 ISSN 2579-7298

Hyperparameter Tuning in Machine Learning to Predicting Student Academic Achievement

Muhammad Arifin[a,1], Soni Adiyono[a,2]

[a] Sistem Informasi, Fakultas Teknik, Universitas Muria Kudus, Indonesia
[1] arifin.m@umk.ac.id *; [2] soni.adiyono@umk.ac.id
* corresponding author

ARTICLE INFO

Article history
Received
Revised
Accepted

Keywords
Hyperparameters
Gradient Boosting Tree
Grid Search
Random Search
Optuna

ABSTRACT

Prediction of student academic achievement is a very important research area; this can be seen from the many researchers who conduct research in this area. To make predictions, a machine learning model is needed. Along with their parameters, the majority of machine learning models have associated hyperparameters. However, knowing the right mix of hyperparameters is essential for robust model performance. A methodical procedure called hyperparameter optimization (HPO) aids in determining the appropriate values for them. In this study we compared four hyperparameters tuning techniques, namely HyperOpt, Random Search, Optuna and Grid Search. The results of the hyperparameters from each of these techniques are then used in machine learning algorithms to predict student academic achievement. Validation uses the 5-fold cross validation method while performance testing uses Mean absolute error. From the experimental results it was found that the hyperparameter technique The best method for predicting student academic achievement in machine learning models is gridsearchcv.

1. Introduction

Predicting academic achievementis one of the main areas in educational data mining [1]. To date, there are 13,653 documents documented in Scopus that discuss students' academic performance. Research on student academic achievementwas first conducted in 1954 by Reed M. Merrill in order to evaluate student academic achievementon probation. Articles related to student academic achievementfrom 2002 to 2021 totaled 13,016 articles. The country with the highest number of articles related to the topic of student academic achievementis the United States with 4,098 articles, followed by India with 815 articles, China, Spain, Australia with 700 articles each. Meanwhile, Indonesia is ranked 8th with 470 articles. The development of research with a focus on predicting the academic achievementof students in the last ten years has also experienced a remarkable increase. There were 1027 articles in Scopus from 2012-2021. Based on these data, the prediction of student academic achievementis an important and interesting research area to study.

Student academic achievement prediction models generally use the GPA variable as a target. Classify GPA into classes [2]–[5]. In addition, some researchers use regression models to improve student academic achievement[6], [7]. Regression models such as graddient boosting regression tree (GBRT), random forest and neural networks involve a number of hyperparameters that must be setup before using them [8].

The goal of GBRT is to enhance the regression achievementof a single model by combining many fitted models. As a result, GBRT uses two algorithms: the decision tree (DT) group's regression tree and gradient boosting, a general metalearning approach used to combine single

regression tree models [9]. We will concentrate on this model since it is currently the best-performing approach for the majority of Kaggle contests [10], [11] and because the achievementis greatly influenced by the selection of the hyperparameters.

Algorithms that use machine learning automatically pick up new information and, as a result, modify their internal parameters in response to new information. These parameters are referred to as "model parameters" or just "parameters" for short. However, there are some settings that must be made beforehand rather than being changed throughout the learning process. These parameters are commonly known as "hyperparameters." While model parameters illustrate how input data is converted into the desired output, hyperparameters indicate the structure and organization of the model itself. Depending on the selection and values of a machine learning model's hyperparameters, its achievementcan significantly affect, A machine learning can get significantly higher accuracy when it makes the right hyperparameter tuning [12], [13].

Tuning hyperparameters is an important step before implementing a prediction algorithm [8], [14]–[16]. Model achievementdepends on hyperparameters model selection [17]. The choice of the hyper-parameter configuration is known to have a significant impact on the achievementof machine learning models [16]. [15], [18] Tuning Hyperparameters for Deep Learning, [12], [16], [19] tuning hyperparameter to machine learning algorithms, [20] to ensemble machine learning, [21] to random forest, [22], [23] to neural networks, [24] to SVM. [25] Manual search is one method for Hyper-Parameter optimization; however, this requires a significant amount of time.

The originality of the paper is found in the comparison of numerous hyperparameter tuning techniques to Predicting Student Academic Achievement. Four hyperparameter techniques are applied in this research; Grid Search, Random Search, Optuna and HyperOpt.

2. Method

In previous studies we have compared several prediction algorithms to predict student academic achievement, researchers discovered that the best technique for this is the Gradient Boosting Regression Tree (GBRT) [6]. For this reason, in this study we tuned hyperparameters on the algorithm. Figure 1. show the steps for this study.

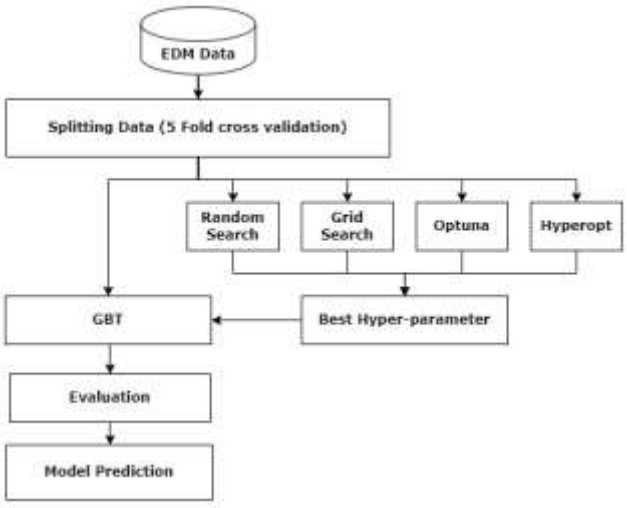

Fig. 1. *Research Steps*

2.1. Hyperparameters

During training, parameters model acquires their values. You cannot manually set this value. From the provided data, the model learns. Model parameters also include, for instance, the linear regression model's coefficients [26].

In contrast, hyperparameters are not determined from the data; instead, researchers need to set them manually. During the model-building phase, researchers must always specify the values for hyperparameters before starting the training process [26].

2.2. Gradient Bosting Tree (GBT)

Other names for the Gradient Boosting Algorithm are Gradient Tree Boosting, Stochastic Gradient Boosting, and GBM. One of the best machine learning models recently established is the Gradient Boosting Machine (GBM). The GBM performs exceptionally well on medium-sized datasets with no more than a few million observations and structured data, where the data is organized into rows and columns. The GBM functions as an ensemble approach by training numerous individual learners, typically decision trees. This is the fundamental information you need to understand about the GBM. However, in a GBM, the trees are trained sequentially, with each tree learning from the mistakes of the preceding ones, unlike in a random forest where the trees are trained in simultaneously. Gradient Descent is used to aggregate the contributions of each individual learned during training to create a single strong ensemble learner from the hundreds or thousands of weak learners (the weights of the individual trees would therefore be a model parameter).

There are numerous hyperparameters available in the GBM that may be adjusted to influence both the general ensemble (such the learning rate) and the individual decision trees (such as the maximum depth of a tree or the number of leaves it has). Because there are complicated interactions between hyperparameters, it is challenging to determine which combination of hyperparameters will function best based solely on theory. Thus, hyperparameter tuning is required because there is no other method to determine the best hyperparameter values than to test a large number of possible combinations on a dataset.

The ensemble model can be categorized into three main groups: Miscellaneous Parameters, Boosting Parameters, and Tree-Specific Parameters. The parameters used for defining a tree are; min_samples_leaf, max_leaf_nodes, min_samples_split, max_depth, min_weight_fraction_leaf and max_features. Parameters for managing boosting is learning_rate, n_estimators and subsample. miscellaneous parameters that have an impact on the general functionality are verbose, presort, warm start random state, loss, and init [27]. We will modify four parameters in this study: max_depth, learning_rate, n_estimators, and subsample similar to the approach taken by [18], [28] for tuning GBRT.

Four parameters will be adjusted in this study: max_depth, learning_rate, subsample, and n_estimators. The n_estimators parameter controls the number of improvement steps, with a higher number typically enhancing performance as gradient boosting can handle overfitting well; its value should be between 1 and infinity. The learning_rate parameter adjusts the contribution of each tree; a higher learning rate means each tree has less impact, creating a trade-off with n_estimators, and its value ranges from 0 to infinity. Max_depth determines the depth of the tree, with a tree of depth h capturing interactions of order h, resulting in up to 2^h leaf nodes and 2^h - 1 split nodes if set to h. The subsample parameter indicates the fraction of the sample used for fitting each base learner; values less than 1.0 lead to Stochastic Gradient Boosting, increasing bias but reducing variance. The subsample value should be between 0 and 1.

2.3. Data Sets

Prediction of student academic achievementusing data related to education. The process of processing educational data is called Educational Data Mining (EDM). An Indonesian university served as the setting for this study. The data set was compiled at the conclusion of the semester of 2022 and consists of 16-week lecture data from every student. In this study, academic data, such as GPA data, demographic data, which include gender and residence, economic data, such as parental income data, and student organization activity data derived from student participation in campus organizations are combined with data from the university's LMS (Moodle) for one semester. These data are combined with Moodle records that have been extracted. GPA is chosen as the goal column for this data set's experimentation because it serves as a general benchmark for assessing academic achievementin students. Student information is taken out of the Moodle LMS, and information from various sources is combined and filtered in accordance with predetermined standards. The LMS records were collected for 19 weeks (one semester beginning in February) and resulted in a total of 199,700 recordings 8,500 enrolled students who participated in lectures. When a student registers as a new student, the academic information system (SIA) provides demographic and academic data,

while registration data provides economic information. The student affairs department's co-curricular data comes in the form of extraction findings from university-based organization decrees. The data's inconsistent values are eliminated. Students with a GPA below 1, those who use the LMS very infrequently, those who have academic information but no LMS record, and so on are some examples of these students. Consequently, 4436 student data sets' worth of information were examined for the study

3. Results and Discussion

Results

To compare the four hyperparameter techniques above, we use the scikit-learn library in Python i.e.; *GridSearchCV, RandomizedSearchCV, Optuna* and *Hyperopt*

3.1. GridSearchCV

The GridSearchCV tool from scikit-learn library in Python was used to tune hyperparameters. It calculates the score of each model based on the scoring index after all parameters are evaluated. This helps identify the best model and parameter combination. This method improves development efficiency by automating the tuning process, but it can be time-consuming, especially as the number of parameters and possible values increases. Key parameter ranges are typically set and refined beforehand to help control this.

The first parameter, called Estimator, is a Scikit-learn machine learning model that acts as our foundational model. The search space is defined by the Python dictionary {param_grid}. There are 256 potential options for our 4-dimensional search space (4 x 4 x 4 x 4). This means that we used Grid Search to train 576 different models. The scoring system that is used to evaluate the achievementof the model. Usual classification terms include "accuracy" or "roc auc." The terms "r^2" or "neg_mean_squared_error" are preferred for regression. Grid search will run n_jobs parallel jobs, which is the number specified below. Set a greater value computer CPU has numerous cores. All cores are used when the value is -1. This will speed up the execution process. The cross-validation fold count, denoted as `cv`, typically ranges from 5 to 15, with common values being 5, 10, or 15. With `cv` set to 5 in this case, each hyperparameter combination is tested five times. This results in a total of 1,280 iterations (256 combinations x 5 folds). We set the vulnerable parameters as n_estimators [50, 100, 200, 300], learning_rate [0.1, 1, 1.5, 2], max_depth [2, 8, 12, 16], subsample [0.9, 0.5, 0.2, 0.1]. The results of experiments using Grid Search are shown in Figure 2.

```
32 om Grid Search " )
33  MSE across all searched param:\n", grid.best_score_)
34  parameters across all searched param:\n", grid.best_params_)

Results from Grid Search

The Best MSE across all searched param:
-0.2491447879628059

The Best parameters across all searched param:
{'learning_rate': 0.02, 'max_depth': 16, 'n_estimators': 300,
'subsample': 0.5}
```

Fig. 2. Results from Grid Search

3.2. RandomizedSearchCV

In addition to Gridsearchcv the scikit-learn library also provides RandomizedSearchCV for tuning hyperparameters. When choosing the best combination of hyperparameters, random search does not examine all possible possibilities. Instead, it uses the RandomizedSearchCV function's n_iter parameter to check a defined number of alternatives that are randomly chosen.

The majority of the variables are the same as those in the GridSearchCV function. In this case, param_distributions, not param_grid, defines the search space. Aside from that, the number of hyperparameter combinations to be chosen at random is specified by the n iter parameter. This is due to the fact that not all hyperparameter combinations defined in the search space are checked by random search. It just takes a random sampling of possible combinations into account. Here, n_iter=10 designates that a random sample of size 10 with 10 unique hyperparameter combinations will be used. Consequently, random search only develops 10 distinct models (previously, 256 models with Grid Search). Controls the randomization of the sample of hyperparameter combinations obtained at each execution. Any integer will do. The total number of iterations in this

example is 100 (10 × 10), which is significantly fewer than in the previous situation (1280 iterations). We set the vulnerable parameters as same with Grid Search. The results of experiments using Random Search are shown in Figure 3.

```
ore across all searched param:\n", Rand.best_score_)
ters across all searched param:\n", Rand.best_params_)

The Best MAE score across all searched param:
-0.25977304025970793

The Best parameters across all searched param:
{'subsample': 0.5, 'n_estimators': 50, 'max_depth': 12,
'learning_rate': 0.1}
```

Fig. 3. Results from Random Search

3.3. *Optuna*

According to the authors [29], Optuna offers several key features: (1) efficient searching and pruning strategies; (2) a define-by-run API that allows users to dynamically create the parameter search space; and (3) an easy setup and flexible architecture suitable for various tasks, from large-scale distributed computing to lightweight experiments using an interactive interface. We set the vulnerable parameters as learning_rate: trial.suggest_float ("learning_rate", 0.1, 0.5), n_estimators: trial.suggest_int("n_estimators", 2, 300), subsample: trial.suggest_float ("subsample", 0.1, 1), max_depth: trial.suggest_int("max_depth", 1, 8). The results of experiments using Optuna are shown in Figure 4.

```
trial = study.best_trial
print('MAE: {}'.format(trial.value))
print("Best hyperparameters: {}".format(trial.params))

MAE: 0.2787275973463638
Best hyperparameters: {'n_estimators': 106, 'subsample': 0.6
661231647189927, 'learning_rate': 0.10019773329797556, 'max_
depth': 8}
```

Fig. 4. Results from Optuna

3.4. Hyperopt

The Scikit-Learn library can automatically adjust its algorithms owing to Hyperopt-Sklearn. Both model selection and hyperparameter tuning are possible with it. The solvers RS, SA, and TPE have been implemented in Hyperopt-Sklearn [30]. We set the vulnerable parameters space as learning_rate: hp.uniform('learning_rate', 0.1, 1), n_estimators: hp.choice('n_estimators', [100, 200, 300]), subsample: hp.choice('subsample', [.5, .75, 1]), and max_depth: hp.quniform('max_depth', 1, 15, 1). The results of experiments using Hyperopt are shown in Figure 5.

```
SCORE:
0.5297682994260802
SCORE:

0.39497870642178734
SCORE:
0.42805850304545395
100%|██████████| 100/100 [05:55<00:00,  3.56s/trial, best
loss: 0.3686409533606595]
{'learning_rate': 0.10943128484457938, 'max_depth': 5.0,
'n_estimators': 1, 'subsample': 2}
```

Fig. 5. *Results from Hyperopt*

The hyperparameter analysis results shown in the Table 1, GridSearchCV emerged as the most effective technique, with the lowest Mean Absolute Error (MAE) of 0.249. The parameters used in GridSearchCV, including a small learning rate (0.02) and a high number of estimators (300), show that this approach successfully optimizes the balance between model complexity and generalization, allowing the model to learn more deeply and carefully. In contrast, RandomizedSearchCV produces a slightly higher MAE, namely 0.259, although parameters such as a higher learning rate (0.1) and a lower number of estimators (50) may cause the model to be less optimal in capturing the complexity of the data. Optuna, with an MAE of 0.278, shows worse performance than GridSearchCV and RandomizedSearchCV. The parameters used, such as larger subsamples and lower max depth, may

limit the model's capacity to effectively capture data patterns. Finally, Hyperopt produced the highest MAE of 0.368, with parameters such as a very small number of estimators and a low max depth, indicating that this model most likely suffers from underfitting and cannot adequately capture the complexity of the data. Overall, GridSearchCV seems to provide the best results and represents a more effective approach in determining hyperparameters for this model.

Table 1. Hyperparameter Techniques Values

Hyperparameter Techniques	*learning_rate*	*subsample*	*n_estimators*	*max_depth*	MAE
GridSearchCV	0.02	0.5	300	16	0.249
RandomizedSearchCV	0.1	0.5	50	12	0.259
Optuna	0.1	0.666	106	8	0.278
Hyperopt	0.109	2	1	5	0.368

Discuss

Based on the experiments that have been carried out, it shows that *GridSearchCV* is the best hyperparameter technique feed with the lowest MAE value. Meanwhile [17] said that Optuna is better when compared to HyperOpt, Optunity and SMAC. [31] Random search is faster when compared to grid search, but cannot guarantee the results. [22] Grid Search is better than Random Search but the best method is Self-Tuning Networks. [32] Grid Search shows better stability than Random Search. However, this difference is not big. [33] recommend more Random Search to search for the best hyperparameters. [34] discover that the Hyperopt technique works better than the Random search and Grid search methods due to its higher mean Gini score, a sign of more accurate predictions. Random Search shows the best achievementwhen compared to TPE, Grid Search, and CMAES [35].

4. Conclusion

The results showed that from the comparison of the four hyperparameter techniques, they had almost the same MAE value. Vulnerable MAE values between techniques are insignificant. GridSearchCV is a technique that has the lowest MAE value, but to achieve this value requires a large estimator value and depth and a very small learning_rate.

References

[1] C. Romero, M. Ventura, Sebastian Pechenizkiy, and R. S. J. . Baker, *Handbook of Educational Data Mining*, 1st ed. United States of America: Springer US, 2010.

[2] R. O. Aluko, E. I. Daniel, O. Shamsideen Oshodi, C. O. Aigbavboa, and A. O. Abisuga, "Towards reliable prediction of academic performance of architecture students using data mining techniques," *J. Eng. Des. Technol.*, vol. 16, no. 3, pp. 385–397, 2018, doi: 10.1108/JEDT-08-2017-0081.

[3] H. Karalar, C. Kapucu, and H. Gürüler, "Predicting students at risk of academic failure using ensemble model during pandemic in a distance learning system," *Int. J. Educ. Technol. High. Educ.*, vol. 18, no. 1, 2021, doi: 10.1186/s41239-021-00300-y.

[4] R. Conijn, C. Snijders, A. Kleingeld, and U. Matzat, "Predicting student performance from LMS data: A comparison of 17 blended courses using moodle LMS," *IEEE Trans. Learn. Technol.*, vol. 10, no. 1, pp. 17–29, 2017, doi: 10.1109/TLT.2016.2616312.

[5] S. Helal *et al.*, "Predicting academic performance by considering student heterogeneity," *Knowledge-Based Syst.*, vol. 161, no. December 2017, pp. 134–146, 2018, doi: 10.1016/j.knosys.2018.07.042.

[6] M. Arifin, Widowati, Farikhin, A. Wibowo, and B. Warsito, "Comparative Analysis on Educational Data Mining Algorithm to Predict Academic Performance," *Proc. - 2021 Int. Semin. Appl. Technol. Inf. Commun. IT Oppor. Creat. Digit. Innov. Commun. within Glob. Pandemic, iSemantic 2021*, pp. 173–178, 2021, doi: 10.1109/iSemantic52711.2021.9573185.

[7] L. W. Santoso and Yulia, "Predicting student performance in higher education using multi-regression models," *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 18,

no. 3, pp. 1354–1360, 2020, doi: 10.12928/TELKOMNIKA.v18i3.14802.

[8] P. Probst, A. L. Boulesteix, and B. Bischl, "Tunability: Importance of hyperparameters of machine learning algorithms," *J. Mach. Learn. Res.*, vol. 20, pp. 1–32, 2019.

[9] C. Qi, A. Fourie, and X. Zhao, "Back-Analysis Method for Stope Displacements Using Gradient-Boosted Regression Tree and Firefly Algorithm," *J. Comput. Civ. Eng.*, vol. 32, no. 5, Sep. 2018, doi: 10.1061/(ASCE)CP.1943-5487.0000779.

[10] M. T. Young, J. Hinkle, A. Ramanathan, and R. Kannan, "HyperSpace: Distributed Bayesian Hyperparameter Optimization," *Proc. - 2018 30th Int. Symp. Comput. Archit. High Perform. Comput. SBAC-PAD 2018*, no. 1, pp. 339–347, 2019, doi: 10.1109/CAHPC.2018.8645954.

[11] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, vol. 13-17-Augu, pp. 785–794, 2016, doi: 10.1145/2939672.2939785.

[12] E. Elgeldawi, A. Sayed, A. R. Galal, and A. M. Zaki, "Hyperparameter tuning for machine learning algorithms used for arabic sentiment analysis," *Informatics*, vol. 8, no. 4, pp. 1–21, 2021, doi: 10.3390/informatics8040079.

[13] R. G. Mantovani, A. L. D. Rossi, E. Alcobaça, J. Vanschoren, and A. C. P. L. F. de Carvalho, "A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves SVM classifiers," *Inf. Sci. (Ny).*, vol. 501, pp. 193–221, Oct. 2019, doi: 10.1016/j.ins.2019.06.005.

[14] H. J. P. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of Tuning Hyperparameters of Machine Learning Algorithms," 2020.

[15] A. H. Victoria and G. Maragatham, "Automatic tuning of hyperparameters using Bayesian optimization," *Evol. Syst.*, vol. 12, no. 1, pp. 217–223, 2021, doi: 10.1007/s12530-020-09345-2.

[16] J. Zhang, Q. Wang, and W. Shen, "Hyper-parameter optimization of multiple machine learning algorithms for molecular property prediction using hyperopt library," *Chinese J. Chem. Eng.*, vol. 52, pp. 115–125, 2022, doi: 10.1016/j.cjche.2022.04.004.

[17] S. Shekhar, A. Bansode, and A. Salim, "A Comparative study of Hyper-Parameter Optimization Tools," *2021 IEEE Asia-Pacific Conf. Comput. Sci. Data Eng. CSDE 2021*, 2021, doi: 10.1109/CSDE53843.2021.9718485.

[18] J. Rijsdijk, L. Wu, G. Perin, and S. Picek, "Reinforcement learning for hyperparameter tuning in deep learning-based side-channel analysis," *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, vol. 2021, no. 3, pp. 677–707, 2021, doi: 10.46586/tches.v2021.i3.677-707.

[19] P. Schratz, J. Muenchow, E. Iturritxa, J. Richter, and A. Brenning, "Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data," *Ecol. Modell.*, vol. 406, no. April 2018, pp. 109–120, 2019, doi: 10.1016/j.ecolmodel.2019.06.002.

[20] J. Wong, T. Manderson, M. Abrahamowicz, D. L. Buckeridge, and R. Tamblyn, "Can Hyperparameter Tuning Improve the Performance of a Super Learner?: A Case Study," *Epidemiology*, vol. 30, no. 4, pp. 521–531, 2019, doi: 10.1097/EDE.0000000000001027.

[21] P. Probst, M. N. Wright, and A. Boulesteix, "Hyperparameters and tuning strategies for random forest," *WIREs Data Min. Knowl. Discov.*, vol. 9, no. 3, May 2019, doi: 10.1002/widm.1301.

[22] M. MacKay, P. Vicol, J. Lorraine, D. Duvenaud, and R. Grosse, "Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions," *7th Int. Conf. Learn. Represent. ICLR 2019*, vol. 1, no. 1, pp. 1–25, 2019.

[23] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing Millions of Hyperparameters by Implicit Differentiation," vol. 108, 2019.

[24] E. Duarte and J. Wainer, "Empirical comparison of cross-validation and internal metrics for tuning SVM hyperparameters," *Pattern Recognit. Lett.*, vol. 88, pp. 6–11, 2017, doi: 10.1016/j.patrec.2017.01.007.

[25] S. Putatunda and K. Rama, "A Modified Bayesian Optimization based Hyper-Parameter Tuning Approach for Extreme Gradient Boosting," *2019 15th Int. Conf. Inf. Process. Internet Things, ICINPRO 2019 - Proc.*, 2019, doi: 10.1109/ICInPro47689.2019.9092025.

[26] H. Ma, X. Yang, J. Mao, and H. Zheng, "The Energy Efficiency Prediction Method Based on Gradient Boosting Regression Tree," *2nd IEEE Conf. Energy Internet Energy Syst. Integr. EI2 2018 - Proc.*, vol. 1, no. 4, 2018, doi: 10.1109/EI2.2018.8581904.

[27] P. Datta, P. Das, and A. Kumar, "Hyper parameter tuning based gradient boosting algorithm for detection of diabetic retinopathy: an analytical review," *Bull. Electr. Eng. Informatics*, vol. 11, no. 2, pp. 814–824, 2022, doi: 10.11591/eei.v11i2.3559.

[28] Z. M. Alhakeem, Y. M. Jebur, S. N. Henedy, H. Imran, L. F. A. Bernardo, and H. M. Hussein, "Prediction of Ecofriendly Concrete Compressive Strength Using Gradient Boosting Regression Tree Combined with GridSearchCV Hyperparameter-Optimization Techniques," *Materials (Basel).*, vol. 15, no. 21, p. 7432, 2022, doi: 10.3390/ma15217432.

[29] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna : A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul. 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.

[30] J. Bergstra, B. Komer, C. Eliasmith, D. Yamins, and D. D. Cox, "Hyperopt: A Python library for model selection and hyperparameter optimization," *Comput. Sci. Discov.*, vol. 8, no. 1, 2015, doi: 10.1088/1749-4699/8/1/014008.

[31] P. Liashchynskyi and P. Liashchynskyi, "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS," no. 2017, pp. 1–11, 2019.

[32] L. Villalobos-Arias, C. Quesada-López, J. Guevara-Coto, A. Mart\'\inez, and M. Jenkins, "Evaluating Hyper-Parameter Tuning Using Random Search in Support Vector Machines for Software Effort Estimation," in *Proceedings of the 16th ACM International Conference on Predictive Models and Data Analytics in Software Engineering*, 2020, pp. 31–40. doi: 10.1145/3416508.3417121.

[33] L. Villalobos-Arias and C. Quesada-López, "Comparative study of random search hyper-parameter tuning for software effort estimation," in *Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering*, Aug. 2021, pp. 21–29. doi: 10.1145/3475960.3475986.

[34] S. Putatunda and K. Rama, "A Comparative Analysis of Hyperopt as Against Other Approaches for Hyper-Parameter Optimization of XGBoost," in *Proceedings of the 2018 International Conference on Signal Processing and Machine Learning - SPML '18*, 2018, pp. 6–10. doi: 10.1145/3297067.3297080.

[35] J. Joy and M. P. Selvan, "A comprehensive study on the performance of different Multi-class Classification Algorithms and Hyperparameter Tuning Techniques using Optuna," *Proc. Int. Conf. Comput. Commun. Secur. Intell. Syst. IC3SIS 2022*, 2022, doi: 10.1109/IC3SIS54991.2022.9885695.