Proceedings of the International Conference on Inventive Computation Technologies (ICICT-2025)
DVD Part Number: CFP25F70-DVD; ISBN: 979-8-3315-1223-1

Optimization Techniques for Machine Learning Models to improve the efficiency of Classification

J. Anita Smiles
Assistant Professor,
Department of Computer Applications, SRM Institute of Science and Technology,
Chennai, Tamil Nadu, India
anitasmiles7@yahoo.com

M.Sakthivanitha
Assistant Professor
Department of Computer Application
Vels Institute of Science, Technology & Advanced Studies Chennai,
Tamil Nadu, India.
sakthivanithamsc@gmail.com

A. Bharathi
Assistant Professor
Department of Computer Application
Vels Institute of Science, Technology & Advanced Studies
Chennai, Tamil Nadu, India
bharathial15@gmail.com

D. Narayani.
Assistant Professor
Department of Computer Applications
*Vels institute of Science Technology and Advanced Studies (VISTAS),*Tamil Nadu, India.
narayanivistas@gmail.com

S. Sudha
Professor,
Department of Computer Applications, Hindustan Institute of Technology and Science, Chennai, Tamil Nadu, India
sudhas@hindustanuniv.ac.in

M.Mohamed Sirajudeen
Associate Professor, PG and Research Department of Computer Science
Nilgiri College of Arts and Science
(Autonomous),Thaloor-643239 ,
The Nilgiris , Tamil Nadu, India

***Abstract*— The objective of the study is to identify the most optimal set of hyperparameters for a machine learning (ML) or deep learning(DL) algorithms that improves its performance on a certain task.. This study uses five machine learning methods like Decision Tree(DT), Random Forest(RF), Gradient Boost Models(XGBoost), Support Vector Machine(SVMs) and K-Nearest Neighbhor(KNN). The model specific parameters were applied to all these ML methods to improve the accuracy of the models. The ML models performance with its hyperparameter tuning are evaluated for performance using the performance metrics like accuracy, precision, Recall and F-score. These findings indicate that XGBoost models performed significantly in terms of accuracy, precision, recall, and F1-score. Gradient boosting models are extremely adaptable, but they are also sensitive to hyperparameters such as the learning rate, number of estimators, and tree depth. Tuning these parameters can dramatically improve performance. The optimal model and tuning method are determined by the dataset, task specifications, and computing power.. The contribution of the study to suggests a suitable with right hyperparameter settings to develop a highly flexible model that can adapt to a variety of datasets. The study and application of model-specific hyperparameters in ML continues to evolve, resulting to advances that improve productivity, durability, and generalization.**

Keywords—Machine Learning, Hyperparameter, Optimization, Decision Tree(DT), Random Forest(RF), Gradient Boost Models(XGBoost), Support Vector Machine(SVMs) and K-Nearest Neighbhor(KNN).

I. INTRODUCTION

The hyperparameters of a learning algorithm are the values that govern the process of learning and decide the final parameters of the models. The goal of hyperparameter optimization(HPO) is to find the ideal hyperparameter settings so that you can receive good results from data as rapidly as possible. Machine learning(ML) is based on algorithms that adjust to changing settings and improve their respective efficiency over time. Hyperparameter adjustment is a vital step in enhancing the effectiveness as well as performance of ML or Deep Learning(DL) models. Considering the enormous dimensionality of hyperparameter spaces and the computational cost of training deep models, efficient optimization strategies are critical. Properly calibrated hyperparameters have a direct impact on the model's ability to learn patterns from data, generalize to new data, and meet specific criteria for real-world applications. Learning rate, batch size, and regularization strength are all important hyperparameters that affect a model's capacity to converge to an optimal solution[1].

A model that is overly sophisticated or lacks proper regularization works well on training data but poorly on validation/test data (overfitting). When the model is too simplistic or hyperparameters such as learning rate or maximum depth are not properly specified, it fails to reflect the data's complexity (underfitting).Proper hyperparameter adjustment ensures that model intricacy and generalization are balanced. Models with hyperparameters that are optimized can scale efficiently to huge datasets or high-dimensional issues without experiencing severe performance deterioration.

Advanced models such as transformers, convolutional neural networks (CNNs), and gradient boosting machines (GBMs) frequently rely extensively on hyperparameters for performance. Some algorithms (e.g., decision trees, random forests) can be simplified by changing hyperparameters such as the maximum depth or minimum samples per leaf without losing performance, hence enhancing interpretability. Hyperparameters might include model-specific, regularization, feature selection, and optimization parameters. This research focuses on assessing the effectiveness of model-specific hyperparameters (MSHs). MSHs are hyperparameters that are specific to one type of ML or DL algorithm. Hyperparameters influence the model's architecture, behavior, and optimization process, all of which have a substantial impact on performance. Common hyperparameters for models include learning rate, number of layers, number of neurons per layer, and so on. Hyperparameters have a significant impact on how well ML models perform with unseen, out-of-sample data.The study of model-specific hyperparameters is important to significantly increase the efficacy of the models and to build a optimized ML models. The major contributions of the study is to

- Identify and explore the MSH's for ML models to improve the performance of the models
- Evaluate the ML models like DT, RF, XGB, SVM and KNN with and without hyperparameter settings to suggest an effective ML model.

II. Literature Survey

Arnold et al., 2024 demonstrate the risks of ignoring model and tuning transparency when comparing machine learning models' capacity to forecast electoral violence from tweets. Hyperparameter tuning and documentation should be included as normal components of robustness assessments for ML models[2].

Bakere et al., 2024 discuss the critical relevance of hyperparameter optimization in complex machine learning models, specifically image classification tasks. Given the impracticality of manual tuning in the face of increasing complexity, the study thoroughly examines eight automated the optimization methods: grid search, random search, Gaussian process Bayesian optimization (BO), Tree Parzenestimator BO, Hyperband, BO/Hyperband hybrid, genetic algorithms, and particle swarm optimization. Assessments consider a variety of model topologies and performance criteria, including accuracy, mean squared error, and optimization time[3].

Raiaan et al., 2024 investigate a variety of widely utilized strategies, including metaheuristic, statistical, sequential, and numerical approaches, for fine-tuning CNN hyperparameters. Our study provides a comprehensive classification of these HPO techniques and explores into the core notions of CNN, describing the role of hyperparameters and their modifications. Furthermore, an extensive literature study of HPO methods in CNN that use the aforementioned algorithms is conducted[4]

Hyperparameters are crucial for measuring prediction performance in ML models. To avoid extremes, they maintain a balance between overfitting and underfitting of research-independent variables. Manual tuning and automated procedures are used to determine the ideal combination and permutation for the model's performance. Li et al., 2024 investigates the pursuit of the optimum fit using different hyperparameters[5]. Fine-tuning hyperparameters for ML algorithms is a computational difficulty due to the problem space's vast size. It emphasizes MSH's as an important topic of study for both practitioners and scholars[6].

III. Proposed Methodology

Model-specific hyperparameters are variables or configurations that are customized to a particular ML or DL model. unlike generic hyperparameters (such as learning rate and batch size), these specialized parameters are strongly related to the model's basic framework or algorithm and have a direct impact on how the framework learns or acts. Each type of ML model has distinct hyperparameters that affect its performance.

A. Decision Trees

Decision trees are a common ML model for classification and regression tasks. Their efficacy and comprehensibility are largely dependent on the correct calibration of MSH's. The key parameters of DT models are as follows

Maximum depth refers to the maximum number of layers in the tree. A deeper tree enables for more complicated patterns to be captured, but it also increases the risk of overfitting. A smaller depth reduces overfitting but may result in underfitting. Experiment with different settings to find the right mix between bias and variance.

Minimum number of samples needed to split an internal node. Larger values create broader splits, which reduces overfitting but may miss smaller patterns. Smaller values allow for deeper growth, but they may lead to overfitting.

Minimum Samples Leaf. The minimal number of samples necessary for a leaf node. Larger values lead to less splits and simpler trees. Smaller numbers can capture finer information but may result in overfitting. It is to be set in proportion to the dataset's size

Maximum Features: The maximum amount of features used to determine the optimal split. Lower values have lower computational cost and the risk of overfitting, but they may miss significant splits. Larger values offer for greater feature selection options, but they also raise the risk of overfitting.

Criteria: The criteria used to assess the quality of a split. Gini impurity is an often used option. Gini impurity, entropy, and MSE.

Splitter: This approach determines the split at each node. The options include Best and Random.

The maximum number of leaf nodes in a tree restricts the number of terminal nodes, preventing overfitting.

Class Weight: Weights connected with classes to manage class imbalance, which assists in boosting accuracy for minority classes.[7]

B. Random Forests

Random Forests are learning models that aggregate numerous decision trees' forecasts to improve accuracy, durability, and generalizability. A RF model's performance is highly dependent on its hyperparameters.

Gradient Boosting Models Number of Estimators: It represents the total number of decision trees and A bigger number of trees improves performance by lowering variation, but it also increases training time. Too many trees can lead to declining benefits and higher computing cost (0 to 1000).

Maximum Depth: The maximum depth of each tree. Deeper trees allow more complex patterns to be captured but increase the risk of overfitting(5 to 50)

Minimum Samples Split: The smallest number of samples necessary to split an internal node. Larger values keep the tree from oversplitting, which reduces overfitting. Smaller numbers allow the tree to grow deeper, potentially overfitting (2, 5, 10).

Minimum Samples Leaf: The minimal number of samples necessary at each leaf node. Larger values result in simpler trees and assist in avoiding overfitting. Smaller numbers allow capturing finer features, but may lead to overfitting. (1-10)

Maximum Features : The maximum amount of features to evaluate while determining the optimum split. Smaller values minimize the complexity of models and overfitting, but they may miss key characteristics. Larger values add complexity and raise the risk of overfitting.

Bootstrap: Whether to create each tree using bootstrapped samples. Randomness, when set to True (the default), increases tree diversity while decreasing overfitting and set to False, all trees are trained on the same dataset, which reduces randomness while enhancing correlation.

Class Weights: Weights are allocated to classes to address class imbalance. It helps to keep the algorithm from being skewed towards the majority class.

Maximum Leaf Nodes : Reduces the amount of leaf nodes in each tree. It minimises overfitting by reducing the tree structure. It may impede the model's capacity to learn intricate patterns.

Minimum Impurity Decrease : A node will only be divided if the impurity decrease is at least this value . It reduces overfitting by discarding splits with insufficient information gain.

Random State: Description A seed for the random number generator. It assures that the results are reproducible [8,9].

C. Gradient Boosting Models

Gradient Boosting Models (GBMs) are collaborative methods that construct decision trees in a sequential order, with each tree attempting to fix the faults of the preceding one. The hyperparameters unique to GBMs govern the characteristics of individual trees, the boosting process, and overall complexity of the model[10].

Maximum Depth: The maximum depth of a tree. Larger depth allows for the capture of more complicated patterns, but it also increases the risk of overfitting. Deeper trees are faster and less prone to overfitting, but they may also underfit.

Minimum Samples Split / Minimum Data in Leaf: The minimum amount of samples needed to split a node. Larger numbers avoid over-splitting and overfitting, whereas smaller values allow trees to develop deeper and capture more features (2-20).

Number of Leaves: The maximum number of leaves in a tree. More leaves complicate the model and increase the danger of overfitting, whereas fewer leaves simplify the model and reduce the likelihood of overfitting.

Subsample: The proportion of training data utilized to create each tree. Values less than 1 introduce randomization to avoid overfitting, whereas values around 1 employ more data per tree, boosting accuracy but raising overfitting risk (0.5-1).

Learning Rate: Reduces the impact of each tree. Smaller values result in slow but more precise learning, while bigger values cause the model to learn quickly but risk overshooting (0.01-0.3).

Number of Estimators: The total number of tree and larger number of trees enhances accuracy but increases training time and the risk of overfitting, whereas too few trees may underfit the data (100-1000).

Boosting Type: Gradient Boosting, Dart, and GOSS

L1 and L2 Regularization penalizes leaf weights to prevent overfitting and bigger values. reduce overfitting, and smaller numbers give more flexibility but risk overfitting (0–10)

Minimum Loss Reduction: The minimum amount of loss required to split further. Larger values inhibit splitting nodes with minimal gain, whereas smaller values encourage finer splits, which adds complexity.(0-1)

Column subsampling refers to the fraction of features used for each tree. Smaller numbers introduce unpredictability, minimizing overfitting, whereas bigger values use more features, raising the danger of overfitting(0.5-1).

Column Subsampling by Level: The fraction of characteristics used at each tree level (XGBoost-specific). Adds an extra layer of unpredictability to tree creation.

Feature Fraction is equivalent to column subsampling for LightGBM(0.5-1).

Random Seed: Regulates randomness for reproducibility, ensuring consistent outcome

Objective function is the loss function that is optimized during training.

Verbosity refers to the level of output logging.It manages debugging and monitoring while training .

D. Support Vector Machines (SVM)

Support Vector Machines (SVMs) are supervised learning models commonly used in classification, regression, and outlier detection. The performance of SVM is heavily influenced by the hyperparameters used, which govern the operation of the decision limit, kernel operation, and the optimization procedure [11].

Regularization Parameter: Specifies the trade-off between achieving a low error on training data and increasing the margin (0.0001–1000).

Kernel Type: A function that converts input data to a higher-dimensional space. The kernel contains linear, poly, rbf, and sigmoid.

Gamma: Measures the impact of a single training example.

Degree denotes the degree of the polynomial kernel function. Higher degrees enable collecting more complicated patterns, but require processing and the danger of overfitting (2-5).

Class Weight: Changes the penalty for misclassifying classes to solve class imbalance. The options include none, balanced, and user-defined.

Probability: Allows for probability estimates by completing extra calculations during training. It is useful for applications that require probability values rather than class labels, but it increases training time.

Tolerance is the terminating criterion for an optimization process. Smaller values produce accurate outcomes but increase training time, whereas bigger values accelerate training at the expense of potentially inferior solutions (0.0001-0.01).

Shrinking Heuristic: Indicates if the shrinking heuristic is employed to accelerate training. True indicates faster convergence, while False indicates somewhat better results but slower training.

Decision Function: Defines how the decision boundary is determined for multiclass problems.

Kernel Coefficient: Limits the impact of higher-order terms in polynomial and sigmoid kernels, with higher values increasing the model's complexity (0-1).

E. K-Nearest Neighbhor

K-Nearest Neighbors (KNN) is a basic but powerful non-parametric technique for classification and regression. Its performance is heavily impacted by hyperparameters that control how neighbors are chosen and weighted[12].

Number of Neighbors: The number of nearest neighbors to consider when generating forecasts. For smaller values of n_neighbors, the model appears sensitive to noise and more prone to overfitting, whereas for bigger values of n_neighbors, the model becomes more refined and robust but risks underfitting (3-20).

Metric: Indicates how distances between points are estimated. The possibilities include Euclidean, Manhattan, Minkowski, Hamming, and custom values. Depending on the feature distribution, alternative distance metrics may represent relationships in the data more effectively.

Minkowski Power Parameter: Defines the power parameter for the Minkowski distance(1-3)

Weights: Determines how much the neighbors contributes to the prediction. The options are uniform, distance, and user-defined. The uniform option is suited for equally dispersed datasets, and in terms of distance, closer neighbors have larger influence.

Algorithm: The technique used to find the nearest neighbors. The possibilities are: auto, ball_tree, kd_tree, and brute. The Ball Tree and KD Tree are faster for large datasets, although their performance is dimensionally dependent.

Leaf Size: The number of points in a leaf in Ball Tree or KD Tree algorithms. The smaller values of these parameters are more exact but slower computations, whilst the bigger values are faster but may diminish accuracy (20-50). Table 1 lists the model specific key parameters for the ML models liks dT, RF, XGB, SVM and KNN

Table 1 Common model specific parameters of ML models

Decision Tree(DT)	Random Forests(RF)	Gradient Boosting Models(XGB)	Support Vector Machines(SVM)	K-Nearest Neighbhor(KNN)
Maximum Depth	Number of Estimators	Learning Rate	Regularization parameter	Number of Neighbors
Minimum samples split	Maximum Depth	Number of Estimators	Kernel type e.g., linear, rbf, poly, sigmoid	Metrics((e.g., Euclidean, Manhattan, Minkowski)
Minimum samples leaf	Minimum samples split	Maximum Depth	Gamma-Kernel coefficient for rbf, poly, and sigmoid kernels	Weights
Criterion(Gini, Entrpoy)	Minimum samples leaf	Minimum samples split	Gamma	Algorithm
Maximum Features	Maximum Features	Sub sample	Class Weight	Leaf Size
Splitter	Bootstrap	Number of Leaves	Probability	Parallel Processing
Maximun Leaf Nodes	Class weights	Learning Rate	Tolerance	-
Class Weight	Maximum leaf nodes	Number of estimators	Shrinking Heuristic	-
-	Minimum Impurity decrease	Boosting type	Decision Function	-
-	Random State	L1 and L2 Regularization	Kernel Coefficient	-
-	-	Column subsampling	-	-
-	-	Column	-	-

		subsampling per level		
-	-	Feature Fraction	-	-
-	-	Objective function	-	-
-	-	Verbosity	-	-

IV. Results And Discussion

The tests were conducted in Python, and this work used two datasets (CIC-IDS2017, UNSW-NB15)[13] to assess the performance of hyperparameter tuning in DT, RF, XGB, SVM, and KNN. This study used performance metrics including as accuracy, precision, recall, and the F1-score.To establish these parameters, data on the distributions of true positives (TP), false positives (FP), and false negatives (FN) are required.

- Accuracy refers to the proportion of correctly identified samples (positive and negative) out of the total number of samples.
$$Accuracy = \frac{TP+FN}{TP+FP+FN+TN} \quad (1)$$
- Precision: How many anticipated positives are actually positives?
$$Precision = \frac{TP}{TP+FP} \quad (2)$$
- Recall (Sensitivity : How many genuine positives were successfully identified?
$$Recall = \frac{TP}{TP+FN} \quad (3)$$
- F1-score: The harmonic mean of precision and recall.
$$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (4)$$

Table 2 (a) : Performance metrics of the ML models

Methods/Measures	Accuracy		Precision	
	Without hyperparameter tuning	With hyperparameter tuning	Without hyperparameter tuning	With hyperparameter tuning
DT	0.86	0.96	0.905263	0.974359
RF	0.865	0.905	0.910526	0.926316
XGB	0.89	0.93	0.951872	0.952632
SVM	0.85	0.895	0.909091	0.925134
KNN	0.855	0.9	0.919355	0.930481

Table 2 (b) : Performance metrics of the ML models

Methods/Measures	Recall		F1-Score	
	Without hyperparameter tuning	With hyperparameter tuning	Without hyperparameter tuning	With hyperparameter tuning
DT	0.971751	0.989583	0.93733	0.981912
RF	0.97191	0.972376	0.940217	0.948787
XGB	0.962162	0.973118	0.956989	0.962766
SVM	0.965909	0.96648	0.936639	0.945355
KNN	0.960674	0.966667	0.93956	0.948229

Table 2 shows the performance measures values of DT, RF, XGB, SVM and KNN ML models with and without hyper-parameter settings .The percentage of accuracy, precision , recall and f1-Scores is found to be less before the hyperparameter tuning. Moreover, the XGB models performance with hyperparameters yields better results than the RF, SVM and KNN.

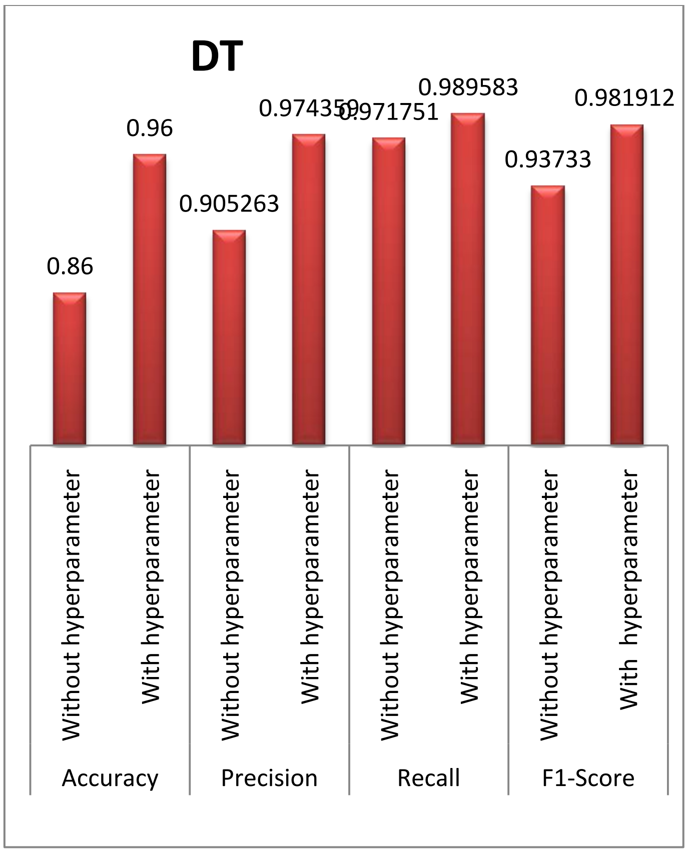

Figure 1. Performance Measures for Decision tree model(With and Without byperparameters)

Figure 1 depicts a graphical representation of the DT model's performance in analyzing model efficiency before and after hyperparameter settings. It is apparent that the effectiveness of data classification is higher for all evaluation parameters that include the essential hyperparameters of DT approaches.

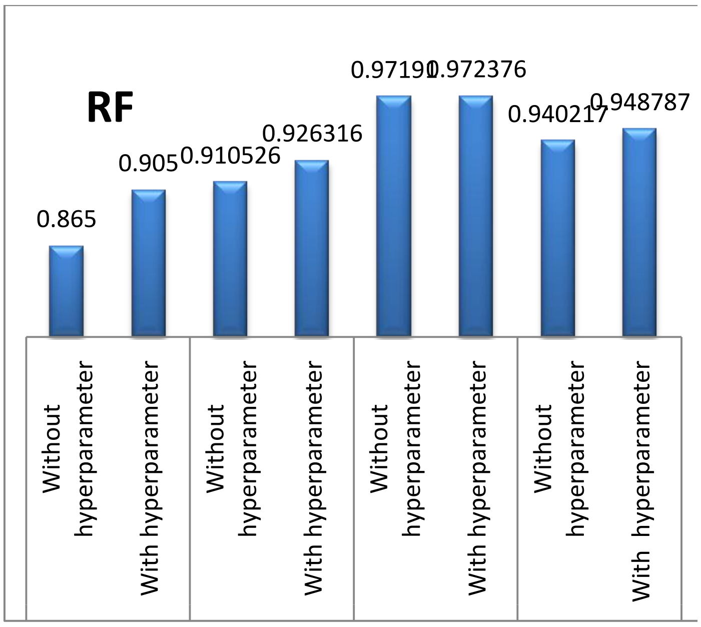

Figure 2. Performance Measures for Random Forests(RF) model(With and Without byperparameters)

RF performance can improve dramatically after hyperparameter tweaking since it is largely reliant on critical parameters that determine the quantity, depth, variety, and decision-making of trees in the ensemble. Figure 2 shows the RFs model's performance in measuring model efficiency before and after hyperparameter changes. It is clear that the effectiveness of data categorization is greater for all assessment factors, including the critical hyperparameters of the RF technique..

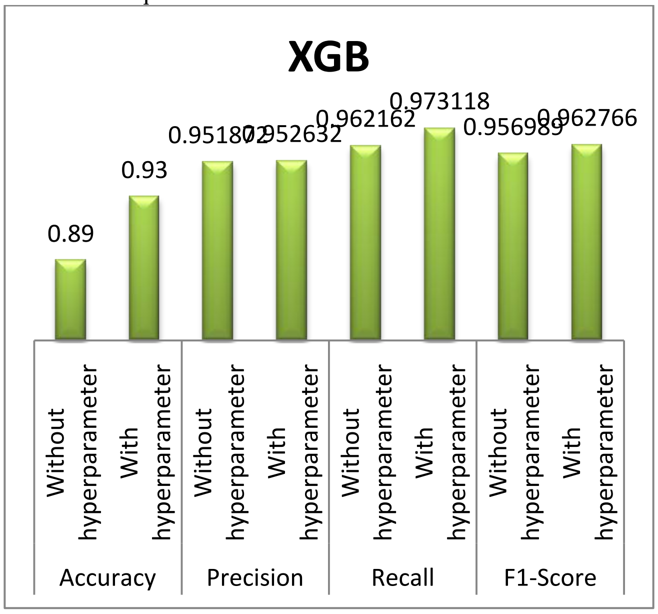

Figure 3. Performance Measures for XGB model(With and Without hyperparameters)

Figure 3 depicts the XGB model's performance in estimating model efficiency before and after hyperparameter adjustments. Data classification clearly outperforms all assessment variables, including the XGB technique's crucial hyperparameters.

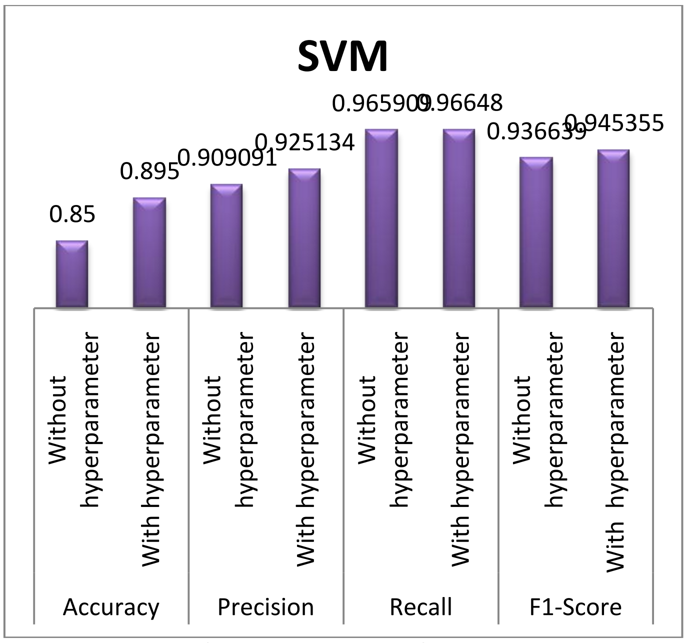

Figure 4. Performance Measures for Support Vector Machine(SVM) (With and Without byperparameters)

SVM are extremely effective at classification and regression tasks, but their performance is significantly dependent on hyperparameter settings. Figure 4 depicts the SVM model's performance in terms of model effectiveness prior to and following hyperparameter adjustments. All evaluation criteria, including the critical hyperparameters of the SVM technique, clearly demonstrate that data classification is more effective.

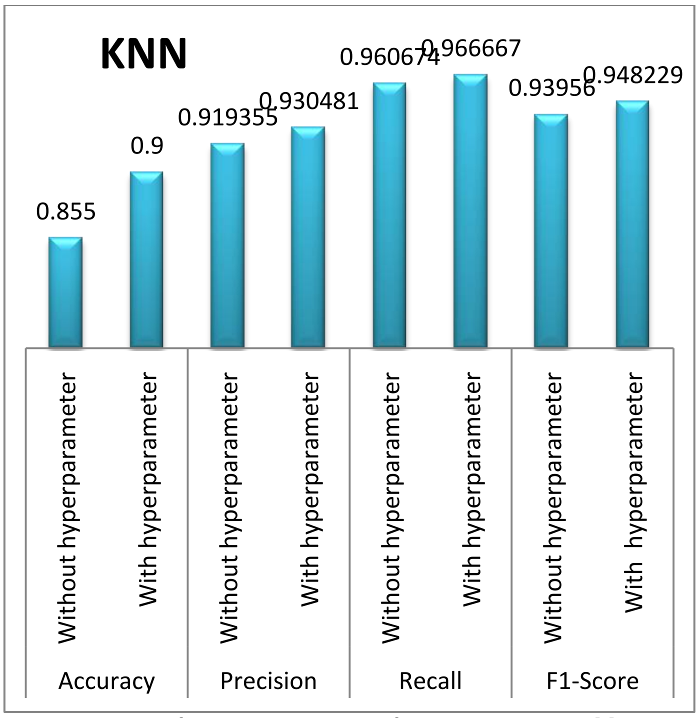

Figure 5. Performance Measures for K-Nearest Neigbhor Model(KNN)(With and Without byperparameters)

The performance of the KNN algorithms before and after hyperparameter adjustment frequently varies dramatically due to the model's sensitivity to its hyperparameters. After hyperparameter optimization, KNN improves prediction accuracy and robustness significantly, although these benefits may come at the expense of slightly lower computational efficiency during inference. Proper tuning, when combined with approaches such as dimensionality reduction or efficient data structures, can successfully balance accuracy and processing costs. Figure 5 depicts the KNN model's performance in terms of model effectiveness before and after hyperparameter changes. All evaluation criteria, including the essential hyperparameters of the KNN approach, show that data categorization is more effective.

Gradient boosting models are extremely adaptable, but they are also sensitive to hyperparameters such as the learning rate, number of estimators, and tree depth. Tuning these parameters can dramatically improve performance. These models feature several hyperparameters (e.g., learning rate, number of trees, max depth) that affect complexity and regularization, and their success is frequently closely related to fine-tuning. Hyperparameter tweaking has the greatest influence on sophisticated models such as Gradient Boosting (XGBoost, LightGBM), and Neural Networks. These models are extremely adaptable and can handle a wide range of datasets, but their success is greatly dependent on determining the proper settings. Hyperparameter adjustment has a smaller influence on simpler models, such as Naive Bayes or Logistic Regression.

V. Conclusion

This study employs five ML techniques: DT, RF, XGB, SVM and KNN. All of these ML algorithms were given model-specific parameters in order to increase their accuracy. The performance of ML models with hyperparameter tuning is measured using measures such as accuracy, precision, recall, and F-score.The results suggest that XGBoost models perform better in terms of accuracy, precision, recall, and F1-score. Furthermore, the proper model and tuning method are determined by the dataset, task requirements, and computing power. HPO is critical for developing efficient, accurate, and generalizable machine learning models. By methodically tuning hyperparameters, practitioners may ensure that their models perform optimally, adapt to varied tasks, and function within resource restrictions. This phase is essential for effectiveness in both practical applications and research.

References

[1] El-Hassani, F.Z., Amri, ., Joudar, NE. et al. A New Optimization Model for MLP Hyperparameter Tuning: Modeling and Resolution by Real-Coded Genetic Algorithm. Neural Process Lett 56, 105 (2024). https://doi.org/10.1007/s11063-024-11578-0

[2] Arnold C, Biedebach L, Küpfer A, Neunhoeffer M. The role of hyperparameters in machine learning models and how to tune them. Political Science Research and Methods. 2024 Oct;12(4):841-8.

[3] Bakare KA, Abubakar SI, Naveen AY, Abdullahi AJ, Gaku MS, Asmau U, Ahmad S. Streamlining the Path from Data to Deployment: Intelligent Methods for Hyperparameter Tuning in Machine Learning. Bima Journal of Science and Technology. 2024 Sep 2;8(2A):192-210.

[4] Raiaan MA, Sakib S, Fahad NM, Al Mamun A, Rahman MA, Shatabda S, Mukta MS. A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks. Decision Analytics Journal. 2024 Apr 24:100470.

[5] Li L, Yang J, Por LY, Khan MS, Hamdaoui R, Hussain L, Iqbal Z, Rotaru IM, Dobrotă D, Aldrdery M, Omar A. Enhancing lung cancer detection through hybrid features and machine learning hyperparameters optimization techniques. Heliyon. 2024 Feb 29;10(4).

[6] Wojciuk M, Swiderska-Chadaj Z, Siwek K, Gertych A. Improving classification accuracy of fine-tuned CNN models: Impact of hyperparameter optimization. Heliyon. 2024 Mar 15;10(5).

[7] Gomiasti FS, Warto W, Kartikadarma E, Gondohanindijo J. Enhancing Lung Cancer Classification Effectiveness Through Hyperparameter-Tuned Support Vector Machine. Journal of Computing Theories and Applications. 2024 Mar 25;1(4):396-406.

[8] Liao M, Wen H, Yang L, Wang G, Xiang X, Liang X. Improving the model robustness of flood hazard mapping based on hyperparameter optimization of random forest. Expert Systems with Applications. 2024 May 1;241:122682.

[9] Habib MA, Abolfathi S, O'Sullivan JJ, Salauddin M. Efficient data-driven machine learning models for scour depth predictions at sloping sea defences. Frontiers in Built Environment. 2024 Feb 9;10:1343398.

[10] Kavzoglu, Taskin, and Alihan Teke. "Advanced hyperparameter optimization for improved spatial prediction of shallow landslides using extreme gradient boosting (XGBoost)." Bulletin of Engineering Geology and the Environment 81, no. 5 (2022): 201

[11] Guido, Rosita, Maria Carmela Groccia, and Domenico Conforti. "A hyper-parameter tuning approach for cost-sensitive support vector machine classifiers." Soft Computing 27, no. 18 (2023): 12863-12881.

[12] Assegie, Tsehay Admassu, Tamilarasi Suresh, Raguraman Purushothaman, Sangeetha Ganesan, and Napa Komal Kumar. "Early prediction of gestational diabetes with parameter-tuned K-Nearest Neighbor Classifier." Journal of Robotics and Control (JRC) 4, no. 4 (2023): 452-457.

[13] https://www.kaggle.com/datasets/yasiralifarrukh/unsw-and-cicids2017-labelled-pcap-data