2021 2nd International Conference on Intelligent Engineering and Management (ICIEM)

A State of the Art Regressor Model's comparison for Effort Estimation of Agile software

Mohit Arora
Dept. of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
mohit.15980@lpu.co.in

Abhishek Sharma
Dept. of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
abhishek.11701051@lpu.in

Sapna Katoch
Dept. of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
sapna.11701025@lpu.in

Mehul Malviya
Dept. of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
mehul.11705660@lpu.in

Shivali Chopra
Dept. of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
shivali.19259@lpu.co.in

2021 2nd International Conference on Intelligent Engineering and Management (ICIEM) | 978-1-6654-1450-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICIEM51511.2021.9445345

***Abstract*— Advances and innovations in the field of software engineering are increasing rapidly. This sensitizes researchers to explore the various cross-cutting concerns incorporated to handle the complexities of various domains of interest. One such thrust area is effort estimation in Agile-inspired software. Estimation has always been challenging in an Agile environment because of its requirement volatility. This paper introduces a critical review of state-of-the-art regression techniques to estimate the efforts of Agile projects. It can be concluded from the obtained results that ensemble estimation techniques outperformed single techniques of estimation. The data have been taken from various companies implementing Agile practices. Different regressors have been trained, tested, cross-validated, and optimized to fill the actual and estimated effort gap. We have used six regression techniques in this paper, Extreme Gradient Boosting (XGB), Decision Tree (DT), Linear Regressor (LR), Random Forest (RF), Adaptive Boosting (AdaBoost) and, Categorical boosting (CatBoost) regressors. Cat Boost regressor wins with the lowest Root Mean Square Error (RMSE) in comparison to other regressors.**

***Keywords*— Agile Estimation, Machine Learning, Regression, Effort, Optimization.**

I. INTRODUCTION

It is inevitable for any software project to continue without a planning process for any phase-related activities. Customer interaction at different stages of the software life cycle, as well as their unquenchable desire for perfection, necessitates improvements and adaptation in Agile-based projects restricts the use of traditional techniques of estimation like Constructive Cost Model (COCOMO), etc. Software projects are inherently heterogeneous and complex because of a buffet of estimation factors [35] which has specific highs and lows in different enterprises. There are different environments and workbenches in which corporate sector stakeholders work and plan various dimensions of projects. Planning being an essential ingredient to pave a happy path for any software project is indispensable. The most essential part of planning is project estimation. All the stakeholders of a typical IT project strive for accurate effort estimation but the gap between actual and estimated effort widens based on dynamic specificities of an Agile-inspired project development. An attempt in this direction has been made using all the prominent regression techniques.

As per the International Society for Parametric Analysis (ISPA) [1], 66.6% of software projects fail to deliver both in time and budget. The root causes of software project failure are an inaccurate estimation of the cost and time of the project, the uncertainty of the system, the number of people needed, and software requirements. Change and sprint-wise calculation are two difficult aspects of estimating scrum-based projects. Most IT companies have implemented hybrid models, which are primarily powered by Agile umbrella techniques. Changes in effort calculation methods can be seen as process models moving from heavy weight models like an iterative waterfall to light weight models such as Agile [2]. All traditional estimation methods, such as top-down estimation, expert judgement, Delphi-Cost estimation, etc. are suited for heavyweight process models in some form or another, but they are not deemed fit for bridging the actual effort and estimated gap of Agile projects. As a result of the volatile nature of Agile based project specifications, analysts began searching for options, eventually settling on machine learning techniques.

II. RELATED WORK

In the 1980s, M Shepperd and M Jorgensen [7] published a review paper that recognized more than ten estimation methods for effort estimation, with regression estimation techniques outperforming empirical estimation techniques. Despite a large number of studies on Machine learning models in software project estimation, conflicting results have been accounted for in terms of estimation accuracy of these ML models. For instance, when a similar ML model is built with distinct datasets [3] [8] or scenarios [9], the estimation accuracy changes. The regression and ML models specified in [3] declares that machine learning models are superior to the regression models, whereas authors in [10] conclude that the regression models are superior to machine learning models. In terms of the association between different ML models like Case-Based Reasoning (CBR) and Artificial Neural Network (ANN), research in [8] suggests that the latter outperforms the former, whereas research in [18] reveals the opposite. Furthermore, the hypothesis of machine learning systems is more complicated than conventional estimation procedures. It is critical to laboriously condense the empirical proof on ML models in ongoing research and practice to promote the use of ML procedures in the Software Development Effort Estimation (SDEE) area. As per Agile State-of-the-Art reports by Collabnet VersionOne, industry professionals are still relying on Expert judgement and Delphi cost estimation methods for estimation rather than machine learning. CBR, ANN, DT [30], Bayesian Network (BN), Support Vector Machine (SVM) [28], Genetic Algorithm (GA), Genetic Programming (GP) [11] [12] [13] [14] etc. are some of the

ML methods have been used for SDEE, but the majority of them have not yet been applied to Agile estimation. The above machine learning systems could be used alone or in combination with non-ML or other ML approaches. For instance, for weighting and choice, GA has been combined with ANN, CBR, and Support Vector Regressor (SVR). For execution, fuzzy logic [15] was combined with DT, ANN, and CBR. For estimation, various datasets have been used, such as ISBSG, PROMISE data repository, JIRA Atlassian repositories, and so on [16] [3]. The three prominent precision measurements that Mean Magnitude Relative Error (MMRE), Percentage Relative Error Deviation (PRED) and Median of the Magnitude of Relative Error (Md MRE) have been used for model evaluation [17]. BN [3][18][19] was found to have the worst MMRE of all ML Techniques, compared to SVR (34%), ANN (37%), AR (49%), CBR (51%), and DT (55%) separately for project estimation, which includes both lightweight and traditional methodologies. While research shows that SVR and ANN [9] outperform other models. This does not signify that we can use them without restriction as increasing the number of hidden layers will eventually escalate in preparation time and can generate over-fitting issues [3]. It has also been seen in literature that ML models have been used in conjunction with COCOMO estimation, regression models, EJ, and Function Point Analysis [20]. In one study, it is also stated that Regression is more accurate than GP. So, based on the acquired data, we've concluded that ML models outperform the non-ML approach. Analysts advise [21] [22] that deciding the best model in a specific setting rather than the best single model is more productive, because estimation models differ from one dataset to the next, making them vulnerable. According to studies on knowledge mining, research strategies produce exact results as compared to single strategies since each strategy has consistency and flaws, and Joining them would mitigate the flaws. Homogeneous (for example Bagging and SVR, RF, Multi-Layer Perceptron (MLP) [23] [25] [27], LR, Radial Basis Function (RBF), Artificial Neuro-Fuzzy Inference Systems (ANFIS) [6], CBR, RF, Stochastic Gradient Boosting (SGB) [26] [28], Classification and Regression Trees (CART), etc.) and heterogeneous effort calculation systems are differentiated by their blend laws and base models. According to the study, Heterogeneous ML procedures are the most common approach for forming ensembles. It was learned that homogeneous models relying on Decision Tree are the most accurate, followed by CBR homogenous models, and finally, homogeneous models relying on SVR. Neuro-Fuzzy, ANN, CBR, DT, Regression and SVR [5] are the most extensively used for classes, with DT, CBR, Regression, SVR. Mix Rules have also been extracted for combining base model endeavors and are divided into two sections: Linear and Non-Linear. The most widely used straight mix rules are mean, mean weighted, and center. The most commonly used non-straight concepts are MLP, SVM, CART, FIS with c implies, and subtractive grouping. In addition to regressors, researchers have also used PSO [33], Hybrid ABC-PSO algorithm [29], Naïve Bayes [31], Deep learning [32], Multiagent techniques [24]. To show a trail of estimation patterns, all the techniques discussed and described in this section are derived from general estimation approaches. [4] [36].

III. APPROACH

We have considered Six Software houses Agile projects data [34] as an input. The proposed approach includes data preprocessing, model selection, testing, and evaluation. The proposed methodology flowchart has been given in Fig. 1.

A. Data Preparation

Upon analyzing, we have found that data is not normalized. Following pre-processing steps have been employed for data transformation.

Step 1: Analyze and load Agile project dataset [34].

Step 2: Perform feature selection by placing project final velocity and number of story points in features, and actual effort in labels.

Step 3: Expanding dataset using k means Synthetic Minority Over-sampling Technique (SMOTE).

Step 4: Box-Cox transformation and normalization of dataset.

Step 5: Scaling of the data of project velocity and story points within the range[0,1], wherein 'D' and 'i' denotes the dataset and an item of the dataset respectively and the normalized value of 'N' of 'i' is computed by using the equation below.

$$N = \frac{(i-\min(D))}{(\max(D)-\min(D))} \tag{1}$$

where min(D) and max(D) are the minimum and maximum values, of dataset D respectively.

B. Dataset Partitioning and Model Selection

After data is normalized and scaled, it is divided into two sets: testing and training.

Step 6: Partitioning of data set into training and testing sets using train_test_split method (80: 20 and random state= 0).

Step 7: Perform Model Selection that is Regression models.

Step 8: Finding the best regressor by setting up hyperparameters using Randomized and Grid search.

Step 9: Fit model on training data.

C. Testing:

In this section model prediction on test data has been performed.

Step 10: Performing prediction on test and train data.

Step 11: Comparing predicting values with original values in the dataset.

D. Performance Evaluation:

In this step, model performance will be evaluated through R_Square, training and testing accuracy, Mean Squared Error (MSE), and Prediction (PRED) accuracy, and Root Mean Square Error (RMSE).

Step 12: Calculate the loss function, that is MSE.

Step 13: Calculate the average variance between the predicted and actual dataset values.

Step 14: Compare models using different comparison metrics such as RSquare, MSE, and RMSE.

IV. RESULT

We have applied six regressors on preprocessed dataset and Fig.2 shows the line plot of predicted vs actual values. Accuracy of CatBoost regressor can be seen in Fig. 2. through near perfect overlapping of predicted and tested curve. However, we have tuned parameters of all the regressors through Randomized and Grid search and the same has been recorded in Table I.

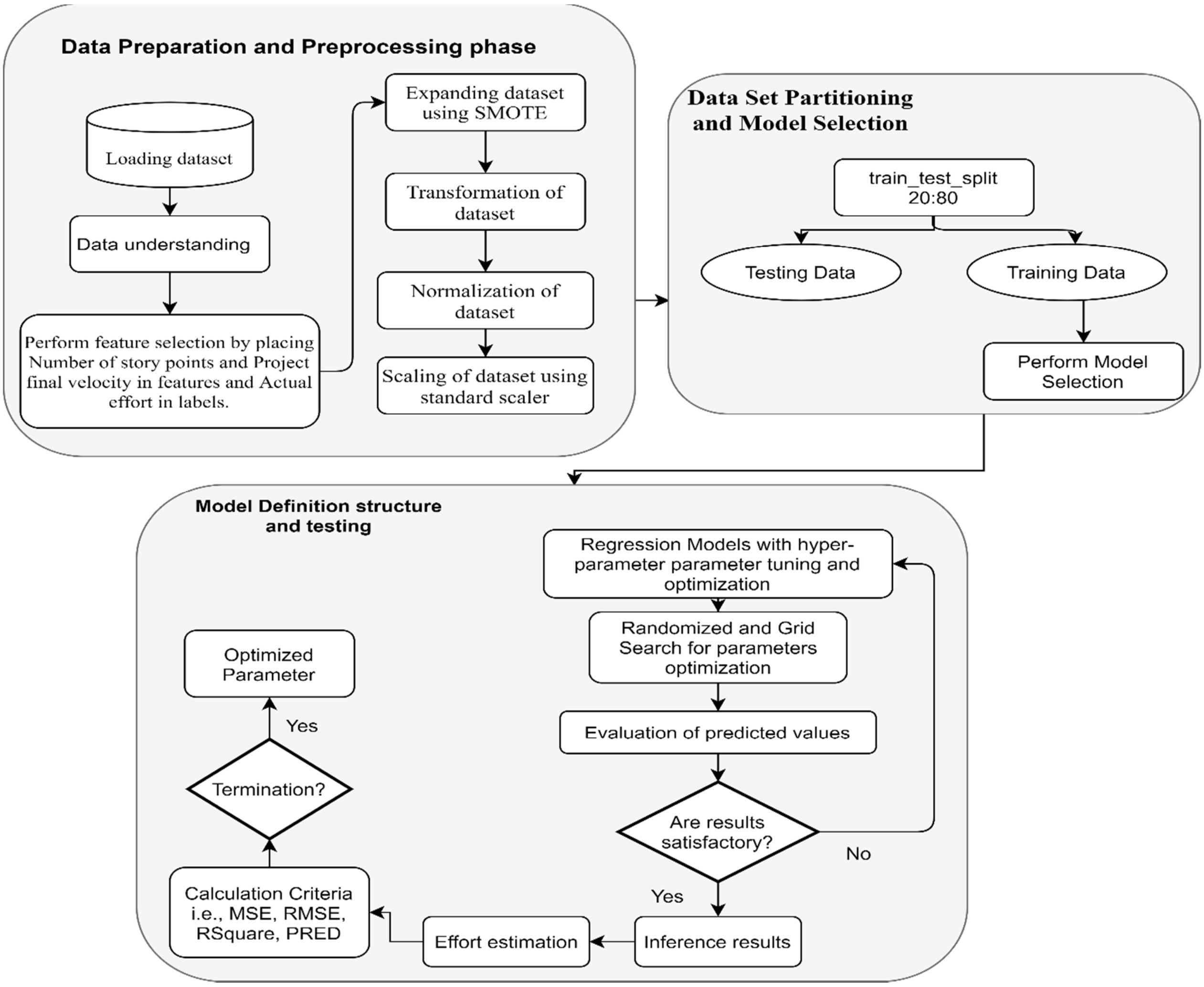

Fig. 1. Proposed Methodology

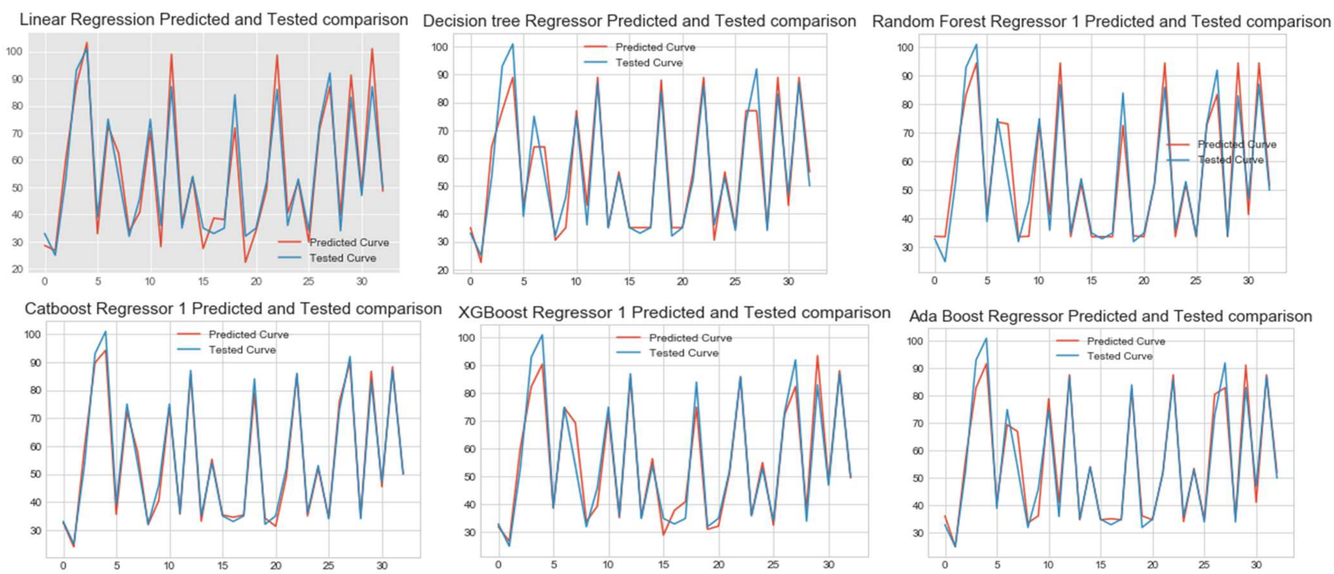

Fig. 2. Comparative Analysis of all regressors

Fig. 2. shows the comparative analysis of various regressor models. CatBoost regressor outperformed all others with an estimation accuracy of 98.52%. This is because it handles the categorical features (project velocity and number of story points) automatically. RF and DT perform average with an There are 162 projects on which regressor models have been applied. The value of RSquare also shows promising results. accuracy of 92.15% and 91.79% respectively. XGBoost also performs well with our dataset with an accuracy of 94.27% with a learning rate of 0.01. To avoid overfitting, we have used SMOTE and trained the regressor model with more data.

Fig. 3. displays prediction accuracy, RSquare, and RMSE. Lower the RMSE and Higher the prediction accuracy determines the best model.

TABLE I. Performance evaluation and hyperparameters of all Regressor models

Algorithms	**Prediction Accuracy**	**RSquare**	**MSE**	**RMSE**	**Hyperparameters**
LR	0.9345	0.93	34.25	5.8523	copy_X is set True, fit_intercept is set True, None is given for n_jobs, normalize is set not True
DT Regressor	0.9215	0.92	41.05	6.4070	Value of min_samples_leaf is 3, value of max_leaf_nodes is set 10, min_samples_split is given 10, max_depth is set 6, criterion is selected as 'mae', random_state= 0
RF Regressor	0.9179	0.9179	42.9 1	6.5505	max_depth is set 2, random_state is 0, n_estimators are set 100
CatBoost Regressor [39]	0.9852	0.9852	7.72	2.7784	Subsample is given 0.8, value of scale_pos_weigh is 0.9, reg_alpha is given 0.07, objective is set 'reg: linear', value of n_estimators is set 76, min_child_weight is set 1, max_depth of regressor is 2, learning_rate is given 0.1, gamma= 1, colsample_bytree is 0.4
XGB Regressor [37]	0.9427	0.9427	29.97 10	5.4745	Value learning_rate is 0.01, n_estimators are set to 1000, loss is 'linear'
AdaBoost Regressor [38]	0.9511	0.9511	25.5580	5.0555	Depth is 6, learning_rate is 0.1, value of iterations is 100

Fig. 3. Performance evaluation of all regressors

V. CONCLUSION AND FUTURE SCOPE

The paper presents a comprehensive review of state-of-the-art regressor models on Agile projects and found that CatBoost regressor outperformed all other regressors. The models are tuned using randomized and grid search. As per our knowledge, no model in the current literature presents such a higher prediction accuracy.

As a future work, more real Agile project data can be used for detailed analysis. Nature-inspired algorithms can be used for hyperparameter tuning of proposed models.

REFERENCES

[1] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, “Neural network models for software development effort estimation: a comparative study,” Neural Comput. Appl., vol. 27, no. 8, pp. 2369–2381, 2016.

[2] R. Popli and N. Chauhan, “Cost and effort estimation in agile software development,” Optim. Reliab. Inf. Technol. (ICROIT), 2014 Int. Conf., pp. 57–61, 2014.

[3] 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.

[4] S. Bilgaiyan, S. Mishra, and M. Das, “A Review of Software Cost Estimation in Agile Software Development Using Soft Computing

Techniques," 2016 2nd Int. Conf. Comput. Intell. Networks, pp. 112–117, 2016.

[5] A. Sharma and R. Ranjan, "Software Effort Estimation using Neuro-Fuzzy Inference System : Past and Present," Int. J. Recent Innov. Trends Comput. Commun., vol. 5, no. 8, pp. 78–83, 2017.

[6] 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, pp. 1–15, 2017.

[7] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," IEEE Trans. Softw. Eng., vol. 33, no. 1, pp. 33–53, 2007.

[8] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobylinski, "An effective approach for software project effort and duration estimation with machine learning algorithms," J. Syst. Softw., vol. 137, pp. 184–196, 2018.

[9] S. M. Satapathy, A. Panda, and S. K. Rath, "Story Point Approach based Agile Software Effort Estimation using Various SVR Kernel Methods," 26th Int. Conf. Softw. Eng. Knowl. Eng., pp. 304–307, 2014.

[10] E. Mendes, I. Watson, C. Triggs, N. Mosley, and S. Counsell, "A comparative study of cost estimation models for web hypermedia applications," Empir. Softw. Eng., vol. 8, no. 2, pp. 163–196, 2003.

[11] M. Usman, E. Mendes, and J. Börstler, "Effort estimation in Agile software development: A survey on the state of the practice," ACM Int. Conf. Proceeding Ser., pp. 1–10, 2015.

[12] M. Azzeh, A. B. Nassif, and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points," IET Softw., vol. 12, no. 1, pp. 19–29, 2018.

[13] Q. M. Yousef and Y. A. Alshaer, "Dragonfly Estimator : A Hybrid Software Projects' Efforts Estimation Model using Artificial Neural Network and Dragonfly Algorithm," Int. J. Comput. Sci. Networ Secur., vol. 17, no. 9, pp. 108–120, 2017.

[14] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, "Negative results for software effort estimation," Empir. Softw. Eng., vol. 22, no. 5, pp. 2658–2683, 2017.

[15] J. M. Alostad, L. R. A. Abdullah, and L. S. Aali, "A Fuzzy based Model for Effort Estimation in Scrum Projects," IJACSA) Int. J. Adv. Comput. Sci. Appl., vol. 8, no. 9, pp. 270–277, 2017.

[16] A. Idri, M. Hosni, and A. Abran, "Systematic Literature Review of Ensemble Effort Estimation," J. Syst. Softw., vol. 1, pp. 1–35, 2016.

[17] S. Bilgaiyan, S. Sagnika, S. Mishra, and M. Das, "A systematic review on software cost estimation in Agile Software Development," J. Eng. Sci. Technol. Rev., vol. 10, no. 4, pp. 51–64, 2017.

[18] L. Radlinski, "A survey of bayesian net models for software development effort prediction," Int. J. Softw. Eng. Comput., vol. 2, no. 2, pp. 95–109, 2010.

[19] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," J. Syst. Softw., vol. 127, pp. 109–119, 2017.

[20] O. D. M. Salmanoglu, T. Hacaloglu, "Effort Estimation for Agile Software Development : Comparative Case Studies Using COSMIC Functional Size Measurement and Story Points," ACM Mensura, pp. 1–9, 2017.

[21] M. Padmaja and D. Haritha, "Software Effort Estimation using Meta Heuristic Algorithm," Int. J. Adv. Res. Comput. Sci., vol. 8, no. 5, pp. 196–201, 2017.

[22] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, "A genetic algorithm based framework for software effort prediction," J. Softw. Eng. Res. Dev., vol. 5, no. 1, pp. 1–33, 2017.

[23] R. de A. Araújo, A. L. I. Oliveira, and S. Meira, "A class of hybrid multilayer perceptrons for software development effort estimation problems," Expert Syst. Appl., vol. 90, pp. 1–12, 2017.

[24] V. S. Dave and K. Dutta, "Neural network based models for software effort estimation: A review," Artif. Intell. Rev., vol. 42, no. 2, pp. 295–307, 2014.

[25] Tung Khuat and Hanh Le, "An Effort Estimation Approach for Agile Software Development using Fireworks Algorithm Optimized Neural Network.," Int. J. Comput. Sci. Inf. Secur., vol. 14, no. 7, pp. 122–130, 2018.

[26] M. Adnan and M. Afzal, "Ontology based Multiagent Effort Estimation System for Scrum Agile Method," IEEE Access, pp. 25993–26005, 2017.

[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," Procedia Comput. Sci., vol. 57, pp. 772–781, 2015.

[28] S. M. Satapathy and S. K. Rath, "Empirical assessment of machine learning models for agile software development effort estimation using story points," Innov. Syst. Softw. Eng., vol. 13, no. 2–3, pp. 191–200, 2017.

[29] T. T. Khuat and M. H. Le, "A Novel Hybrid ABC-PSO Algorithm for Effort Estimation of Software Projects Using Agile Methodologies," J. Intell. Syst., vol. 27, no. 3, pp. 489–506, 2018.

[30] S. Porru, A. Murgia, S. Demeyer, M. Marchesi, and R. Tonelli, "Estimating Story Points from Issue Reports," Proc. 12th Int. Conf. Predict. Model. Data Anal. Softw. Eng. - PROMISE 2016, pp. 1–10, 2016.

[31] K. Moharreri, A. V. Sapre, J. Ramanathan, and R. Ramnath, "Cost-Effective Supervised Learning Models for Software Effort Estimation in Agile Environments," 2016 IEEE 40th Annu. Comput. Softw. Appl. Conf., pp. 135–140, 2016.

[32] 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. Softw. Eng., vol. 14, no. 8, pp. 1–12, 2016.

[33] I. Manga and N. V. Blamah, "A particle Swarm Optimization-based Framework for Agile Software Effort Estimation," Int. J. Eng. Sci., vol. 3, no. 6, pp. 30–36, 2014.

[34] S. K. Tipu and S. Zia, "An Effort Estimation Model for Agile Software Development," Adv. Comput. Sci. its Appl., vol. 2, no. 1, pp. 314–324, 2012.

[35] R. Popli and N. Chauhan, "Agile estimation using people and project related factors," 2014 Int. Conf. Comput. Sustain. Glob. Dev. INDIACom 2014, pp. 564–569, 2014

[36] M. Arora, S. Chopra, and P. Gupta, "Estimation of regression test effort in agile projects," Far East J. Electron. Commun., vol. 3, no. II, pp. 741–753, 2016.

[37] T. Chen and C. Guestrin, ''XGBoost: A scalable tree boosting system,''in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, San Francisco, CA, USA, 2016, pp. 785–794.

[38] Y. Freund and R. Schapire, "A Decision-Theoretic Generalization of Online Learning and an Application to Boosting", Journal of Computer and System Sciences, vol. 55, pp. 119-139, 1997.

[39] Liudmila Prokhorenkova et al., "CatBoost: unbiased boosting with categorical features", Advances in neural information processing systems, 2018.

[40] Sandhu AK, Batth RS., "Software reuse analytics using integrated random forest and gradient boosting machine learning algorithm", Software: Practice and Experience., 2020, 1-13., Wiley. https://doi.org/10.1002/spe.2921

[41] A. K. Sandhu and R. S. Batth, "Integration of Artificial Intelligence into software reuse: An overview of Software Intelligence," 2021 2nd International Conference on Computation, Automation and Knowledge Management (ICCAKM), Dubai, United Arab Emirates, 2021, pp. 357-362, doi: 10.1109/ICCAKM50778.2021.9357738