JOURNAL OF NORTHEASTERN UNIVERSITY

Volume 25 Issue 04, 2022 ISSN: 1005-3026 https://dbdxxb.cn/ Original Research Paper

HYBRID RANDOM-GRID OPTIMIZATION TECHNIQUES FOR A MALARIA PREDICTION USING MACHINE LEARNING ALGORITHMS

Yusuf Aliyu Adamu and Jaspreet Singh

School of Engineering GD Goenka University Sohna 12203 Haryana, India

aliyuadamuyusuf@gmail.com

ABSTRACT

Algorithms for machine learning have been commonly used in many applications and fields. Diseases like malaria can be predicted using machine learning algorithms to create a suitable and precise model that predicts when, how, and where the outbreak will occur. The model performance can be improved using various strategies, such as ensemble methods or fine-tuning the hyper-parameters. Choosing a proper hyper-parameter configuration impacts the model accuracy, so it must fit well to improve the performance even though different optimization techniques exist, each with its benefits and disadvantages when used to solve various tasks. When working with hyper-parameter optimization strategies, a greater understanding of how machine learning models work is usually necessary. In this study, a hybridized form of Random-Grid optimization is proposed and applied to the three standard machine learning algorithms; K-nearest Neighbors, Random Forest, Support Vector Machine and their ensemble. It works by doing a randomized search to obtain the best hyper-parameters and then using it in a grid search to minimize the time it will take to search for the optimal combination of hyper-parameters. The grid and the random search work together to obtain the optimal hyper-parameters to improve the model accuracy by specifying the number of iterations to be performed when looking for the optimal model. The proposed technique is compared with well-known techniques such as Bayesian optimization, the Grid search, Random Search, genetic algorithm, and the particle swamp and evaluated using the malaria dataset obtained from the World Health Organization. The proposed technique improved the prediction accuracy for each base learner through an experimental study, and the ensemble method gives better results for HRGO-ensemble, HRGO-RFC, HRGO-SVM, and HRGO-KNN with 97%, 96%, 95 and 93%, respectively.

Keywords: Malaria, Machine Learning Algorithms, Hyper-parameter, Genetic Algorithm, Particle Swarm Optimization, Bayesian Optimization.

1.0 INTRODUCTION

A parasite that typically infects mosquitoes that feed on humans causes malaria, a dangerous and occasionally fatal infectious disease. According to the world health organization, there were more than 200 million reported malaria cases globally [1]. The year 2020 saw a surge in malaria infections in Africa due to the global coronavirus pandemic that claimed many lives. The primary factor contributing to the high fatality rate is the lack of immediate and proper assessment of malaria treatment [2]. Numerous investigations indicated that non-climatic and atmospheric elements play a substantial role in predicting the frequency of malaria outbreaks

Submitted: 16/11/2022 **Accepted**: 05/12/2022

in society. Technology improvements enable early prediction and preventive mechanisms, giving government and non-government organizations more time to prepare and stop the spread of diseases that could otherwise claim many lives.

In several application domains, such as user behaviours, computer vision, health care, recommendation systems, and natural language processing, machine learning has recently emerged as one of the most efficient techniques [3]. It has many benefits like robustness, relatively cheap computation, generality, and better performance. Different algorithms are appropriate for solving various types of issues [4]. Owing to its adaptability and efficiency in resolving data analytics-related issues [5]. However, building an effective model requires a series of processes because of its complexity and time-consuming nature in identifying an appropriate algorithm and tuning its hyper-parameters [6]. Researchers employed numerous methods as predictive models to improve model accuracy by reducing variance and limiting model bias. As a result, algorithms perform better than others at producing correct results based on the available problems to be solved [7]. Researchers also developed enhanced ensemble techniques to increase the efficiency performances of the classifiers [8]. Researchers are constantly working to boost the accuracy of their methods and procedures. The right combination of hyper-parameter tweaking improves the model's accuracy and efficiency.

There are two parameters: the model parameter, which is set up and modified as the actual training process progresses, and the hyper-parameter, which works before training a model. Hyper-parameters specify the model architecture and cannot be predicted effectively from data learning [9]. Hyper-parameters are variables used to set up a model or specify how to minimize loss function [10]. Constructing a model architecture with the ideal hyper-parameter tuning is the primary component of creating an efficient learning model, especially for deep neural networks and tree-based models [11]. However, the tuning approach varies between algorithms because it may be discrete, categorical, or continuous [12]. The hyperparameter can be tuned manually [13]. It is ineffective for some issues due to the complicated nature of models, many hyper-parameters, non-linear hyper-parameter interactions, and time-consuming model evaluations. These circumstances lead to hyper-parameter optimization (HPO) [14]. Automating the hyper-parameter tuning process and enabling its efficient application in machine learning models are the primary goal of hyper-parameter optimization [6].

It is crucial to pick the ideal optimization method to find the best hyper-parameters. Conventional methods are not appropriate for all optimization situations since many hyper-parameter optimization problems are not convex and could result in the local optimum rather than the global one [15]. The most popular conventional optimization technique is the gradient descent-based approach, which continuously adjusts hyper-parameters by determining their slope [16]. Decision theoretic approaches, metaheuristic algorithms, multi-fidelity techniques, and Bayesian methods are better for optimization problems because they can more accurately identify conditional, categorical, and discrete hyper-parameters [10]. Using the optimization technique to alter the hyper-parameters of the models improved their computational complexity and performance. "Hyper-parameter optimization" aims to efficiently automate the hyper-parameter tuning process using machine learning models to solve real-world issues [5]. After

the hyper-parameter optimization process, the model architecture would be optimal as expected. Optimization methods are necessary to increase the performance of machine learning models and decrease the time spent tuning hyper-parameters, especially for models with more hyper-parameters. It also helps find the optimal model for a given task when the same threshold of hyper-parameters is used [10].

The grid search (GS) method creates hyper-parameters search space, identifying the possible hyper-parameter combinations inside a fixed region of hyper-parameters value [17]. Random Search (RS) chooses combinations of hyper-parameters randomly from the search space while utilizing constrained resources and execution time [18]. Hyper-parameters are treated independently in both grid and random search. Training a learning model requires an appropriate amount of time and space. Bandit-based algorithms like the hyper-band technique, a more excellent version of random search, frequently use multi-fidelity optimization techniques to handle jobs with limited resources [19]. Although using proper hyper-parameter optimization techniques, machine learning models perform much better. Though each optimization technique has pros and cons, the choice is made based on the task.

As the number of parameters increases, so does the number of evaluations, making GS an inefficient HPO method for high dimensionality hyper-parameter configuration unless configuration space for hyper-parameter is limited [20]. Because each evaluation is independent, RS is simple to parallelize, efficient, capable of exploring a more extensive search space, and resource-allocated. Reducing the possibility of wasting time on a limited, irrelevant search space. Its limitation is that each iteration's evaluation is independent of previous evaluations [18]. GS and RS issues can be solved by creating new optimization techniques using records from previous evaluations to determine the subsequent evaluation and obtain optimal hyper-parameters.

Proper optimization algorithm selection necessitates understanding the various ML models and problems. This paper makes the following contributions:

1. It examines standard ML algorithms and their key hyper-parameters.
2. It examines standard HPO techniques to assist in their application to various ML models through appropriate algorithm selection in practical problems.
3. It combines the strength of the Grid and Random search to propose a newly hybridized form of Random-grid optimization (HRGO), which will help to know the appropriate hyper-parameters combination to design a better model.
4. It ensemble the classifiers used and compares it with an optimized ensemble version of the proposed algorithm

The experiment was carried out using a malaria dataset from the world health organization. It works by doing a randomized search to obtain the best hyper-parameters and then using it in a grid search to minimize the time it will take to find the optimum combination. The grid and a random search work together to obtain the best hyper-parameters to improve the model performance. The proposed HRGO technique is used to observe the effect of hyperparameters on the individual model and the ensemble. Tuning hyperparameters of the model inside the

optimal ensemble will result in better prediction accuracy. The performances are compared with well-known optimization techniques such as Bayesian optimization, Grid search, Random Search, genetic algorithm, and particle swamp optimization. Random Forest, SVM, and KNN are the machine learning algorithms used and the stacking method of an ensemble for the experiment. The paper begins with an introduction and briefly explains optimization techniques in the second section of related work and the algorithms employed. It then moves on to section three, where the techniques and approaches used for the performance metrics are analyzed, and the fourth section describes the finding and discussions before concluding.

2.0 RELATED WORK

Statistical modelling and Machine learning algorithms are commonly used for the prediction process. All of these methods, along with computational engineering models, are vital for forecasting and decision-making [21]. Climatic factors play a significant role in malaria transmission [22]. Rainfall, be it plenty or inadequate, has been discovered to have an impact beyond just the risk of malaria [23]. Temperature and rainfall are the key factors attributed to malaria spread [24].

Generally, formulating machine learning models requires tuning a hyper-parameter using optimization techniques to fit a model to specific problems. An algorithm for machine learning includes supervised semi-supervised, unsupervised, or reinforcement learning. Classification or regression problems are examples of supervised learning, which refers to the methods that relate input features to a target set of labelled data [25]. K-nearest neighbour, linear models, deep learning, decision tree-based, support vector machines, and Naive Bayes algorithms are all supervised learning [26]. Unsupervised learning algorithms, which include dimension reduction and clustering, are used to recognize patterns in unlabeled data [27]. Semi-supervised learning techniques are generic models, self-training, and other techniques [28]. Different learning algorithms are combined to improve the model's performance, such as bagging, voting, stacking, AdaBoost, and XGBoost [29]. When employing different configurations of hyper-parameters, it is possible to generate an optimal predictive model function that varies based on the model's architecture. Different loss functions, including information gain, cross-entropy, hinge loss, contrast loss, and the square of Euclidean distance, are used in supervised learning [30]. In order to create various models, machine learning algorithms employ various hyper-parameter setups.

K-Nearest Neighbors; The data point is classed by calculating the separations between the data points and determining which class most of its k-Neighbors be. The model will underfit if the value of k is small and overfit if it is too large, requiring more computation time. Therefore, k is the paramount hyper-parameter to consider [31], depending on the problem to deal with during prediction. While in a support vector machine, the concept of data point mapping is linearly separable from a low dimension into a high-dimensional space [32]. The hyper-plane is the root cause of the categorization boundary that separates data points [33]. A kernel

function measures the similarity between data points, which would be the hyper-parameters to be tuned [34]

Furthermore, the Decision Tree classifier is a tree-based structure for making decisions [35]. It has three components: the root node, which represents the overall data; the sub-node; and the leaf node. Techniques are employed to split the data while considering several features [36]. The maximum depth of a tree is one of the primary hyper-parameters of the decision tree algorithm. If the tree has more subtrees as it goes down, an accurate decision can be obtained [37]. Other hyper-parameters would be tuned to have a more effective split by setting measuring functions that can either be Gini impurity or information gain [38]. The model performance can increase by combining several decision trees, like an extra tree, random forests, and extreme gradient boosting models. In Random Forest, decision trees are constructed on several randomly selected subsets of data and choose the class with the highest majority vote to be the overall classification outcome [39]. XGBoost is an ensemble model that uses boosting and gradient descent methods to improve the performances and the speed of a decision tree [40]. An extra tree uses the whole sample data to formulate a decision tree and choose the feature set randomly [41].

Finally, naïve Bayes uses the Bayes theorem concept [42]. *Smoothing parameters* are the continuous hyper-parameters that are in the Bayes theorem. In this case, there is no need to tune the hyper-parameter. However, various fields are employed in the deep learning model, such as natural language processing, computer vision, and machine translation. The idea of artificial neural networks serves as the foundation for these models. Deep belief networks, deep neural networks, feedforward networks, recurrent neural networks, and convolutional networks are types of deep learning architecture [43]. Since the models all have comparable hyper-parameters that need to be tweaked, they gain more from HPO because their underlying neural networks are similar.

Babysitting is one of the optimization techniques that are manual and called the grad student descent or trial and error method because it has a simple workflow that deals with the fundamental hyper-parameter tuning technique [13]. After designing the model, the user will give possible hyper-parameters based on their guessing, experience, or previously evaluated outcomes. The process will repeatedly continue until the desired result is satisfied by the user by reaching optimal hyperparameter values. However, this technique is not feasible for some problems due to complex models, several hyper-parameters, time spent for model evaluation, and nonlinear hyper-parameters interaction [14]. These features have led to the development of techniques for automatically optimizing hyperparameters [44]. The gradient descent method minimizes the cost function to reduce mistakes and increase model correctness. By computing the gradient of the variable, a random point is chosen and arbitrarily chosen a direction to identify the optimal and advance towards the optimal, which is in the opposite direction of the

most significant gradient, to find the next point [45]. Since it is challenging to reduce errors, optimization stops when there is no improvement, at which point a local minimum.

The grid search employs an exhaustive search or brute-force approach to evaluate all potential grid-given combinations to investigate the hyper-parameter configuration space [5]. It analyses the Cartesian product of a specific set of user-supplied data [11]. Whenever the available hyper-parameter configuration space is limited, it works well. While in a random search, the technique picks and trains a specified number of observations using a random selection between the lower and higher boundaries as potential hyper-parameter values [46]. When the configuration region is enormous, a global optimum can be quickly found, even on a minimal budget. Furthermore, since every evaluation is independent, resources can then be shared concurrently, lowering the possibility of spending time on the search space. Bayesian optimization is an iterative procedure for hyper-parameter optimization that bases its computations on the value already tested to establish the following potential evaluation points [47]. First, it generates the appropriate hyper-parameter configuration using two elements, the acquisition function surrogate model [48]. Then, the surrogate model is updated each time the objective function is analyzed. Finally, it uses the previous result to detect the optimal hyperparameter combinations. That makes it more efficient, and methods are effective on stochastic, non-convex, and non-continuous objective functions. The core disadvantage is that it is difficult to reach a global optimum if it fails to stabilize between exploration and exploitation. Instead, it might just go to the local minimum. Furthermore, the model cannot work parallelized because the intermediate results depend on one another [12]. According to the posterior distribution, exploitation entails sampling in the existing location where the global optimum is the most likely to occur, while exploration entails sampling instances in areas that still need to be checked.

The genetic algorithm is based on evolutionary theory as it is one of the heuristic methods. It tests the survival capability and adaptability of individuals. Those with the best are likely to pass them on to the next generation and inherit their parents' worst and best characteristics [49]. Weaker ones are likely to disappear, and better ones have offspring more capable of surviving. After so many generations, only those with better adaptability will be globally optimal [50]. A particle swarm optimization approach uses an evolutionary algorithm widely [51]. It emerged from biological populations that reflect people's social behaviours [52]. It enables groups of particles to move across the search space in an essentially random pattern [14]. This algorithm determines the best solutions by recognizing and exchanging information with each particle in the group. After each iteration, they share information as they operate independently. That makes it easy to parallelize, which leads to an improvement in model efficiency [14]. A global optimum can be achieved only by performing a good population initialization through prior knowledge or by employing approaches designed primarily for discrete hyperparameters [53]. Finally, Hyperband dynamically chooses a manageable number of configurations [54]. It balances model resource usage and its performance mostly with limited resources and time,

which makes it more efficient [55]. However, every hyper-parameter is handled separately., without considering their correlations [56]. As a result, it is inefficient for algorithms like DBSCAN, logistic regression, and SVM that use conditional hyperparameters.

3.0 METHODOLOGY

In this section, we develop a proposed model called hybrid Random-grid optimization (HRGO) to enhance the performance of the models, and the outcomes of applying hyper-parameter optimization techniques are contrasted, analyzed, and optimized on three models due to their adjustable hyper-parameters. Possible hyper-parameters used in SVM are the kernel type and penalty parameter. KNN only uses K as the primary hyper-parameter to determine each sample's nearest Neighbors. Random Forest has a variety of multiple hyper-parameters to adjust. The model performances of different classifiers, k-nearest Neighbors (KNN), Random Forest classifier (RFC), Support Vector Machine (SVM), and their stacking ensemble using logistic regression are compared with the optimized HRGO of all the classifiers. The malaria dataset obtained from the World Health Organization contains different features such as the percentage of the population using at least basic sanitation and drinking water, the total number of yearly malaria reported cases, a total Incidence of malaria, and climatic features of an average temperature and average Rainfall. Algorithms used 80 per cent and 20 per cent for training and testing, respectively. Finally, Grid Search, Random Search, Particle Swarm Optimization, Genetic Algorithm, and Bayesian Optimization were also the techniques implemented during the comparison to examine the effects of using hyper-parameter optimization approaches on the classifiers. The machine learning models and HPO were analyzed using Python libraries like Sklearn [57], Skopt [58], Hyperopt [59], Optunity [60], BOHB [61], and TPOT [62]. The machine learning algorithms are compared using various criteria such as accuracy, macro average, precision, recall, and weighted average to discover which works better in this specific use case and which is most likely to perform equally well given a similar dataset.
Characters define as False Positive (FP), False Negative (FN), True Positive (TP), True Negative (TN).

The performance of the classifier's default configuration is measured by considering how classifiers identify an instance into the category of TP, TN, FP, and FN. Also measured accuracy, precision, and recall using the classification report. Accuracy is the key performance indicator for classifiers with the appropriate hyper-parameter setting. Therefore, the same configuration space for the hyper-parameter is maintained to evaluate all the optimization techniques. For example, KNN has only one hyper-parameter to utilize by setting it to a specific range. They additionally configured for Random Forest and SVM in the same configuration area.

3.1 Proposed HRGO Algorithm

This Algorithm shows how the hybrid form of Random and Grid search can work together in choosing the best combination of hyper-parameters named hybrid random grid optimization (HRGO).

Table 1: Symbols and their description used in the proposed algorithm

S/N	Symbol	Description
1	S	Set of all values
2	S_o	Value to be selected in the search space
3	f (S_o)	The Function of value
4	S_{new}	New independent value after the increment
5	S_z	Last value in search space
6	K	Value to increment

Table 1 indicates symbols and their description that are used in our newly develop HRGO techniques. The steps are:

Step 1: Select an initial and current value for each S_o at random where S_oES.

Step 2: Calculate f (S_o) and put K=0

Step 3: if S_o == optimal value, stop with progress

Step 4: Generate new independent values say S_{new}(K+1) E S in accordance with the selected probability distribution

if f(S_{new}(K+1) < f (S_z)

Set S_{z-1} = S_{new}(K+1)

Else

S_{z-1} = S_z

Step 5: The process will end when the predetermined number of evaluations has been met.

Step 6: If the results are optimal, choose that option; otherwise chose S_{new}, end if

Step 7: Try on all selected values of S_o, S_{new}(k+1)

Step 8: Return the best results after the allotted number of iterations.

3.2 How the Algorithm Work

Since grid search and random search techniques performed independent experiments in creating models with different hyper-parameters in isolation, we can use the information from one random search to improve the grid search experiment. First, use a randomized search to obtain the best hyper-parameter and then use it in a grid search to minimize the time it will take to search for the optimum hyper-parameters. In this case, the two techniques work together to obtain the best hyperparameters combination that improves the model performance by specifying the number of iterations to be performed when looking for the best model. The random search allows values to be sampled randomly from a statistical distribution for each hyper-parameter. The hyper-parameter values of the model will be set by sampling the defined distribution for each iteration. The grid search can then be enhanced to thoroughly search for an optimal solution following a random search to focus on finding the best value for the crucial hyper-parameter by reducing the results. Finally, a combination of all the hyper-parameter values obtained from a random search with some additional values within the regions where the model will perform well is utilized and used to build a model by evaluating the hyper-parameter and selecting the ones that give the best result.
All of the employed optimization techniques have their performance metrics compared:

1. For all optimization approaches, the number of iterations is 100.
2. The experiment is rerun with different random seeds, and we use majority voting to obtain the best result.
3. The most accurate model architecture with the best hyper-parameter configuration is selected. Certain constraints have been taken into account when comparing the optimization algorithms chosen.
4. All optimization methods use the same configuration space for the k-nearest Neighbors, where only one parameter needs to be optimized and is set between 1 to 50 for the assessment of each optimization method.

Our concern is improving the model performance's accuracy, which is only possible when an appropriate selection of hyper-parameters is made in the models or their ensemble. Since predicting the best values for hyper-parameters is difficult, there is a need to try all the possible values to have the optimal one. The HRGO method proposed is used in all the selected classifiers and compared their performances with default running and individual optimization techniques.

Table 2: Hyper-parameter setting space for the Proposed HRGO Technique

Classifiers	Hyper-parameters	Data Type	Specification of search space
HRGO-KNN	n-neighbors	Int	[1, 100]

HRGO-RFC	n_estimators max_depth min_sample_split max_sample_leave criterion max_features	Discrete Discrete Discrete Discrete Categorical Discrete	start=200, stop=2000, num=10 (10,1000,10) [1,3,4,5,7,9] [1,2,4,6,8] [‘Gini’, ‘entropy’] 'auto','sqrt','log2'
HRGO-SVM	C Kernel gamma Randon_state	Float Float Int Int	[1.0] ‘linear’,’rbf’,’poly’,’sigmoid’,’pre computed’ ‘auto’, ‘scale’ Instance or None

Table 2 above describes all the hyper-parameters’ configuration and the configuration search space for all the classifiers used for the newly proposed techniques.

4.0 RESULT AND DISCUSSION

Figure 1 indicates how variables used in the analysis correlated with one another. It also shows a reasonable correlation between malaria incidence and Rainfall and moderately correlated with Temperature, Basic drinking water services, and sanitation services, and less correlation exists with other features. Three models were trained and assessed as a baseline model with their default hyper-parameter setting. We measured the performances of our model by calculating the accuracy, AUC, weighted average, Macro average, precision, recall, and F1 score.

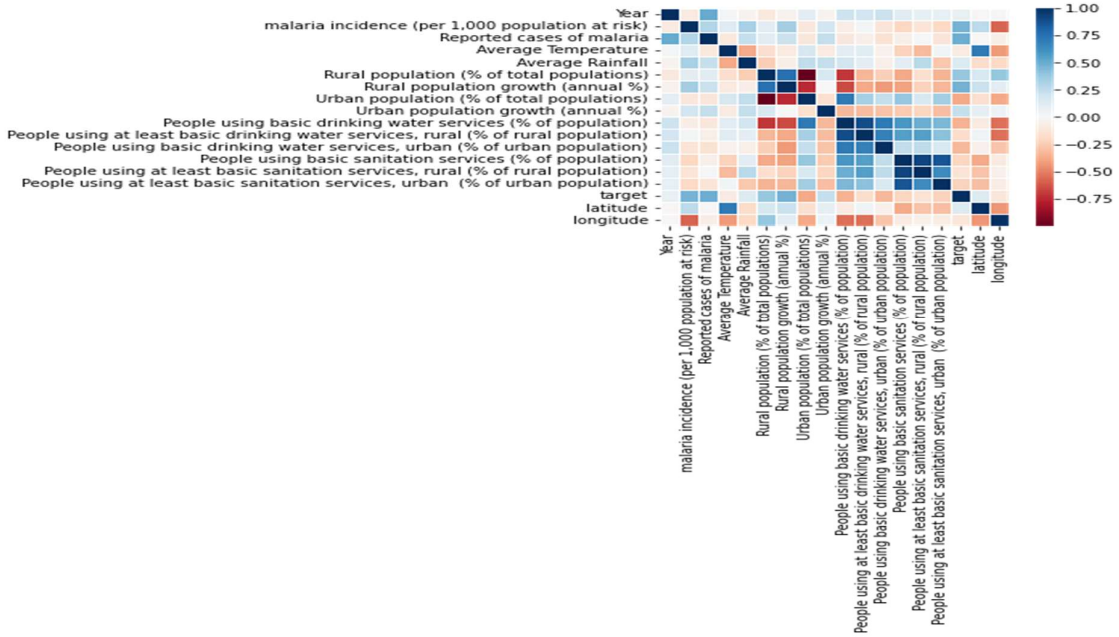

Figure 1: Heat map Correlation of the variables used for analysis

The results provided in Table 3 summarize the confusion matrix for all the classifiers and found that adopting optimization approaches is necessary because using the default hyper-parameters setting does not produce the best model performance in our results. It indicates how the models classified instances correctly and incorrectly. It also shows that the proposed HRGO ensemble can classify the data into TP, TN, FP, and FN more than all other classifiers. It is found that HRGO-SVM, HRGO-RFC, and HRGO-KNN using the Table 2 hyper-parameters configuration has more chance of classifying the instances. Each classifier has its advantage depending on what the user is looking for in the analysis. The ensemble of classifiers produces a better categorization of the instances, followed by Random Forest classifiers, SVM, and KNN.

Table 3: Default classifiers and proposed Confusion Matrix

		PREDICTED															
		RFC		KNN		SVM		RF_Kn_Svm ENSEMBLE		HRGO-RFC		HRGO-KNN		HRGO-SVM		HRGO_ ENSEMBLE	
		1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
Actual	1	190	11	164	37	181	20	192	9	196	5	191	10	201	0	201	2
	0	9	90	39	60	55	44	8	91	8	91	10	89	14	85	7	90

The performance of the model's default hyper-parameters and HRGO is shown in Table 4; the result of the proposed technique clearly shows that HRGO outperforms all other classifiers, particularly HRGO-ensemble with accuracy, recall, precision, macro average, F1- score, weighted average and AUC with 97%, 99%, 96%, 96%, 97%, 97% and 97% respectively. Based on the default configuration, the ensemble is optimal with 96% F1-score, 94% macro average, 94% weighted average, 96% recall, 96% precision, and 94 % accuracy, followed by the Random Forest classifier with 93% SVM and KNN with 75% each.

Table 4: The performance classification of classifiers using default hyper-parameters and proposed method

Classifiers	Accuracy %	Recall %	Precision %	Macro average %	F1-Score %	Weighted average %	AUC %
Random Forest	93	95	95	92	95	93	93
KNN	75	82	81	71	81	75	93
SVM	75	90	77	73	83	74	67
RF_Knn_Svm ENSEMBLE	94	96	96	94	96	94	94
HRGO-RFC	96	98	96	95	97	96	96
HRGO-KNN	93	95	95	92	95	93	92

HRGO-SVM	95	100	93	97	97	96	93
HRGO_Ensemble	97	99	96	96	97	97	97

Table 5 summarizes the default configuration, selected optimization techniques and proposed HRGO methods applied to machine learning algorithms and provides the accuracy of each optimization evaluated. Each optimization technique is evaluated based on accuracy, indicating that the accuracies vary across each technique.

Table 5: Accuracy performance analysis of applying the HPO on the classifiers

Optimization	Random Forest	SVM	K-Nearest Neighbor
Default Configuration	93%	75%	75%
Grid Search	96%	78%	94%
Random Search	86%	%	94%
Genetic Algorithm	96%	77%	95%
Swarm Particle Algorithm	95%	77%	93%
Bayesian Optimization	87%	78%	93%
Proposed HRGO	96%	95%	93%

The accuracy of default configuration is 93% for Random Forest, 75% for each KNN, and SVM, which increased when using any optimization techniques. Comparing the overall performance of all the individual classifiers and their optimization techniques shows that the proposed HRGO method gives more 96%, 95%, and 93% accuracy than all other optimization techniques when using Random Forest, SVM, and KNN, respectively. Therefore, it is essential to utilize optimization techniques to obtain optimal results.

5.0 CONCLUSION

With so many applications in the realm of research, machine learning algorithms have emerged as the method for solving data-related issues. Machine learning models' hyper-parameters must be tweaked to fit a particular dataset for solving practical problems. Although manually adjusting hyper-parameter is challenging and highly expensive. It has become vital since the rate at which data use has dramatically grown in real-world applications. The primary concern for choosing appropriate hyper-parameters is to increase the model performance. The five optimization techniques chosen are grid search, random search, Genetic algorithm, Bayesian optimization, and particle swarm. Each of the techniques has an accuracy higher compared to the default configuration. Our experiment shows that the accuracy increased when using the proposed HRGO technique, which gives 97%, 96%, 95, and 93% for HRGO-ensemble, HRGO-RFC, HRGO-SVM and HRGO-KNN respectively. Optimization approaches are selected based on the algorithm and the hyper-parameters that ensure the model is well-fitted. The newly developed technique can be used well for predicting malaria by giving better accuracy than the existing techniques. The effect of tuning the hyperparameter of three models and their ensemble is observed.

REFERENCES

1. WHO (2019). Malaria. Available: http://www.who.int/malaria/en/
2. WHO (2019) Overview of malaria treatment. Available: http://www.who.int/malaria/areas/treatment/overview/en/.
3. M.I. Jordan, T.M. Mitchell (2015). Machine learning: Trends, perspectives, and prospects, Science 255–260. https://doi.org/10.1126/science.aaa8415.
4. Bonow, R.O.; Mann, D.L.; Zipes, D.P.; Libby, P. Braunwald's (2011). Disease E-Book: A Textbook of Cardiovascular Medicine; Elsevier Health Sciences: Amsterdam, the Netherlands.
5. M.-A. Zoller and M. F. Huber (2019), Benchmark and Survey of Automated Machine Learning Frameworks, arXiv preprint arXiv:1904.12054. https://arxiv.org/abs/1904.12054.
6. R. E. Shawi, M. Maher, S. Sakr (2019). Automated machine learning: Stateof-the-art and open challenges, arXiv preprint arXiv:1906.02287. http://arxiv.org/abs/1906.02287.
7. Yusuf Aliyu.Adamu, Singh J (2021). Malaria Prediction Model Using Machine Learning Algorithms. Turkish Journal of Computer and Mathematics Education, Vol.12 No.10, 7488- 7496. Doi-10.17762/turcomat.v12i10.5655.
8. Yusuf Aliyu Adamu, Jaspreet Singh, (2021). Malaria prediction model using advanced ensemble machine learning techniques. Jour. of Med. P'ceutical & Allied. Sci. V 10 - I 6, 1701, P3794-3801. doi: 10.22270/jmpas.V10I6.170
9. M. Kuhn and K. Johnson, (2013). Applied Predictive Modeling., Springer ISBN: 9781461468493.
10. G.I. Diaz, A. Fokoue-Nkoutche, G. Nannicini, H. Samulowitz, (2017). An effective algorithm for hyperparameter optimization of neural networks, IBM J. Res. Dev. 61, 1–20. https://doi.org/10.1147/JRD.2017.2709578
11. F. Hutter, L. Kotthoff, and J. Vanschoren, Eds., (2019). Automatic Machine Learning: Methods, Systems, Challenges, Springer ISBN: 9783030053185.
12. N. Decastro-Garc´ıa, A. L. Mu˜noz Casta˜neda, D. Escudero Garc´ıa, and ´ M. V. Carriegos, (2019). Effect of the Sampling of a Dataset in the Hyperparameter Optimization Phase over the Efficiency of a Machine Learning Algorithm, Complexity. https://doi.org/10.1155/2019/6278908.
13. S. Abreu, (2019). Automated Architecture Design for Deep Neural Networks, arXiv preprint arXiv:1908.10714. http://arxiv.org/abs/1908.10714.
14. O. S. Steinholtz, (2018). A Comparative Study of Black-box Optimization Algorithms for Tuning of Hyper-parameters in Deep Neural Networks, M.S. thesis, Dept. Elect. Eng., Lule˚a Univ. Technol.,.
15. G. Luo, (20`6). A review of automatic selection methods for machine learning algorithms and hyper-parameter values, Netw. Model. Anal. Heal. Informatics Bioinforma. 5, 1–16. https://doi.org/10.1007/s13721- 016-0125-6.

16. D. Maclaurin, D. Duvenaud, R.P. Adams, (2015). Gradient-based Hyperparameter Optimization through Reversible Learning, arXiv preprint arXiv:1502.03492. http://arxiv.org/abs/1502.03492.
17. J. Bergstra, R. Bardenet, Y. Bengio, and B. K´egl, (2011). Algorithms for hyperparameter optimization, Proc. Adv. Neural Inf. Process. Syst., 2546–2554.
18. B. James and B. Yoshua, (2012). Random Search for Hyper-Parameter Optimization, J. Mach. Learn. Res. 13 (1), 281–305.
19. L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, Hyperband, (2012). A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18, 1–52.
20. M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. De Moor, Easy Hyperparameter Search Using Optunity, arXiv preprint arXiv:1412.1114, (2014). https://arxiv.org/abs1412.1114.
21. Li H, (2018). Ensemble learning for overall power conversion efficiency of the all-organic dye-sensitized solar cells. IEEE Access 6:34118–26. Doi-10.1109/ACCESS.2018.2850048.
22. M. Woube, (1997). Geographical distribution and dramatic increases in incidences of malaria: consequences of the resettlement scheme in Gambela, SW Ethiopia," Indian Journal of Malariology, vol. 34, no. 3, pp. 140–163.
23. A. D. Kassa and B. B. Beyene, (2014). Climate variability and malaria transmission—fog era district, Ethiopia, 2003–2011," Science Journal of Public Health, vol. 2, no. 3, pp. 234–237.
24. M. C. Thomson, S. J. Mason, T. Phindela, and S. J. Connor, (2005). "Use of rainfall and sea surface temperature monitoring for malaria early warning in Botswana," The American Journal of Tropical Medicine and Hygiene, vol. 73, no. 1, pp. 214–221.
25. Y. Kim, (2014). "Convolutional neural networks for sentence classification," in Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751.
26. R. Caruana, A. Niculescu-Mizil, (2006). An empirical comparison of supervised learning algorithms, ACM Int. Conf. Proceeding Ser. 148, 161–168. https://doi.org/10.1145/1143844.1143865.
27. J. A. Hartigan and M. A. Wong, (1979). "Algorithm AS 136: A k-means clustering algorithm," Journal of the Royal Statistical Society. Series C (Applied Statistics), vol. 28, pp. 100–108.
28. K. P. Bennett and A. Demiriz, (1999). "Semi-supervised support vector machines," in Advances in Neural Information processing systems, pp. 368–374.
29. T.Chen, C.Guestrin, (2016). XGBoost: a scalable tree boosting system, arXiv preprint arXiv:1603.02754. http://arxiv.org/abs/1603.02754.
30. C. Gambella, B. Ghaddar, J. Naoum-Sawaya, (2019). Optimization Models for Machine Learning: A Survey, 1–40. http://arxiv.org/abs/1901.05331.

31. W. Zuo, D. Zhang, K. Wang, (2008). On kernel difference-weighted nearest neighbor classification, Pattern Anal. Appl. 11, 247–257. https://doi.org/10.1007/s10044-007-0100-z.
32. A. Smola, V. Vapnik, (1997). Support vector regression machines, Adv. Neural Inf. Process. Syst. 9, 155-161.
33. L. Yang, R. Muresan, A. Al-Dweik, L.J. Hadjileontiadis, (2018). Image-Based Visibility Estimation Algorithm for Intelligent Transportation Systems, IEEE Access. 6, 76728–76740. https://doi.org/10.1109/ACCESS.2018.2884225.
34. O.S. Soliman, A.S. Mahmoud, 92012). A classification system for remote sensing satellite images using support vector machine with non-linear kernel functions, 2012 8th Int. Conf. Informatics Syst. INFOS, BIO-181-BIO-187.
35. S. Rasoul, L. David, (1991). A Survey of Decision Tree Classifier Methodology, IEEE Trans. Syst. Man. Cybern. 21, 660–674.
36. D.M. Manias, M. Jammal, H. Hawilo, A. Shami, P. Heidari, A. Larabi, R. Brunner, (2019). Machine learning for performance aware virtual network function placement, 2019 IEEE Glob. Commun. Conf. GLOBECOM- Proc. 12–17. https://doi.org/10.1109/GLOBECOM38437.2019.9013246.
37. L. Yang, A. Moubayed, I. Hamieh, A. Shami, (2019). Tree-based intelligent intrusion detection system in the internet of vehicles, IEEE Glob. Commun. Conf. GLOBECOM 2019 - Proc. https://doi.org/10.1109/GLOBECOM38437.2019.9013892.
38. S. Sanders, C. Giraud-Carrier, (2017). Informing the use of hyperparameter optimization through meta-learning, Proc. - IEEE Int. Conf. Data Mining, ICDM. 1051–1056. https://doi.org/10.1109/ICDM.2017.137.
39. M. Injadat, F. Salo, A.B. Nassif, A. Essex, A. Shami, (2018). Bayesian Optimization with Machine Learning Algorithms Towards Anomaly Detection, 2018 IEEE Glob. Commun. Conf. 1–6. https://doi.org/10.1109/glocom.2018.8647714.
40. T.Chen, C.Guestrin, XGBoost: a scalable tree boosting system, arXiv preprint arXiv:1603.02754. http://arxiv.org/abs/1603.02754.
41. K. Arjunan, C.N. Modi, (2016). An enhanced intrusion detection framework for securing network layer of cloud computing, ISEA Asia Secure. Prev. Conf. 2017, ISEASP 2017. (2017) 1–10. https://doi.org/10.1109/ISEASP.2017.7976988.
42. I. Rish, (2001). An empirical study of the naive Bayes classifier, IJCAI 2001 Work. Empir. methods Artif. Intell.,, 41-46.
43. W. Yin, K. Kann, M. Yu, and H. Sch¨utze, (2017). Comparative Study of CNN and RNN for Natural Language Processing, arXiv preprint arXiv:1702.01923. https://arxiv.org/abs1702.01923.
44. I. Ilievski, T. Akhtar, J. Feng, C.A. Shoemaker, (2017). Efficient hyperparameter optimization of deep learning algorithms using deterministic RBF surrogates, 31st AAAI Conf. Artif. Intell. AAAI 2017. 822–829.
45. Y. Bengio, (2000). Gradient-based optimization of hyperparameters, Neural Comput. 12 (8) 1889-1900.

46. B. James and B. Yoshua, (2012). Random Search for Hyper-Parameter Optimization, J. Mach. Learn. Res. 13 (1), 281–305.
47. J. Snoek, H. Larochelle, R. Adams, (2012). Practical Bayesian optimization of machine learning algorithms Adv. Neural Inf. Process. Syst. 4, 2951-2959.
48. S. Sanders, C. Giraud-Carrier, (2017). Informing the use of hyperparameter optimization through meta-learning, Proc. - IEEE Int. Conf. Data Mining, ICDM. 1051–1056. https://doi.org/10.1109/ICDM.2017.137.
49. S. Lessmann, R. Stahlbock, S.F. Crone, (2005). Optimizing hyperparameters of support vector machines by genetic algorithms, Proc. 2005 Int. Conf. Artif. Intell. ICAI'05. 1, 74–80.
50. F. Itano, M.A. De Abreu De Sousa, E. Del-Moral-Hernandez, (2018). Extending MLP ANN hyper-parameters Optimization by using Genetic Algorithm, Proc. Int. Jt. Conf. Neural Networks. 1–8. https://doi.org/10.1109/IJCNN.2018.8489520.
51. X. Yan, F. He, Y. Chen, (2017). A Novel Hardware / Software Partitioning Method Based on Position Disturbed Particle Swarm Optimization with Invasive Weed Optimization, 32 340–355. https://doi.org/10.1007/s11390-017-1714-2.
52. Q. Yao et al., (2018). Taking Human out of Learning Applications: A Survey on Automated Machine Learning, arXiv preprint arXiv:1810.13306, http://arxiv.org/abs/1810.13306.
53. M.Y. Cheng, K.Y. Huang, M. Hutomo, (2018). Multi objective Dynamic Guiding PSO for Optimizing Work Shift Schedules, J. Constr. Eng. Manag. 144 1–7. https://doi.org/10.1061/(ASCE)CO.1943- 7862.0001548.
54. L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, (2012). A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18, 1–52.
55. K. Eggensperger, F. Hutter, H.H. Hoos, K. Leyton-Brown, (2015). Efficient benchmarking of hyperparameter optimizers via surrogates, Proc. Natl. Conf. Artif. Intell. 2, 1114–1120.
56. J. Wang, J. Xu, and X. Wang, (2018). Combination of Hyperband and Bayesian Optimization for Hyperparameter Optimization in Deep Learning, arXiv preprint arXiv:1801.01596. https://arxiv.org/abs1801.01596.
57. F. Pedregosa et al., (2011). Scikit-learn: Machine learning in Python, J. Mach. Learn. Res., 12 2825–2830.
58. Tim Head, MechCoder, Gilles Louppe, et al., (2018). scikitoptimize/scikitoptimize: v0.5.2. https://doi.org/10.5281/zenodo.1207017.
59. J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, D.D. Cox, (2015). Hyperopt: A Python library for model selection and hyperparameter optimization, Comput. Sci. Discov. 8,. https://doi.org/10.1088/1749- 4699/8/1/014008.
60. M. Claesen, J. Simm, D. Popovic, Y. Moreau, and B. De Moor, (2014). Easy Hyperparameter Search Using Optunity, arXiv preprint arXiv:1412.1114, https://arxiv.org/abs1412.1114.

61. S. Falkner, A. Klein, F. Hutter, (2018). BOHB: Robust and Efficient Hyperparameter Optimization at Scale, 35th Int. Conf. Mach. Learn. ICML 2018. 4, 2323–2341.
62. R. S. Olson and J. H. Moore, (2019). TPOT: A tree-based pipeline optimization tool for automating machine learning, Auto Mach. Learn, 151- 160. https://doi.org/10.1007/978-3-030-05318-5-8