ISSN 2278-3091
Volume 8, No.1.5, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse5181.52019.pdf
https://doi.org/10.30534/ijatcse/2019/5181.52019

Optimizing Software Effort Estimation Models Based On Metaheuristic Methods: A Proposed Framework

[1]Robert Marco, [2]Nanna Suryana Herman, [3]Sharifah Sakinah Syed Ahmad
[1]Department of Computer Science,
Universitas Amikom Yogyakarta, Indonesia.
robertmarco@amikom.ac.id
[2]Professor, Department of Information & Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia.
nsuryana@utem.edu.my
[3]Associate Professor, Department of Information & Communication Technology,
Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia.
sakinah@utem.edu.my

ABSTRACT

Software effort estimation are part of the field of project management in software that is very important for development efforts. Software development planning is something very complex and serious, which determines the success of a software project. Because of the lack of good requirements and information, it causes software project failures. Although there are many studies that aim to solve the problem of noisy, irrelevant and excessive data to achieve accuracy. The purpose of this study is to combine metaheuristic optimization techniques as a framework for using Machine Learning models. By proposing a hybrid estimation model based on a combination of the Satin Bowerbird Optimizer (SBO) algorithm and Support Vector Regression (SVR) to improve the accuracy of software estimation efforts. This study is to determine the effort estimation and duration estimation. The proposed framework is based on theoretical concepts. the proposed model will be tested using a heterogeneous dataset, namely the ISBSG dataset. the results of the study are expected to be used as decision making as the initial planning of software project development.

Key words : Software effort estimation, Satin Bowerbird Optimizer (SBO) algorithm, Support Vector Regression (SVR), Parameter optimization, metaheuristic.

1. INTRODUCTION

According to the IEEE Computer Society, that Software Engineering (SE) is an approach to systematically measuring the application of software engineering [1]. In SE there are several parts that are used in software engineering management, from planning to measuring software [2]. Accuracy and reliability of Software Efforts Estimation (SEE) in the early stages of project planning must be able to allocate resources and good scheduling [3]. Success and failure in software development efforts lies in the planning aspect which is an important and accurate part of project completion [3]. This condition is different from other engineering disciplines where the results of the engineering process provide a real form. The measurement of the volume of abstract software engineering work is only based on the usefulness of the engineering results.

The large number of interrelation factors that are very influential in effort and development. Improving estimation techniques will make it easier to control duration and budget [4][5][6]. Poor estimation techniques can lead to budget waste and inappropriate scheduling in software projects [7][8]. The importance of problem solving in software project management, so that a large number of studies propose models in software effort estimation to overcome software development problems [9]. Software development is increasingly aware of the need for a better model, so the importance of proposing new models to predict software development efforts [10]. Or otherwise produces the right results regardless of uncertainty [8]. It is important to advance software development, by making a definite model so that in the process of developing it will produce appropriate software, in a way that minimizes waste of duration and budget.

There have been many estimation methods that have been proposed to solve software project problems, so as to increase the accuracy of estimates. The estimation method can be classified in several types, including: machine learning methods[11][12]; expert judgment; Case-Based Reasoning (CBR) methods; regression-based methods; parametric models; dynamics-based models; and composite methods [13]; and algorithmic method [14].

Many researchers have proposed several techniques to improve accuracy in estimating software efforts using machine learning. A number of studies have tried to modify

new models using machine learning to improve accuracy in software effort estimation [3][15][16][17]. Using a based feature selection [18][19][20][21], or parameter optimization [22][23][24]. Some prediction techniques have been suggested but none have proved consistently successful in predicting software development efforts [11]. Estimation methods have long been introduced, but this approach still has the potential to make estimates on software accurately and stably. So software development using machine learning to produce more reliable and accuracy development is still needed in this field of research.

Problems that occur based on previous studies, are in the early stages of software project development. Often considered an obstacle, namely the problem of time and cost framework. But significantly that the costs estimation and duratin estimation are inversely proportional. So in this study, the first problem that the most accurate estimation is to estimate the effort and duration. Because the estimation model of the effort and duration is proposed as a decision-making tool in developing software so that it is released from errors that cause negative implications or failures in software projects. While the second problem, is the prediction process must be based on historical information. So in developing the model for effort estimation and duration estimation must use data mining. Because the presence of noisy, irrelevant, and redundant data on the dataset will greatly affect ML performance, because poor data quality arises due to missing values, outliners and missing values can cause uncertainty and inconsistency. According to Huang et al (2006), estimation problems are complex problems and have features such as nonlinear relationships; Measurement of software metrics; and software processes that are inaccurate and uncertain changes rapidly, no model has proven to be the perfect solution [35]. The implementation of software at an early stage can significantly improve the success of software projects if it can make precise and accurate effort and duration estimation. So it is necessary to do a comprehensive approach, by doing the data preparation stage until implementation to produce accurate and reliable software estimates.

In software efforts estimation several types of algorithms can be applied, including Genetic Algorithm [18]; Support Vector Machine [25][26]; Fuzzy [27]; Support Vector Regression [18][28]; Artificial Neural Network [29]; Adaptive Regression [30]. Approaches for comparison of these models are often invalid and may make things worse. Identified several theoretical problems with a study comparing different estimation models in several common datasets to produce the best models [31]. No specific classifier can do the best accuracy results for all data sets [32].

In a few years, many optimization algorithms have been used as enhancements and adjustments to effort estimator parameters. There are two categories of optimization methods in general, including: 1) Mathematical methods, like: Dynamic Programming (DP) and Quasi-Newton (QN) [33]; 2) Metaheuristic algorithm[33], like: Genetic algorithm (GA) [18][28][34][35], Bee Colony Optimization (BCO) and Ant Colony Optimization (ACO) [36], satin bower bird optimization algorithm (SBO) [3], Particle Swarm Optimization (PSO) [37][38], COCOMO [39][40][41], Cuckoo Search (CS) [42]. There are so many optimization algorithms that have been submitted in an effort to develop software, because each optimization algorithm has different adaptation and performance capabilities.

The Meta-heuristic algorithm can effectively solve problems in non-linear optimization [43]. Implementation of this algorithm can be calculated in various ways to solve the optimization problem [33][43]. To increase effort predictions by exploring parameter settings is one of the functions of using meta-heuristics [44]. Using the meta-heuristic approach in finding the best feature parts, by adopting a classifier to select features optimally using the wrapper model [24]. In addition, significantly optimization of using metaheuristics can have the ability to find full search space and high-quality solutions in a reasonable period of time using global search capabilities [45]. Metaheuristic method designed to overcome this problem [46]. Metaheuristic optimization method gives good results than traditional and non-evolutionary methods in terms of increasing accuracy using the feature selection method.

In overcoming limitations and narrowing the gap between the findings of recent research and the potential for dissemination in the application of machine learning algorithms to estimate the effort and duration of developing software efforts early in the project life cycle. A comprehensive approach is used to ensure its usefulness and the accuracy of exceptional estimates and resilience to in-data noise, irrelevance, and redundant. According to the results obtained in the literature review, several studies only focused on adjusting individual algorithms for best performance and accuracy in Machine Learning (ML) models, such as the use or improvement of algorithms, such as neural networks, case-based reasoning, support vector regression, decision trees, and etc. So that the application of statistics and machine learning algorithms are used to effort estimation and estimated duration. Contributions in this study, will integrate the data preprocessing, meta learning, feature selection; and Optimization parameter aims to convert heterogeneous data into homogeneous data to improve accuracy for effort estimation and duration estimation. The aim of the study was to overcome the limitations of the gap, in developing the SEE model in the early stages of software project planning using the meta-heuristic approach and the application of machine learning to effort and estimation of duration. In this study, we will use the ISBSG dataset, which is the most popular dataset and has the most reliable data source..

2. RELATED WORK

Feature selection is a process of removing irrelevant and excessive features. Because in large data the use of feature selection to overcome a large number of input features by looking for subset feature space. The search method is chosen to do a search and evaluators submit values for each feature section [47]. The function of selecting features to do relevant and most informative data extraction, so that the classification

is more suitable in the feature set [36]. Most of the problems in classification can be solved by selecting features, but still need a new approach to determine the sub-feature options in increasing accuracy [48]. The main purpose of features selection in learning is to find the features so that it produces high accuracy [26][49]. The performance of feature selection techniques is strongly influenced by the characteristics of the dataset, this has an impact on the problem of accuracy and time complexity of various feature selection techniques.

In the prediction system is very influential on the data collection used. Although there have been many models proposed to solve this problem. However, there are still many models that significantly and consistently have accurate results that funds have uncertainty about prediction techniques [50]. Attribute noise, incomplete, and inconsistent in the software measurement dataset lowers the performance of machine-learning [51]. Data quality will decrease when used on heterogeneous and inconsistent datasets [52]. Irrelevant and inconsistent project effects on downhill estimates by designing frameworks, where all projects are clustered [53]. The existence of each dataset that is not normally distributed will imply an effort to develop an accurate method [54]. The choice of features is used to speed up the performance of algorithms in mining data and improve data quality, by reducing the dimensions of the feature space, removing data that is excessive, irrelevant, and noisy [55][56]. A collection of relevant dataset features can improve accuracy [56]. The selection of features will explore the effects of attributes that are not relevant [57]. The machine learning model is greatly influenced by the level of accuracy by using the dataset [58]. Data preparation used to build machine learning models is needed, by selecting, cleaning, reducing, transforming and feature selection [51][58[59]. The use of efficient machine learning algorithms is an important task in features selection as reduction dimensions. However, the proliferation of this feature selection technique raises the difficulty of choosing the algorithm for selecting the features that are most suitable for an application, resulting from the selection of different features [60].

There are several ensemble methods that have been proposed, such as bagging, boosting, random sampling techniques [61]; and stacking [62][63]. Using the ensemble method in performing different data collection will achieve better accuracy than individual techniques [63]. Boosting [64][65] or bagging [22][65[66] is a representative approach that combines preprocessing data oversampling and under sampling with ensemble classifiers. Integrating bagging with under-sampling is stronger than over-sampling [22]. Bagging provides a large advantage in accuracy, with testing on real datasets and simulations using classification, regression trees and subset selection in linear regression [66]. Bagging is a method that can handle class imbalances, and can improve performance in noisy data environments [67]. The application using the ensemble approach serves to predict the average using strong machine learning, stabilize the model, reduce the influence of noise in the data and have an impact on the abnormal behavior of the algorithm. So that it can be concluded that the ensemble method that has the best performance is bagging and stacking.

All this time, the SEE technique has experienced a lot of instability in producing precise approximate accuracy [50], so that to overcome this problem an ensemble learning approach needs to be applied [68]. Ensemble to predict software project development efforts by combining more than one SDEE technique used. To analyze the accuracy of the ML model by using MMRE and MdMRE that have lower values, while the Pred (25) is higher, this shows a more accurate estimate. According to Wen et al (2012), it was shown that ANN and SVR were the most accurate (median MMRE around 35% and Pred (25) around 70%), followed by CBR, Decision Tree (DT), and Genetic programming (median MMRE and Pred (25) around 50%), while Bayesian Networks (BN) have the worst accuracy (median MMRE around 100% and Pred (25) around 30%) [11]. Whereas Idri et al (2016), the performance of the ensemble effort estimation technique that SVR is the most accurate (median Pred (25) 50% and MMRE 48.6%), ANN (median Pred (25) 40% and MMRE 49.9%), while Neuro Fuzzy (NF) is the least accurate (Median Pred (25) 31% and MMRE 79.5%) [69]. based on the literature review, it can be concluded that ANN and SVR are ML techniques that have the best and most accurate accuracy in predicting software development.

While Genetic method algorithms can improve performance in ML and feature selection [18][28]. The parameters of the basic COCOMO model can be improved by applying simple genetic algorithms [40]. Combining the GA and SVM methods can improve predictions more accurately by finding the best SVM regression parameters by the proposed model [70]. The Satin Bowerbird Optimizer (SBO) algorithm compared to the five most famous new algorithms (Ant Lion Optimization (ALO), Partial Swarm Optimization (PSO), Fire Fly optimization (FA), GA and Artificial Bees Colony (ABC)) has the best performance than other algorithms, both in test functions and is statistically superior [3].

3. LITERATURE

3.1 Satin Bowerbird Optimizer (SBO)

The satin bower bird optimization algorithm is one type of algorithm that simulates the life of a type of satin bower bird [71]. Bowerbirds during autumn and winter, will leave forest habitat and move to open forests to find food. However, in the spring, they gather together and inhabit the area, because at that time it is the mating season for them. During the season they will make different materials such as flowers, fruits, shiny objects, branches, and dramatic movements to attract women's attention which is a variable in this regard. That male birds use natural instincts and imitate other males to build their nests [71].

According to satin bowerbird's life principle, here are the steps of the SBO algorithm:

1. A Set of Random Bower Generations:

At the initial stage, the SBO will create a random population and position for the bowers. In this case the position must be optimized, because it is a n-parameter dimensional vector. In the optimization problem, each bower parameter is the same as a variable. The value will be randomly initialized taking into account the lower and upper limit parameters [43][71].

2. Calculating the Probability of Each Population Member:
The Probability equation below, fit_i is the suitability of the i and n solutions is the number of bower.

$$\mathrm{fit_i} = \begin{cases} \frac{1}{1 + \mathrm{f(x_i)}}, \mathrm{f(x_i)} \geq 0 \\ 1 + |\mathrm{f(x_i)}|, \mathrm{f(x_i)} < 0 \end{cases} \quad (1)$$

Then, each bower will calculate the probability according to the equation. 2.

$$\mathrm{prob_i} = \frac{\mathrm{fit_i}}{\sum_{n=1}^{NB} \mathrm{fit_n}} \quad (2)$$

$f(x_i)$ as a cost function value in i_{th} position or i_{th} bower. This function must be optimal, because calculating the final fitness must have a value greater or equal to zero, then calculate the suitability for a value that is less than zero.

   a. Because , $f(x_i)$ = 0 as a conformity value of one
   b. Fitness is always a positive value

3. Elitism:
Elitism is maintained at each stage of the optimization process to allow the best solutions to be intended as elite iterations. Because it has the highest fitness, it must influence other positions.

4. Determining new changes in any position:
Each new change must be calculated according to Equation. 3.

$$\mathrm{X_{ik}^{new}} = \mathrm{X_{ik}^{old}} + \lambda_k \left(\left(\frac{\mathrm{X_{jk}} + \mathrm{X_{elite,k}}}{2} \right) - \mathrm{X_{ik}^{old}} \right) \quad (3)$$

In equation, $\mathrm{x_i}$ is $\mathrm{i_{th}}$ bower/solution vector and $\mathrm{x_{ik}}$ is k member of this vector. $\mathrm{x_j}$ as a target solution in the current iteration. The value of j is calculated based on the probability. $\mathrm{x_{elite}}$ shows the elite position, which is stored in each algorithm cycle. The parameter λ_k determines the attraction in the bower goal. λ_k determines the number of steps calculated for each variable.

$$\lambda_k = \frac{a}{1 + p_j} \quad (4)$$

5. Mutation:
At the end, random changes are used for certain probabilities. Random changes are applied to x_{ik} with a certain probability. The mutation process, a normal distribution (N) is employed with average of x_{ik}^{old} and variance of σ^2, as seen in Equation. 5.

$$\mathrm{X_{ik}^{new}} \sim \mathrm{N}(\mathrm{X_{ik}^{old}}, \sigma^2) \quad (5)$$

$$\mathrm{N}(\mathrm{X_{ik}^{old}}, \sigma^2) = \mathrm{X_{ik}^{old}} + \sigma * \mathrm{N}(0,1)) \quad (6)$$

In Equation. 6, the σ value is the proportion of the width of space, calculated in Equation. 7.

$$\sigma = \mathrm{Z} * (\mathrm{var_{max}} - \mathrm{var_{min}}) \quad (7)$$

In Equation. 7, var_{max} and var_{min} are each upper and lower limit assigned to the variable. The Z parameter as percent difference between upper and lower limits which are variables.

3.2 Support Vector Regression (SVR)

Support Vector Regression (SVR) is a new generation of Machine Learning algorithms, which are suitable for predictive data modeling problems [72]. SVR is a Support Vector Machines based approach [72].

Suppose given training data $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \epsilon \mathbb{R}^d$ denotes an input vector and $y_i \epsilon \mathbb{R}$ corresponding target value. In the regression of ε-SV, it aims to find the function $f(x)$ which has the most deviation ε from the actual target y_i for all training data, and at the same time is as light as possible. In other words, do not care about errors as long as they are less than ε, but will not accept greater deviations from this [73]. This type of loss function defines the margin around the actual output. The idea is that errors smaller than a certain threshold ε > 0 are rejected. That is, errors in margins are considered zero. On the other hand, errors caused by points outside the margin are measured by variables ξ and ξ^* [73]. Analogously with the "soft margin" function used in SVM (Support Vector Machine), by introducing slack variables ξ and ξ^* to overcome the unavoidable constraints of optimization problems.

For pedagogical reasons, this is linear function f, which is presented in the Equation. 8.

$$\mathrm{f(x)} = \langle \mathrm{w}, \mathrm{x} \rangle + \mathrm{b} \text{ with } \mathrm{w} \epsilon \mathbb{R}^{\mathrm{d}}, \mathrm{b} \epsilon \mathbb{R} \quad (8)$$

Where $\langle .,. \rangle$ denotes the dot product in $\mathbb{R}^d$. For case nonlinear regression $f(x) = \langle w, \emptyset(x) \rangle + b$, where $\emptyset$ are some nonlinear functions that map the input space to a higher dimensional feature space $\mathbb{R}^d$. ε-SV, weight vector w and threshold b are selected to optimize the problem.

$$\text{minimize}_{w,b,\xi,\xi^*} \quad \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \qquad (9)$$

$$\text{Subject to} \begin{cases} (\langle w, \emptyset(x_i)\rangle + b) - y_i \leq \varepsilon + \xi_i. \\ y_i - (\langle w, \emptyset(x_i)\rangle + b) \leq \varepsilon + \xi_i^*(1) \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

The constant C > 0 determines the trade-off between the complexity of the model, that is the flatness of $f(x)$ and the amount until deviations greater than ε are tolerated. ξ and ξ^* are slack variables and measure the cost of errors at training points. ξ measuring deviations exceeds the target value with more than ε and ξ^* measuring deviations that are more than ε below the target value.

In study using SVR with both linear and RBF (Radial Basis Function) kernels. The Polynomial kernel as

$K(x_i, x_j) = \langle 1 + (x_i), (x_j)\rangle^p$ and RBF kernel as

$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right), \gamma > 0.$

The parameters C and ε significantly influence ε-SVR generalization performance.

3.3 The ISBSG Dataset

The ISBSG repository, is a collection of many multi-organization repositories that can produce very heterogeneous databases [74]. The subset of the ISBSG dataset is very heterogeneous, indicating that the characteristics of the underlying dataset affect the inhibition of different estimation techniques [75][76]; in order to obtain minimal homogeneity data in the sample, it is necessary to data preparation before analyzing [76]. Because if the data collected is large and heterogeneous data, there will be a strong possibility of instability and data cannot be generalized [77]. Accurate estimates of development efforts do not seem to be achieved in a collection of data sets consisting of heterogeneous projects [78]. Heterogeneous project data will produce poor performance on a single mathematical model [79]. Based on Phannachitta et al (2017), it states that testing on 8 public datasets has inconsistent results in the range of 10.71% and 73.33% [80].

During the data preprocessing process, heterogeneous properties in the dataset will disappear, because heterogeneous datasets will affect the results in generalization [81]. Because the dataset preprocessing will be able to restore missing data and normalize the dataset [20][82]; and can change the format of datasets to be easier and more effective when they are processed [83].

The impact of preparation data significantly influences the performance of machine learning methods [59]. In addition, data preparation has a very important role in developing an effective model [59]. The use of preparation data used for data mining and machine learning will produce intelligence or business knowledge in the selection of features [45]. Because large and heterogeneous datasets can cause a decrease in the level of accuracy, so the data preparation process is needed to obtain homogeneity from the dataset. Because the model is built by the use of cross-industry ISBSG datasets (Heterogeneous datasets) that contain the latest software projects in several organizations that are geographically dispersed, so the ISBSG dataset is worthy of being used to develop models in approximate software. It can be recommended to extract the subset that is suitable for each software effort estimation practice.

3. PROPOSED FRAMEWORK

The framework proposed in this study has the main difference between the proposed framework and other frameworks are on the learning scheme consisting of: 1) Data preprocessing; 2) Feature selection; 3) Parameter optimization; 4) Learning algorithm; and 5) Meta learning. Combining metaheuristic optimization techniques using machine learning to be used as feature selection and optimization parameters.

3.1 Data Preprocessing

Preprocessing data is an important part of the estimation problem because it can significantly affect the quality of training [38]. Referring to problems, related to noise, class imbalances, inconsistent and irrelevant datasets. An important specification of the attributes of a software project, which makes estimating efforts difficult and complicated. There have been many techniques and methods proposed in solving optimization problems to predict software efforts. But it is not proven consistently and there is uncertainty, this is because it is influenced by the data set used [50]. The cause of poor performance of machine learning classifiers is the noise attribute, incomplete, and inconsistent in the dataset [51]. Data quality will decrease when used in heterogeneous and inconsistent datasets [52]. Stages in data preprocessing are important processes in building Machine Learning models, as follows: selection, cleaning, reduction, transformation and selection of features [59].

3.2 Proposed Framework

Figure 1 represents the framework proposed in this study using Scaling as data preprocessing; Satin Bowerbird Optimization (SBO) and Support Vector Regression (SVR) as feature selection; while Satin Bowerbird Optimization (SBO) as parameters optimization; Bagging (B) as meta learning; and selecting 4 algorithms to conduct experiments. In addition, Satin Bowerbird Optimization and SVR are also used to optimize parameters and improve performance estimation of software efforts. Approach techniques in estimations used in the improvement and comparison of this study, including: Multi Layer Perceptron (MLP), Generalized Linear Models (GLM), Classification and Regression Tree (CART), and Radial Basis Function (RBF).

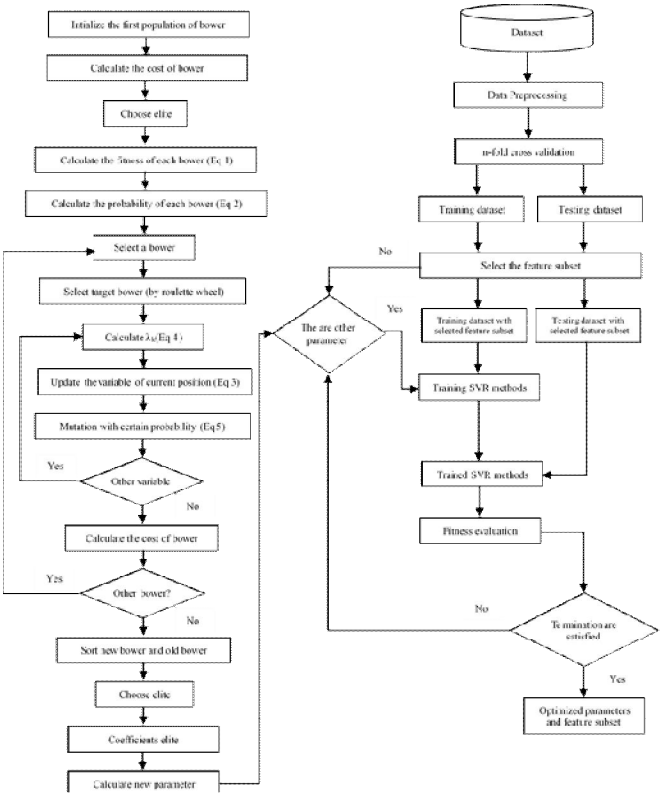

Figure 1: Proposed framework Software effort estimation

In the table 1, Comparison of the framework with previous research. Wahono (2015) used a feature selection, meta learning, parameter optimization and selected 10 algorithm (LR, LDA, NB, k-NN, K*, NN, SVM, C4.5, CART, RF) [91]. Oliviera at al (2010), used a feature selection, parameters optimization and selected 4 algorithm (RBF, MLP, SVR, M5P) [92]. Pospieszny et al (2017), used a smart data preparation, feature selection and selected 3 algorithm (SVM, MLP, GLM) [58]. Hosni at al (2017), used data preprocessing, parameters optimization and selected 4 algorithm (k-NN, SVR, MLP, DT) [12].

Table 1: Framework comparison

Framework	Data Preparation	Feature Selection	Classifier	Meta Learning	Parameter Optimization
[91]	-	GA and PSO	10 Algorithm	Bagging	GA
[92]	-	GA	4 Algorithm	-	GA and SVM
[58]	Selection, Cleaning, Reduction, and Transformation	Lasso, Stepwise regression and pearson correlation	3 Algorithm	-	-
[12]	Select features, Weight the features, and Remove outliers.	-	4 Algorithm	-	GSO and PSO
Proposed Framework	Scaling	SBO and SVR	4 Algorithm	Bagging	SBO

The accuracy of the models proposed using algorithms shows that the most influential factor is the approach in dataset selection, cleaning and preparation. So that the use of preprocessing data is very important to produce normally distributed data, which has an impact on the level of accuracy produced. This study focuses more on verification of the proof-of-concept approach with multi-organization datasets, considered a measure approach to the accuracy of machine learning for effort and duration estimation model.

In the proposed framework, metaheuristic optimization methods, specifically SBO dan SVR, are used as feature selection and optimization parameters. Based on previous research, that this method has proven effective for solving problems that arise in machine learning. Feature selection techniques for selecting more valuable attributes and removing non-attractive attributes [84]; A SVR based feature selection [18][28].

Ensemble learning methods combine several basic classifiers to build strong classifiers, such as bagging, boosting, sampling, and stacking are applied to classifications with data imbalance problems. Bagging and Boosting is an ensemble method by adopting sampling techniques in each iteration to classify imbalance data [85]. Boosting techniques have a lower performance than bagging techniques when working in noisy and imbalance data [67]. while stacking techniques have the ability to build high-level prediction models based on first-level predictions [86]. Scaling is to avoid attributes in a larger numeric range dominating attributes in a smaller numeric range [34]. To improve the efficiency and excellence of software development, he can utilize data mining techniques to analyze and predict large amounts of data [93].

The amount of research that has done the method of selecting features and parameters optimization, they are only limited to solving problems for noisy, irrelevant and redundant data to achieve accuracy. This research will conduct feature selection based on parameter optimization using a bagging techniques, to distribute heterogeneous data into homogeneous, for class imbalance problems in terms of increasing accuracy in effort estimation and duration estimation in the software development effort estimation field. Ensure its usefulness and the accuracy of exceptional estimates and robustness for noise in the data. To get good accuracy, computers / systems must be trained properly with the help of training data sets [94][95].

In the next stage, it will be elaborated in evaluating the accuracy of the models proposed in this study. in the process of evaluating the output using values on MMRE and PRED (25). In this stage, training and testing data will be normalized using SVR by trying 4 proposed algorithms (such as MLP, GLM, CART and RBF) to obtain optimal accuracy results. this is different from a similar process carried out previously, that the optimization parameters obtained using the default system. the next step is to calculate the MMRE and PRED values (25) to find out the value of effort estimation and duration estimation.

3.3 Model Validation

The ISBSG dataset is applied, normalization of dependent variables, cross validation approaches and Mean Absolute Residual Error (MAR) with Median and Mean of Absolute Residuals (MdAR) are used as accuracy criteria [58]. Several methods have been proposed to evaluate the accuracy of the predictive value of accuracy in software. Besides being measured using various metrics on the value of accuracy, there are several accuracy assessments on popular software effort estimation, such as: n-fold cross validation, holdout and leave-one-out cross validation (LOOCV) [11][88].

3.4 Model Evaluation

There are three metrics used to evaluate the performance of the software effort estimation, such as: Mean Magnitude Relative Error (MMRE), Magnitude Relative Error (MRE), and percentages of the PRED, which are calculated as follows [89][90]:

$$\text{MRE} = \frac{|\text{estimated} - \text{actual}|}{\text{actual}} \quad (10)$$

$$\text{MMRE} = \frac{\sum_{i=1}^{N} \text{MRE}}{N} \quad (11)$$

$$\text{PRED(X)} = \frac{A}{N} \quad (12)$$

A is the number of projects with $MRE \leq X$ and N N the number of projects in the set. MRE must be less than 0.25 (≤0.25) to be accepted by the software effort estimation model. whereas MMRE must have a minimum value and PRED (25) has a maximum value [90].

4. CONCLUSION

The large number of previous studies prioritizes accurate accuracy, regardless of the estimation process that takes a long time. then based on that problem, the development of software projects will produce good accuracy if it produces a fast, efficient and practical time. So the need to implement the machine learning algorithm to measure effort and duration of estimation. the proposed framework will present a holistic approach to building models in estimating efforts and duration in the early stages of software development. the stages in this study process include: data preprocessing, feature selection, optimization parameters, meta learning and 4 (four) machine learning algorithms using the ISBSG dataset. In addition, classification problems also involve a number of features, this is because not all available features are equally important. Good and accurate classification must require small features. for the type of validation used to measure the accuracy of the estimated overall model using n-fold cross validation. Whereas to evaluate accuracy to estimate software engineering using; MMRE, and PRED (25).

REFERENCES

1. P.Bourque p, R.Dupuis, A.Abran, J.W.Moore, and L.Tripp, **Guide to the Software Engineering Body of Knowledge**, *IEEE Software* , Vol. 16, Issue. 6, 1999. https://doi.org/10.1109/52.805471
2. N. Garcia-Diaz, C. Lopez-Martin, and A. Chavoya, **"A Comparative Study of Two Fuzzy Logic Models for Software Development Effort Estimation**," *Procedia Technol.*, vol. 7, pp. 305–314, 2013.
3. S. H. Samareh Moosavi and V. Khatibi Bardsiri, **"Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation,"** *Eng. Appl. Artif. Intell.*, vol. 60, no. May 2016, pp. 1–15, 2017. https://doi.org/10.1016/j.engappai.2017.01.006
4. I. Kalichanin-Balich and C. Lopez-Martin, **"Applying a feedforward neural network for predicting software development effort of short-scale projects,"** *8th ACIS Int. Conf. Softw. Eng. Res. Manag. Appl. SERA 2010*, pp. 269–275, 2010.
5. G. R. Finnie, G. E. Wittig, and J.-M. Desharnais, **"A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models,"** *J. Syst. Softw.*, vol. 39, no. 3, pp. 281–289, 1997.
6. S. J. Huang, N. H. Chiu, and Y. J. Liu, **"A comparative evaluation on the accuracies of software effort estimates from clustered data,"** *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 879–888, 2008.
7. H. Lee, **"A structured methodology for software development effort prediction using the analytic hierarchy process,"** *J. Syst. Softw.*, vol. 21, no. 2, pp. 179–186, 1993.
8. J. Ryder, **"Fuzzy modeling of software effort prediction,"** *Inf. Technol. Conf. 1998. IEEE*, pp. 53–56, 1998.
9. A. Heiat, **"Comparison of artificial neural network and regression models for estimating software development effort,"** *Inf. Softw. Technol.*, vol. 44, no. 15, pp. 911–922, 2002. https://doi.org/10.1016/S0950-5849(02)00128-3
10. S. G. Macdonell and A. R. Gray, **"A Comparison of Modeling Techniques for Software Development Effort Prediction,"** *Springer-Verlag*, pp. 869–872, 1997.
11. J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, **"Systematic literature review of machine learning based software development effort estimation models,"** *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012.

12. M. Hosni, A. Idri, A. Abran, and A. Bou, **"On the value of parameter tuning in heterogeneous ensembles effort estimation,**" *Soft Comput.*, 2017.
13. K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, **"Data mining techniques for software effort estimation: A comparative study,"** *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 375–397, 2012.
14. N. Saini and B. Khalid, **"Effectiveness of Feature Selection and Machine Learning Techniques for Software Effort Estimation,"** *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 34–38, 2014. https://doi.org/10.9790/0661-16193438
15. M. Azzeh, D. Neagu, and P. I. Cowling, **"Analogy-based software effort estimation using Fuzzy numbers,"** *J. Syst. Softw.*, vol. 84, no. 2, pp. 270–284, 2011.
16. S. Aljahdali and A. F. Sheta, **"Software effort estimation by tuning COOCMO model parameters using differential evolution,"** *ACS/IEEE Int. Conf. Comput. Syst. Appl. - AICCSA 2010*, pp. 1–6, 2010.
17. V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, **"Increasing the accuracy of software development effort estimation using projects clustering,"** *IET Softw.*, vol. 6, no. 6, p. 461, 2012.
18. A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. L. Cornelio, **"GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation,"** *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1155–1166, 2010. https://doi.org/10.1016/j.infsof.2010.05.009
19. Q. Liu, J. Xiao, and H. Zhu, **"Feature selection for software effort estimation with localized neighborhood mutual information,"** *Cluster Comput.*, vol. 3456789, no. 1, 2018.
20. J. Huang, Y. F. Li, J. W. Keung, Y. T. Yu, and W. K. Chan, **"An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the ISBSG data,"** *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 442–449, 2017. https://doi.org/10.1109/QRS.2017.54
21. P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, **"A stability assessment of solution adaptation techniques for analogy-based software effort estimation,"** *Empir. Softw. Eng.*, vol. 22, no. 1, pp. 474–504, 2017.
22. J. Błaszczyński and J. Stefanowski, **"Neighbourhood sampling in bagging for imbalanced data,"** *Neurocomputing*, vol. 150, no. PB, pp. 529–542, 2014.
23. H. Velarde, C. Santiesteban, A. Garcia, and J. Casillas, **"Software Development Effort Estimation based-on multiple classifier system and Lines of Code,"** *IEEE Lat. Am. Trans.*, vol. 14, no. 8, pp. 3907–3913, 2016.
24. S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, **"Particle swarm optimization for parameter determination and feature selection of support vector machines,"** *Expert Syst. Appl.*, vol. 35, no. 4, pp. 1817–1824, 2008.
25. M. Azzeh and A. B. Nassif, **"A hybrid model for estimating software project effort from Use Case Points,"** *Appl. Soft Comput. J.*, vol. 49, pp. 981–989, 2016.
26. İ. Babaoglu, O. Findik, and E. Ülker, **"A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine,"** *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3177–3183, 2010. https://doi.org/10.1016/j.eswa.2009.09.064
27. M. Azzeh, D. Neagu, and P. Cowling, **"Improving Analogy Software Effort Estimation using Fuzzy Feature Subset Selection Algorithm,"** *PROMISE ACM*, pp. 71–78, 2008.
28. P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, **"A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation Chromosome design,"** *ACM*, pp. 1788–1792, 2008.
29. [29] S. Aljahdali, A. F. Sheta, and narayan C. Debnath, **"Estimating Software Effort and Function Point Using Regression , Support Vector Machine and Artificial Neural Networks Models,"** *IEEE Access*, 2015.
30. S. M. Satapathy, **"Empirical Assessment of Machine Learning Models for Effort Estimation of Web-based Applications,"** *ISEC '17, ACM*, pp. 74–84, 2017.
31. B. Kitchenham and E. Mendes, **"Why Comparative Effort Prediction Studies may be Invalid,"** *PROMISE '09 Proceedings of the 5th International Conference on Predictor Models in Software Engineering*, 2009. https://doi.org/10.1145/1540438.1540444
32. Q. Song, Z. Jia, M. Shepperd, S. Ying, and J. Liu, **"A general software defect-proneness prediction framework,**" *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 356–370, 2011.
33. A. Kaveh and V. R. Mahdavi, "**Colliding bodies optimization: A novel meta-heuristic method,**" *Comput. Struct.*, vol. 139, pp. 18–27, 2014.
34. C. L. Huang and C. J. Wang, **"A GA-based feature selection and parameters optimizationfor support vector machines,"** *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231–240, 2006.
35. S. J. Huang and N. H. Chiu, **"Optimization of analogy weights by genetic algorithm for software effort**

estimation," *Inf. Softw. Technol.*, vol. 48, no. 11, pp. 1034–1045, 2006. https://doi.org/10.1016/j.infsof.2005.12.020

36. P. Shunmugapriya and S. Kanmani, **"A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid),"** *Swarm Evol. Comput.*, vol. 36, pp. 27–36, 2017.
37. J. Mercieca and S. G. Fabri, **"A Metaheuristic Particle Swarm Optimization Approach to Nonlinear Model Predictive Control,"** vol. 5, no. 3, pp. 357–369, 2012.
38. V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, **"A PSO-based model to increase the accuracy of software development effort estimation,"** *Softw. Qual. J.*, vol. 21, no. 3, pp. 501–526, 2013.
39. P. Agrawal and S. Kumar, "**Early phase software effort estimation model,"** *2016 Symp. Colossal Data Anal. Netw.*, pp. 1–8, 2016. https://doi.org/10.1109/CDAN.2016.7570914
40. R. K. Sachan *et al.*, **"Optimizing Basic COCOMO Model Using Simplified Genetic Algorithm,"** *Procedia Comput. Sci.*, vol. 89, pp. 492–498, 2016.
41. O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, **"COCOMO-Based Effort Estimation,"** *Kluwer Acad. Publ.*, pp. 265–281, 2003.
42. E. E. Miandoab and F. S. Gharehchopogh, **"A Novel Hybrid Algorithm for Software Cost Estimation Based on Cuckoo Optimization and K-Nearest Neighbors Algorithms,"** *Eng. Technol. Appl. Sci. Res.*, vol. 6, no. 3, pp. 1018–1022, 2016.
43. R. Kishore and D. . Gupta, **"Software Effort Estimation using Satin Bowerbird Algorithm,"** vol. 5, no. 3, pp. 216–218, 2012.
44. A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, **"Using tabu search to configure support vector regression for effort estimation,"** *Empir. Softw. Eng.*, vol. 18, no. 3, pp. 506–546, 2013.
45. S. C. Yusta, **"Different metaheuristic strategies to solve the feature selection problem,"** *Pattern Recognit. Lett.*, vol. 30, no. 5, pp. 525–534, 2009.
46. J. Reca, J. Martínez, C. Gil, and R. Baños, **"Application of several meta-heuristic techniques to the optimization of real looped water distribution networks,"** *Water Resour. Manag.*, vol. 22, no. 10, pp. 1367–1379, 2008.
47. D. Oreski, S. Oreski, and B. Klicek, **"Effects of dataset characteristics on the performance of feature selection techniques,**" *Appl. Soft Comput. J.*, vol. 52, pp. 109–119, 2017. https://doi.org/10.1016/j.asoc.2016.12.023
48. H. K. Bhuyan and N. K. Kamila, **"Privacy preserving sub-feature selection in distributed data mining,"** *Appl. Soft Comput. J.*, vol. 36, pp. 552–569, 2015.
49. M. Ramaswami and R. Bhaskaran, **"A Study on Feature Selection Techniques in Educational Data Mining,"** *J. Comput.*, vol. 1, no. 1, pp. 7–11, 2009.
50. M. J. Shepperd and G. Kadoda, **"Comparing software prediction techniques using simulation,"** *IEEE Trans. Softw. Eng.*, vol. 27, no. 11, pp. 1014–1022, 2001.
51. C. Catal, O. Alan, and K. Balkan, **"Class noise detection based on software metrics and ROC curves,"** *Inf. Sci. (Ny).*, vol. 181, no. 21, pp. 4867–4877, 2011.
52. V. Khatibi Bardsiri and E. Khatibi, **"Insightful analogy-based software development effort estimation through selective classification and localization,"** *Innov. Syst. Softw. Eng.*, vol. 11, no. 1, pp. 25–38, 2015. https://doi.org/10.1007/s11334-014-0242-2
53. V. Resmi, S. Vijayalakshmi, and R. S. Chandrabose, **"An effective software project effort estimation system using optimal firefly algorithm,"** *Cluster Comput.*, 2017.
54. E. Kocaguneli and T. Menzies, **"Software effort models should be assessed via leave-one-out validation,"** *J. Syst. Softw.*, vol. 86, no. 7, pp. 1879–1890, 2013. https://doi.org/10.1016/j.jss.2013.02.053
55. J. Novaković, P. Strbac, and D. Bulatović, "**Toward optimal feature selection using ranking methods and classification algorithms,"** *Yugosl. J. Oper. Res.*, vol. 21, no. 1, pp. 119–135, 2011.
56. M. Hosni, A. Idri, and A. Abran, **"Investigating Heterogeneous Ensembles with Filter Feature Selection for Software Effort Estimation,"** *ACM*, no. 2, 2017.
57. N. Acir, Ö. Özdamar, and C. Güzeliş, **"Automatic classification of auditory brainstem responses using SVM-based feature selection algorithm for threshold detection,"** *Eng. Appl. Artif. Intell.*, vol. 19, no. 2, pp. 209–218, 2006.
58. P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobyliński, **"An effective approach for software project effort and duration estimation with machine learning algorithms,"** *J. Syst. Softw.*, 2017.
59. J. Huang, Y. F. Li, and M. Xie, **"An empirical analysis of data preprocessing for machine learning-based software cost estimation,"** *Inf. Softw. Technol.*, vol. 67, pp. 108–127, 2015. https://doi.org/10.1016/j.infsof.2015.07.004
60. N. Bidi and Z. Elberrichi, "**Feature selection for text classification using genetic algorithms,"** *Proc. 2016 8th Int. Conf. Model. Identif. Control. ICMIC 2016*, pp. 806–810, 2017.

61. E. Kocaguneli, T. Menzies, and J. W. Keung, **"On the value of ensemble effort estimation,**" *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, 2012.
62. D. H. Wolpert, **"Stacked Generalization,"** vol. 5, pp. 241–259, 1992.
63. T. Wang, W. Li, H. Shi, and Z. Liu, **"Software Defect Prediction Based on Classifiers Ensemble,"** *J. Inf. Comput. Sci.*, vol. 16, no. December, pp. 4241–4254, 2011.
64. R. E. Schapire, "**The Strength of Weak Learnability (Extended Abstract),"** *Mach. Learn.*, vol. 227, no. October, pp. 28–33, 1989.
65. Y. Liu, E. Shriberg, A. Stolcke, and M. Harper, **"Using machine learning to cope with imbalanced classes in natural speech: evidence from sentence boundary and disfluency detection.,"** *Interspeech*, no. 1, pp. 2–5, 2004.
66. L. Breiman, **"Bagging predictors,"** *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.
67. T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, **"Comparing boosting and bagging techniques with noisy and imbalanced data,"** *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 41, no. 3, pp. 552–568, 2011.
68. M. Azzeh, A. B. Nassif, S. Banitaan, and F. Almasalha, **"Pareto efficient multi-objective optimization for local tuning of analogy-based estimation,"** *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2241–2265, 2015. https://doi.org/10.1007/s00521-015-2004-y
69. A. Idri, M. Hosni, and A. Abran, **"Systematic literature review of ensemble effort estimation,"** *J. Syst. Softw.*, vol. 118, pp. 151–175, 2016.
70. J.-C. Lin, C.-T. Chang, and S.-Y. Huang, **"Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression,"** *2011 Int. Symp. Comput. Sci. Soc.*, pp. 349–352, 2011.
71. M. A. Mostafa, A. F. Abdou, A. F. A. El-gawad, and A. F. Abdou, **"SBO-based selective harmonic elimination for nine levels asymmetrical cascaded H-bridge multilevel inverter,"** *Aust. J. Electr. Electron. Eng.*, vol. 00, no. 00, pp. 1–13, 2018.
72. Corazza, A. *et al.* (2013) **'Using tabu search to configure support vector regression for effort estimation'**, *Empirical Software Engineering*, 18(3), pp. 506–546. doi: 10.1007/s10664-011-9187-3.
73. Smola Cotroneo, D. *et al.* (2016) **'Prediction of the Testing Effort for the Safety Certification of Open-Source Software: A Case Study on a Real-Time Operating System'**, *Proceedings - 2016 12th European Dependable Computing Conference, EDCC 2016*, pp. 141–152. doi: 10.1109/EDCC.2016.22.
74. D. Déry and A. Abran, **"Investigation of the Effort Data Consistency in the ISBSG Repository Investigation of the Effort Data Consistency in the ISBSG Repository,"** *researchGate*, no. January 2005, 2014.
75. C.-J. Hsu and C.-Y. Huang, **"Comparison of weighted grey relational analysis for software effort estimation,"** *Softw. Qual. J.*, vol. 19, no. 1, pp. 165–200, 2011.
76. M. Fernández-diego, **"Discretization Methods for NBC in Effort Estimation : An Empirical Comparison based on ISBSG Projects,**" pp. 103–106, 2012.
77. C. Lokan and E. Mendes, **"Investigating the use of chronological split for software effort estimation,"** *IET Softw.*, vol. 3, no. 5, p. 422, 2009.
78. V. Khatibi, D. Norhayati, A. Jawawi, A. Khatibi, and E. Khatibi, "**Engineering Applications of Arti fi cial Intelligence LMES : A localized multi-estimator model to estimate software development effort,"** *Eng. Appl. Artif. Intell.*, pp. 1–17, 2013.
79. R. Malhotra, "**Software Effort Prediction using Statistical and Machine Learning Methods,"** vol. 2, no. 1, pp. 145–152, 2011.
80. P. Phannachitta, J. Keung, K. E. Bennin, A. Monden, and K. Matsumoto, **"Filter-INC: Handling effort-inconsistency in software effort estimation datasets,"** *Proc. - Asia-Pacific Softw. Eng. Conf. APSEC*, pp. 185–192, 2017.
81. S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, **"Duplex output software effort estimation model with self-guided interpretation,"** *Inf. Softw. Technol.*, vol. 94, pp. 1–13, 2018.
82. Y. S. Seo and D. H. Bae, **On the value of outlier elimination on software effort estimation research**, vol. 18, no. 4. 2013.
83. T. Iliou, M. Nerantzaki, and G. Anastassopoulos, **"A Novel Machine Learning Data Preprocessing Method for Enhancing Classification Algorithms Performance,"** pp. 1–5, 2015.
84. N. Cerpa, M. Bardeen, C. A. Astudillo, and J. Verner, **"Evaluating different families of prediction methods for estimating software project outcomes,"** *J. Syst. Softw.*, vol. 112, pp. 48–64, 2016.
85. Y. Arafat, S. Hoque, and D. Farid, **"Cluster-based Under-sampling with Random Forest for Multi-Class Imbalanced Classification,"** *Int. Conf. Software, Knowledge, Inf. Manag. Appl.*, pp. 1–6, 2017.
86. B. Sluban and N. Lavrač, **"Relating ensemble diversity and performance: A study in class noise detection,"** *Neurocomputing*, vol. 160, pp. 120–131, 2015.
87. D. Zhu, "**A hybrid approach for efficient ensembles,"** *Decis. Support Syst.*, vol. 48, no. 3, pp. 480–487, 2010.

88. A. Idri and A. Abran, **"Analogy-based software development effort estimation : A systematic mapping and review,"** *Inf. Softw. Technol.*, 2014.
89. I. (Norwegian S. of M. Myrtveit, E. (Buskerud U. C. Stensrud, and M. (Bournemouth U. Shepperd, **"Reliability and validity in comparative studies of software prediction models,"** *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, 2005.
90. V. Khatibi Bardsiri and E. Khatibi, **"Model to estimate the software development effort based on in-depth analysis of project attributes,"** *IET Softw.*, vol. 9, no. 4, pp. 109–118, 2015.
91. Wahono R. S, **Software defect prediction framework based on hybrid metaheuristic optimization methods.** Thesis, Universiti Teknikal Malaysia Melaka, 2015.
92. Oliveira, A. L. I. *et al.* (2010) **'GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation',** *Information and Software Technology*. Elsevier B.V., 52(11), pp. 1155–1166. doi: 10.1016/j.infsof.2010.05.009.
93. Rohini B. Jadhav, Shashank D. Joshi, Umesh G. Thorat, and Aditi S. Joshi, **A Software Defect Learning and Analysis Utilizing Regression Method for Quality Software Development,** *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, No.4, July – August 2019. https://doi.org/10.30534/ijatcse/2019/38842019
94. Bhanuprakash Dudi, and V.Rajesh, **Medicinal Plant Recognition based on CNN and Machine Learning**, *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, No.4, July – August 2019. https://doi.org/10.30534/ijatcse/2019/03842019
95. T. Hariguna, & Akmal,"Assessing students' continuance intention in using multimedia online learning", TELKOMNIKA (Telecommunication Computing Electronics and Control), 17(1), 187–193, 2019. http://doi.org/10.12928/TELKOMNIKA.v17i1.10328