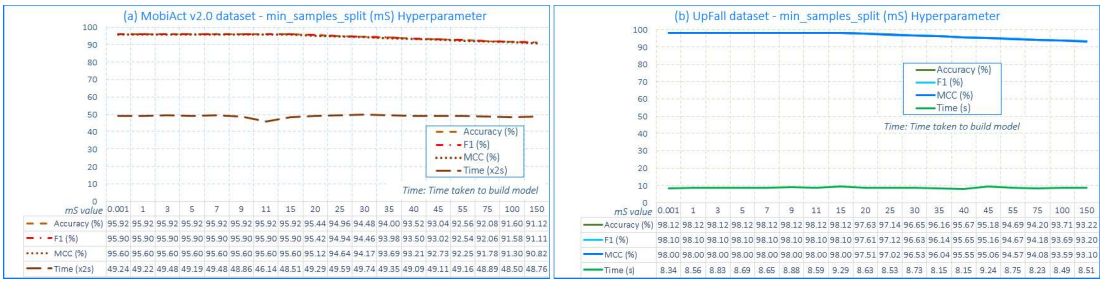

Figure 6: Increasing the minimum number of samples required to split into child nodes leads to a decrease in RF performance

4.6 Combined Tuning of Multiple Hyperparameters

In this section, we move beyond evaluating each hyperparameter independently, as described in previous sections, and investigate the simultaneous adjustment of two to five hyperparameters to assess their combined influence on the model's performance and building time. We manually select hyperparameter values based on survey results from Sections 4.1 to 4.5. The default values of hyperparameters are shown in Table 1 as cells with a dark background (blue) and light text (yellow). Specifically, we choose max_depth (M_D) = 19, NumTree (N_T) = 150, max_features (M_F) = 20, min_samples_leaf, (m_L) = 2, and min_samples_split (m_S) = 1 as they have the most positive effect on RF performance

Table 1 presents the experimental results based on the MobiAct v2.0 and UP-Fall datasets. These results indicate that combining the simultaneous adjustment of two hyperparameters, numTree, and max_features, results in the best RF performance. Additionally, combining the three hyperparameters $M_D, N_T,$ and M_F also yields good results, similar to the combination of numTree and max_features. In contrast, combining the hyperparameters min_samples_leaf and min_samples_split yields unexpected results, with the model having the lowest performance of all combinations. Regarding time, model building is faster when hyperparameters are set to their default values than when adjusting the combination of hyperparameters.

The simultaneous adjustment of numTree and max_features is crucial in optimizing the model and often produces the most positive results. Although the combination of max_depth and

Table 1: Combined tuning of multiple hyperparameters results

No.	Code	Hyperparameters value					Metrics				Note
		MD	NT	MF	mL	mS	Accuracy	F1-score	Accuracy	F1-score	
		1	1	1	1	1	(MobiAct)	(MobiAct)	(UP-Fall)	(UP-Fall)	
1	00000	0	100	0	1	0.001	95.918	95.867	98.124	98.122	All defaults
2	11000	19	150	0	1	0.001	95.950	95.900	98.321	98.314	MD and NT are tuning
3	10100	19	100	20	1	0.001	95.931	95.893	98.469	98.465	MD and MF are tuning
4	10010	19	100	0	2	0.001	95.702	95.646	**97.975**	97.973	
5	10001	19	100	0	1	1	95.912	95.859	98.124	98.122	
6	01100	0	150	20	1	0.001	**95.956**	**95.917**	**98.568**	**98.565**	Best performance
7	01010	0	150	0	2	0.001	95.778	95.723	**97.975**	**97.971**	
8	01001	0	150	0	1	1	95.880	95.825	98.321	98.314	
9	00110	0	100	20	2	0.001	95.899	95.861	98.370	98.365	
10	00101	0	100	20	1	1	95.829	95.786	98.469	98.465	
11	00011	0	100	0	2	1	**95.658**	**95.597**	**97.975**	**97.971**	Worst performance
12	11100	19	150	20	1	0.001	95.924	95.885	**98.568**	**98.565**	
13	11010	19	150	0	2	0.001	95.766	95.712	97.976	97.973	
14	11001	19	150	0	1	1	95.950	95.900	98.321	98.314	
15	10110	19	100	20	2	0.001	95.855	95.814	98.370	98.365	
16	10101	19	100	20	1	1	95.931	95.893	98.469	98.465	
17	10011	19	100	0	2	1	95.702	95.646	**97.975**	97.973	
18	01110	0	150	20	2	0.001	95.912	95.873	98.420	98.414	
19	01101	0	150	20	1	1	95.924	95.885	**98.568**	**98.565**	
20	01011	0	150	0	2	1	95.778	95.723	**97.975**	**97.971**	
21	00111	0	100	20	2	1	95.899	95.861	98.370	98.365	
22	11110	19	150	20	2	0.001	95.816	95.775	98.420	98.414	
23	11101	19	150	20	1	1	**95.956**	**95.917**	**98.568**	**98.565**	Best performance
24	11011	19	150	0	2	1	95.766	95.712	**97.975**	**97.971**	
25	10111	19	100	20	2	1	95.855	95.814	98.370	98.365	
26	01111	0	150	20	2	1	95.912	95.873	98.420	98.414	NT, MF, mL and mS tuning
27	11111	19	150	20	2	1	95.816	95.775	98.420	98.414	Tuning of 5 hyperparameters

numTree is slightly less effective than the numTree and max_features combination, it significantly reduces model building time. It is worth noting that adjusting min_samples_leaf and min_samples_split together can reduce the model's performance. With a powerful computer system, the number of trees should be beyond 200 to keep the modelstable, which is also supported by theoretical evidence from Probst and Boulesteix [17].

5 Discussion

This research aims to identify the range of hyperparameter values that can performance enhance random forests in fall detection systems. Our findings suggest that each hyperparameter has a different impact on the performance and training time of RF, and inappropriate selection of hyperparameter values may lead to decreased model performance compared to default values.

Out of the five hyperparameters analyzed, max_depth (M_D) exerts the most substantial influence on RF classification performance. When the value of max_depth is set too small, the model performs poorly. However, optimal performance is maintained when $M_D > 15$ (for the MobiAct v2.0 dataset) and $M_D > 7$ (for the UP-Fall dataset), as shown in Figure 2.

Optimizing the tree depth can enhance the model's performance, but this comes at the expense of longer training time. Furthermore, the research underscores the significance of having adequate data samples for model training since insufficient samples may result in inadequate information for evaluation. Overall, these results indicate that a balance between tree depth and data samples is critical for attaining optimal model performance.

The default value of numTree is 100 trees that can also produce a model with good performance. However, increasing N_T beyond 100 trees can improve performance but at the cost of longer training time, which can be overcome by using powerful computers. When N_T exceeds 35 trees, RF performance becomes good and stabilizes on both datasets, as shown in Figure 3. This finding is consistent with the assessment of Probst et al. [1, 16]. Therefore, the optimal value of numTree should be chosen within the range of [35, 150] to balance performance and training time optimization.

The max_features hyperparameter plays a crucial role in the RF algorithm, especially for datasets with a large number of samples, as it significantly affects the training time. Increasing the value of max_features lead to a rapid increase in training time. In this research, with datasets consisting of 44 features, the default value of "none" outperforms the other default values of "sqrt", "log2", and "auto". As the maximum number of features is increased from 1 to 20, the performance of the random forest increases. However, once max_features exceed 20, the performance of the random forest is stable and better (Figure 4).

The simultaneous adjustment of numTree and max_features is crucial in optimizing the model and often produces the most positive results. Although the combination of max_depth and numTree is slightly less effective than the numTree and max_features combination, it significantly reduces model building time. It is worth noting that adjusting min_samples_leaf and min_samples_split together can reduce the model's performance. With a powerful computer system, the number of trees should be beyond 200 to keep the model stable, which is also supported by theoretical evidence from Probst and Boulesteix [17].

The experimental results demonstrate a correlation between the hyperparameters min_samples_leaf and min_samples_split, as both follow the same distribution and define the minimum number of samples for splitting. Figures 5 and 6 illustrate that to achieve high performance with RF, it is advisable to select small values for m_L and m_S, preferably below five.

Generally, tuning hyperparameters can have either positive or negative effects on the performance of the RF algorithm. Selecting optimal values for hyperparameters can improve RF performance and vice versa. Setting a large number of trees can enhance performance and maintain accuracy during training, but it comes at the expense of increased computational costs. Hyperparameters such as max_features, max_depth, and node size act as controls for the randomness of the RF. Of these, max_depth and numTree are the most influential hyperparameters, as per our theoretical analysis and experimental findings.

6 Conclusion

This study evaluates the impact and determines the optimal range of hyperparameters on the performance of the random forest classification algorithm in the fall detection system. Based on the test results on human activity simulation datasets collected by accelerometers, we drew the following conclusions:

(i) The number of trees (N_T) is the most crucial hyperparameter of the RF algorithm. Models with N_T < 35 yield poor classification results, while increasing N_T to 100 or higher significantly improves model performance. Therefore, to enhance the RF algorithm's performance, we should choose several trees > 100 for this hyperparameter, as supported by related literature and our experiments.

(ii) Simultaneously tuning numTree and max_features, or max_Depth and numTree, or max_Depth and numTree and max_features can improve RF performance.

(iii) The hyperparameters min_samples_leaf and min_samples_split should be selected as defaults for achieving high performance in RF. In particular, when these two hyperparameters are adjusted simultaneously, the performance of the RF is at its worst.

While our research has shown promising results in developing a random forest algorithm for fall detection, there are still certain limitations. These limitations include using only two accelerometer datasets, limiting the number of hyperparameters considered, and focusing only on the Weka toolkit without experimentation with other toolkits, such as Scikit-learn. Additionally, the hyperparameter tuning combination was limited to a manual method based on the survey results of each hyperparameter individually.

Future studies can address these limitations by expanding the range of datasets, extending the scope of surveys to multiple parameters, and utilizing various popular toolsets to evaluate the impact of hyperparameters. Also, we want to use genetic algorithms to find optimal combinations of hyperparameters for random forests to improve the performance of fall detection systems. With these lines of research in mind, we aim to develop more advanced and effective fall detection systems that enhance the safety and well-being of populations requiring special care.

Acknowledgment

The authors sincerely thank the Hellenic Mediterranean University- Department of Electrical and Computer Engineering - Biomedical Informatics and eHealth Laboratory (BMI) for sharing the MobiAct dataset. Great thanks to Martínez-Villaseñor et al. for sharing the UP-Fall dataset.

References

[1] S. Bernard, L. Heutte, and S. Adam, "Influence of Hyperparameters on Random Forest Accuracy," *Multiple Classifier Systems: 8th International Workshop, MCS 2009, Reykjavik, Iceland, June 10-12, 2009. Proceedings 8*, Springer, pp. 171–180, 2009.

[2] B. Boehmke and B. Greenwell, *Hands-on Machine Learning with R*. Chapman and Hall/CRC, 2019.

[3] L. Breiman, "Random Forests," *Machine Learning*, 45(1):5-32, 2001.

[4] L. Breiman, "Bagging Predictors," *Machine Learning*, 24(2):123-140, 1996.

[5] C. Chatzaki, M. Pediaditis, G. Vavoulas, and M. Tsiknakis, "Human Daily Activity and Fall Recognition using a Smartphone's Acceleration Sensor," International Conference on Information and Communication Technologies for Ageing Well and e-Health, Springer, pp. 100-118, 2016.

[6] P. Contreras, J. Orellana-Alvear, P. Muñoz, J. Bendix, and R. Célleri, "Influence of Random Forest Hyperparameterization on Short-Term Runoff Forecasting in an Andean Mountain Catchment," *Atmosphere*, 12(2):238, 2021.

[7] D.-M. Ge, L.-C. Zhao, and M. Esmaeili-Falak, "Estimation of Rapid Chloride Permeability of SCC using Hyperparameters Optimized Random Forest Models," *Journal of Sustainable Cement-Based Materials*, 0(0):1-19, July 2022. doi: 10.1080/21650373.2022.2093291.

[8] C. Joo, H. Park, J. Lim, H. Cho, and J. Kim, "Development of Physical Property Prediction Models

for Polypropylene Composites with Optimizing Random Forest Hyperparameters," *International Journal of Intelligent Systems*, 37(6):3625–3653, 2022.

[9] K. M. Kelkar and J. W. Bakal, "Hyper Parameter Tuning of Random Forest Algorithm for Affective Learning System," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), IEEE, pp. 1192-1195, 2020.

[10] K. Lai, S. N. Yanushkevich, V. Shmerko, and M. Hou, "Capturing Causality and Bias in Human Action Recognition," *Pattern Recognition Letters*, 147:164-171, 2021.

[11] H.-L. Le, D.-N. Nguyen, T.-H. Nguyen, and H.-N. Nguyen, "A Novel Feature Set Extraction Based on Accelerometer Sensor Data for Improving the Fall Detection System," *Electronics*, 11(7):1030, 2022.

[12] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts, "Understanding Variable Importances in Forests of Randomized Trees," *Advances in Neural Information Processing Systems*, Vol. 26, 2013.

[13] L. Martínez-Villaseñor, H. Ponce, J. Brieva, E. Moya-Albor, J. Núñez-Martínez, and C. Peñafort-Asturiano, "UP-Fall Detection Dataset: A Multimodal Approach," *Sensors*, 19(9):1988, 2019.

[14] N. Mohapatra, K. Shreya, and A. Chinmay, "Optimization of the Random Forest Algorithm," *Advances in Data Science and Management: Proceedings of ICDSM 2019*, Springer, pp. 201-208, 2020.

[15] P. Probst, M. N. Wright, and A.-L. Boulesteix, "Hyperparameters and Tuning Strategies for Random Forest," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 9(3):e1301, 2019.

[16] P. Probst, "tuneRanger: Tune Random Forest of the Ranger Package," *R Package Version*, Vol. 2, 2018.

[17] P. Probst and A.-L. Boulesteix, "To Tune or Not to Tune the Number of Trees in Random Forest," *The Journal of Machine Learning Research*, 18(1):6673-6690, 2017.

18] "RandomForest." https://weka.sourceforge.io/doc.dev/ (accessed Oct. 26, 2022).

[19] E. Scornet, "Tuning Parameters in Random Forests," *ESAIM: Proceedings and Surveys*, 60:144-162, 2017.

[20] "sklearn.ensemble.RandomForestRegressor — scikit-learn 1.1.2 Documentation." https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html (accessed Oct. 26, 2022).

[21] J. Thorn, "Random Forest: Hyperparameters and How to Fine-Tune Them," *Medium*, Sep. 26, 2021. https://towardsdatascience.com/random-forest-hyperparameters-and-how-to-fine-tune-them-17aee785ee0d (accessed Oct. 26, 2022).

[22] H. Tyralis, G. Papacharalampous, and A. Langousis, "A Brief Review of Random Forests for Water Scientists and Practitioners and Their Recent History in Water Resources," *Water*, 11(5):910, 2019.

[23] G. Vavoulas, C. Chatzaki, T. Malliotakis, M. Pediaditis, and M. Tsiknakis, "The Mobiact Dataset: Recognition of Activities of Daily Living Using Smartphones," International Conference on Information and Communication Technologies for Ageing Well and e-Health, SCITEPRESS, pp. 143-151, 2016.

[24] L. Wang, X. Zhou, X. Zhu, Z. Dong, and W. Guo, "Estimation of Biomass in Wheat Using Random Forest Regression Algorithm and Remote Sensing Data," *The Crop Journal*, 4(3):212-219, June 2016. doi: 10.1016/j.cj.2016.01.008.

[25] M. N. Wright, T. Dankowski, and A. Ziegler, "Unbiased Split Variable Selection for Random Survival Forests using Maximally Selected Rank Statistics," *Statistics in Medicine*, 36(8):1272–1284, 2017.

[26] L. Yang and A. Shami, "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice," *Neurocomputing*, 415:295-316, Nov. 2020. doi: 10.1016/j.neucom.2020.07.061.

[27] N. Zhu, C. Zhu, L. Zhou, Y. Zhu, and X. Zhang, "Optimization of the Random Forest Hyperparameters for Power Industrial Control Systems Intrusion Detection Using an Improved Grid Search Algorithm," *Applied Sciences*, 12(20):10456, 2022.

Hong-Lam Le received his B.E degree in Electrical and Electronics Engineering from Ho Chi Minh City University of Technology and Education (HCMUTE) in 2006, and M.E degree in Telecommunications Engineering from Hanoi University of Science and Technology (HUST) in 2015. He is working as a PhD student in the Information Technology Institute, Vietnam National University in Hanoi. Currently, he is a lecturer in Faculty of Electronic Engineering, Vinh University of Technology Education, Vinh city, Vietnam. His research interests include machine learning, data analysis, IoT, and sensor data processing in machine learning.

Thanh-Tuoi Le received the B.E degree in Information Technology from Vinh University (TDV) in 2003 and M.E degree in Information Technology with a specialization in Information System at the Military Technical Academy (MTA) in 2010. He has been working with the Department of Information Technology, Vinh University of Technology Education, Vinh City, Vietnam. Since 2022, he has been pursuing a Ph.D. degree in the Department of Information Technology, Hanoi National University of Education. His current research interests are in machine learning methods and their applications in molecular biology data mining.

Thi-Thu-Hien Vu received the B.E degree in Information Technology from Vinh University of Education (TDV) in 2000 and M.E the degree in Computer Science at Military Technical Academy (MTA) in 2010. Currently, she is the Vice Dean of Information Technology Department at Vinh University of Technology Education, with many years of experience in educating and researching topics such as cloud computing, embedded system, and applications.

Doan-Hieu Tran received his BSc in Information Technology from the Ho Chi Minh City University of Transport in 2010 and a MSc in Information Systems from the Posts and Telecommunications Institute of Technology (PTIT) in 2014. At present, he is a Deputy Head of the Information Technology Management Department at the Ho Chi Minh City University of Banking, Vietnam. He has over 10 years of teaching and research experience in the fields of financial risk analysis, applied computer science, computer networks and communications, and Python programming for data analysis.

Dinh Van Chau received his BSc from Hanoi University of Science and Technology (HUST), Vietnam, in 1998; MSc from De La Salle University, The Philippines, in 2005 and PhD at Tokyo Institute of Technology, Japan, in 2009. He is currently the Acting Rector of Electric Power University. He has many years of teaching and research experience in the fields of Engineering Physics, Nanotechnology, Energy Saving and Sustainable Development. He was principle investigator and participated in many domestic and foreign projects.

Ngo Thi-Thu-Trang received her B.E degree of Telecommunications and Electronics Engineering from Vietnam National University, Hanoi (VNUH) in 2002, and M.E degree of Computer and Communication Engineering from Chungbuk National University, Korea in 2005, and PhD in Communication Engineering from Posts and Telecommunications Institute of Technology (PTIT) in 2021. Now, she is a lecturer in Telecommunications Faculty 1 of PTIT. Her research interests include digital signal processing, optical communication, and broadband networks.