Software Defect Prediction Based on Tree-structured Parzen Estimator Using Machine Learning Classifiers

Faiza Khan[1], Sultan Almari[2,*], Muhammad Haseeb Khan[3], and Summrina Kanwal[4]

[1]Riphah International University, Faculty of Computing, Islamabad 45211, Pakistan
[2]Department of Computing and Informatics, Saudi Electronic University, Riyadh 11673, Saudi Arabia
[3]Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur, Pakistan
[4]Center for Applied Intelligent Systems Research, Halmstad University, Sweden
Email: khanfaiza706@gmail.com(F.K.); salamri@seu.edu.sa (S.A.); summrina@gmail.com(S.K.)
[*]Corresponding author
Manuscript received January 10, 2023; revised January 25; accepted February 9, 2023; published June 21, 2024.

***Abstract*—Software testing is the most significant task in software development and it takes maximum amount of time, cost, and effort. Therefore, to decrease these resources SDP is utilized to improve the work of the SQA process with the help of predicting faulty or defective components. Numerous methods have been proposed by researchers to predict defective components but these methods generate partial results when applied to imbalanced data sets. An imbalanced dataset has non-uniform class distribution with very limited illustrations of a precise class as compared to that of the other class. The usage of imbalanced datasets leads to off-target predictions of the smaller class, that are usually considered to be more significant than the mainstream class. Thus, handling imbalanced data and HPO efficiently is important for the successful development of a capable bug prediction model. In this paper SDP model is anticipated that utilizes different machine learning classifiers with Tree-structured Parzen Estimator Method (TPE) as hyperparameter optimizer to enhance defect prediction accuracy through HPO and SMOTE algorithm to solve class imbalance issue. The proposed method was evaluated on eighteen software defect datasets from the promise repository. Experimental results demonstrated that the proposed technique achieved improved accuracy than when the classifiers are used with default parameters.**

***Keywords*—Software bug prediction (SDP), Tree-structured Parzen Estimator Method (TPE), Synthetic Minority Over-sampling Technique (SMOTE), Hyperparameter optimization (HPO)**

I. Introduction

Software Defect Prediction (SDP) helps in detecting possible upcoming faults. Effective defect prediction helps in finding the parts in software that had caused the flaws in software and helps in decreasing maintenance costs and the resources required. As with time, the complexity of software is increasing, defect prediction (DP) timely, and Software Quality Assurance (SQA) of projects is becoming a tough task. To solve this issue in SDP, different machine learning (ML) methods have been utilized by researchers. But still, most of the software defect data is imbalanced and also mostly ML classifiers use their default setting. These issues of SDP have gained a lot of interest from researchers in the Software Engineering (SE) community. Imbalanced Data means that the amount of faulty class (minority class) is fewer than the non-faulty class (majority class). This imbalanced data misleads classifiers while learning the faulty class appropriately and later the results obtained are unfair and imprecise. A better DP model is trained on the same number of occurrences of faulty and non-faulty classes [1]. Hyperparameters (HP) are parameters that are optimized for ML classifiers to enhance their performance. Hyperparameter optimization (HPO) or HP tuning is the method of optimizing the HP of ML models or the method of finding the finest HP values. Every classifier has diverse features that need to be optimized [2]. It has been assessed by Fu, Menzies *et al.* [3] that 80% of the utmost cited SDP studies depend on the defaulting parameters. The default parameter configurations greatly affect the performance of DP models due to which they underperform.

The main inspiration behind our anticipated algorithm for HP and class imbalance is from Malhotra and Jain [1], Shen, Cai *et al.* [4], Khan, Kanwal *et al.* [2], Tantithamthavorn, McIntosh *et al.* [5], and Kanwar, Awasthi *et al.* [6]. Malhotra and Jain [1] used six oversampling methods and four undersampling techniques on different datasets to resolve the class imbalance issue. Shen, Cai *et al.* [4] utilized a Bayesian optimization algorithm to optimize the HP of the random forest model. Khan, Kanwal *et al.* [2] used Artificial Immune Network (AIN) to optimize the HP of seven ML classifiers. Tantithamthavorn, McIntosh *et al.* [5] examined caret automated parameter optimization, grid search (GS), random search (RS), genetic algorithm (GA), and differential evolution (DE) on 20 ML classifiers to optimize defect prediction models. Kanwar, Awasthi *et al.* [6] utilized a Bayesian optimization algorithm to optimize the HP of the LightGBM algorithm and borderline-SMOTE technique to solve the class imbalance issue. Feng, Keung *et al.* [7] investigated the stability of SMOTE-based oversampling methods to improve the firmness of SMOTE-based oversampling methods in SDP. Furthermore, it is stated that when algorithms like GA, RS, GS, and DE are utilized for HPO problems it maximizes computational costs and causes over-fitting. However, in these studies, they did not study the class imbalance issue and HPO of the classifier together instead they only focus on HPO or class imbalance problem. We intended to overcome these difficulties by experimenting with SMOTE to tackle class imbalance issues and Tree-structured Parzen Estimator Method (TPE) for HPO and we have used the defect prediction dataset.

The contributions made in this paper are to tackle the class imbalance issue using SMOTE on the SDP dataset and to examine the effect of HPO of ML classifiers using TPE on SDP using accuracy as performance measure.

The structure of the paper is organized as follows. Section II explains related work, Section III delivers the background information about the methods utilized in this paper. The anticipated technique is described in Section IV. The results

of our experimental study are described in Section V and the conclusion and upcoming work are shown in Section VIII.

II. Literature Review

Numerous studies have anticipated using ML methods for SDP. In maximum studies, the HP of these ML classifiers are left to their defaulting values. Nevertheless, in very limited studies the outcome of HPO on defect prediction has been inspected. Malhotra and Jain [1] inspected six oversampling and four undersampling approaches with 15 ML methods and results were validated statistically utilizing the Friedman test and Nemenyi post hoc analysis. They have reported that these resampling methods significantly enhanced the performance of ML methods. Khan, Kanwal *et al.* [2] investigated ML classifiers with the Artificial Immune Network (AIN) for HPO. They used seven machine learning classifiers and the method was evaluated on a bug prediction dataset. They have reported that the HPO of ML classifiers, using AIN for SDP had performed better than classifiers when their HP are not optimized. Shen, Cai *et al.* [4] investigated a Bayesian optimization algorithm to optimize the HP of the random forest model on the SDP dataset. They have stated that their proposed method outperformed. Tantithamthavorn, McIntosh *et al.* [5] examined automatic parameter optimization on 20 ML methods using GS, RS, GA, and DE for defect prediction models on SDP datasets from the National Aeronautics and Space Administration (NASA) repository. They have reported that automatic parameter optimization enhances accuracy by up to 40%. Kanwar, Awasthi *et al.* [6] used a Bayesian optimization algorithm to optimize the HP of the LightGBM algorithm and borderline-SMOTE technique to solve the class imbalance issue. And they have further compared their proposed model with XGBoost and random forest (RF) algorithm. They have reported that their proposed model had achieved an accuracy of 99.12%. Feng, Keung *et al.* [7] investigated the stability of SMOTE-based oversampling methods to improve the firmness of SMOTE-based oversampling methods in SDP. They have reported that the performance of stable SMOTE-based oversampling methods is improved than SMOTE-based oversampling methods. Balogun, Lafenwa-Balogun et al. [8] used SMOTE and homogeneous ensemble (Bagging and Boosting) methods for SDP. They have used Decision Tree (DT) and Bayesian Network (BN) as base classifiers. They have reported that their proposed method outperformed other methods. They have achieved an accuracy of 86.8%. Bahaweres, Agustian *et al.* [9] investigated a combination of Neural Network (NN) and SMOTE in which the HP of SMOTE and NN are optimized utilizing RS to tackle the class imbalance issues in six NASA datasets. They have reported that Bal increases by 25.48% and Recall by 45.99% compared to the original NN. They have also compared their proposed method performance with Traditional ML-based SMOTE. Xie, Xie *et al.* [10] used SMOTE to create a balanced sample dataset and oversample the defective components in the unbalanced sample dataset. They have also proposed a method that uses regression after classification, and the proposed method is applied to support vector machines (SVM) to classify components. The experiment was evaluated on open-source datasets. They have stated that the accuracy anticipated method is improved than the prediction by regression alone.

Based on the findings from literature review of the SDP problem it is specified that in maximum defect prediction studies, HP values are not optimized in ML classifiers due to which these classifiers did not perform well. It was evident from the literature that 80% of the SDP studies depend on defaulting HP values and also HPO was not discovered in the imbalance problem of SDP. In divergence, our study objectives are to combine SMOTE and several ML classifiers with HPO using TPE. We have also inspected the impact of HPO on classifiers by comparing the prediction accuracy of these classifiers before and after HPO and also before and after class imbalance problems.

III. Background

Here we have deliberated the methods, TPE, ML classifiers, and HP in detail that are utilized in this paper.

A. Synthetic Minority Oversampling Technique (SMOTE)

SMOTE was first presented by Chawla, Bowyer *et al* [11] which is an oversampling technique that is utilized to solve the issue of class imbalance. To resolve the class imbalance problem, SMOTE technique usually enhances the quantity of data to the minority class with synthetic data. Synthetic data was attained from k-NN (k-nearest neighbor). It has 2 parameters that are tuned such as ratio, and the number of neighbors. The ratio parameter is the amount of major and minor classes whereas the number of neighbors is the amount of nearby neighbors to generate syntactic data. Synthetic data can be produced through different processes such as on numerical and category scales. For numerical data calculation, Euclidian distance is used; while for categorical data calculation mode values are used [9, 12]. The algorithm starts by selecting KNNs then synthetic data is created by taking the variance among the feature vector of the example under deliberation and its nearest neighbor. Then it increases the variance by a random figure among 0 and 1 and enhances it to the feature vector under contemplation. So, a random point is carefully chosen with the line segment among two precise features. Therefore, SMOTE widens the data region of the alternative class instances and forces the result area of the class to develop more generally. The flowchart of SMOTE is described in Fig. 1 below:

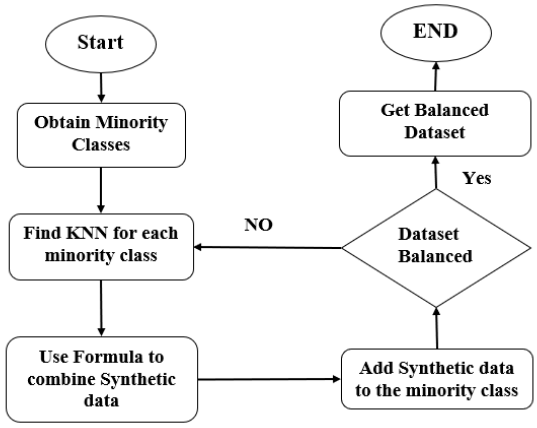

Fig. 1. Flow Chart of SMOTE Algorithm.

B. Machine learning (ML) classifiers

In this study, we have utilized five ML classifiers such as

KNN, SVM, NB, RF, and XGBoost.

KNN is a classification-based ML classifier that classifies an instance to the adjacent class based on its maximum number of neighbors. The KNN stores all the accessible data and categorizes an innovative data point founded on the resemblance measure e.g., distance functions. This means that when new data appears it can be simply classified into a well-suited group by KNN [2]. In this we have utilized three distance metrics namely Euclidean, Manhattan, and Minkowski metrics for KNN. SVM is a geometrically-inspired ML classifier that utilizes a hyperplane to distinguish two classes by selecting the finest by separating hyper-plane to categorize data linearly. It is trained to create a model and after the model is evaluated. If the example data is not separated linearly then approximately additional approaches are castoff. In this paper RBF, Polynomial, Sigmoid, and Linear kernel methods are used. NB is a probability-based ML classifier based on Bayes Theorem. It works by defining the connection between the probabilities of an incident that is presently happening with the probability of an alternative incident that has previously happened. RF is an ML classifier that creates a forest with numerous trees. In RF, the amount of trees is directly compared to the resultant accuracy; that is if the amount of trees in the forest is bigger, the accuracy of the results is also greater. When generating a forest of trees, the RF initially constructs each tree separately via bagging and feature randomness [13, 14]. XGBoost stands for Extreme Gradient Boosting. It offers parallel tree boosting and is an important ML library for regression, and classification problems.

We have chosen these classifiers for SDP because they operate differently and have many hyperparameters that need to be optimized.

C. Tree-structured Parzen Estimator Approach (TPE)

The TPE optimization is mostly similar to Bayesian optimization [15]. The TPE algorithm is accessible as a sequential model-based optimization (SMBO) method that will tackle the problem of HPO. The TPE method takes an evaluation function f(θ) that alters the setting area into a non-parametric Parzen-window density estimate. This setting area is demonstrated by uniform distribution, discrete uniform distribution, or logarithmic uniform distribution. Later, these variations of settings will contribute to the TPE. For the recursive method, the TPE approaches $f(\theta)$ in setting area. The expected improvement (EI) standard is deployed to find the finest HP $\theta*$ from the examination area. This algorithm states that the probability distribution $p(\theta|y)$ by dividing the setting area into good and bad HP instances as [14–16]:

$$P(\theta|y) = \begin{cases} HP_{good}(\theta) if y < y^* \\ HP_{bad}(\theta)\ if\ y > y^* \end{cases}$$

where $HP_{good}(\theta)$and $HP_{bad}(\theta)$ are Parzen estimators utilized to evaluate the compactness designed by utilizing the observations θ if (θ_i) is fewer than or bigger than $y*$, correspondingly. In this $y < y*$ shows that the value of the $f(\theta)$ is fewer than the threshold, and $y > y*$ signifies that the value of the $f(\theta)$ is higher than the threshold. The ideal HP value θ is expressed as [16]:

$$\theta^* = argmin \frac{HP_{bad}(\theta)}{HP_{good}(\theta)}$$

The flow chart of the TPE approach is shown in Fig. 2 below [16]:

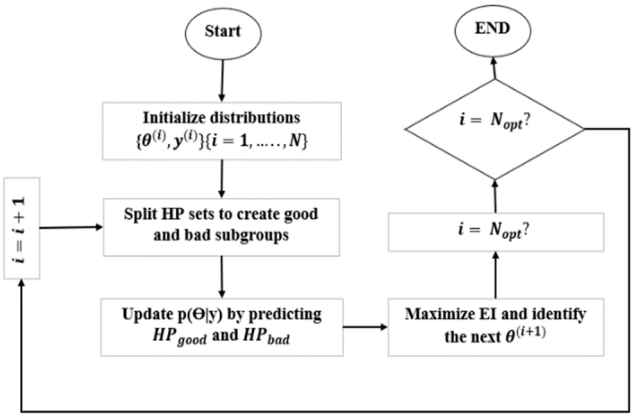

Fig 2: Flow chart of the TPE [16].

D. Software Defect Prediction (SDP) Dataset:

The dataset utilized in this model is an object-oriented (OO) class-level open-source dataset. This dataset is taken from the PROMISE repository of empirical SE data (http://promisedata.googlecode.com). It is containing 20 OO metrics as independent features and defects of class as the dependent variable. The feature information or metrics utilized in this dataset is provided in Table 1 below [17]:

Table 1. The arrangement of Channels

S.No	Attributes	Explanation
1	WMC	Weighted methods per class
2	DIT	Depth of Inheritance Tree
3	NOC	Number of Children
4	CBO	Coupling between object classes
5	RFC	Response for a Class
6	LCOM	Lack of cohesion in methods
7	LCOM3	Lack of cohesion in methods
8	NPM	Number of Public Methods
9	DAM	Data Access Metric
10	MOA	Measure of Aggregation
11	MFA	Measure of Functional Abstraction
12	CAM	Cohesion Among Methods of Class
13	IC	Inheritance Coupling
14	CBM	Coupling Between Methods
15	AMC	Average Method Complexity
16	Ca	Afferent couplings
17	Ce	Efferent couplings
18	CC	Cyclomatic complexity
19	Max(CC)	The greatest value of CC
20	Avg(CC)	The arithmetic mean of the CC

E. Hyperparameter optimization (HPO):

HP are constraints that are adjusted for ML classifiers to increase their classification accuracy. These constraints typically disturb the learning, construction, and assessment of ML classifiers. HPO is the procedure of finding the finest HP of ML classifiers and is also recognized as HP tuning [2, 18]. The HP of the considered classification methods that are adjusted are given in Table 2 below. To adjust the HP of these classifiers, TPE was utilized. Details of the HP for these classifiers using TPE are shown in Table 2 below:

Table 2. HP of the ML CLASSIFIERS that are optimized

ML Classifiers	Hyperparameters name	Default value	Optimized parameters value range
KNN	n_neighbors	5	5,7,9,11,13,15
	Weights	Uniform	Uniform, distance
	Metric	Minkowski	Minkowski, Euclidean, manhattan
SVM	C	1.0	0.1,1,100,1000
	Gamma	1	1, 0.1, 0.01, 0.001, 0.0001
	Kernel	RBF	rbf, poly, sigmoid, linear
	Degree	3	1,2,3,4,5,6
NB	var_smoothing	1e-9	1e-9, 1e-6, 1e-12
RF	n_estimators	100	100, 200, 300, 400,500,600
	max_depth	None	1, 15,50
	Criterion	Gini	gini, entropy
Xgboost	max_depth	6	3, 18, 1
	Gamma	0	1,9
	reg_alpha	0	40, 180, 1
	reg_lambda	1	0, 1
	colsample_bytree	1	0.5, 1
	min_child_weigh t	1	0, 10, 1
	n_estimators	100	180
	Seed	0	0

F. Performance Metrics:

The proposed model was assessed by using the following performance metric.

Accuracy: It is the number of correct guesses made as a ratio to all guesses made. The formula for accuracy is given below:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

IV. PROPOSED SMOTE-TPE HPO APPROACH:

We have proposed a TPE algorithm for the HPO of ML classifiers such as SVM, KNN, NB, RF, and XGBoost and balanced our data with SMOTE algorithm. This optimization method objective is to enhance the prediction accuracy of ML classifiers. The process of the proposed methodology is shown in Fig 3.

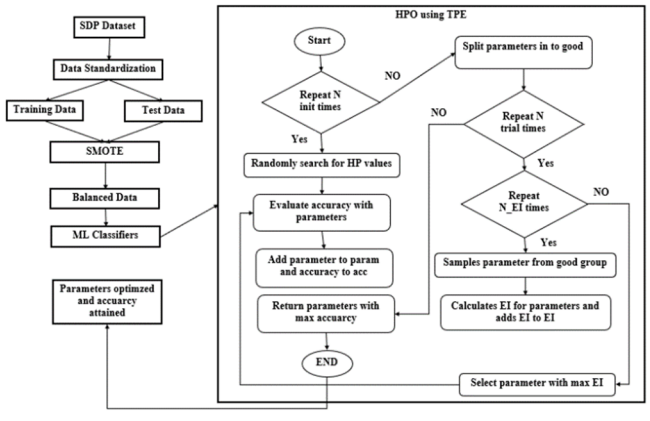

Fig. 3. Workflow of our proposed technique.

Our proposed methodology describes that the data preprocessing was first carried out employing Data Standardization in which the data was standardized to resize the distribution of values in this the mean of the detected values is 0 and the standard deviation is 1. We have used this because it will help in increasing the classifier accuracy. The expression for data standardization is shown below:

$$y = \frac{(x - \mu)}{\sigma}$$

where x is the instance, μ is the mean and σ is the standard deviation. After normalization, data was separated into 70% training and 30% testing data. Later we applied SMOTE to solve the class imbalance issue to the training and test data. For the defective class synthetic instances are generated so that it balances the non-defective class. The tools we used for SMOTE are imblearn.over_sampling. When the data is balanced using SMOTE then the training data was utilized to train the ML classifiers and testing data was utilized to compute the classification accuracy. When the classifier was proficient with training data utilizing 10-fold cross-validation, then testing was performed on the testing data and the outcomes were assessed utilizing classification accuracy. To increase the classification accuracy HPO was used, and, for this reason, the TPE optimization algorithms were used. For HPO we have used Tpe.suggest which executes a Bayesian-based reiterative search. The search strategy of TPE comprises two phases. During the first phase, it randomly identifies an HP value. These HP values are chosen by activation functions. The selected HP value combinations are utilized to train a model, which is assessed with the test data to check that the selected HP value can influence the accuracy. The second stage continues for n_init iterations which are 20 in our tests and constructs a function founded on the Bayesian rule that is shown below:

$$P(accuracy|parameter) = \frac{P(parameter|accuarcy) * P(parameter)}{P(accuarcy)}$$

where P(accuracy|parameter) is the probability of classification accuracy (accuracy) that is attained with a set of HP values (parameter). Grounded on this classification accuracy, HP values are scattered into good and bad parts. In our experiment, parameter γ = 0.25 is used which means that 25% of all HP value combinations fit in good part, while the rest (75%) fit in bad part, also γ permits us to fix the scope of a good part. Founded on how HP values are scattered, the accuracy threshold (accuracy') is considered. Thus, P(accuracy|parameter) is anticipated to give an enhancement only if accuracy $\geqslant$ acc'. P(parameter|accuracy) is shown below:

$$P(parameter|accuracy) = \begin{cases} P_{good}(parameter) if accuarcy < accuracy' \\ P_{bad}(parameter)\ if\ accuarcy > accuracy' \end{cases}$$

The aim of the second phase of TPE is that the expected improvement (EI) ratio is maximized formula for this is given below:

$$EI = \frac{P_{good}(parameter)}{P_{bad}(parameter)}$$

The EI is maximized by selecting the HP values (parameter) with higher probability P_{good}(parameter) and lower probability P_{bad}(parameter). It is completed by gathering N_EI HP values (n_EI = 24) and selecting the finest one with the highest EI enhancement. Then, the highest EI enhancement is learned and utilized in the succeeding repetition. In the upcoming repetition, TPE computes the classification accuracy and allocates the HP values into good and bad parts, but during this period, it utilizes all prior HP values organized with the new one. This procedure is recurring till the resolute number of trials (n_trials = 50) is touched. The default TPE parameters that are utilized in our experiment are n_init = 20, γ = 0.25, n_EI = 24, and n_trials set to 50.

V. Results and Discussion

Table 3. Results of ML classifiers in accuracy and F1-score before HPO

Dataset	KNN	SVM	RF	NB	XgBoost
ant-1.7	0.76	0.72	0.79	0.66	0.80
Camel-1.0	0.76	0.76	0.78	0.66	0.79
Camel-1.6	0.76	0.78	0.80	0.57	0.77
Data_arc	0.81	0.70	0.79	0.76	0.63
Data_ivy-2.0	0.83	0.85	0.85	0.71	0.87
Data_prop-6	0.85	0.72	0.88	0.58	0.89
Data_redaktor	0.84	0.79	0.86	0.57	0.81
Jedit-3.2	0.83	0.80	0.85	0.73	0.64
Jedit-4.2	0.87	0.75	0.88	0.67	0.83
Log4j-1.1	0.82	0.78	0.79	0.73	0.64
Lucene-2.0	0.71	0.65	0.68	0.67	0.50
Poi-2.0	0.85	0.88	0.90	0.54	0.86
Synapse-1.0	0.84	0.91	0.87	0.75	0.85
Synapse-1.2	0.75	0.79	0.83	0.73	0.63
Velocity-1.6	0.78	0.75	0.80	0.57	0.61
Xalan-2.4	0.83	0.91	0.90	0.61	0.81
Xerces-1.2	0.81	0.83	0.81	0.55	0.81
Xerces-1.3	0.83	0.81	0.87	0.74	0.81

Table 4. Results of ML classifiers in accuracy and F1-score after HPO

Dataset	KNN	SVM	RF	NB	XgBoost
ant-1.7	0.86	0.76	0.88	0.70	0.81
Camel-1.0	0.92	0.90	0.97	0.91	0.96
Camel-1.6	0.76	0.86	0.87	0.60	0.80
Data_arc	0.82	0.72	0.89	0.86	0.64
Data_ivy-2.0	0.87	0.94	0.93	0.72	0.88
Data_prop-6	0.88	0.74	0.91	0.60	0.90
Data_redaktor	0.87	0.90	0.93	0.59	0.84
Jedit-3.2	0.85	0.84	0.87	0.75	0.67
Jedit-4.2	0.91	0.82	0.92	0.70	0.86
Log4j-1.1	0.86	0.80	0.82	0.78	0.66
Lucene-2.0	0.73	0.69	0.71	0.70	0.52
Poi-2.0	0.87	0.91	0.93	0.56	0.88
Synapse-1.0	0.87	0.93	0.90	0.76	0.89
Synapse-1.2	0.79	0.81	0.85	0.70	0.66
Velocity-1.6	0.81	0.78	0.82	0.60	0.65
Xalan-2.4	0.86	0.92	0.91	0.64	0.84
Xerces-1.2	0.83	0.88	0.89	0.60	0.84
Xerces-1.3	0.87	0.84	0.91	0.77	0.84

This segment provides the results of the HPO of ML classifiers utilizing TPE for SDP to increase its prediction accuracy. We conducted our experiment with an SDP dataset to categorize the information as defective or nondefective. The result was assessed utilizing classification accuracy. Below Tables 3, and 4 describes the accuracy of SDP using ML classifiers such as SVM, KNN, NB, RF, and XGBoost before and after HPO. This procedure continues to evaluate according to their association with the accuracy. As expected, the method anticipated by us produced acceptable results for all eighteen datasets. From Table 4 it is evident that for HPO the prediction accuracy of ML classifiers is significantly enhanced for nearly all the datasets. To calculate the variance in the accuracy of every ML classifier, we compared the accuracy of the ML classifiers utilizing default values and HPO values shown in Tables 3, and 4. Overall, accuracy is improved by the HPO of ML classifiers using TPE rather than using defaulting parameter values. In summary, we can see XGBoost based TPE contains highest accuracy of 96% on Camel-1.0 dataset. this shows that XGBoost based TPE can outperform in terms of performance from other algorithms.

VI. Conclusion and Future Work

This paper anticipated SMOTE algorithm to enhance ML classifiers performance on class imbalance issue of SDP. Separately HP values of all ML classifiers are adjusted using a TPE to find the finest combination of HP. Best HP values are utilized for assessment. The dataset utilized is 18 promise repository datasets. On the basis of results XGBoost performance was increased significantly (accuracy by 96%) and all ML classifiers outperformed compared to the original ML classifiers without SMOTE and TPE HPO. Yet, each dataset has a diverse winning HPO values, obtained accuracy and the differences are not significant. For further research, Bayesian optimization and evolutionary algorithms can be utilized to evade unfairness in the results of HPO on SDP.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The experiments were conceived and planned by Faiza Khan, Sultan Almari, Muhammad Haseeb Khan, and Summrina Kanwal. Faiza Khan and Muhammad Haseeb Khan executed the experiments. Summrina Kanwal, Faiza Khan, and Muhammad Haseeb Khan were involved in both planning and conducting the simulations. Sultan Almari, Faiza Khan, and Summrina Kanwal contributed to sample preparation and the interpretation of the results. Faiza Khan took the primary responsibility for writing the manuscript. All authors played a crucial role in providing valuable feedback and contributing to the research, analysis, and manuscript formation.

References

[1] R. Malhotra and J. Jain, "Predicting defects in imbalanced data using resampling methods: An empirical investigation," *PeerJ Computer Science*, vol. 8, e573, 2022.

[2] F. Khan, S. Kanwal, S. Alamri, and B. Mumtaz, "Hyper-parameter optimization of classifiers, using an artificial immune network and its application to software bug prediction," *IEEE Access*, vol. 8, pp. 20954-20964, 2020.

[3] W. Fu, T. Menzies, and X. Shen, "Tuning for software analytics: Is it really necessary?" *Information and Software Technology*, vol. 76, pp. 135-146, 2016.

[4] Y. Shen, S. Hu, S. Cai, and M. Chen, "Software defect prediction based on Bayesian optimization random forest," in *Proc. 2022 9th International Conference on Dependable Systems and Their Applications (DSA)*, IEEE, August 2022, pp. 1012-1013.

[5] C. Tantithamthavorn, S. McIntosh, A. E. Hassan, and K. Matsumoto, 2018, "The impact of automated parameter optimization on defect

prediction models," *IEEE Transactions on Software Engineering*, vol. 45, no. 7, pp. 683-711.

[6] S. Kanwar, L. K. Awasthi, and V. Shrivastava, "Cross-Project Defect Prediction by Using Optimized Light Gradient Boosting Machine Algorithm," *Communication and Intelligent Systems*, Springer, Singapore pp. 933-946, 2022.

[7] S. Feng, J. Keung, X. Yu, Y. Xiao, and M. Zhang, "Investigation on the stability of SMOTE-based oversampling techniques in software defect prediction," *Information and Software Technology*, vol. 139, p. 106662, 2021.

[8] A. O. Balogun, F. B. Lafenwa-Balogun *et al.*, "SMOTE-based homogeneous ensemble methods for software defect prediction," in *Proc. International Conference on Computational Science and its Applications*, July 2020, pp. 615-631, Springer, Cham.

[9] R. B. Bahaweres, F. Agustian, I. Hermadi, A. I. Suroso, and Y. Arkeman, "Software defect prediction using neural network based SMOTE," in *Proc. 2020 7th International Conference on Electrical Engineering, Computer Sciences and Informatics (EECSI)*, October 2020, pp. 71-76, IEEE.

[10] G. Xie, S. Xie, X. Peng, and Z. Li, "Prediction of Number of Software Defects based on SMOTE," *International Journal of Performability Engineering*, vol. 17, no. 1, 2021.

[11] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2002.

[12] C. Pak, T. T. Wang, and X. H. Su, 2018, "An empirical study on software defect prediction using over-sampling by SMOTE," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28(06), 811-830.

[13] B. Mumtaz, S. Kanwal, S. Alamri, and F. Khan, "Feature selection using artificial immune network: An approach for software defect prediction," *Intelligent Automation & Soft Computing*, vol. 29, no. 3, 2021.

[14] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Advances in Neural Information Processing Systems*, vol. 24, 2011.

[15] J. Bergstra, D. Yamins, and D. Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," in *Proc. International conference on machine learning*, February 2013, pp. 115-123, PMLR.

[16] M. Massaoudi, H. Abu-Rub, S. S. Refaat, M. Trabelsi, I. Chihi, and F. S. Oueslati, 2021, "Enhanced deep belief network based on ensemble learning and tree-structured of parzen estimators," *An Optimal Photovoltaic Power Forecasting Method*, vol. 9, pp. 150330-150344.

[17] D. Aggarwal, "Software defect prediction dataset," *Figshare, Dataset*, 2021. https://doi.org/10.6084/m9.figshare.13536506.v1

[18] S. Kanwal, A. Hussain, and K. Huang, "Novel Artificial Immune Networks-based optimization of shallow machine learning (ML) classifiers," *Expert Systems with Applications*, vol. 165, pp. 113834, 2021.