Support Vector Regression Based on Grid-Search Method for Agile Software Effort Prediction

Abdelali ZAKRANI
dept of Industrial Engineering
ENSAM, University of Hassan II
Casablanca, Morocco
abdelali.zakrani@univh2c.ma

Assia NAJM
dept of Mathematics & Informatics
Faculté des Sciences Ben M'sik
Casablanca, Morocco
assia.najm@gmail.com

Abdelaziz MARZAK
dept of Mathematics & Informatics
Faculté des Sciences Ben M'sik
Casablanca, Morocco
marzak@hotmail.com

***Abstract*—The existing literature on software development effort estimation is extensive and focuses particularly on traditional software projects, while few studies have been devoted to agile projects, especially estimating the effort needed for completing the whole software projects. Story points and team velocity are among the commonly effort drivers used for planning and predicting when a software will be completed. In this paper, we propose an improved model for estimating the software effort based on support vector regression (SVR) optimized by grid search method (GS). The story point and velocity were used as inputs of the prediction model. The empirical evaluation is carried out using 21 historical agile software projects through leave-one-out cross validation method. The obtained results demonstrate that our approach was able to improve the performance of SVR technique. Moreover, it outperforms, in terms of Pred(0.25), MMRE and MdMRE, some recent methods reported in the recent literature.**

***Index Terms*—agile software effort estimation, support vector regression, grid search, story point, velocity.**

I. Introduction

Despite the wide adoption of agile methodologies in software development companies. These latter still struggle to deliver software projects on time, within the budget and scope. Standish Group has conducted a study on 50 000 projects around the world, ranging from tiny enhancements to massive systems re-engineering implementations, and found that on average 9% of agile projects failed and 52% of project run over budget or time and only 39% succeeded [1]. Accurate and reliable effort estimates can contribute significantly in planning and monitoring software project to ensure it complete in time and within budget [2]. For this purpose, a considerable amount of literature has been published on software development effort estimation (SDEE) over the last three decades [3][4]. These studies aimed at developing accurate and reliable tools and methods to assist project managers in their planning and scheduling of the software under development. However, the greater part of these studies on SDEE were concerned particularly with traditional software projects and a little research work has been devoted for estimation in agile projects.

In their systematic literature review of SDEE studies, Jørgensen and Shepperd identified up to 11 estimation approaches proposed in 304 selected journal papers [3]. These approaches are based on different techniques including expert judgment [5][6], statistical analysis of historical project data [7]–[9], and machine learning techniques [10]–[13]. Most of these approaches estimate the effort needed for developing a complete software system, relying on a set of attributes that characterize the software project developed using traditional methodologies such as V-model and waterfall.

In agile development methods, software development work is broken into small increments developed through a repeated cycle of iterations (or sprints in scrum framework). At the end of the iteration, a working product is demonstrated to stakeholders. This minimizes overall risk and allows adaptation to changing requirements at any point during a project's life [14]. In each iteration, a set of user stories is planned to be implemented, the development team analyzes, designs, codes, tests and delivers a functioning increment of the product. In agile development, A user story (US) is the unit of which software features are estimated and developed. A user story describes a feature of the software. To each US is assigned a priority and a level of effort based on the team's capability and capacity [15]. The commonly used approach to estimate the US effort is story points employing one of existing methods like planning poker which measures the effort required following Fibonacci scale (1, 2, 3, 5, 8, 13, 21, and so on) [16]. These story points are also used to compute the team velocity, which represents the total of points completed and accepted by the customer at the end of the iteration. The velocity, in turn, is used to determine the number of features that should be scheduled for the next iteration [17].

According to recent studies [18][19], most frequently used techniques to estimate effort in agile software development (ASD) context are expert judgement, planning poker and use case points (UCP). Whereas, story points, UCP and function points are the most frequently used size measure during effort estimation [19]. In fact, these estimation techniques are subjective since they rely greatly on expert assessment to generate an estimate. This may lead to inaccuracy and more importantly inconsistencies between estimates [20].

Up to now, few studies have investigated the use of machine learning techniques to estimate effort in ASD context [21][22][23]. In addition, none of these studies has employed a grid search optimization method to improve the performance of effort estimation model. In this paper, we propose an improved model for estimating the software effort based on

support vector regression (SVR) optimized by grid search method (GS). The story point and the velocity were used as inputs of the prediction model. The empirical evaluation uses twenty-one historical agile software projects and leave-one-out cross-validation method for model selection. The accuracy is assessed by three accuracy measures Pred(0.25), MMRE and MdMRE.
To summarizes the objectives of this study, three research questions (RQs) are discussed:
RQ1: *Does the use of grid search method optimization improve the accuracy of the estimation generated by SVR?*
RQ2: *Does SVR optimized by GS method outperforms SVR using other optimization methods?*
RQ3: *is SVR-GS method more accurate in terms of the three accuracy criteria than regression models?*
The remaining part of the paper proceeds as follows: Section 2 gives a brief review of effort estimation techniques used within ASD with particular focus on machine learning methods. Section 3 presents the proposed model and optimization method. The empirical design is described in Section 4. The results are presented and discussed in section 5. Finally, Section 6 concludes the paper.

II. Previous Work

Abrahamsson *et al.* proposed a method for predicting development effort based on user stories. The authors extracted the predictors from US and used them to train the model. They considered the construction of three types of models: regression models, neural networks, and support vector machine. Their empirical evaluation used 1325 user stories and leave-one-out cross validation procedure. Their results showed that the proposed effort estimation models worked reasonably well if the user stories are written in a structured manner [24].

The work in [25] reported on the performance of estimation models built using Story Points and COSMIC Function Points and demonstrated that, for the organization in which the data was collected, the use of COSMIC Function Points leads to estimation models with much smaller variances. The study also showed that the use of COSMIC Function Points allows objective comparison of productivity across tasks within a Scrum environment.

The authors in [23] developed a belief network effort estimation model. They claimed that the model is relatively small and simple and all the input data are easily elicited, insofar the impact on agility is minimal. The model was validated using a dataset of 160 tasks from real agile projects. The results indicated a very good accuracy in terms of Pred (Pred(0.25) =100%).

Zia *et al.* proposed an effort estimation model for ASD [26]. The model was based on user story and developed to accommodate most of the characteristics of agile methodology, especially, adaptation and iteration. the model was assessed using empirical data collected from 21 software projects. The experimental results demonstrated that the model made acceptable prediction accuracy in terms of MMRE and Pred.

In [27], four types of neural networks namely: General Regression Neural Network (GRNN), Probabilistic Neural Network (PNN), Group Method of Data Handling (GMDH) Polynomial Neural Network (PNN) and Cascade-Correlation Neural Network (CCNN), were evaluated and compared using different accuracy measures. The results showed that cascade network outperformed other networks.

Tung and Hanh [28] introduced a novel formula based on team velocity and story points factors. The parameters of the formula were optimized by combining artificial bee colony and particle swarm optimization algorithms. Their empirical results indicated that their approaches outperformed the Zia *et al.*'s regression method [26][27] in terms of MMRE, MdMRE, Pred(8%), R^2 and MAR.

Satapathy *et al.* employed SVR with various kernel to improve the estimation of agile software using story points approach. The model then was optimized and evaluated using data about project from Zia work. The results showed that SVR with RBF kernel produced the most accurate estimates in terms of Pred(0.25) and MMRE. Nevertheless, the optimization method used in their work can still be enhanced to fortify the accuracy estimates. Our study focuses on tuning the SVR with RBF kernel by grid search optimization method.

This section has attempted to provide a brief summary of the literature relating to effort estimation in agile software development. The next section presents our SVR model based on grid search optimization method.

III. The Proposed Model: Grid based SVR

This section presents an overview of the SVR models designed in this paper particularly SVR with RBF kernel and illustrates how these models were trained and optimized by grid search method.

A. Overview of SVR Technique

Support Vector Machines (SVM) as described in [29] have shown to deliver promising solutions in various classification and regression tasks thanks to their ability to avoid local minima, improved generalization capability, and sparse representation of the solution. SVM are based on Structural Risk Minimization (SRM) principle and thus tries to control the upper bound of generalization risk while reducing the model complexity. In addition, they do not suffer from over fitting problem and local minimization issues and hence offer enhanced generalization capability.
For regression tasks, Vapnik *et al.* proposed in [30] an SVM called ε-support vector regression (ε-SVR), which performs prediction tasks from the ε-insensitive loss function. Because

the ε parameter is useful if the approximation accuracy is specified beforehand, it is better to find a procedure to optimize this accuracy without depending a priori on a value set. This procedure was studied by Schölkopf *et al.* in [31]. They proposed a new formulation, called ν-support vector regression (ν-SVR), that automatically minimizes the ε-insensitive loss function and changes the SVR formulation by using a new ν parameter whose value is between [0,1]. In addition to minimizing the ε value, the ν parameter is used for controlling the number of support vectors, since the value of ε influences the choice of support vectors. In this study, a special form of SVM i.e., Support Vector Regression (SVR) is utilized for modeling the input–output functional relationship or regression purpose and is explained next.

Given a set of input–output sample pairs $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ where $x_i \in \mathbb{R}^p$ and $y_i \in \mathbb{R}$, the objective of ν-SVR technique is to approximate the nonlinear relationship given in (1), such that $f(x)$ should be as close as possible to the target value y and should be as flat as possible in order to avoid over-fitting.

$$f(x) = w^T \cdot \Phi(x) + b \tag{1}$$

where w^T is the weight vector, b is the bias and $\Phi(x)$ represents the transformation function that maps the lower dimensional input space to a higher dimensional space. The primal objective of the problem thus reduces to (2), in order to ensure that the approximated function meets the above two objectives of closeness and flatness.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|w\|^2 + C\{\gamma \cdot \varepsilon + \frac{1}{2}\sum_{i=1}^{n}(\xi + \xi^*)\} \\ \text{subject to} \quad & y_i - \langle w^T \cdot \Phi(x)\rangle - b \leq \varepsilon + \xi_i^*, \\ & \langle w^T \cdot \Phi(x)\rangle + b - y_i \leq \varepsilon + \xi_i^*, \\ & \xi, \xi^* \geq 0 \end{aligned} \tag{2}$$

where ε is a deviation of a function $f(x)$ from its actual value and, ξ, ξ_i^* are additional slack variables introduced by [32], which determines that, deviations of magnitude ξ above ε error are tolerated. The constant C known as regularization parameter determines the trade-off between the flatness of f and tolerance of error above ε.

The SVR algorithm involves the use of Lagrangian multipliers, which rely solely on dot products of $\Phi(x)$. This can be accomplished via kernel functions, defined $K(x_i, x_j) = (\Phi(x_i), \Phi(x_j))$. Thus, the method avoids computing the transformation $\Phi(x)$ explicitly. The details of the solution can be found in [33].

In this study, we consider SVR with RBF kernels since it has been shown a good performance [34][35]. The RBF kernel is calculated as $K(x_i, x_j) = exp(-\gamma\|x_i - x_j\|^2)$. In most cases, the parameters C and ε significantly impact ε-SVR generalization performance. For RBF kernels, the parameter γ must also be carefully selected.

B. Grid Search Method

It is well known that the parameter settings could have a significant impact on the estimation accuracy of trained effort estimation techniques. Therefore, building an accurate model requires selection of optimal values of its learning parameters [34]. However, finding optimal values is complicated task and various approaches have been proposed in the literature to address this issue, such as grid search (GS) [36], particle swarm optimization (PSO) [37] and genetic algorithm (GA) [38].

In order to enable SVR models, developed in this study, to achieve a higher prediction accuracy over the agile software project dataset, we employed grid search (GS) as optimization method combined with cross-validation procedure. The main idea behind the grid search method is that different pairs of parameters are tested and the one with the highest cross validation accuracy is selected. The major advantage of GS method is its high learning accuracy and the ability of parallel processing on the training of every SVR, because they are independent of each other. Although GS method can find the optimum parameters, the computational complexity is very big obviously, and the time spent is very large, especially for large sample data. In our case, the dataset used is very small and we limited the search space to the most promising values guided by previous studies [39].

The GS method finds the best configuration of SVR models by evaluating every possible combination of Table 1 with respect to mean square error (MSE) based error function using 5-fold cross-validation approach. The best configuration of each technique that minimizes MSE is then selected.

TABLE I
GRID SEARCH PARAMETER FOR SVR MODELS

Techniques	**Parameters**

SVR	Type= {eps-regression}
	Kernel function = { RBF }
	Complexity = {150,1000,10000}
	Kernel parameter ={0.0001, 0.001, 0.005}
	Epsilon={1e-07, 2e-05, 3e-05}

IV. Empirical Design

This section describes the empirical design of this study including: (1) the description of the selected dataset, (2), the accuracy measures used to evaluate the proposed SVR models and (3) the experimental process followed to construct and compare the different configuration of SVR models.

A. Dataset Description

The dataset used in this paper contains 21 agile software projects developed by six software houses. Each project is described by two attributes. The first attribute is the total story points and the second attribute is the team velocity. Table 2 summarizes descriptive statistics of the selected dataset, including size of dataset, number of attributes, median, mean, minimum, maximum, skewness and kurtosis of effort. The values of the two last measures show that the effort is not normally distributed with the presence of few outliers.

TABLE II
DESCRIPTION STATISTICS OF THE DATASET

Characteristic		Value
Number of software projects		21
Number of attributes		2
Effort	Min	21
	Max	112
	Mean	56.43
	Median	52
	Skewness	0.6049
	Kurtosis	-0.7679

B. Performance Measures

We employ the following criteria to assess and compare the accuracy of the effort estimation models. A common criterion for the evaluation of effort estimation models is magnitude of relative error (MRE), which is defined as

$$MRE = \frac{|Effort_{actual} - Effort_{estimated}|}{Effort_{actual}} \tag{3}$$

The MRE values are calculated for each project in the dataset, while mean magnitude of relative error (MMRE) computes the average over N projects as illustrated in Eq. 4.

$$MMRE = \frac{1}{N}\sum_{i=1}^{N} MRE_i \tag{4}$$

Generally, the acceptable target value for MMRE is 25%. This indicates that on the average, the accuracy of the established estimation models would be less than 25%. Another widely used criterion is the Pred(l) which represents the percentage of MRE that is less than or equal to the value l among all projects. This measure is often used in the literature and is the proportion of the projects for a given level of accuracy. The definition of Pred(l) is given as follows:

$$Pred(l) = \frac{k}{N} \tag{5}$$

Where N is the total number of observations and k is the number of observations whose MRE is less or equal to l. A common value for l is 0.25, which is also used in the present study. The Pred(0.25) represents the percentage of projects whose MRE is less or equal to 0.25. The Pred(0.25) value identifies the effort estimates that are generally accurate whereas the MMRE is fairly conservative with a bias against overestimates [40][41]. For this reason, MdMRE has been also used as another criterion since it is less sensitive to outliers (Eq. 6).

$$MdMRE = median(MRE_i) \quad i \in \{1, 2, \ldots, N\} \tag{6}$$

C. Validation Methods

Leave-one-out cross validation (LOOCV) method was used to evaluate the generalization ability of the estimation models in testing phase while 5-fold cross validation was used in training phase to tune the SVR parameters. LOOCV was chosen because it has been shown to give an almost unbiased estimator of the generalization properties of learning models, and therefore provides a sensible criterion for model selection and comparison. In addition, LOOCV method can generate the same results in a particular dataset if the experiment is replicated, which is not the case for k-fold cross validation [41]. On other hand, 5-folds CV was chosen to prevent model from been overfitted to data during training phase which might affect its generalization ability.

V. RESULTS AND DISCUSSION

This section reports and discusses the results of empirical experiments performed using SVR models presented in Section II and following the design described in the previous section.
To carry out the empirical experiments, different R packages were used to develop an R prototype employed to construct the proposed model. As indicated previously, SVR parameters were tuned using grid search optimization method with the values given in Table I. 5-fold cross validation was used to select the best model according to mean squared error (MSE). Once the model is trained, it is tested on the observation left out (LOOCV was used for testing). The experiment was repeated a certain number of times. The results reported in this paper are the best results obtained. We evaluated the accuracy of the predicting effort using SVR-RBF-GS based on criteria MMRE, MdMRE and Pred(0.25). Table III shows the obtained results of the SVR-RBF with and without any optimization method. i.e. with default parameters, and SVR-RBF with GS method.
As it can been seen from Table III, the use of grid search optimization method improved significantly the performance of the SVR-RBF model in terms of all accuracy measures. This not surprising since the GS method searches the whole parameter space and ensures to find the optimal solution if such a solution exists.

TABLE III
COMPARISON BETWEEN SVR-RBF WITH AND WITHOUT GS OPTIMIZATION

Technique	MMRE	MdMRE	Pred(0.25)
SVR-RBF	0.0620	0.0426	100
SVR-RBF-GS	0.1640	0.0893	80.952

To further investigate the efficiency of the proposed model for estimating effort in ASD, we want to compare them against recent results reported in scientific articles: [21], [22], [26]–[28]. Unfortunately, these papers' results used different validation methods and accuracy measures. For example,

TABLE IV
COMPARISON OF GS-BASED SVR-RBF MODEL TO OTHER METHODS

Technique	MMRE	MdMRE	Pred (0.25)	Ref.
SVR-RBF optimized by Grid Search (SVR-RBF-GS)	0.0620 0,0613 (MMER)	0.0426 0.0408 (MdMER)	100 Pred(8%)=66.667	
Support Vector Regression with RBF kernel (SVR-RBF)	0.0747	NA	95.9052	[22]
ABC-PSO	0.0569 0,0564 (MMER)	0.0333 0,0344 (MdMER)	100 Pred(8%)=66.667	[28]
Zia *et al.* regression model	0.0719	0.0714	57.14	[26]
General Regression Neural Network (GRNN)	0.3581	NA	85.9182	[27]
Probabilistic Neural Network (PNN)	1.5776	NA	87.6561	
GMDH Polynomial Neural Network (GMDH-PNN)	0.1563	NA	89.6689	
Cascade-Correlation Neural Network (CCNN)	0.1486	NA	94.7649	
Decision Tree (DT)	NA	NA	38.09	[21]
Stochastic Gradient Boosting (SGB)	NA	NA	85.71	
Random Forest (RF)	NA	NA	66.67	

ref. [21] used MMER and MdMER instead of MMRE and MdMRE, while ref. [28] computed Pred at level 0.08 instead of 0.25 which is commonly used in effort estimation. In addition, ref. [28] did not describe the validation method used to evaluate ABC-PSO model or it may have used all data for training and testing. Nevertheless, we have attempted to follow the most used empirical design in effort estimation to make a fair comparison with above-mentioned papers. Table 4 compares the results obtained by our GS-based SVR-RBF model to previous results obtained by the references indicated in Ref. column. These results show that the proposed method outperforms all other techniques, except ABC-PSO algorithm, in terms of all accuracy measures available. Remarkably, the GS-based SVR-RBF produced comparable results to ABC-PSO in terms of Pred(0.25) and Pred(0.08). However, our method made slightly higher MMRE and MdMRE (+0.005 and +0.009). These results highlight the potential role that GS-based SVR can play in providing more accurate effort estimation in agile software development.

VI. CONCLUSION

This study proposed and investigated an improved method for agile software effort estimation. The proposed method applies a grid search optimization to tune the hyperparameter of SVR-RBF technique. The proposed model was evaluated and compared with most recent results reported in recent articles. The comparison was performed using 21 agile software projects and three accuracy measures (MMRE, MdMRE and Pred(0.25)). Returning to the research questions (RQs 1-3) posed at the beginning of this study, it is now possible to state that:

RQ1: *Does the use of grid search method optimization improve the accuracy of the estimation generate by SVR?*
The results obtained suggest that the use of GS method improves greatly the performance of SVR-RBF model in terms of MMRE and Pred(0.25). Pred was increased by 19% while the error was decreased by 0.1.

RQ2: *Does SVR optimized by GS method outperforms SVR using other optimization methods?*
The results confirmed that GS-based SVR-RBF is more accurate than SVR-RBF optimized by various methods in [23].

RQ3: *is SVR-GS method more accurate in terms of the three accuracy criteria than regression models?*
The finding suggest that GS-based SVR-RBF performs better than Zia' regression in terms of the three accuracy measures. However, GS-based SVR produced similar results to ABC-PSO based regression model.

Finally, it is difficult to claim that GS-based SVR-RBF is the best technique in all situation since different results might be obtained when using: different validation methods and different datasets. Therefore, further research is required to confirm or infirm the results obtained in this study.

REFERENCES

[1] S. Hastie and S. Wojewoda, *Standish group 2015 chaos report - q&a with jennifer lynch*, Web Page, 2015. [Online]. Available: https://www.infoq.com/articles/standish-chaos-2015.

[2] A. Trendowicz and R. Jeffery, *Software Project Effort Estimation: Foundations and Best Practice Guidelines for Success*. Springer Publishing Company, Incorporated, 2014, p. 469.

[3] M. Jørgensen and M. Shepperd, "A systematic review of software development cost estimation studies," *IEEE Trans. Software Eng.*, vol. 33, no. 1, pp. 33–53, 2007.

[4] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation," *Information & Software Technology*, vol. 91, pp. 1–21, 2017.

[5] M. Jørgensen and T. Halkjelsvik, "The effects of request formats on judgment-based effort estimation," *Journal of Systems and Software*, vol. 83, no. 1, pp. 29–36, 2010.

[6] M. Jorgensen, "Practical guidelines for expert-judgment-based software effort estimation," *IEEE Software*, vol. 22, no. 3, pp. 57–63, 2005.

[7] B. W. Boehm, Clark, Horowitz, Brown, Reifer, Chulani, R. Madachy, and B. Steece, *Software Cost Estimation with Cocomo II with Cdrom*. Prentice Hall PTR, 2000, p. 540.

[8] S. Basri, N. Kama, H. M. Sarkan, S. Adli, and F. Haneem, "An algorithmic-based change effort estimation model for software development," in *23rd Asia-Pacific Software Engineering Conference, APSEC 2016*, pp. 177–184.

[9] D. Nandal and O. P. Sangwan, "Software cost estimation by optimizing cocomo model using hybrid batgsa algorithm," *International Journal of Intelligent Engineering and Systems*, vol. 11, no. 4, pp. 250–263, 2018.

[10] A. Zakrani and A. Idri, "Applying radial basis function neural networks based on fuzzy clustering to estimate web applications effort," *International Review on Computers and Software*, vol. 5, pp. 516–524, 2010.

[11] F. A. Amazal, A. Idri, and A. Abran, "Software development effort estimation using classical and fuzzy analogy: A cross-validation comparative study," *International Journal of Computational Intelligence and Applications*, vol. 13, no. 3, 2014.

[12] A. Idri, A. Zakrani, M. Elkoutbi, and A. Abran, "Fuzzy radial basis function neural networks for web applications cost estimation," in *4th International Conference on Innovations in Information Technology*, pp. 576–580.

[13] T. R. Benala and R. Mall, "Dabe: Differential evolution in analogy-based software development effort estimation," *Swarm and Evolutionary Computation*, vol. 38, pp. 158–172, 2018.

[14] A. Moran, *Agile Risk Management*. Springer Publishing Company, Incorporated, 2014, p. 101.

[15] N. C. Haugen, "An empirical study of using planning poker for user story estimation," in *AGILE 2006 (AGILE'06)*, 9 pp.-34.

[16] M. Usman, E. Mendes, F. Weidt, and R. Britto, "Effort estimation in agile software development: A systematic literature review," in *10th International Conference on Predictive Models in Software Engineering, PROMISE 2014*, pp. 82–91.

[17] M. Cohn, *Agile Estimating and Planning*. Prentice Hall PTR, 2005.

[18] S. Bilgaiyan, S. Sagnika, S. Mishra, and M. Das, "A systematic review on software cost estimation in agile software development," *Journal of Engineering Science and Technology Review*, vol. 10, no. 4, pp. 51–64, 2017.

[19] M. Usman, J. Börstler, and K. Petersen, "An effort estimation taxonomy for agile software development," *International Journal of Software Engineering and Knowledge Engineering*, vol. 27, no. 4, pp. 641–674, 2017.

[20] M. Choetkiertikul, H. K. Dam, T. Tran, T. T. M. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Software Eng.*, pp. 1–1, 2018.

[21] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," *Innovations in Systems and Software Engineering*, vol. 13, no. 2-3, pp. 191–200, 2017.

[22] S. M. Satapathy, A. Panda, and S. K. Rath, "Story point approach based agile software effort estimation using various svr kernel methods," in *26th International Conference on Software Engineering and Knowledge Engineering*, vol. 2014-January, pp. 304–307.

[23] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," *Journal of Systems and Software*, vol. 127, pp. 109–119, 2017.

[24] P. Abrahamsson, I. Fronza, R. Moser, J. Vlasenko, and W. Pedrycz, "Predicting development effort from user stories," in *2011 International Symposium on Empirical Software Engineering and Measurement*, pp. 400–403.

[25] C. Commeyne, A. Abran, and R. Djouab, "Effort estimation with story points and cosmic function points - an industry case study," *Software Measurement News*, vol. 21, no. 1, pp. 25–36, 2016.

[26] Z. Zia, S. K. Tipu, and S. K. Zia, "An effort estimation model for agile software development," *Advances in Computer Science and its Applications*, vol. 2, no. 1, pp. 314–324, 2012.

[27] A. Panda, S. M. Satapathy, and S. K. Rath, "Empirical validation of neural network models for agile software effort estimation based on story points," in *3rd International Conference on Recent Trends in Computing, ICRTC 2015*, vol. 57, pp. 772–781.

[28] K. T. Tung and L. T. M. Hanh, "A novel hybrid abc-pso algorithm for effort estimation of software projects using agile methodologies," *J. Intelligent Systems*, vol. 27, no. 3, pp. 489–506, 2018.

[29] V. N. Vapnik, *The nature of statistical learning theory*. Springer-Verlag, 1995, p. 188.

[30] V. Vapnik, S. E. Golowich, and A. Smola, *Support vector method for function approximation, regression estimation and signal processing*, 1996.

[31] B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett, "New support vector algorithms," *Neural Computation*, vol. 12, no. 5, pp. 1207–1245, 2000.

[32] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[33] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[34] M. Hosni, A. Idri, A. Abran, and A. B. Nassif, "On the value of parameter tuning in heterogeneous ensembles effort estimation," *Soft Computing*, pp. 1–34, 2017.

[35] A. Idri and I. Abnane, "Fuzzy analogy based effort estimation: An empirical comparative study," in *17th IEEE International Conference on Computer and Information Technology, CIT 2017*, pp. 114–121.

[36] X. Ma, Y. Zhang, and Y. Wang, "Performance evaluation of kernel functions based on grid search for support vector regression," in *IEEE Conference on Robotics, Automation and Mechatronics*, 2015, RAM:283–288.

[37] H. Zhang, M. Wang, and X. Huang, *Parameter selection of support vector regression based on particle swarm optimization*, 2010.

[38] A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. Cornélio, "Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," *Information & Software Technology*, vol. 52, no. 11, pp. 1155–1166, 2010.

[39] P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, "Software effort estimation using machine learning techniques with robust confidence intervals," in *7th International Conference on Hybrid Intelligent Systems, HIS 2007*, pp. 352–357.

[40] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion mmre," *IEEE Trans. Software Eng.*, vol. 29, no. 11, pp. 985–995, 2003.

[41] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," *Journal of Systems and Software*, vol. 86, no. 7, pp. 1879–1890, 2013.