Logistic Regression Hyperparameter Optimization for Cancer Classification

Ahmed Arafa
Computer Science & Engineering Dept.
Faculty of Electronic Engineering
Menoufia, Egypt.
ahmed.arafa@el-eng.menofia.edu.eg

Marwa Radad
Computer Science & Engineering Dept.
Faculty of Electronic Engineering
Menoufia, Egypt.
marwa_abbas2003@yahoo.com,

Mohammed Badawy
Computer Science & Engineering Dept.
Faculty of Electronic Engineering
Menoufia, Egypt.
mohamed.badawi@el-eng.menofia.edu.eg

Nawal El-Fishway
Computer Science & Engineering Dept.
Faculty of Electronic Engineering
Menoufia, Egypt.
nelfishawy@hotmail.com

***Abstract*—**

In machine learning, optimization of hyperparameters aims to find the best values of model hyperparameters yielding an optimal model with minimum prediction error. It is the most important step that directly affects the performance of learned model. Many techniques have been proposed to optimize hyperparameters for different predictive models. In this paper, the performance of grid search, random search, Bayesian Tree Parzen Estimator (TPE) and Simulated Annealing (SA) optimization techniques is evaluated to determine the best hyperparameters for a logistic regression model when used in cancer classification. Wisconsin Breast Cancer Dataset (WBCD) has been used to evaluate the previously mentioned optimization techniques. The results show that Bayesian TPE outperformed other techniques in terms of number of iterations and running time. The number of iterations to get optimal parameters in TPE is less than SA by 75.75 %, and random search by 77.1%. While the time taken by TPE is better than SA, random search and grid search by 79.9%, 86.1% and 99.9% respectively. The resulted optimal hyperparameter values have been utilized to learn a logistic regression model to classify cancer using WBCD dataset. The optimized model succeeded in classifying cancer with 98.2% for test accuracy, 0.962 for kappa statistic and 0.963 for MCC metrics when evaluated using 10-fold cross validation.

Keywords*— *Hyperparameter Optimization, Random Search Grid Search, Tree Parzen Estimator, Simulated Annealing

I. INTRODUCTION

In the world of machine learning, there exist two main types of parameters that determines the performance of the predictive model. These types are model parameters and hyperparameters. Model parameters are model coefficients such as weights for logistic regression model or neural network that can be estimated from the training data and resulted during model training [1]. Contrariwise, hyperparameters, also called tuning parameters [2], are set of options and settings that are independent of the training data and must be determined before training the model. Examples on hyperparameters are the penalty type and regularization strength for logistic regression, number of neighbors in KNN (K-Nearest Neighbors), kernel type in support vector machines (SVM) and number of trees in Random Forest.

The performance of machine learning models mainly depends on the settings of their hyperparameters [3]. These parameters can be determined manually based on experience through multiple tests or it can be determined automatically using one of the widely used optimization algorithms [4]. Recently, new trends in machine learning depends on the automatic adjustment of hyperparameters, so, many techniques have been proposed and applied for this purpose. These techniques are classified as black-box and multi-fidelity optimization techniques [5]. Figure 1 shows the classification of hyperparameter optimization techniques. Black box optimization doesn't use the gradient of the objective function because the function is inaccessible or the gradient is expensive to be calculated. So, only the function output which is already known, is used to estimate the hyperparameters [6]. Contrariwise to black-box, instead of considering a single expensive evaluation for the objective function, multi-fidelity optimization utilizes many cheap low-fidelity evaluations to ignore the regions with low values while keeping expensive approximations for promising regions [7]. For hyperparameter optimization with large data sets, cheap approximations can be obtained by training the model using only a randomly chosen subset of the training dataset [8].

In this paper, the performance of common black-box optimization techniques is evaluated to optimize logistic regression hyperparameters. These hyperparameters include penalty and learning rate. This study includes grid search, random search, Bayesian Tree Parzen Estimator (TPE) and Simulated Annealing (SA) techniques. The models are evaluated using Wisconsin Breast Cancer Dataset (WBCD) [9]. A model for cancer classification using the resulted optimal parameters is introduced. The performance of the optimized model has been investigated according to other research work.

The rest of this paper is organized as follow: section 2 explains the logistic model parameters. Section 3 reviews the black box optimization techniques. Section 4 introduces

the workflow. Section 5 shows the results and discussion. Section 6 is the conclusion and future work.

II. Logistic Regression Model and Hyperparameters

Logistic regression is a machine learning algorithm used for classification problems. It utilizes the odds ratio to model the value of a binary or a multinomial dependent variable. The hypothesis used by logistic regression is given by:

$$h(\theta^{T}X) = P\,(Y = 1, X1, X2, \cdots, Xn) = \frac{1}{1+e^{-\theta^{T}x}} \quad (1)$$

Where Y is the dependent variable to be modelled, $X1, X2, \cdots, Xn$ are the given independent predictors and θ is the weight vector for given predictors. Given m as sample size, logistic regression utilizes the cross-entropy function as a cost function which is expressed as:

$$\boldsymbol{J(\theta)} = \frac{-1}{m}\sum_{i=1}^{m}\begin{bmatrix} \boldsymbol{y^{(i)}\, log\left(h_\theta(x^{(i)})\right)} + \\ \boldsymbol{\left(1-y^{(i)}\right) log\left(1-h_\theta(x^{(i)})\right)} \end{bmatrix} \quad (2)$$

Optimizing the cost function is required to obtain the best weights yielding maximum performance for learning model. Actually, many algorithms are used for this purpose. In this paper, averaged stochastic gradient descent is used where the updated weights in each iteration are given by:

$$\boldsymbol{\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)} \quad (3)$$

Where $\boldsymbol{\alpha}$ is the learning rate, which really controls the learning process. The learning rate may be one of four categories supported by Python libraries which are constant, optimal, inverse scaling and adaptive learning rates [10]. For each category, initial learning rate **eta0** is required to specify the learning rate value in each iteration during the learning process. Also, to avoid overfitting logistic regression is regularized by adding a new term to equation (2) that is called penalty. There are three main types of penalties: L1, L2 and Elastic net [11,12,13]. The terms added are $\boldsymbol{\lambda \sum_{j=1}^{n} |\theta_j|}$ for L1, $\boldsymbol{\lambda \sum_{j=1}^{n} \theta_j^2}$ for L2 and $\boldsymbol{\lambda 1 \sum_{j=1}^{n} |\theta_j| + \lambda 2 \sum_{j=1}^{n} \theta_j^2}$ for Elastic net and $\boldsymbol{\lambda}$ is the regularization strength. Logistic regression is regularized in python libraries by choosing the appropriate penalty type.

Table 1: Logistic regression hyperparameters.

Hyperparameter	Type	Values	Description
penalty_type	Categoric	{l1, l2, elasticnet}	Type of regularization
l1_ratio	Continuous	[0,1]	Mixing for elastic net
learning_rate	Categoric	{constant, optimal, invscaling, adaptive}	Type of learning rate
eta0	Continuous	[0,2]	Initial learning
Alpha	Continuous	[0.00001,0.01]	Regularization strength
Warm_start	Categoric	{True, False}	Reuse previous fit

However, when the elastic net is selected, then a new parameter that called 1_ratio is used to determine regularization strength. Also, warm Strat is another setting that is used to allow or disallow the reuse of the previous fit. Table 1 lists the most important hyperparameter for logistic regression model used in this paper.

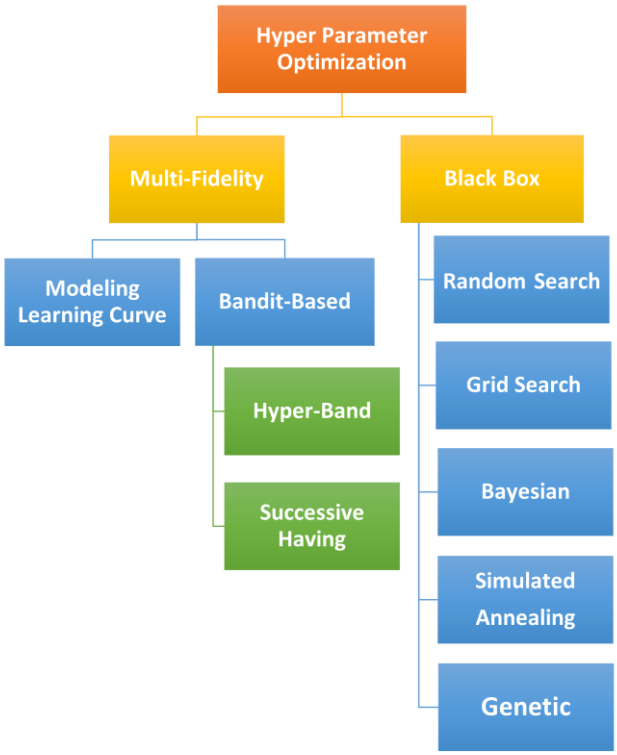

Figure 1: Classification of Hyperparameter Optimization techniques [5].

III. Black-Box Optimization Techniques

Black-box optimization is a general optimization technique used to optimize functions with just a known output. So, it doesn't have to make many assumptions about the problem being optimized that making it widely applicable to different areas [14]. Black-box algorithms are widely applied in machine learning for hyperparameter optimization by minimizing the model's cost function (e.g., cross-entropy function) over the hyperparameter space. This section introduces the most common Black-box techniques which are applied in this paper.

A. Grid Search

Grid search simply selects optimal hyperparameters by testing all possible combinations among all given hyperparameters [15]. It is guaranteed to find the optimal solution if it exists in its hyperparameter space. Many contributions have utilized grid search for hyperparameter optimization [16, 17]. Despite it is widely used in hyperparameter optimization, it has many limitations making it unfavorable. First of them is the inability to work over hyperparameters with continuous distribution as this result in infinite combinations among hyperparameters.

Discretizing the continuous range with regular intervals may help solving this problem. However, grid search may lose the optimal solution [18]. Another limitation of grid search is the expensive cost in terms of time and computational resources required to search all hyperparameters combinations in its grid, especially with high dimensional hyperparameter space and large datasets [19].

B. Random Search

As its name implies, and on the contrary to grid search, not all combinations of the hyperparameters are tested to get the optimal solution. The number of combinations to be tested are selected as a random subset of the overall hyperparameters space and explicitly passed to random search with alternative number of iterations. Random search is simply a black-box optimization technique that is proven theoretically and empirically to compete grid search [20]. In another research [21], the classification performance of random search equates the performance of some meta-heuristic optimization techniques namely Genetic Algorithm, Particle Swarm, and Estimation Distribution Algorithm with lower computational cost. Random search is not guaranteed to return the optimal solution, but it can find a near optimal solution with much less cost in terms of search time when compared to grid search.

C. Bayesian Optimization

Bayesian optimization is another black-box optimization technique. It utilizes Bayes probability theorem by setting a prior probability distribution over the function being optimized and combine it with the sample information (also, called evidence) to get a posterior function [22]. Bayesian optimization reduces the number of iterations to get the optimal hyperparameters by using all information of previous evaluations of the function being optimized [23]. Recently, Bayesian optimization has gained a high popularity in the field of hyperparameter optimization, especially for deep learning architectures [24]. In addition, many contributions have been done to utilize Bayesian optimization in different applications [25, 26]. For modeling the objective function, Bayesian optimization utilizes many probability distributions. Where Gaussian process (GP) is assumed to be the most suited distribution. It is used as a prior distribution for Bayesian optimization [27]. Recently, new models are gaining more popularity and proved its effectiveness. This makes it supported in hyperparameter optimization libraries such as python's Hyper-opt.

C.1. Bayesian Tree Parzen Estimator

One of these models is the Tree Parzen Estimator (TPE) which replaces the prior probabilistic distribution by a non-parametric density [28]. As a result of using the density estimator, TPE is able to use both continuous and discrete hyperparameter spaces [29]. Actually, TPE is structured as a tree keeping all conditional dependencies, allowing it to support hyperparameter spaces with conditional variables [30].

C.2. Simulated Annealing

Another model utilized by Bayesian optimization is Simulated Annealing (SA). It is a probabilistic and metaheuristic technique that is capable of optimizing a given function in a process simulating the material annealing. Recently SA has been used in hyperparameter optimization in different application areas [31,32]. During optimization of a function, SA moves randomly. If this movement improves the solution, then SA accepts it [33].

IV. WORKFLOW AND DATASET

In this paper, Wisconsin Breast Cancer Dataset (WBCD) has been used. It contains 32 numeric features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass [9]. WBCD contains 569 samples with 212 among them classified as Malignant (M) while the rest 357 are classified as Benign (B).

The proposed system starts with loading the dataset and preparing. All features were scaled to the range [0,1] using the min-max scaler by applying the following formula: $x` = \frac{x - min(x)}{max(x) - min(x)}$ (4)

Where x is the feature value before scaling, $x`$ is the feature value after scaling, $min(x)$ is the minimum feature value and $max(x)$ is the maximum feature value. After pre-processing, the dataset was split through train-test split procedure. In this work, the train-test split ratio has been chosen as 80-20%. The test set is preserved unseen for evaluation of the final optimized model. The training set is used by cross validation procedure and the optimizer for hyperparameter estimation. During the training stage, after initializing the parameter space, the training set is divided to 10 folds using k-fold cross validation. Then, one of the four used optimizers (Grid search, Random search, TPE, SA) is chosen to fit the model and estimate the hyperparameters at the different number of iterations. Finally, the best hyperparameters estimated by the optimizer are used to retrain the logistic model and estimate its performance using the previously reserved test set. These steps and their sequence are displayed in Figure 2.

V. RESULTS AND DISCUSSION

An important parameter for TPE, SA and random search is the number of iterations. It represents number of combinations that are sampled to get an estimation of the hyperparameters. When the number of iterations increases, the quality of the solution increase, but also the time required to find the solution increase. So, the performance of these algorithms should be studied with increasing the number of iterations. The time required to achieve the optimal solution by the different optimizers has been evaluated. All experiments are carried out on a second-generation machine with 8 GB RAM, and 2.4 GHZ Core i5 processor and 64-bit windows 10 as operating system. Each experiment is repeated for 10 times and their results are averaged.

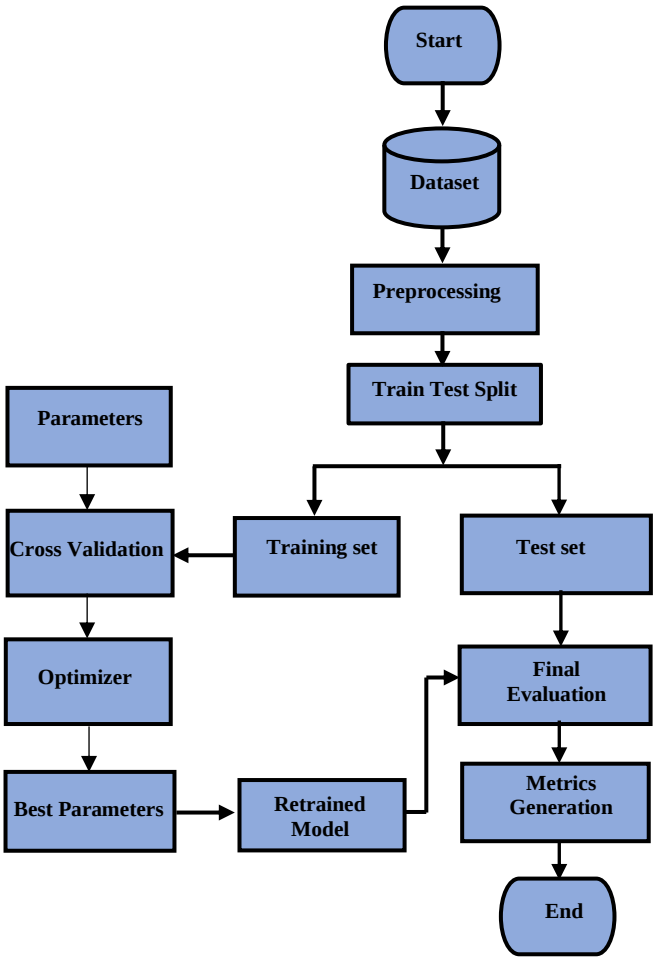

Figure 2: The proposed system.

A. *Performance metrics variation with iterations*

In this section, the variation of the performance metrics for the optimized model has been studied with increasing number of iterations. Confusion matrix parameters which are True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) are the main parameters from which classification metrics such as Accuracy, Precision, Recall and F1 score are computed [34]. Another metric is Matthew's Correlation Coefficient (MCC) which is used in evaluating classification models with imbalanced dataset to indicate how much the predictive model is better than random guess [35]. Another metric is cohesion's Kappa which measures the agreement between predicted and true classes [36]. Table 2 lists these metrics with their expressions as calculated based on confusion matrix parameters.

Several experiments have been carried out to evaluate the mentioned parameters at different iterations. The result showed that by increasing the number of iterations, the performance has been improved as a result of reaching more better values of the optimized hyperparameters. Also, the results showed that at the 8th iteration, the performance of TPE in terms of test accuracy, kappa an MCC reached its maximum value, then settled down while SA reached at the 17th iteration, but it fell down, then regained its performance and settled down at the 33rd iteration and finally random search reached its maximum performance then settled down at the 35th iteration. Figures 3,4 and 5 shows the variation of test accuracy, kappa and MCC metrics with the number of iterations for TPE, SA and Random search algorithms. Table 3 lists the values of hyperparameters obtained by each algorithm after reaching steady state which represents the optimal hyperparameters for each algorithm.

Table 2: Classification metrics.

Metric	Expression
Accuracy	$\frac{TP + TN}{TP + TN + FP + FN}$
Precision	$\frac{TP}{TP + FP}$
Recall	$\frac{TP}{TP + FN}$
F1 Score	$\frac{2 * precision * Recall}{Precision + Recall}$
MCC	$\frac{TP.TN - FP.FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$
Kappa	$\frac{Po - Pe}{1 - Pe}$

Table 3: Hyperparameter result of the different optimizers.

Hyperparameter	Optimized values of Hyperparameters			
	Grid	Random	Anneal	TPE
penalty_type	Elastic net	L2	L2	L2
l1_ratio	0.1	-	-	-
learning_rate	Adaptive	Adaptive	Adaptive	Optimal
eta0	1.71	1.118	1.712	0.04099
Alpha	0.00031	0.000433	0.00053	0.000255
Warm_start	True	False	True	False

B. *Iterations and time required to get the optimal solution*

An important factor in the comparison between optimization algorithms is the speed of finding an optimal or near optimal solution. A number of experiments have been carried out to measure the number of iterations to reach the best solution for each algorithm. The number of iterations required to reach the maximum test accuracy have been recorded. The results showed that TPE reached to the best performance at 8th iteration only while simulated annealing reached at the 33rd iteration and finally the random search reached at the 35th iteration. This means that applying TPE, to get the optimal hyperparameters contributed in reduction of iterations by 75.75 % compared to SA and by 77.1 % when compared to random search. Figures 6 compares the number of iterations to get the

optimal performance for TPE, SA and Random search algorithms. Also, the time required for reaching optimal performance is recorded. The results show that TPE takes 0.81 seconds to reach the optimal performance while SA takes 4.0391 seconds and Random search takes 5.85 seconds. Finally, Grid search needs 3705.29 seconds to reach the optimal solution which is a very long time when compared to other algorithms. This means that applying TPE resulted in reduction of time required to reach maximum performance by 79.9%, 86.1%, and 99.9% compared to SA, Random search and grid search respectively. Figure 7 compares training time of TPE, SA and Random search algorithms.

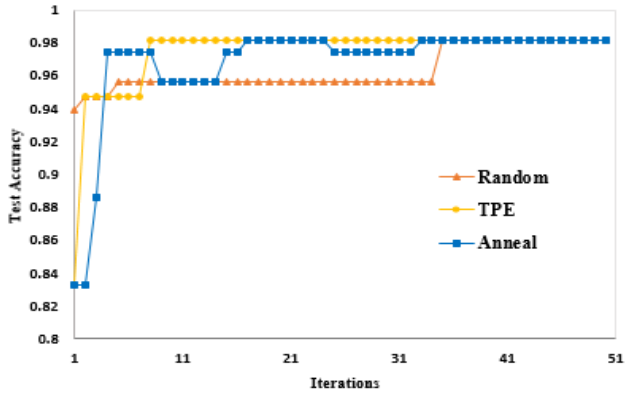

Figure 3: Test Accuracy variation with Iterations for each optimizer.

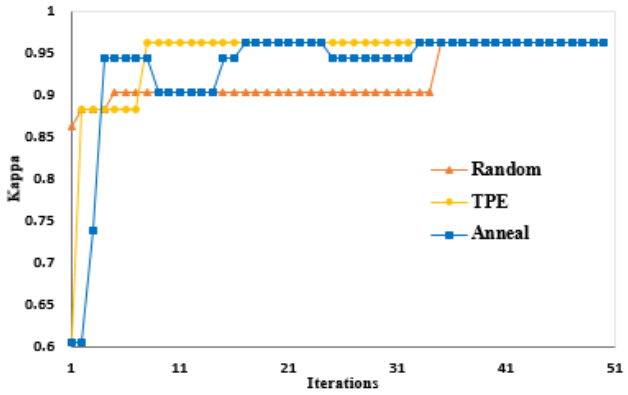

Figure 4: Kappa statistic variation with Iterations for each optimizer.

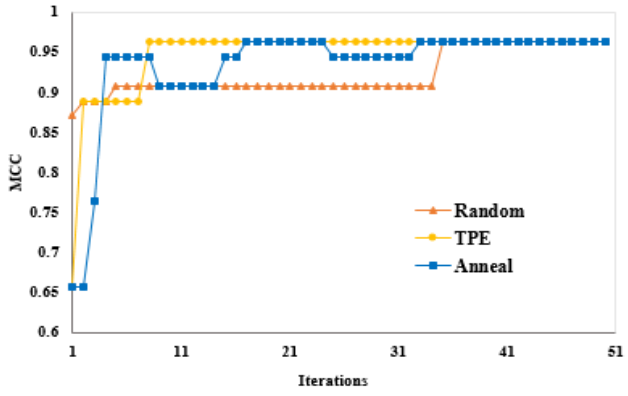

Figure 5: MCC variation with Iterations for each optimizer.

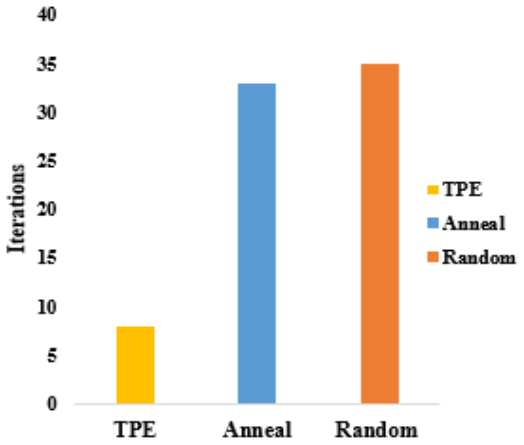

Figure 6: Iterations taken by each optimizer to get Optimal.

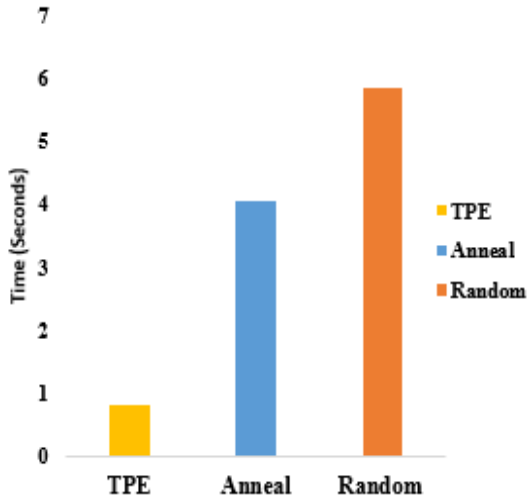

Figure 7: Time taken by each optimizer to get Optimal.

C. Training Time variation with iterations

In this section, the relation between the number of iterations and the training time of the logistic regression model when tuned by TPE, SA and Random search optimizers is studied. A number of experiments have been carried out to measure the training time taken by each algorithm with varying iterations. To measure the time at each iteration, each experiment is repeated for 10 times and their time is averaged. Figure 8 illustrates the variation of time with a number of iterations for TPE, SA and Random search algorithms. The results show that with increasing the number of iterations, training time increased in a linear fashion with the three algorithms but with different slopes. Increasing number of iterations had the minimum effect on the Random search that had the line with minimum slope. SA algorithm had the worst time performance as a result of the maximum increase in time with increasing the number of iterations. Also, TPE had a training time closer to that of the Random search but much better than the SA

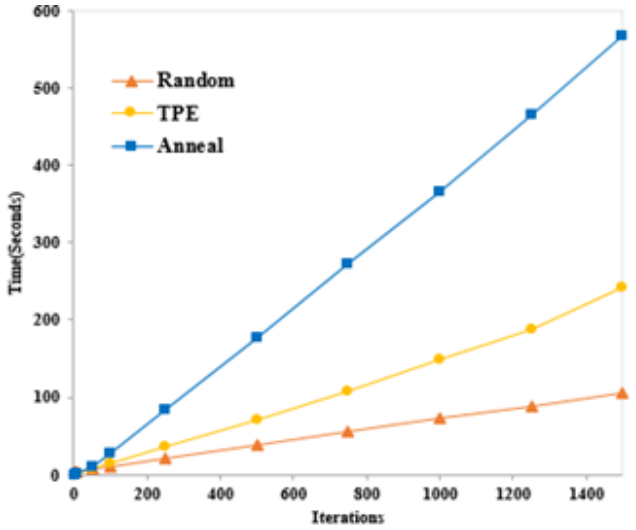

Figure 8: Training Time variation with Iterations for each optimizer.

D. Evaluation of the final optimized model

After tunning the logistic regression hyperparameters, the resulted optimal hyperparameters are used again to build a model for cancer classification using WBCD data set and logistic regression. This model is tested using the previously reserved test set and all metrics in Table 2 are measured. The resulted evaluation metrics of this model is listed in Table 4.

Table 4: Final Evaluation metrics of the optimized model.

Train Acc.	Test Acc.	Kappa	MCC	F1 Score	Precision	Recall
0.982	0.9825	0.962	0.963	0.99	0.98	0.98

To assess the effect of tunning the hyperparameters, the optimized model is compared with the non-optimized model using the performance metrics in Table 2. The results showed that the optimized model outperformed the non-optimized model by 1.15 %, 1.75 %, 3.7%, 3.8%, 3%, 2% and 2% in training accuracy, test accuracy, kappa, MCC, f1 score, precision and recall, respectively. Figure 9 draws a comparison between the optimized and non-optimized models according to the discussed metrics.

Also, as a classifier it has been compared with other classifiers using WBCD dataset. Vivek K. et al. [37] introduced an implementation of different classification techniques for cancer using WBCD dataset. This implementation included Boost M1, Decision Table, J-Rip, J48, Lazy IBK, Lazy K-star, Logistic Regression, Multilayer–Perceptron, Random Forest and Random Tree algorithm. They evaluated these algorithms using 10-fold cross-validation. Also, Md. Imran in [38] compared the performance of Naive Bayes (NB), Support Vector Machine (SVM) and Artificial Neural Network (ANN) each with different configuration using WBCD and 10-fold cross-validation. The proposed optimized logistic regression classifier outperforms SVM with linear kernel in [38] by 1.53%, by 2.37% when compared to ANN with radius basis function, and by 2.34 % when compared to Gaussian NB. Moreover, the proposed optimized logistic regression model outperforms some of Vivek K. et al. [37] classifiers by 1.12%. 0.83%, 15.44%, 15.44% ,15.16 %, and 12.15% when compared to logistic regression, MLP, J48 Free, Ada-Boost M1, Decision Table and J-Rip. On the contrary, their Lazy k-star, Lazy IBK and Random Forest algorithms outperforms the proposed optimized model by 0.89%. Figure 10 draws illustrates a comparison between the proposed optimised model and all other classifiers.

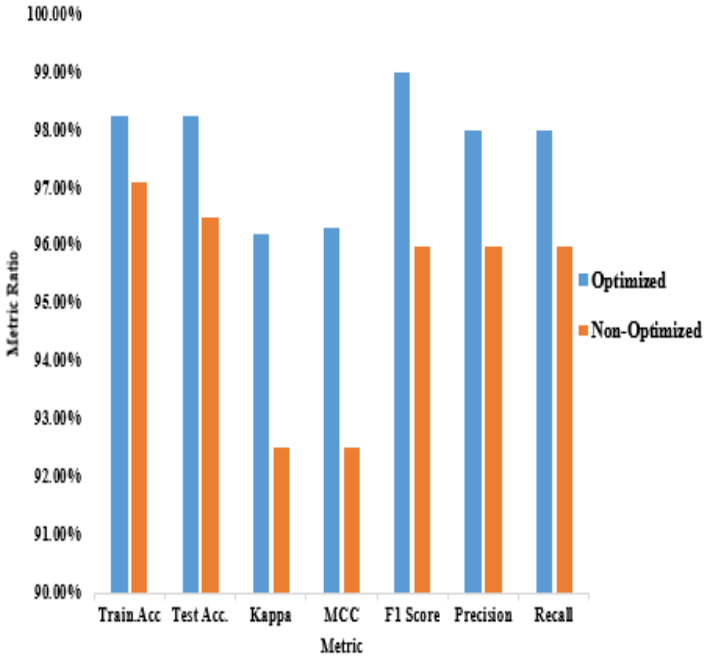

Figure 9: Evaluation Metrics of optimized and non-optimized models.

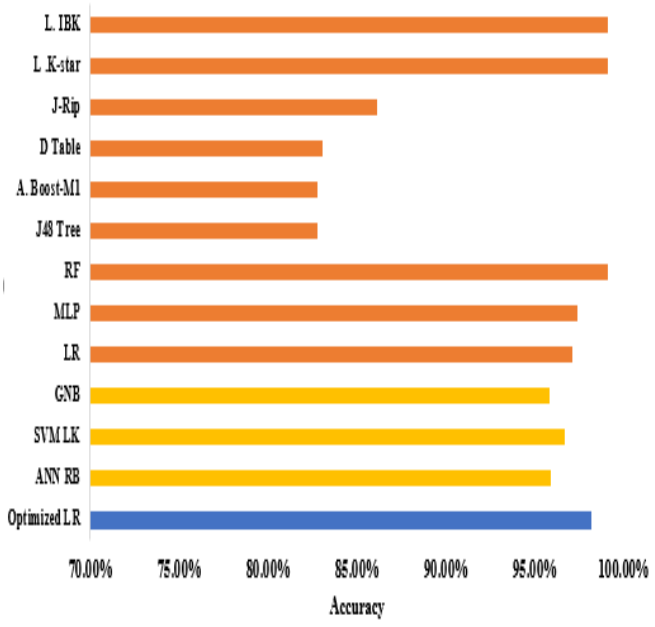

Figure 10: Test Accuracy of Optimized Model VS Other Classifiers.

VI. Conclusion and Future work

In this paper, the performance of common black-box techniques namely Random search, Grid search, TPE and SA using WBCD dataset has been studied. TPE was proved to outperform other algorithms in terms of iterations and time requiring to get an optimized solution for logistic regression hyperparameters. TPE reduced number of iterations to get the optimized hyperparameter values by 75.75 % compared to SA and by 77.1 % when compared to Random search. Also, it reduced time taken 79.9 % compared to simulated annealing and by 86.1 % when compared to random search and by 99.9 % when compared to grid search. The resulted hyperparameter values have been utilized to learn a logistic regression model to classify cancer using WBCD dataset. The optimized model succeeded in classifying cancer with 98.2% for test accuracy, 0.962 for kappa statistic and 0.963 for MCC metrics when evaluated using 10-fold cross validation. The future work will include other optimization techniques namely genetic algorithms and more complicated classifiers such as Deep Neural Networks (DNN) applied to higher dimensional datasets.

References

[1] Yang, L and Shami, A. "On Hyperparameter Optimization of Machine Learning Algorithms: Theory and Practice". Neurocomputing, vol. 415, PP. 295-316, July 2020.

[2] Kuhn M. and Johnson K. "Over-Fitting and Model Tuning". In: Applied Predictive Modeling, Springer, New York, NY, vol.1, PP. 64-65, 2013.

[3] Van Rijn, J. N and Hutter, F. "Hyperparameter Importance Across Datasets". ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, United Kingdom. PP. 2367–2376, August 2018.

[4] P. Probst, A. Boulesteix and B. Bischl, "Tunability: Importance of Hyperparameters of Machine Learning Algorithms". Journal of Machine Learning Research, vol. 20, PP. 1-31, 2019.

[5] R. Shawi, and S. Sakr, "Automated Machine Learning: Techniques and Frameworks" In: Big Data Management and Analytics, 9th European Summer School, eBISS 2019 Berlin, Germany, PP. 40-69, June 30 – July 5, 2019.

[6] H. Ghanbari and K. Scheinberg, "Black-Box Optimization in Machine Learning with Trust Region Based Derivative Free Algorithm". ArXiv, vol abs/173.06925, 2017.

[7] K. Kandasamy, K. Raju, W. Neiswanger, B. Paria, C.R. Collins, J. Schneider, B. Poczos and E. P. Xing, "Tuning Hyperparameters without Grad Students: Scalable and Robust Bayesian Optimization with Dragonfly", Journal of Machine Learning Research, vol.21, PP. 1–27, Mar 2020.

[8] Hu, Y.Q., Yu, Y., Tu, W.-W., Yang, Q., Chen, Y., and Dai, W. "Multi-Fidelity Automatic Hyper-Parameter Tuning via Transfer Series Expansion". Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, PP.3846–3853, 2019.

[9] https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic) last visited 28/382021.

[10] https://scikit-learn.org/stable/modules/sgd.html, last visited 26/3/2021.

[11] R.J. Tibshirani, "Regression shrinkage and selection via the lasso". Journal of the Royal Statistical Society. Series B (Methodological) vol. 58 (1), PP.267–288, 1996.

[12] E. Hoerl and R.W. Kennard. "Ridge Regression: Biased Estimation for Nonorthogonal Problems", Technometrics Vol. 12 (1), PP.55–67. 1970.

[13] H. Zou and T. Hastie," Regularization and Variable Selection via the Elastic Net". Journal of the Royal Statistical Society: Series B (Statistical Methodology), vol. 67 (2), PP. 301−320, 2005.

[14] D. Golovin and Benjamin Solnik and Subhodeep Moitra and G. Kochanski and John Karro and D. Sculley "Google Vizier: A Service for Black-Box Optimization", ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, PP. 1487-1495, August, 2017.

[15] 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(1), PP. 160–180, Jul 2019.

[16] B. H. Shekar, and G, Dagnew, "Grid Search-Based Hyperparameter Tuning and Classification of Microarray Cancer Data". Second International Conference on Advanced Computational and Communication Paradigms (ICACCP), PP. 1-8, 2019.

[17] I. Syarif, A. Pr and G. Wills, "SVM Parameter Optimization using Grid Search and Genetic Algorithm to Improve Classification Performance", TELKOMNIKA, vol 14, PP. 1502-1509, 2016.

[18] C. Matache and J. Passerat, and B. Kainz, "Efficient Design of Machine Learning Hyperparameter Optimizers", MEng Individual Project, Imperial College London, PP.17-18, 2019.

[19] J. Waring, C. Lindvall and R. Umeton, " Automated machine learning: Review of the state-of-the-art and opportunities for healthcare", Artificial Intelligence in Medicine, vol 104, PP. 1-12, April 2020.

[20] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," Journal of Machine Learning Research., vol. 13, PP. 281-305, Mar. 2012.

[21] R. G. Mantovani, A. L. D. Rossi, J. Vanschoren, B. Bischl and A. C. P. L. F. de Carvalho, "Effectiveness of Random Search in SVM hyper-parameter tuning," 2015 International Joint Conference on Neural Networks (IJCNN), Killarney, Ireland, PP. 1-8, 2015.

[22] E. Brochu, Vlad M. Cora and N. D. Freitas, "A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modelling and Hierarchical Reinforcement Learning". ArXiv, vol abs/1012.2599, 2010.

[23] J. Snoek, H. Larochelle and R P. Adams ,"Practical Bayesian optimization of machine learning algorithms" , Proceedings of the 25th International Conference on Neural Information Processing Systems ,PP. 2951–2959, December 2012.

[24] M. Feurer, F. Hutter "Hyperparameter Optimization." In Automated Machine Learning. The Springer Series on Challenges in Machine Learning. Springer, Cham, PP. 3-33, may 2019.

[25] G. E. Dahl, T. N. Sainath and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, PP. 8609-8613, 2013.

[26] G. Melis, C. Dyer and P. Blunsom, "On the State of the Art of Evaluation in Neural Language Models", 6th International Conference on Learning Representations (ICLR) ,2018.

[27] J. Wu, X. Chen, H. Zhang, L. Xiong, H. Lei and S. Deng," Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization", Journal of Electronic Science and Technology, vol 17 (1), PP. 26-40, Mar 2019.

[28] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kegl, "Algorithms for Hyper-Parameter Optimization", Proceedings of the 24th International Conference on Neural Information Processing Systems, PP. 2546–2554, December 2011.

[29] S. Falkner, A. Klein and F. Hutter, "BOHB: Robust and Efficient Hyperparameter Optimization at Scale", Proceedings of the 35th International Conference on Machine Learning, PP. 80:1437-1446, 2018.

[30] I. Dewancker, M. McCourt, and S. Clark, "Bayesian optimization primer", 2015. URL https://app. sigopt. com/static/pdf/SigOpt_ Bayesian_Optimization_Primer. pdf.

[31] C. Tsai, C. Hsia, S. Yang, S. Liu and Z. Fang, “Optimizing hyperparameters of deep learning in predicting bus passengers based on simulated annealing", Applied Soft Computing, vol 88, 2020.

[32] N. Pathik and P. Shukla, "Simulated Annealing Based Algorithm for Tuning LDA Hyper Parameters", In: Theories and Applications. Advances in Intelligent Systems and Computing, Springer, Singapore. vol 1154, PP. 515-521, June 2020.

[33] A. Gülcü and Z. KUş, "Hyper-Parameter Selection in Convolutional Neural Networks Using Microcanonical Optimization Algorithm," IEEE Access, vol. 8, PP. 52528-52540, Feb 2020.

[34] D.M.W Powers, “Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation”, Journal of Machine Learning Technologies, vol. 2 (1), PP.37–63,2011.

[35] Y. Jiao and P. Du, “Performance measures in evaluating machine learning based bioinformatics predictors for classifications”, Quantitative Biology, vol.4, PP. 320–330, 2016.

[36] M. Grandini, E. Bagli and G. Visani, “Metrics for Multi-Class Classification: An Overview”, ArXiv, vol abs/2008.05756, August 2020.

[37] V. Kumar, B.K. Mishra, M. Mazzara, D. N. Thanh and A. Verma, “Prediction of Malignant and Benign Breast Cancer: A Data Mining Approach in Healthcare Applications. In: Advances in Data Science and Management”. Lecture Notes on Data Engineering and Communications Technologies, vol. 37, Singapore, Springer, 2020.

[38] M. I. H. Showrov, M. T. Islam, M. D. Hossain and M. S. Ahmed, "Performance Comparison of Three Classifiers for the Classification of Breast Cancer Dataset," 2019 4th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, PP. 1-5, 2019.