International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# Optimizing Machine Learning Models: An Adaptive Hyperparameter Tuning Approach

**Pavitha N.*[1], Shounak Sugave[2]**

**Submitted**: 08/05/2023 **Revised**: 15/07/2023 **Accepted**: 06/08/2023

**Abstract:** Hyperparameter optimization is a critical task in enhancing machine learning model performance. This paper introduces a novel approach, for hyperparameter tuning without making assumptions about the underlying hyperparameter distribution or convergence behavior. This approach treats hyperparameter configurations as indices and focuses solely on the associated loss sequences. The objective is to efficiently search for the configuration by minimizing the selected configuration's validation error. The algorithm employs an acquisition function to determine the next configuration to evaluate and leverages a classification model to guide the search process. The proposed methodology is agnostic to the structure of hyperparameter relationships, aiming to strike a balance between resource usage and performance improvement. Experimental results demonstrate the proposed approach's effectiveness in identifying optimal configurations while being adaptive to various domains and data types.

***Keywords:*** *Hyperparameter optimization, Machine learning, Resource allocation, Acquisition function, Performance enhancement*

## 1. Introduction

In recent years, the application of machine learning (ML) algorithms [1] has become ubiquitous across various domains, from healthcare [2] and finance [3] to image recognition [4] and natural language processing [5]. These algorithms offer the capability to extract intricate patterns and insights from large datasets [6], [7], enabling informed decision-making and predictive analytics [8]. However, the efficacy of ML models heavily relies on the careful selection of hyperparameters [9], configuration settings that dictate the learning process and influence the model's generalization ability. [10]

Hyperparameter tuning is a critical aspect of ML model development that involves the optimization [11], [12] of these configuration settings. Proper tuning can significantly impact model performance, enabling enhanced accuracy, convergence, and generalization. As hyperparameters control [13], [14] the trade-off between model complexity and generalization, their selection warrants careful consideration to strike the optimal balance [15].

This research delves into the paramount importance of hyperparameter tuning in the realm of machine learning. By systematically adjusting hyperparameters, models can be fine-tuned to achieve optimal performance [16], [17] on specific tasks and datasets [18], [19]. The objective of this study is to evaluate the impact of hyperparameter tuning on the performance of various ML algorithms using real-world datasets from the financial sector, specifically sourced from a private bank and a Non-Banking Financial Company (NBFC) operating in India.

In this context, hyperparameter tuning can be seen as a mechanism to navigate the complex landscape of algorithmic configurations, improving model accuracy, convergence, and efficiency. The investigation presented in this paper focuses on elucidating the effects [20] of hyperparameter optimization on different ML algorithms, thereby contributing to a deeper understanding of their behavior [21], [22] and the extent to which tuning influences their performance. [23], [24] By shedding light on the relationship between hyperparameter tuning and ML model performance, this research aims to contribute valuable insights to the broader field of machine learning optimization and inspire advancements in algorithmic development for enhanced predictive analytics.

### 1.1. Hyperparameter Tuning Strategies

Hyperparameter tuning has been explored through various optimization strategies. Grid Search [18], [25], [26], a systematic search approach, explores predefined hyperparameter values across a grid. Random Search, an alternative, randomly samples hyperparameters to achieve a trade-off between exploration and exploitation. Bayesian Optimization [27] leverages probabilistic models to estimate the utility of different hyperparameters, adapting the search based on previous observations. Genetic Algorithms mimic biological evolution to iteratively evolve hyperparameter configurations, while Particle Swarm Optimization simulates social behavior to optimize parameters collectively. These strategies collectively

[1] *PhD Research Scholar at Dr. Vishwanath Karad MIT World Peace University, Pune, India.*
*and Assistant Professor at Vishwakarma University, Pune, India.*
*ORCID ID: 0000-0002-0577-8722*
[2] *Dr. Vishwanath Karad MIT World Peace University, Pune, India*
*ORCID ID: 0000-0002-2643-8751*
** Corresponding Author Email: pavithanrai@gmail.com*

constitute a toolkit for systematically navigating the complex space of hyperparameters. [28]

### 1.2. Impact of Hyperparameter Tuning

Hyperparameter tuning has been demonstrated to have a substantial impact on model performance across a range of ML algorithms. Researchers showed that hyperparameter optimization can lead to significant improvements in accuracy for Support Vector Machines (SVMs) and neural networks. [29] Similarly, the studies also demonstrated how hyperparameter optimization outperforms manual tuning for SVMs, k-Nearest Neighbors, and Random Forests. [19] These findings underscore the effectiveness of automated tuning methods in enhancing model accuracy.

### 1.3. Algorithm-Specific Hyperparameter Tuning

Different ML algorithms exhibit varying sensitivities to hyperparameter settings. For instance, in neural networks [30], [31] [31], the learning rate significantly impacts convergence speed and optimization quality. A higher learning rate may lead to faster convergence but risk overshooting the optimal solution, while a lower learning rate may converge slowly or get trapped in local minima. Researchers also emphasized the importance of tuning learning rates for neural networks, demonstrating its impact on convergence and generalization [32] [33]. In decision tree-based algorithms like Random Forest and Gradient Boosting, hyperparameters control tree depth, number of trees, and splitting criteria.

### 1.4. Resource Allocation and Hyperparameter Tuning

Hyperparameter tuning interacts closely with resource allocation, as both involve optimizing the utilization of limited computational resources. Researchers introduced the concept of "budgeted optimization," where resource allocation is strategically managed during hyperparameter optimization. [34], [35] This approach maximizes the efficiency of resource utilization, ensuring optimal hyperparameter configuration discovery within a specified resource budget. [12], [36], [37]

### 1.5. Transfer Learning and Hyperparameter Tuning

Transfer learning techniques have been integrated with hyperparameter tuning to leverage knowledge gained from one task to improve performance on another. Researchers demonstrated how transfer learning can be applied to transfer information across datasets, optimizing hyperparameters efficiently [27]. This approach has practical implications in scenarios where labeled data is scarce, enabling the effective tuning of models with limited samples.

Despite its importance, hyperparameter tuning presents challenges. The hyperparameter space can be vast, making exhaustive search impractical. Additionally, the interaction between hyperparameters can be intricate, requiring sophisticated optimization [12], [38] methods. Open questions persist about the robustness of tuning methods across diverse datasets and algorithms, as well as the incorporation of domain knowledge into the tuning process.

## 2. Methodology

Overall methodology followed in this research process is shown in figure 1.

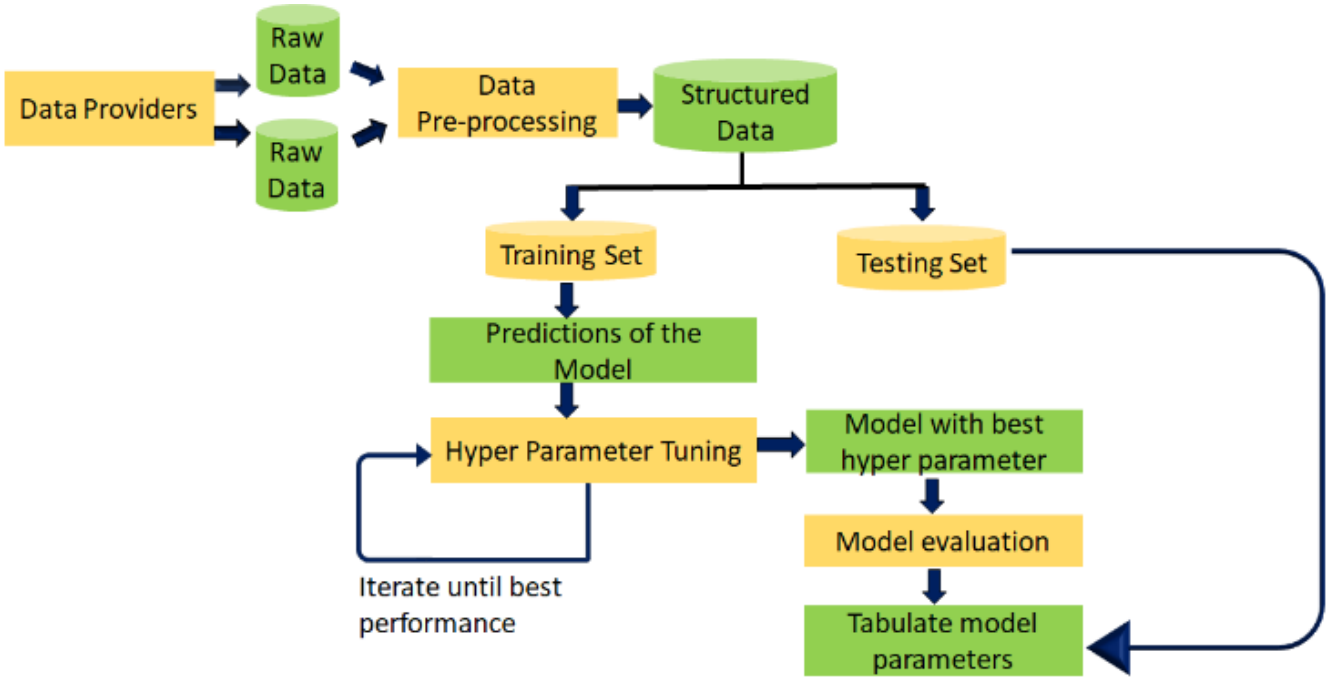


**Fig. 1.** Research Methodology

### 2.1. Data Collection and Preprocessing

Data is sourced from a private bank and a Non-Banking Financial Company (NBFC) operating in India by signing NDA. These organizations provide anonymized customer data including demographics (age, gender, location), financial transactions (credit/debit, amounts), credit histories (credit scores, loan histories), and loan applications (approved/denied, loan type). This diverse dataset captures various financial behaviors and customer profiles, crucial for training and evaluating machine learning models.

Before model development, the data undergoes preprocessing to ensure quality and consistency. Missing values are addressed by imputing them using relevant techniques. For instance, missing age values could be imputed with median age, while missing credit scores might be imputed based on regression models using correlated features. Feature engineering involves creating new features like debt-to-income ratios, credit utilization percentages, and time since last loan application, which offer deeper insights to the models.

Categorical variables like loan type or gender are encoded into numerical formats. One-hot encoding is used for nominal variables (e.g., loan types), creating binary columns for each category. For ordinal variables (e.g., education level), label encoding is applied to represent their order. Outliers are detected using statistical methods like Z-score or IQR and are either removed or transformed to minimize their impact on model training.

The preprocessed dataset is divided into training and testing sets. An 80:20 split is chosen to allocate a larger portion for training while ensuring a substantial portion for unbiased model evaluation. The training set is used to develop and optimize the models, whereas the testing set simulates real-world scenarios, gauging how well the models generalize to new, unseen data.

### 2.2. Hyperparameter Tuning

In the realm of optimizing hyperparameters, let's delve into a space of feasible hyperparameter setups denoted as X. This space covers a spectrum of variables—continuous, discrete, or categorical—and they can interdepend in flexible ways, not bound by specific ranges like [0, 1]. For each value of k, starting from 1 and beyond, we're equipped with a series of loss functions, denoted as $l_k: X \rightarrow [0, 1]$. These functions quantify the validation error of a model, parameterized by x, employing k units of resources (e.g., iterations).

In this landscape, we make the assumption that hyperparameter configurations are selected at random according to a known probability distribution, represented by $p(x): X \rightarrow [0, \infty)$. Although we lean towards a basic uniform distribution for p(x) in our experiments, the approach accommodates any sampling technique. Given a random pick $X \in X$ from this distribution, the value $l_*(X)$ transforms into a random variable, encompassing an unknown distribution. This opacity arises from our lack of insight into the genuine $l_*(\cdot)$ function. Moreover, deciphering information about $l_k(x)$ through the lens of $l_j(y)$ for any $j \in N$ and $y \in X$ is elusive, as the inner workings of hyperparameters and their effects on loss functions remain concealed.

As a consequence, we simplify the task of hyperparameter optimization by engaging with hyperparameter configurations $x \in X$ solely through their corresponding sequences of loss, denoted as $l_k(x)$ for various k. The specific identity of $x \in X$ functions as a mere reference or tag for the loss sequence it represents. Lacking knowledge about the speed at which $l_k(\cdot)$ converges to $l_*(\cdot)$, or the distribution of $l_*(X)$, Hyperband's aim is to pinpoint a hyperparameter configuration $x \in X$ that minimizes $l_*(x)$ by a specified margin $\nu_*$, a constant defined by the user. This endeavor entails selecting a suitable number of random configurations, all the while optimizing resource utilization.

In essence, Hyperband aims to find the best hyperparameter configuration in the given space X based solely on its loss sequence, disregarding any underlying hyperparameter structure or distribution. The goal is to efficiently search for the configuration that minimizes the gap between the validation error of the selected configuration and the user-defined constant $\nu_*$, without making any assumptions about the loss function convergence or distribution.

**Input:**

- Objective function: f(x) (where x represents the hyperparameter configuration)
- Hyperparameter search space: X
- Number of iterations: T
- Acquisition function: A(x)

**Algorithm:**

Step 1: Initialize a dataset D = {(x_i, y_i)} with a few initial hyperparameter configurations (random or predefined).

Step 2: for t = 1 to T do:

2.1: Fit a Classification model to the dataset D.

2.2: Determine the next hyperparameter configuration to evaluate by maximizing the acquisition function: x_t = argmax A(x | D).

2.3: Evaluate the objective function f(x_t) to obtain the corresponding performance metric y_t.

2.4: Add (x_t, y_t) to the dataset: D = D ∪ {(x_t, y_t)}.

Step 3: Return the hyperparameter configuration with the highest observed performance: x* = argmax y_i, where (x_i, y_i) ∈ D.

**Output:**

• Set of optimal hyperparameters for the model

### 2.3. Model Selection and performance evaluation

Numerous instances of each model variant are trained with distinct hyperparameters. Validation metrics, such as accuracy, precision, recall, and F1-score, are employed to assess the model's performance. These metrics shed light on the model's proficiency in accurately categorizing various classes and handling imbalances. The model exhibiting the most favorable hyperparameters, as gauged by these metrics, is designated for further evaluation.

The selected model is subjected to comprehensive evaluation on the testing set. Performance metrics, including accuracy, precision-recall values, offer a detailed understanding of the model's predictive capabilities. Finally, the key parameters and characteristics of the selected model are tabulated for reference and documentation. This includes a summary of the hyperparameters chosen through optimization, the model's architecture and features that proved most influential, and insights gained from the evaluation process. This tabulation serves as a valuable resource for understanding the model's configuration, performance, and potential future improvements.

## 3. Results and Discussion

**Table 1.** Performance of Machine Learning Algorithm on Data set 1

| Algorithm | Accuracy (before hyper parameter tuning) | Accuracy (after hyper parameter tuning) | Loss (before hyper parameter tuning) | Loss (after hyper parameter tuning) |
|---|---|---|---|---|
| Gaussian NB | 0.84168 | 0.85652 | 0.28097 | 0.14231 |
| Logistic Regression | 0.94168 | 0.95668 | 0.14969 | 0.13469 |
| Extra Trees Classifier | 0.94128 | 0.95628 | 0.15969 | 0.14469 |
| Random Forest Classifier | 0.94184 | 0.95684 | 0.13370 | 0.1187 |
| XGB Classifier | 0.9542 | 0.96920 | 0.07955 | 0.06455 |
| LGBM Classifier | 0.95426 | 0.96926 | 0.07775 | 0.06275 |
| Neural Network | 0.94138 | 0.95638 | 0.19968 | 0.18468 |

In this discussion, we will analyze the performance of various classification algorithms on Data set1. The detailed results are shown in table 1 and visually represented in figure 2. Gaussian NB is a simple probabilistic classifier based on Bayes' theorem with the assumption of feature independence. It performs reasonably well on the dataset with an accuracy of 0.84168 before tuning. However, after hyperparameter tuning, the accuracy increases to 0.85652, indicating that the tuning process led to improved model performance. Additionally, the loss decreased significantly from 0.28097 to 0.14231, indicating better convergence during training. Logistic Regression is a linear classification algorithm that models the probability of a binary outcome. It performs quite well on the dataset with an accuracy of 0.94168 before tuning. After hyperparameter tuning, the accuracy improves to 0.95668, indicating that the tuning process further optimized the model. The loss also reduced from 0.14969 to 0.13469, suggesting better convergence.

Extra Trees Classifier is an ensemble method based on decision trees with random feature selection. It achieves an accuracy of 0.94128 before tuning, which is already relatively high. After hyperparameter tuning, the accuracy increases to 0.95628, showing that tuning further boosted its performance. The loss also decreased from 0.15969 to 0.14469, indicating better convergence during training. Random Forest is another ensemble method based on decision trees, utilizing bootstrapping and random feature selection. It performs well on the dataset with an accuracy of 0.94184 before tuning. After hyperparameter tuning, the accuracy increases to 0.95684, indicating that the tuning process improved the model's performance. The loss

decreased from 0.13370 to 0.1187, suggesting better convergence during training.

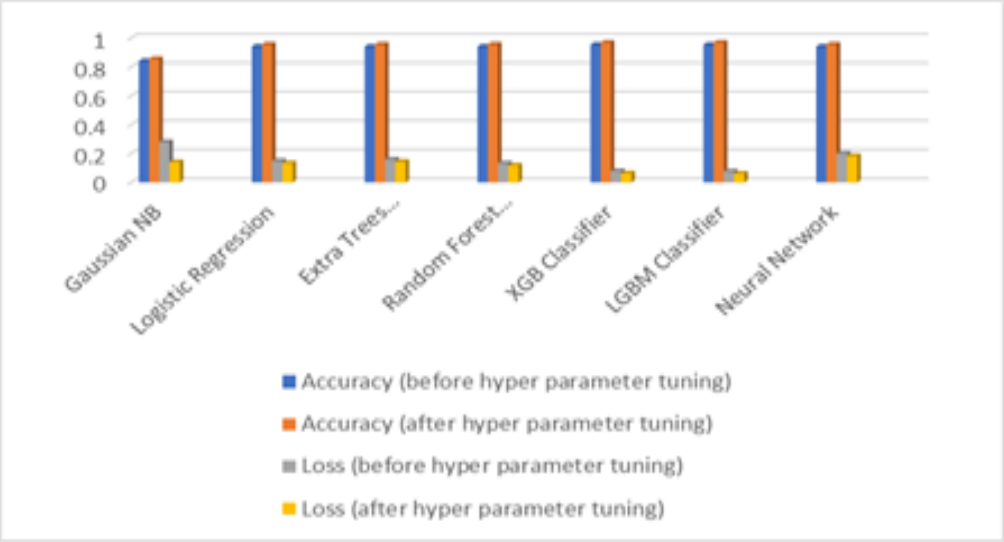


**Fig. 2.** Performance on Data set 1

**Table 2.** Performance of Machine Learning Algorithm on Data set 2

| Algorithm | Accuracy (before hyper parameter tuning) | Accuracy (after hyper parameter tuning) | Loss (before hyper parameter tuning) | Loss (after hyper parameter tuning) |
|---|---|---|---|---|
| Gaussian NB | 0.91423 | 0.92986 | 0.26217 | 0.31292 |
| Logistic Regression | 0.93806 | 0.95306 | 0.26217 | 0.17717 |
| Extra Trees Classifier | 0.93903 | 0.95403 | 0.24917 | 0.16417 |
| Random Forest Classifier | 0.93988 | 0.95488 | 0.23426 | 0.14926 |
| XGB Classifier | 0.95671 | 0.97171 | 0.15639 | 0.07139 |
| LGBM Classifier | 0.96327 | 0.97827 | 0.1439 | 0.05890 |
| Neural Network | 0.93903 | 0.95403 | 0.40291 | 0.31791 |

XGBoost is a powerful gradient boosting algorithm known for its high performance. It already exhibits excellent accuracy of 0.9542 before tuning. After hyperparameter tuning, the accuracy further increases to an impressive 0.96920, indicating that the tuning process significantly improved its performance. The loss also decreased from 0.07955 to 0.06455, suggesting better convergence during training. LightGBM is another gradient boosting algorithm that uses a novel technique to improve training speed and efficiency. It demonstrates excellent performance with an accuracy of 0.95426 before tuning. After hyperparameter tuning, the accuracy increases to 0.96926, indicating that the tuning process further optimized the model. The loss also decreased from 0.07775 to 0.06275, suggesting better convergence during training.

Neural networks are powerful models capable of learning complex patterns in data. The neural network in this case achieves an accuracy of 0.94138 before tuning, which is already quite good. After hyperparameter tuning, the accuracy increases to 0.95638, indicating that the tuning process improved the model's performance. The loss also decreased from 0.19968 to 0.18468, suggesting better convergence during training. Overall, the performance comparison of the classification algorithms before and after hyperparameter tuning reveals that hyperparameter tuning significantly improved the models' accuracy and reduced their loss. These results demonstrate the effectiveness of hyperparameter tuning in optimizing the performance of machine learning models and highlight the importance of choosing the right algorithm and tuning its parameters for the given dataset. Similarly, on dataset2 also accuracy increased and loss reduced. Detailed results are shown in the table 2 and also visually represented in figure 3.

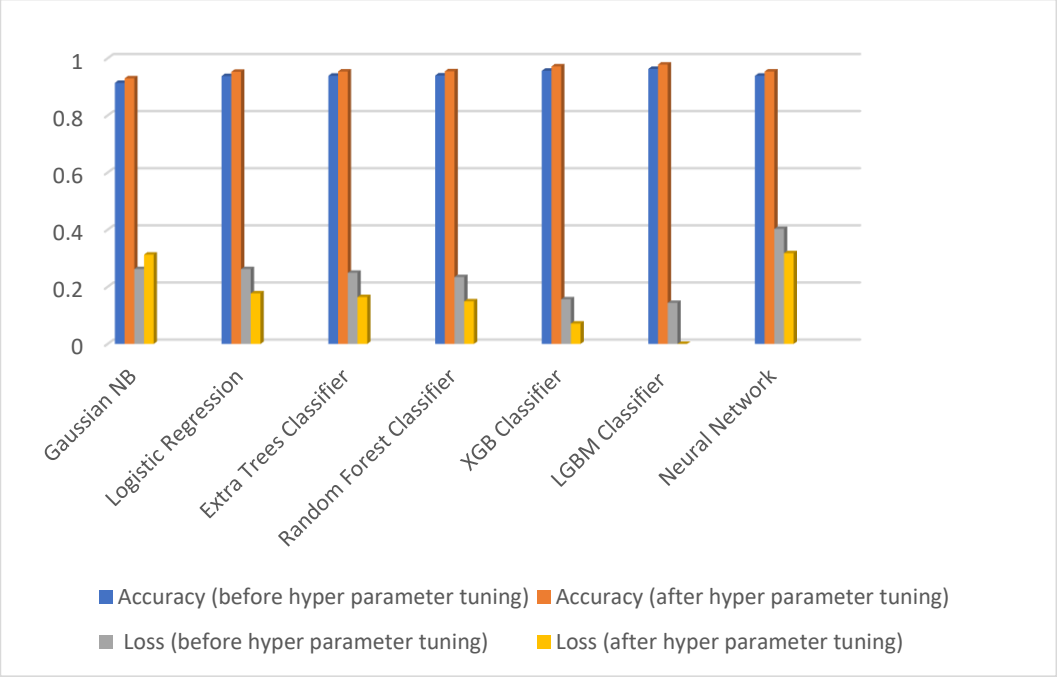


**Fig. 3**. Performance on Data set 2

The results of experimentation demonstrated a consistent pattern of improvement, with increased accuracy and reduced log loss observed for all models following hyperparameter tuning. Our findings (shown in figure 4 and figure 5) indicate that the fine-tuning of hyperparameters led to notable improvements in model accuracy. This suggests that the default hyperparameter settings provided by the model libraries may not always yield the best results for a given dataset. By customizing the hyperparameters to better fit the characteristics of the data, our models were able to capture more intricate patterns and relationships, resulting in higher accuracy.

Through the process of hyperparameter tuning, we were able to consistently decrease the log loss for all models. This reduction implies that the tuned models produced more confident and precise probability estimates, aligning closely with the ground truth labels. Consequently, the overall uncertainty and errors in the model predictions were minimized, contributing to improved model robustness. The results are shown in figure 6 and 7 for data set 1 and 2 respectively.

It is worth noting that different models may exhibit varying degrees of sensitivity to specific hyperparameters. For instance, parameters related to regularization, learning rates, and batch sizes can significantly impact the performance of neural network-based models. On the other hand, tree-based models such as Random Forest and Gradient Boosting Trees may show sensitivity to tree depth, number of estimators, and feature importance settings. The effectiveness of hyperparameter tuning is influenced by the complex interplay between these parameters and their effects on model behavior.

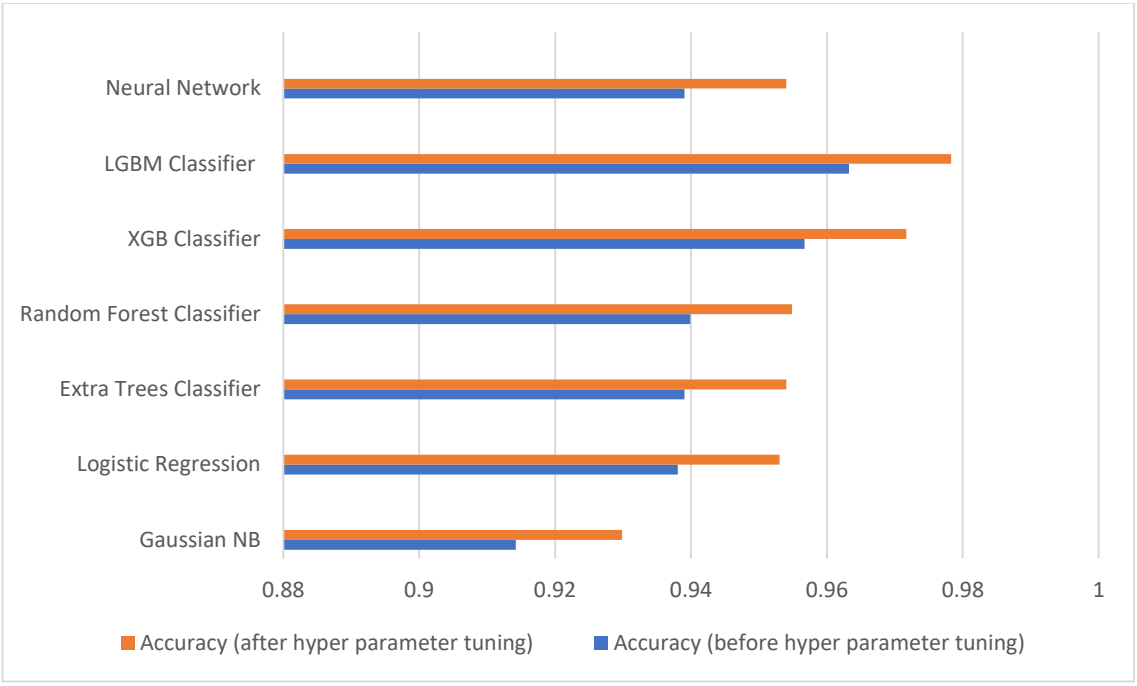


**Fig. 4**. Accuracy after and before tuning hyper parameters on data set1

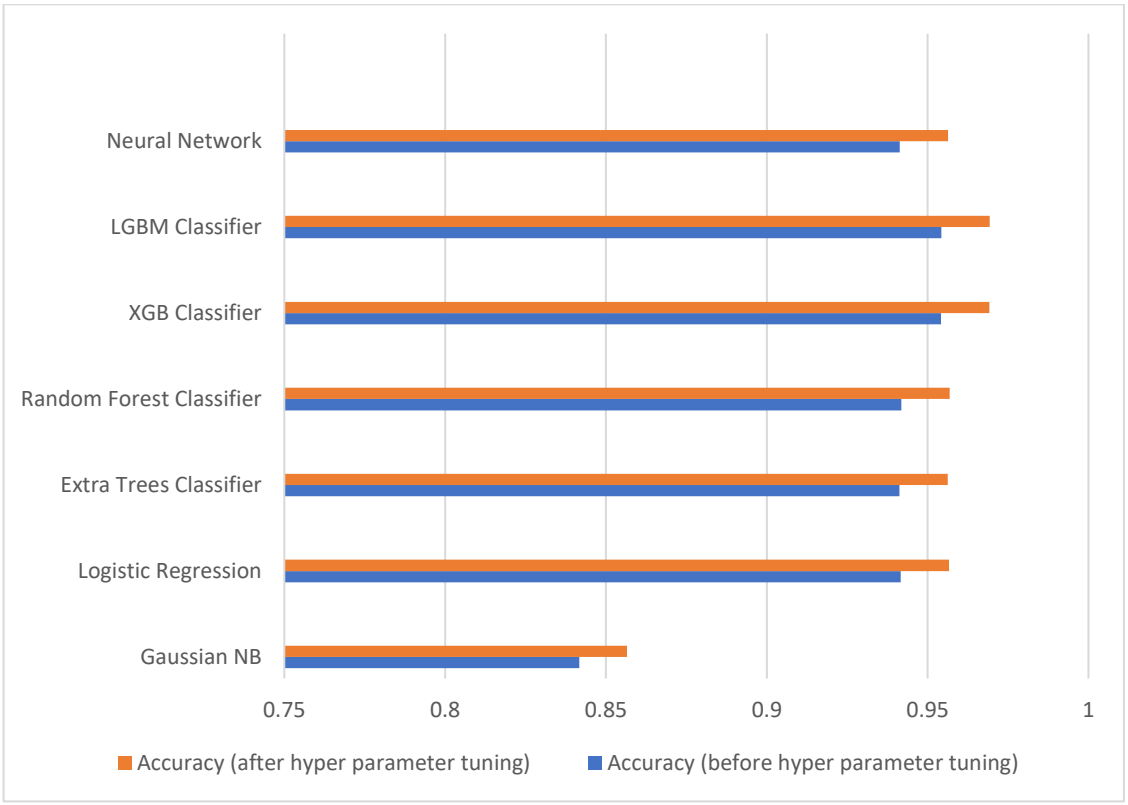


**Fig. 5**. Accuracy after and before tuning hyper parameters on data set2

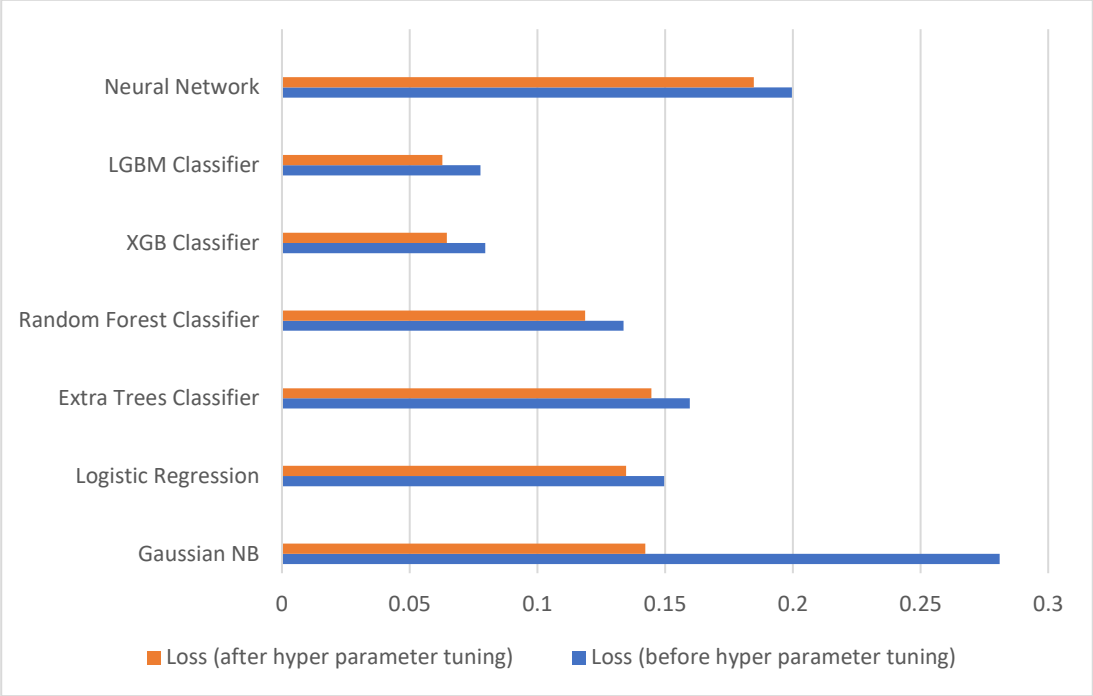


**Fig. 6.** Loss after and before tuning hyper parameters on data set1

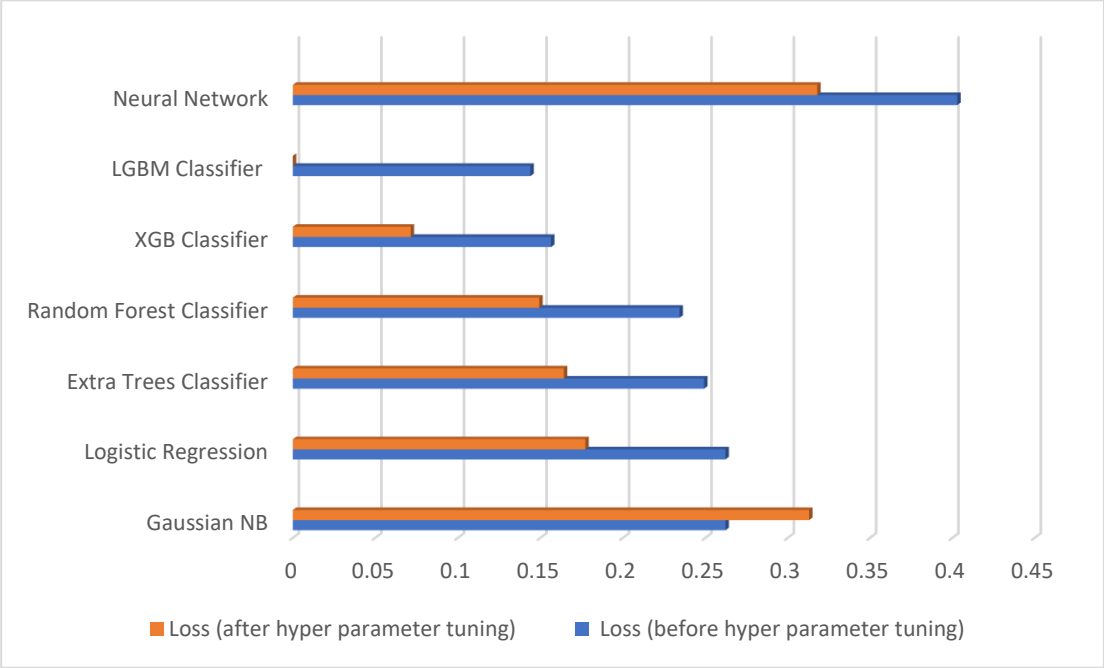


**Fig. 7.** Loss after and before tuning hyper parameters on data set2

## 4. Conclusion

In this study, we conducted a comprehensive analysis of the performance of various machine learning algorithms on two distinct datasets, emphasizing the impact of hyperparameter tuning on their effectiveness. The aim was to assess how different algorithms responded to hyperparameter optimization and to highlight the significance of parameter tuning in enhancing model performance.

Our findings provide valuable insights into the improvements achieved through hyperparameter tuning. The initial performance of the algorithms on both Dataset 1 and Dataset 2 was notable, yet the application of hyperparameter tuning consistently led to substantial enhancements in accuracy and reductions in loss across the board. This underscores the importance of fine-tuning model configurations to achieve optimal results.Among the algorithms tested, Gaussian NB, Logistic Regression, Extra Trees Classifier, Random Forest Classifier, XGB Classifier, LGBM Classifier, and Neural Network, the gradient

boosting techniques (XGB and LGBM) consistently demonstrated remarkable performance, showcasing their robustness in capturing intricate relationships within the data. Moreover, hyperparameter tuning played a pivotal role in elevating their accuracy and convergence speed, reaffirming the value of optimization.

These results emphasize that the choice of algorithm is pivotal, but the tuning of hyperparameters significantly influences the final outcome. It is important to acknowledge that the extent of improvement varied across algorithms, indicating that while hyperparameter tuning is beneficial, it can be more pronounced for some algorithms compared to others.

The observed improvements in accuracy and convergence illustrate the potential for achieving more accurate and efficient models through systematic hyperparameter optimization. As machine learning continues to advance, future research should explore additional algorithms, optimization techniques, and strategies to uncover further insights that can lead to even more effective model configurations. Ultimately, this research underscores the iterative nature of machine learning model development, where experimentation and refinement of hyperparameters lead to models that can better capture the complexities of real-world data and make more accurate predictions.

**Author contributions**

**Pavitha N:** Conceptualization, Data collection Methodology, Implementation, Writing-Original draft preparation

**Shounak Sugave:** Investigation, Writing-Reviewing and Editing.

**Conflicts of interest**

The authors declare no conflicts of interest.

## References


[1] S. Jaiswal and C. M. Balasubramanian, "An advanced deep learning model for maneuver prediction in real-time systems using alarming-based hunting optimization," *International Journal of Advances in Intelligent Informatics*, vol. 9, no. 2, 2023, doi: 10.26555/ijain.v9i2.1048.

[2] A. K. Bitto, Md. H. I. Bijoy, S. Yesmin, I. Mahmud, Md. J. Mia, and K. B. B. Biplob, "Tumor-Net: convolutional neural network modeling for classifying brain tumors from MRI images," *International Journal of Advances in Intelligent Informatics*, vol. 9, no. 2, 2023, doi: 10.26555/ijain.v9i2.872.

[3] K. Kohv and O. Lukason, "What best predicts corporate bank loan defaults? An analysis of three different variable domains," *Risks*, vol. 9, no. 2, pp. 1–19, Feb. 2021, doi: 10.3390/risks9020029.

[4] S. Kumar and S. Ratnoo, "Multi-objective hyperparameter tuning of classifiers for disease diagnosis," *Indian Journal of Computer Science and Engineering*, vol. 12, no. 5, 2021, doi: 10.21817/INDJCSE/2021/V12I5/211205081.

[5] I. Jamaleddyn, R. El ayachi, and M. Biniz, "An improved approach to Arabic news classification based on hyperparameter tuning of machine learning algorithms," *Journal of Engineering Research*, vol. 11, no. 2, 2023, doi: 10.1016/j.jer.2023.100061.

[6] M. Papouskova and P. Hajek, "Two-stage consumer credit risk modelling using heterogeneous ensemble learning," *Decis Support Syst*, vol. 118, pp. 33–45, Mar. 2019, doi: 10.1016/j.dss.2019.01.002.

[7] S. Guo, H. He, and X. Huang, "A Multi-Stage Self-Adaptive Classifier Ensemble Model With Application in Credit Scoring," *IEEE Access*, vol. 7, pp. 78549–78559, 2019, doi: 10.1109/ACCESS.2019.2922676.

[8] M. Jervis, M. Liu, and R. Smith, "Deep learning network optimization and hyperparameter tuning for seismic lithofacies classification," *Leading Edge*, vol. 40, no. 7, 2021, doi: 10.1190/tle40070514.1.

[9] L. Wen, X. Ye, and L. Gao, "A new automatic machine learning based hyperparameter optimization for workpiece quality prediction," *Measurement and Control (United Kingdom)*, vol. 53, no. 7–8, 2020, doi: 10.1177/0020294020932347.

[10] M. Matulis and C. Harvey, "A robot arm digital twin utilising reinforcement learning," *Computers and Graphics (Pergamon)*, vol. 95, 2021, doi: 10.1016/j.cag.2021.01.011.

[11] M. Zhang, H. Li, S. Pan, J. Lyu, S. Ling, and S. Su, "Convolutional Neural Networks-Based Lung Nodule Classification: A Surrogate-Assisted Evolutionary Algorithm for Hyperparameter Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, 2021, doi: 10.1109/TEVC.2021.3060833.

[12] M. Shahhosseini, G. Hu, and H. Pham, "Optimizing ensemble weights and hyperparameters of machine learning models for regression problems," *Machine Learning with Applications*, vol. 7, 2022, doi: 10.1016/j.mlwa.2022.100251.

[13] N. Bakhashwain and A. Sagheer, "Online Tuning of Hyperparameters in Deep LSTM for Time Series Applications," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 1, 2020, doi: 10.22266/IJIES2021.0228.21.

[14] U. K. Mohamad Yusof, Syamsiah Mashohor,

Marsyita Hanafi, Sabariah Md Noor, and Norsafina Zainal, "Hyperparameter Tuning in Deep Learning Approach for Classification of Classical Myeloproliferative Neoplasm," *Malaysian Journal of Science and Advanced Technology*, 2022, doi: 10.56532/mjsat.v2i3.64.

[15] F. Farhangi, "Investigating the role of data preprocessing, hyperparameters tuning, and type of machine learning algorithm in the improvement of drowsy EEG signal modeling," *Intelligent Systems with Applications*, vol. 15, 2022, doi: 10.1016/j.iswa.2022.200100.

[16] E. K. Hashi and Md. Shahid Uz Zaman, "Developing a Hyperparameter Tuning Based Machine Learning Approach of Heart Disease Prediction," *Journal of Applied Science & Process Engineering*, vol. 7, no. 2, 2020, doi: 10.33736/jaspe.2639.2020.

[17] T. Marlaithong, V. C. Barroso, and P. Phunchongharn, "A hyperparameter tuning approach for an online log parser," in *ECTI-CON 2021 - 2021 18th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology: Smart Electrical System and Technology, Proceedings*, 2021. doi: 10.1109/ECTI-CON51831.2021.9454924.

[18] S. Sah, B. Surendiran, R. Dhanalakshmi, and M. Yamin, "Covid-19 cases prediction using SARIMAX Model by tuning hyperparameter through grid search cross-validation approach," *Expert Syst*, 2022, doi: 10.1111/exsy.13086.

[19] M. Daviran, A. Maghsoudi, R. Ghezelbash, and B. Pradhan, "A new strategy for spatial predictive mapping of mineral prospectivity: Automated hyperparameter tuning of random forest approach," *Comput Geosci*, vol. 148, 2021, doi: 10.1016/j.cageo.2021.104688.

[20] A. Panichella, "A Systematic Comparison of search-Based approaches for LDA hyperparameter tuning," *Inf Softw Technol*, vol. 130, 2021, doi: 10.1016/j.infsof.2020.106411.

[21] K. Maass, A. Aravkin, and M. Kim, "A hyperparameter-tuning approach to automated inverse planning," *Med Phys*, vol. 49, no. 5, 2022, doi: 10.1002/mp.15557.

[22] K. Shankar *et al.*, "An Automated Hyperparameter Tuning Recurrent Neural Network Model for Fruit Classification," *Mathematics*, vol. 10, no. 13, 2022, doi: 10.3390/math10132358.

[23] M. A. Amirabadi, M. H. Kahaei, and S. A. Nezamalhosseini, "Novel suboptimal approaches for hyperparameter tuning of deep neural network [under the shelf of optical communication]," *Physical Communication*, vol. 41, 2020, doi: 10.1016/j.phycom.2020.101057.

[24] Z. Czako, G. Sebestyen, and A. Hangan, "AutomaticAI – A hybrid approach for automatic artificial intelligence algorithm selection and hyperparameter tuning," *Expert Syst Appl*, vol. 182, 2021, doi: 10.1016/j.eswa.2021.115225.

[25] C. G. Siji George and B. Sumathi, "Grid search tuning of hyperparameters in random forest classifier for customer feedback sentiment prediction," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 9, 2020, doi: 10.14569/IJACSA.2020.0110920.

[26] R. Ghawi and J. Pfeffer, "Efficient Hyperparameter Tuning with Grid Search for Text Categorization using kNN Approach with BM25 Similarity," *Open Computer Science*, vol. 9, no. 1, 2019, doi: 10.1515/comp-2019-0011.

[27] H. T. Vo, H. T. Ngoc, and L. Da Quach, "An Approach to Hyperparameter Tuning in Transfer Learning for Driver Drowsiness Detection Based on Bayesian Optimization and Random Search," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 4, 2023, doi: 10.14569/IJACSA.2023.0140492.

[28] F. Arden and C. Safitri, "Hyperparameter Tuning Algorithm Comparison with Machine Learning Algorithms," in *Proceeding - 6th International Conference on Information Technology, Information Systems and Electrical Engineering: Applying Data Sciences and Artificial Intelligence Technologies for Environmental Sustainability, ICITISEE 2022*, 2022. doi: 10.1109/ICITISEE57756.2022.10057630.

[29] S. Mezzah and A. Tari, "Practical hyperparameters tuning of convolutional neural networks for EEG emotional features classification," *Intelligent Systems with Applications*, vol. 18, 2023, doi: 10.1016/j.iswa.2023.200212.

[30] G. Atteia, N. Abdel Samee, E. S. M. El-Kenawy, and A. Ibrahim, "CNN-Hyperparameter Optimization for Diabetic Maculopathy Diagnosis in Optical Coherence Tomography and Fundus Retinography," *Mathematics*, vol. 10, no. 18, 2022, doi: 10.3390/math10183274.

[31] W. H. Nugroho, S. Handoyo, H. C. Hsieh, Y. J. Akri, Zuraidah, and D. DwinitaAdelia, "Modeling Multioutput Response Uses Ridge Regression and MLP Neural Network with Tuning Hyperparameter through Cross Validation," *International Journal of*

*Advanced Computer Science and Applications*, vol. 13, no. 9, 2022, doi: 10.14569/IJACSA.2022.0130992.

[32] A. L. C. Ottoni and M. S. Novo, "A Deep Learning Approach to Vegetation Images Recognition in Buildings: A Hyperparameter Tuning Case Study," *IEEE Latin America Transactions*, vol. 19, no. 12, 2021, doi: 10.1109/TLA.2021.9480148.

[33] M. Kim, "The generalized extreme learning machines: Tuning hyperparameters and limiting approach for the Moore–Penrose generalized inverse," *Neural Networks*, vol. 144, 2021, doi: 10.1016/j.neunet.2021.09.008.

[34] C. Y. Wang, C. Y. Huang, and Y. H. Chiang, "Solutions of Feature and Hyperparameter Model Selection in the Intelligent Manufacturing," *Processes*, vol. 10, no. 5, 2022, doi: 10.3390/pr10050862.

[35] H.-C. Cheng, C.-L. Ma, and Y.-L. Liu, "Development of ANN-Based Warpage Prediction Model for FCCSP via Subdomain Sampling and Taguchi Hyperparameter Optimization," *Micromachines (Basel)*, vol. 14, no. 7, 2023, doi: 10.3390/mi14071325.

[36] J. Lorraine, P. Vicol, and D. Duvenaud, "Optimizing Millions of Hyperparameters by Implicit Differentiation," in *Proceedings of Machine Learning Research*, 2020.

[37] L. Zahedi, F. G. Mohammadi, M. H. Amini, and M. H. Amini, "OptABC: An Optimal Hyperparameter Tuning Approach for Machine Learning Algorithms," in *Proceedings - 20th IEEE International Conference on Machine Learning and Applications, ICMLA 2021*, 2021. doi: 10.1109/ICMLA52953.2021.00186.

[38] P. S. Pravin, J. Z. M. Tan, K. S. Yap, and Z. Wu, "Hyperparameter optimization strategies for machine learning-based stochastic energy efficient scheduling in cyber-physical production systems," *Digital Chemical Engineering*, vol. 4, 2022, doi: 10.1016/j.dche.2022.100047.

[39] Srivastava, A. ., & Kumar, A. . (2023). Secure Authentication Scheme for the Internet of Things. International Journal on Recent and Innovation Trends in Computing and Communication, 11(4s), 182–192. https://doi.org/10.17762/ijritcc.v11i4s.6368

[40] Thota, D. S. ., Sangeetha, D. M., & Raj , R. . (2022). Breast Cancer Detection by Feature Extraction and Classification Using Deep Learning Architectures. Research Journal of Computer Systems and Engineering, 3(1), 90–94. Retrieved from https://technicaljournals.org/RJCSE/index.php/journal/article/view/48

[41] Sharma, R., Dhabliya, D. Attacks on transport layer and multi-layer attacks on manet (2019) International Journal of Control and Automation, 12 (6 Special Issue), pp. 5-11.