E-ISSN 2581 – 7957
P-ISSN 2277 – 3916

CVR Journal of Science and Technology, Volume 25, December 2023
DOI: 10.32377/cvrjst2508

Optimizing Hyper Parameters for Enhanced Performance: A Comparative Study on K-Nearest Neighbor and Support Vector Machine Models

Arun Kumar Katkoori[1] and R. Prakash Kumar[2]
[1]Sr. Asst. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: arun.katkoori@gmail.com
[2]Sr. Asst. Professor, CVR College of Engineering/ECE Department, Hyderabad, India
Email: prakash.rachmagdu@gmail.com

Abstract: **Numerous applications and areas benefit from the use of machine learning algorithms. The hyperparameters needed to fit a suitable model must be tuned to address different issues. The choice of the best configuration directly affects the performance of the model. User-defined hyper-parameters are those that are set before the training process is carried out. In machine learning, optimizing the hyperparameters is a process that can reduce the cost function. This paper presents a couple of methods that are used to optimize the K-Nearest Neighbor algorithm and the Support Vector Machine model. We have performed several experiments on the different optimization techniques to evaluate their accuracy and complexity.**

Index Terms: **Hyper parameters, Optimization Techniques, KNN, SVM, Random Search, Grid Search.**

I. Introduction

Machine learning has been widely used in various applications, such as advertising and recommendation systems. These algorithms are generally good at handling data analytics-related challenges. ML techniques can be used for different kinds of datasets or issues [1]. Developing an efficient model can be a time-consuming process, as it involves choosing the best algorithm and tuning its hyperparameters. There are two kinds of parameters that are used in developing machine learning models. The first is called model parameters, which can be updated and initialised through the learning process. The other is called hyper-parameters, which are defined by the model architecture. ML models cannot directly be estimated from data learning, and these must first be set [2].

One of the most important factors that are considered when developing machine learning algorithms is hyperparameters [3]. These are settings that are set before the training process, and they can be used to control the model's behavior. They also help determine how well the model can learn from new data. A hyperparameter is different from a machine learning model's parameter, which are acquired from data gathered during the training phase. For instance, the learning rate, the number of trees in a randomly generated forest, and the regularisation term are examples of hyperparameters [4].

In machine learning, optimizing hyperparameters is an important step. Doing so can significantly affect the model's performance. There are various techniques for doing so, such as random search [5], grid searching [6], and Bayesian optimization [7], among others. Grid searching is a type of algorithm that involves trying out a set of predefined hyperparameters and picking the one that gives you the best validation set performance. Random search, on the other hand, is a process that involves randomly sampling the combinations. For instance, in Bayesian optimization, a probabilistic model is used to guide the search.

There are many advantages to implementing HPO techniques in machine learning frameworks.

By utilizing HPO to identify the ideal set of hyperparameters, user can greatly enhance the performance and accuracy of ML model. User can also fine-tune the parameters to make the model more capable of learning from and generalising new data.

One of the most common issues that an ML model encounters when it comes to training data is overfitting. This occurs when the model fails to perform well on fresh data. With the help of HPO, you can identify the appropriate hyperparameters that will help minimize overfitting.

Utilizing hyperparameter optimization (HPO) techniques can help find the ideal hyperparameters efficiently, resulting in reduced training times for models. This is especially advantageous when dealing with large datasets or complex models. The use of HPO techniques can help ensure the consistency and reproducible nature of your ML models, even if the same set of parameters is used by different developers. This is because their hyperparameters are precisely optimized, utilizing an objective and systematic approach.

Hyperparameter optimization techniques are different from standard optimization methods in several aspects.

- ➢ Traditional optimization techniques are commonly used to improve a single function with a small number of variables. On the other hand, HPO involves optimizing several hyperparameters in a search area, which can be more challenging.
- ➢ An objective function that is well-defined can be improved through traditional optimization techniques. In HPO, an objective function is usually focused on the performance of a machine learning model in a validation set.
- ➢ The computational cost of implementing traditional optimization methods is typically high when there are many factors involved. Due to the complexity of the training pipeline, it can

Received on 17-10-2023, Revised on 11-11-2023, Accepted on 09-12-2023.

also have a significant impact on the overall cost of computing.

Decision-theoretic approaches are used to define a search space for hyper-parameters. They allow users to identify the ideal combination of these characteristics inside the search space and choose the most effective one. A particular type of search method known as grid search is used to find the optimal configuration of these characteristics.

Due to the limited number of resources and the time needed to perform a search, a decision-theoretic method known as random search is used. It randomly chooses the most appropriate hyper-parameters from the search space.

II. Literature Survey

The paper [8] analyzed the various algorithms used in the development of neural networks to find the best hyperparameters. The three algorithms used in the study were the Genetic Algorithm, the Grid Search, and the Bayesian Algorithm. The experiments were conducted on a dataset from Santander for customer transaction prediction. The parameters used in the analysis included the number of hidden layers, size, loss function, optimizers, activation function, and drop-out. The results indicated that the grid search performed better than the genetic algorithm, while the latter performed better than the Bayesian algorithm. The best hyperparameters were found in the first two algorithms, the Grid Search, and the Bayesian. The former performed well because it had the same number of secret layers and an identical optimizer.

Another paper [9] proposes an IDS framework that uses the k-nearest-neighbor tuning method to improve the performance of its semi-supervised learning. The first step is to identify the nearest neighbor of each unlabeled datapoint. After that, the data points are classified into either attack or normal classes according to the statistical information collected from these neighboring data points. The paper presents a comprehensive analysis of the model's robustness using a standard dataset NSL-KDD. The results of the study show that the proposed framework performs better than the KNN algorithms that are based on IDS.

Authors [10] introduced several advanced techniques and discussed their application to the algorithms. It also provides a variety of frameworks and libraries for addressing the issue. The paper conducted experiments to evaluate the effectiveness of different optimization techniques and presents practical optimization examples. The results of these studies will help researchers and industrial users identify the optimal configurations of their models.

Authors [11] evaluated six machine-learning and statistical models in terms of their predictive performance. The results revealed that the former outperformed the latter, except for SVM. In addition, non-spatial partitioning models exhibited overoptimistic performance when paired with spatial autocorrelation.

This paper [12] aims to analyze the various popular algorithms for optimizing hyperparameters in a grid search, random search, and genetic algorithm for building a neural network framework. The results of the study are based on the proposed models' accuracy and execution time.

The general method for choosing SVM hyperparameters is known as the CMA evolution strategy. It can handle various kernel parameters and doesn't require the availability of differentiable kernels or the separability of data. In addition, the selection criteria typically have multiple local optima, making it more suited for model selection than the gradient-based approach [13].

To improve the SVM parameters, authors [14] used two different methods: grid search and genetic algorithm. In their experiment, it was revealed that the grid search method is very reliable when it comes to finding optimal combinations within the given ranges, but it was very slow when it came to performing optimization in low dimensional datasets. The use of GA in SVM parameter optimization can help solve the issue of grid search. Compared to grid search, GA is more stable. The average running time of this method on nine datasets was 16 times faster. In fact, the results of the GA were significantly better than those of grid search in eight of the datasets.

Manual and grid searches are commonly used for optimizing hyper-parameters. In this paper, the authors [15] show that trials that are randomly chosen are more efficient than those conducted on a grid. We also compare the effectiveness of these two strategies with a previous study that used both to configure deep belief networks and neural networks. They found that random searches are more efficient than neural networks that are configured with a grid search. They can find better models in a fraction of the time. In contrast, random searches are more effective when they are granted the same computational budget.

III. Methodology

Fig. 1 illustrates the key components and the flow of hyperparameter tuning using optimization techniques within the context of building and evaluating machine learning models. These techniques help you systematically explore and optimize hyperparameters to improve the model's performance on dataset.

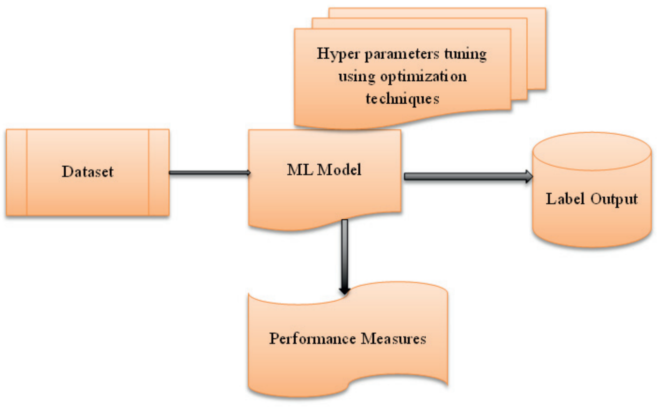


Figure 1. Block diagram of proposed method

The dataset represents the collection of data points with features and labels, serving as the foundation for training and evaluating machine learning models. The machine learning model that takes the input data from the dataset and produces predictions as output. The ML model generates predictions based on the input data. These predictions are then compared to the actual labels in the dataset to assess the model's performance. Performance measures [16] are metrics used to evaluate the quality of the model's predictions, such as accuracy, precision, recall, F1-score, mean squared error, etc.

Here four datasets are used i.e., balance scale dataset, IRIS dataset, breast cancer dataset, and digits dataset. They are frequently utilized to test and benchmark the performance of various algorithms and models in the evaluation of classification-related tasks.

i) Balance Dataset:

It displays a balanced scale with varying weights on its sides to determine its fairness. The dataset features four attributes, which represent the distance and weight of objects on the balance's right and left sides. The target variable of the dataset is the class that determines whether or not the scale is balanced.

ii) IRIS dataset:

This dataset is used in the research of statistics and machine learning. It contains the measurements of various features of an iris flower, such as its sepal width, length, and petal width. The four features represent the iris flower's dimensions. The target is the species.

iii) Breast Dataset:

This dataset is frequently utilized for classification purposes related to the diagnosis of breast cancer. It features digitized images of a fine needle as an FNA. The attributes of this dataset include various features such as the mean radius, texture, and smoothness. The classification parameter target determines whether the tumor is benign or malignant.

iv) Digits Dataset:

The Digits dataset is frequently utilized for identifying handwritten digits. It comprises images of the digits. The features that represent them have pixel values. The target digit is the one depicted in the image.

Hyperparameters are model settings or configurations that are not learned from the data but must be specified before training. Examples include learning rates, the number of hidden layers in a neural network, regularization strength, etc. Hyperparameter tuning is the process of finding the optimal combination of hyperparameters to maximize the model's performance on the given task. This is important because different hyperparameter settings can have a significant impact on the model's effectiveness.

The KNN [17] and SVM [18] models were then created using the scikit-learn framework. We identified the necessary hyperparameters for each model. For SVM, the hyperparameters are gamma, C, and kernel. On the other hand, for KNN, the hyperparameters are n neighbors.

A. Q-Random Search

Here, Random Search [19] is utilized to find the optimal hyperparameters suitable for SVM and KNN models. A quick-random search introduces various constraints and refinements. For instance, the range or intervals that random sampling may be restricted to may be considered more meaningful for the analysis.

The three hyperparameters of SVM that are used in the decision function are "C," "gamma," and "kernel." The "C" parameter determines the optimal balance between the training points correctly classified and the smooth decision boundary. The "gamma" parameter indicates how far the influence of one training example can go.

TABLE I.
PSEUDO CODE FOR Q-RANDOM SEARCH USING SVM

```
INPUT: maximum number of iterations( maxitr)
1.   param_grid = {'c': [1, 3, 5,7],
     'gamma': [0.1, 0.01, 0.001],
      'kernel': ['linear', 'rbf']
2.best_para={'c':lst['c'][1],'gamma':lst['gamma'][1],'kernel':lst['kernel'][0]}
3.svm_model=svm.SVC(C=best_para['c'],gamma=best_para['gamma'],
  kernel=best_para['kernel'])
4.   best_acc=accuracy(svm_model )
5.   For i in maxitr :
                     c=random(param_grid['c'])
                     gamma=random(param_grid['gamma')
                              kernel=random(param_grid['kernel'])
                              model=svm(c,gamma,kernel)
                              present_acc=accuracy(model)
                                  if(present_acc)> best_acc:
                                           best_acc= present_acc
                                           best_para={'c':c,'gamma':gamma,'ker
                              nel':kernel}
                              End if
                     End for
6.   Return best_para
```

The optimal combination of hyperphys will be determined by evaluating each iteration through a loop called "maxitr". The random value of each parameter will be used in the creation of a new SVM. Its accuracy is then calculated by using the function "accuracy".

TABLE II.
PSEUDO CODE FOR Q-RANDOM SEARCH USING KNN

```
INPUT: maximum number of iterations( maxitr)
1.   para_grid = {'n_neighbors': [1:10]}
2.   Random Search for KNN
     def random_search_knn(maxitr, para_grid_knn):
3.    Initialize the best accuracy and best parameters
          best_acc = 0
          best_para = {}
4.   for in range(maxitr)
           n_neighbors =
     random.choice(para_grid_knn['n_neighbors'])
           model = KNeighborsClassifier(n_neighbors=n_neighbors)
5.   present_acc = accuracy_knn(model)
6.   if present_acc > best_acc:
             best_acc = present_acc
            best_para = {'n_neighbors': n_neighbors}
      End if
      End for
7.   Return best_para
```

The function "accuracy" updates the hyperphys' values if the new model's accuracy exceeds the current best. It then returns the most accurate set.

The given table II pseudocode provides a method that can be used to search for hyperparameter values in a KNN algorithm. It randomly chooses the optimal values for the n_neighbors parameter from the grid "para_grid_knn". Each iteration of the KNN algorithm, a model is created with a randomly chosen n_neighbors value. The algorithm's accuracy is calculated by using a hypothetical accuracy-knn function. If the current model exceeds the previous best, its corresponding hyperparameters and accuracy are updated. The algorithm then returns the best configuration after the random search.

B. Q-Grid Search

For Grid Search, a set of parameters are created that are used to train the models. Then select the best ones for the validation set and random sampling for the training.

The Q-grid (Quick) search strategy adopts a more efficient approach, which eliminates the need to evaluate every possible combination within a predefined grid. Instead, it uses a randomized sampling method to evaluate each combination. This method helps reduce the computational demands while maintaining a level of exhaustiveness. The new strategy avoids the need to analyze every possible combination in each grid, especially when dealing with large search spaces. It focuses on exploring different hyperparameter configurations to find promising ones.

The first value of each hyperparameter in the dictionary of "param grid" is the initial value of the model that will be used to create a SVM. The model's accuracy is then calculated using the "accuracy" function. The best acc parameter is used to represent the initial accuracy. The algorithm iterates through the various hyperparameters in the grid and creates new SVM models with the current set of values. The model's accuracy is then calculated using the "accuracy" function.

The function considers the new model's accuracy and updates the hyperparameters with the new values. It then returns the best set of values, which gives the highest accuracy.

Table IV pseudocode describes a method that can be used to search for hyperparameter values in a KNN algorithm. It takes into account the maximum number of iterations that can be performed to find the most appropriate values for the n_neighbors parameter from the grid "para_grid_knn".The KNN algorithm creates a model with the chosen n_neighbors value in every iteration. Its accuracy is computed by using a function known as an accuracy_knn. If the current model's accuracy exceeds the previous best, the corresponding hyperparameters are updated. The algorithm then performs an iterative process to find the optimal configuration.

TABLE III.
PSEUDO CODE FOR Q-GRID SEARCH USING SVM

```
INPUT: maximum number of iterations( maxitr)
1.  param_grid = {'c': [1, 3, 5,7],
                  'gamma': [0.1, 0.01, 0.001],
                  'kernel': ['linear', 'rbf']
2.  best_para={'c':lst['c'][1],'gamma':lst['gamma'][1],'kernel':lst['kernel'][0]}
3.  svm_model=svm.SVC(C=best_para['c'],gamma=best_para['gamma'],
    kernel=best_para['kernel'])
4.  best_acc=accuracy(svm_model )
    For c in param_grid['c']
        For gamma in param_grid['gamma']
            For kernel in param_grid['kernel']
                model=svm(c,gamma,kernel)
                present_acc=accuracy(model)
                if(present_acc)> best_acc:
                    best_acc= present_acc
best_para={'c':c,'gamma':gamma,'kernel':kernel}
                End if
            End for
        End for
    End for
5.  Return best_para
```

TABLE IV.
PSEUDO CODE FOR Q-GRID SEARCH

```
INPUT: maximum number of iterations( maxitr)
1. param_grid = {'n_neighbors': [1:10]}
2.  Grid Search for KNN
    def grid_search_knn(maxitr, para_grid_knn):
3.  Initialize the best accuracy and best parameters
        best_acc = 0
        best_para = {}
4.  for in range(maxitr)
        n_neighbors = grid.choice(para_grid_knn['n_neighbors'])
        model = KNeighborsClassifier(n_neighbors=n_neighbors)
5.  present_acc = accuracy_knn(model)
6.  if present_acc > best_acc:
            best_acc = present_acc
            best_para = {'n_neighbors': n_neighbors}
    End if
    End for
7.  Return best_para
```

IV. Results

Tables V and VI provide a comparative analysis of different SVM configurations, KNN and their performance on the different datasets. It demonstrates the impact of hyperparameter tuning on accuracy and computation time, which can be crucial when selecting the best SVM model for a particular problem.

TABLE V.
SVM Classifier's Performance on Different Datasets

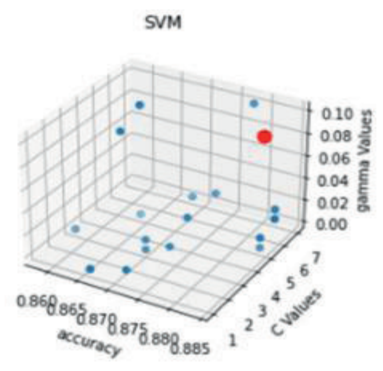
Dataset	Name	Accuracy	Computation time (sec.)	c	gamma	kernel
Balance Scale Dataset	Default	0.87	0.004115	1.0	scale	rbf
	grid search	0.872	0.39201	1	0.01	rbf
	Random	0.884	0.20400	4.64158	0.00599	linear
	Q-grid search	0.884	0.13948	5	0.1	linear
	Q-random search	0.884	0.2372	5	0.1	linear
IRIS Dataset	Default	0.85833	0.00373	1.0	Scale	rbf
	grid search	0.96666	0.362037	3	0.1	linear
	Random	0.925	0.14787	215.4434	0.35938	linear
	Q-gird search	0.9666	0.1012547	3	0.01	linear
	Q-random search	0.96666	0.124896	3	0.01	linear
Breast Cancer	Default	0.90789	0.00808	1.0	scale	rbf
	grid search	0.94736	19.35496	5	0.1	linear
	Random	0.96052	17.96333	1.291549	2.7825	linear
	Q-gird search	0.95614	13.00778	3	0.01	linear
	Q-random search	0.95614	15.81649	3	0.01	linear
Digits Dataset	Default	0.97496	0.013361	1.0	scale	rbf
	grid search	0.98817	0.86540	3	0.001	rbf
	Random	0.96105	0.446697	1.29154	0.04641	linear
	Q-gird search	0.988178	0.913914	3	0.001	rbf
	Q-random search	0.988178	0.940677	7	0.001	rbf

E-ISSN 2581 – 7957
P-ISSN 2277 – 3916

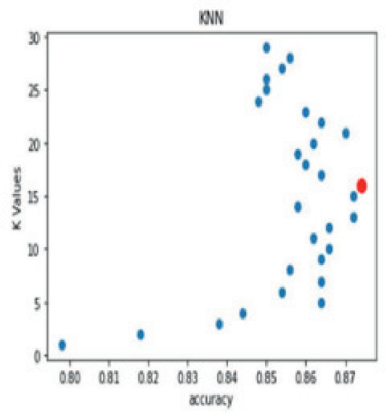
TABLE VI.
KNN CLASSIFIER'S PERFORMANCE ON DIFFERENT DATASETS

Dataset	Name	Accuracy	Computation time (sec.)	No. of neighbours
Balance Scale Dataset	Default	0.838	0.004738	3.0
	grid search	0.86	1.6074438	18
	Random	0.86	0.58122706	23
	Q-gird search	0.874	0.63590097	16
	Q-random search	0.872	0.62832236	15
IRIS Dataset	Default	0.925	0.0060422	3.0
	grid search	0.933333	1.6042816	1
	Random	0.9333	1.08904	1
	Q-gird search	0.9333	0.71362185	1
	Q-random search	0.93333	0.2778687	1
Breast Cancer Dataset	Default	0.61538	0.0025372	3.0
	grid search	0.65034	1.5867512	5
	Random	0.66433	1.0021665	19
	Q-gird search	0.69930	0.4683482	4
	Q-random search	0.69930	0.6204957	4
Digits Dataset	Default	0.93421	0.0049138	3.0
	grid search	0.93421	2.0322103	3
	Random	0.93421	1.2464041	3
	Q-gird search	0.93421	1.5173799	3
	Q-random search	0.93421	0.6609208	3

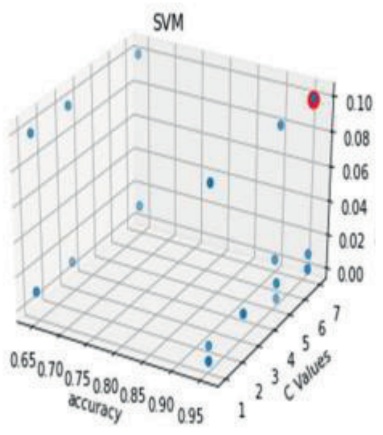
Fig. 2 shows the SVM Classifier's C and gamma value, which are optimized using Q-random and Q-grid search techniques for 5 datasets. Also, the KNN Classifier is evaluated using Q-random and Q-grid-based search optimization techniques for 5 datasets. Red color dot indicates the optimum value.



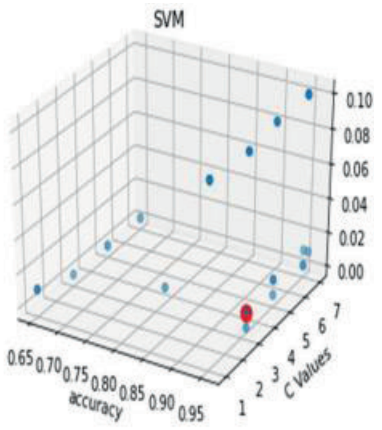
a) Balanced dataset



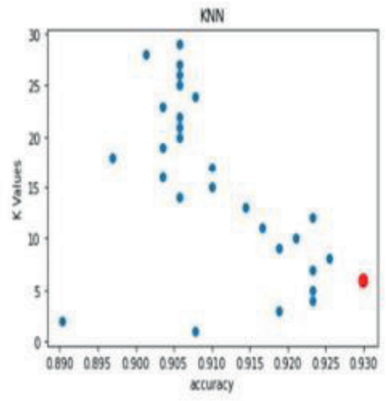
b) Balanced dataset



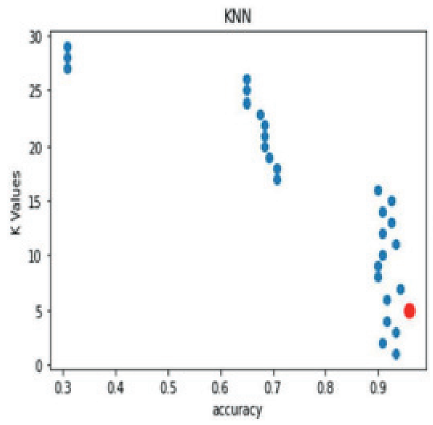
e) Breast Cancer Dataset



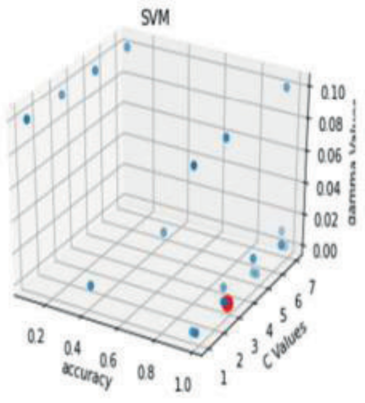
c) Iris Dataset



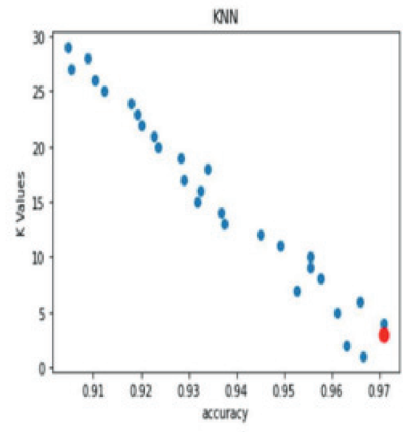
f) Breast Cancer Dataset



d) Iris dataset



g) Digits dataset



h) Digits dataset

Figure 2. SVM's and KNN's classifiers hyper parameters optimization using Q-random and Q-grid Search techniques applied on Balanced, Iris, Breast cancer and Digits datasets.

The results of the study revealed that the Q-random algorithm is better than the grid and random search methods when it comes to tuning hyperparameters in ML models. The accuracy of the models that were trained using the different algorithms was evaluated.

According to the findings, the Q-search optimization algorithm is more accurate than the grid or random search methods. This can be attributed to the algorithm's low number of iterations, which aids in achieving improved hyperparameter tuning.

It is noted that since the Q-random optimization algorithm only iterates around 30 times, it is more accurate and efficient than the random search method. It also has better performance in terms of tuning time and processing resources. The researchers attributed the improved accuracy of the Q-random algorithm to its intelligent sampling technique, which helps it focus on ideal hyperparameters at a faster rate than the random search method.

V. Conclusions

The importance of tuning machine learning models is acknowledged in this study, as it directly affects their performance in various applications. The study also analyzed the different KNN and SVM configurations on different datasets.

- ➢ The study investigated optimizing hyperparameters using various optimization techniques, such as random and grid searches. It found that the q-random algorithm performed better than both random and grid searches in tuning accuracy.
- ➢ The efficiency and accuracy of the training models were found to be better with the q-random algorithm. This was due to its low number of iterations. This algorithm can effectively fine-tune hyperparameter values.
- ➢ The efficiency of the q-random algorithm when it comes to tuning time and computational resources was also better than that of random searches. Its ability to sample various hyperparameters allowed it to improve its accuracy and convergence.
- ➢ The findings of this study have practical applications in the areas of machine learning. It shows that the q-random algorithm can be utilized to optimize hyperparameters, which makes it an ideal alternative to traditional techniques.

The study demonstrates that optimizing the performance of learning models involves considering the various hyperparameters. The q-random approach is an ideal choice, as it offers high efficiency and accuracy. As machine learning advances, more effective optimization techniques will be needed to help develop AI applications that can be used in various sectors.

References

[1] Cherian, I. ., Agnihotri, A. ., Katkoori, A. K. ., & Prasad , V. . (2023). Machine Learning for Early Detection of Alzheimer's Disease from Brain MRI. International Journal of Intelligent Systems and Applications in Engineering, 11(7s), 36–43.

[2] Rajesh, Dr. V. and Bhanuprakash Dudi. "Performance analysis of leaf image classification using machine learning algorithms on different datasets." (2021).

[3] Dudi, Bhanuprakash, and V. Rajesh. "Medicinal plant recognition based on CNN and machine learning." International Journal of Advanced Trends in Computer Science and Engineering 8.4 (2019): 999-1003.

[4] Prakash, D. Bhanu, K. Arun Kumar, and R. Prakash Kumar. "Hyper-parameter optimization using metaheuristic algorithms." CVR Journal of Science and Technology 23.1 (2022): 37-43.

[5] Li, B. (n.d.). Random Search Plus: A more effective random search for Random Search Plus: A more effective random search for machine learning hyperparameters optimization machine learning hyperparameters optimization.

[6] Liashchynskyi, P., & Liashchynskyi, P. (2019). Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS.

[7] Zhang, Y., Apley, D.W. & Chen, W. Bayesian Optimization for Materials Design with Mixed Quantitative and Qualitative Variables. Sci Rep 10, 4924 (2020).

[8] Alibrahim, Hussain & Ludwig, Simone. (2021). Hyperparameter Optimization: Comparing Genetic Algorithm against Grid Search and Bayesian Optimization. 1551-1559. 10.1109/CEC45853.2021.9504761.

[9] Wazirali, R. An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation. Arab J.Sci Eng 45, 10859–10873(2020).

[10] Li Yang, Abdallah Shami, On hyperparameter optimization of machine learning algorithms: Theory and practice, Neurocomputing, Volume 415, 2020, Pages 295-316, ISSN 0925-2312.

[11] Patrick Schratz, Jannes Muenchow, Eugenia Iturritxa, Jakob Richter, Alexander Brenning, Hyperparameter tuning and performance assessment of statistical and machine-learning algorithms using spatial data, Ecological Modelling, Volume 406, 2019, Pages 109-120, ISSN 0304-3800.

[12] Liashchynskyi, Petro B. and Pavlo Liashchynskyi. "Grid Search, Random Search, Genetic Algorithm: A Big Comparison for NAS." ArXiv abs/1912.06059 (2019).

[13] Frauke Friedrichs, Christian Igel, Evolutionary tuning of multiple SVM parameters, Neurocomputing, Volume 64, 2005, Pages 107-117, ISSN 0925-2312.
[14] Syarif, Iwan & Prugel-Bennett, A. & Wills, Gary. (2016). SVM Parameter Optimization using Grid Search and Genetic Algorithm to Improve Classification Performance. TELKOMNIKA (Telecommunication Computing Electronics and Control). 14.
[15] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. J. Mach. Learn. Res. 13, null (3/1/2012), 281–305.
[16] Kumar, K.A., Boda, R. A computer-aided brain tumor diagnosis by adaptive fuzzy active contour fusion model and deep fuzzy classifier. Multimed Tools Appl 81, 25405–25441 (2022).
[17] L. R. Somula and M. Meena, "K-Nearest Neighbour (KNN) Algorithm based Cooperative Spectrum Sensing in Cognitive Radio Networks," 2022 IEEE 4th International Conference on Cybernetics, Cognition and Machine Learning Applications (ICCCMLA), Goa, India, 2022, pp. 1-6, doi: 10.1109/ICCCMLA56841.2022.9988996.
[18] Katukuri Arun Kumar, Ravi Boda, A Multi-Objective Randomly Updated Beetle Swarm and Multi-Verse Optimization for Brain Tumor Segmentation and Classification, The Computer Journal, Volume 65, Issue 4, April 2022, Pages 1029–1052.
Esmaeili, Ahmad, Zahra Ghorrati, and Eric T. Matson. 2023. "Agent-Based Collaborative Random Search for Hyperparameter Tuning and Global Function Optimization" Systems 11, no. 5: 228.