Software Effort Estimation using Machine Learning Technique

Mizanur Rahman[1]
Faculty of Computing
Universiti Malaysia Pahang
26600, Pekan, Pahang, Malaysia

Partha Protim Roy[2]
Research Assistant
United International University
Dhaka, Bangladesh

Mohammad Ali[3]
Developer
Edusoft Consultants Ltd
Dhaka, Bangladesh

Teresa Gonçalves[4]
Associate Professor
Department of Computer Science
University of Évora, Portugal

Hasan Sarwar[5]
Professor
Department of Computer Science and Engineering
United International University
Satarkul, Badda, Dhaka, Bangladesh

***Abstract*—Software engineering effort estimation plays a significant role in managing project cost, quality, and time and creating software. Researchers have been paying close attention to software estimation during the past few decades, and a great amount of work has been done utilizing a variety of machine-learning techniques and algorithms. In order to better effectively evaluate predictions, this study recommends various machine learning algorithms for estimating, including k-nearest neighbor regression, support vector regression, and decision trees. These methods are now used by the software development industry for software estimating with the goal of overcoming the limitations of parametric and conventional estimation techniques and advancing projects. Our dataset, which was created by a software company called Edusoft Consulted LTD, was used to assess the effectiveness of the established method. The three commonly used performance evaluation measures, mean absolute error (MAE), mean squared error (MSE), and R square error, represent the base for these. Comparative experimental results demonstrate that decision trees perform better at predicting effort than other techniques.**

***Keywords*—Software effort estimation; K-nearest neighbor regression; machine learning; decision tree; support vector regression**

List Of Abbreviation

ML	Machine Learning	KNN	K-nearest neighbour
SVR	Support Vector Regression	DT	Decision Tree
ANN	Artificial Neural Network	RF	Random Forest
CBR	Case Base Reasoning	LR	Linear regression
MSE	Mean Square Error	MAE	Mean Absolute Error
RMSE	Root Mean Square Error	SEE	Software Effort estimation
COCOMO	Constructive Cost Model	LASSO	Least Absolute Shrinkage and Selection

I. Introduction

Software teams and businesses have long had substantial difficulty with software effort estimation, which should be taken into account at the outset of a software project[1]. For software project success and risk reduction, accurate software work estimation is essential. The practice of estimating the amount of time and money it will take to produce a software process or product is known as effort estimation. In order to effectively budget, plan, control, and manage the project, proper estimating may require accurately projecting software costs. In order to allocate resources effectively and prepare for the development of software, precise cost and time estimation are crucial. Project planning determines whether a project will succeed or fail because during this stage, the time and financial restrictions necessary to finish the project successfully are estimated [2]. Since the 1940s, when the computer industry first started to take off, the idea of software effort estimation has gained popularity. Research in this area is still ongoing [3].

Numerous estimation methods are categorized into three broad groups in the literature on software effort estimation: algorithmic, non-algorithmic, and machine learning [4]. Algorithmic approaches utilize statistical and mathematical concepts for software project estimation. COCOMO-II, Putnam Software Life cycle Management (SLIM), SEER-SEM, and True Planning are some examples of estimate techniques. The fundamental input to these models is the size of the software being evaluated, It is usually quantified in a metric like function points, source lines of code, or use case points. Models that don't rely on algorithms rely on subjective evaluations and interpretations of data. Data from prior projects are analyzed by these models. Expert judgment, planning poker, wideband Delphi, and the work breakdown structure (WBS) are all examples of non-algorithmic approaches. Machine learning is an alternative to algorithmic model building. Artificial neural networks (ANN), case-based reasoning (CBR), support vector regression (SVR), decision trees (DT), fuzzy models, Bayesian networks, and genetic algorithms are all examples of machine learning estimation approaches [5].

Several datasets have been proposed that are used to measure the effort estimation of software development. These datasets were proposed quite a long time ago. At present the effort engagement in software development has changed a lot. One perspective is that in this post-COVID era, many companies have moved to a hybrid development approach. In hybrid mode, developers are not bound to come to the office physically every day. They can work from home. The office visit is expected only if there is a need for so. Another perspective is that an extreme level of change requirements

is needed due to the dynamic nature of clients' businesses, updates to new technology, and extreme competition in the business. Considering these two scenarios, In this study, we have developed a new dataset and we have successfully shown that this new dataset is capable of providing future insight into software efforts. We have used three machine learning models, namely, SVM, KNN, and DT. We have shown that both these models are able to predict the software effort for the future. We claim that this is the first instance of using such a dataset in the prediction of effort in software development. We have also shown that these new parameters are perfectly able to predict the future.

Here is how the article progresses for the remainder. In section II, we review the many methods of effort estimation that have been published by scholars. The section III explains how we conducted our research. In section IV, we detail the performance evaluation of the several competing ML approaches we employed based on the trial outcomes. Thereafter, the essay finishes with Section VI.

II. Related Work

The majority of software effort estimates are made using a variety of approaches that have been put out by scholars over the past few decades. The first three methods to calculate the effort required to create widely used software were the function point-based model, the constructive cost model (COCOMO), and the Putnam program life cycle model (SLIM), and they each provided a specific formula for calculating the effort required from historical data[6][7]. Machine learning (ML) algorithms have been widely utilized to estimate the software development effort [8]. These ML Algorithms allow professionals to devote more time to other client-pleasing aspects of software systems and less time to analyzing new projects. Many scholars have utilized machine learning to estimate effort over the past ten years.

A systematic study to investigate the use of ML models in predicting Software Development Life Cycle (SDLC) effort reports CBR – Case Based Reasoning, Neural Network, and Decision Tree as the most frequently used algorithms [9]. In another report, the datasets that were considered are Albrecht, China, Desharnais, Kemerer, Kitchenham, Maxwell, and Cocomo81. Here various stacking models were used. These are stacking using generalized linear models (S-GLM), stacking using S-DT, stacking using SVM, and stacking using RF [10]. A survey of 35 selected studies on effort estimation accuracy implemented on both public and non-public domain datasets suggests ensemble effort estimation as a better technique than solo techniques [5]. The major goal of this research [11] is to empirically compare the performance of several Machine Learning (ML) algorithms in order to identify a performance model for evaluating the software effort. Seven datasets have been used for Effort Estimation, and various ML approaches have been applied. The LASSO approach with the China dataset gave the best performance when compared to the other algorithms, according to the results and trials with several ML algorithms for software effort estimation.

To determine the precise software effort, Abdelali et al. [12] constructed an RF model and experimentally adjusted the effectiveness by altering the important parameters. Specifically, ISBSG, Tukutuku, and COCOMO datasets were employed. The 30% hold-out validation approach was used to manage the evaluation. Three performance indicators, such as the MdMRE, MMRE, and Pred (0.25), are used to assess and identify the well-performed technique. When the generated RF model was compared to the traditional regression tree, it was clear that the upgraded RF model outperformed it. Nassif et al. [13] experiment with fuzzy models to estimate software effort. In order to compare and implement three fuzzy logic models, namely Sugeno, Mamdani with constant output, and Sugeno with linear output, regression analysis was done. These models were used to estimate the program effort. The ISBSG dataset was used for model training and testing, and the Scott-Knott, mean inverted balance relative error, and other performance measures were used to assess the models' effectiveness. The Sugeno fuzzy model with linear output performed better when compared to other fuzzy models created to help in regression and analysis.

The soft computing techniques of linear regression (LR), SVR, ANN, RF, DT, and bagging methodology were used by Sharma and Vijayvargiya [14] to predict the time and resources required for software projects using the benchmark datasets. It was decided that the results from the RF and choice tree were the most helpful. The cost-benefit analysis led to enhanced cognitive performance. This strategy, however, was not evaluated using a deep learning classifier.

A gradient boosting regressor model was suggested by [15]. The stochastic gradient descent, KNN, DT, bagging, RF, Adaboost, and gradient-boosting neighbors are all compared to the model. MSE, root mean square error (RMSE), and R2 were used to evaluate the model by authors. They displayed the outcomes using the China and Cocomo81 datasets.

The goal of this [16] work is to give an approach for accurately estimating the time and resources required to complete a software development project using only a subset of that organization's past project data. Two techniques, the correlation matrix, and the decision tree were utilized to determine the optimal prediction parameters. Each test's results were double-checked by using two different methods. The outcomes of the two analyses were identical, and the same three parameters were chosen for prediction. For certain variables, multiple prediction models were constructed and trained. According to the findings of the tests, Evolutionary SVM is the most accurate predictor.

The AdaBoost ensemble learning method and RF are used in this [17] study, while the Bayesian optimization method is used to calculate the model's hyperparameters. The SEE model was built using the PROMISE repository and the ISBSG dataset. Under 3-fold cross-validation, the created model was thoroughly compared with four machine learning approaches. The RF method based on AdaBoost ensemble learning and Bayesian optimization clearly outperforms this approach. Furthermore, the AdaBoost-based model assigns a feature relevance rating, making it a viable tool for predicting software effort.

III. Methodology

There has been extensive study into the use of machine learning (ML) based prediction methods for software devel-

opment effort estimation to improve predictions. The goal of this machine learning method is to minimize the loss function while simultaneously optimizing the support vector boundaries by transferring non-linear separable patterns in the input into higher feature space. Fig. 1 shows the methodology of software effort estimation. Three common machine-learning techniques are described below.

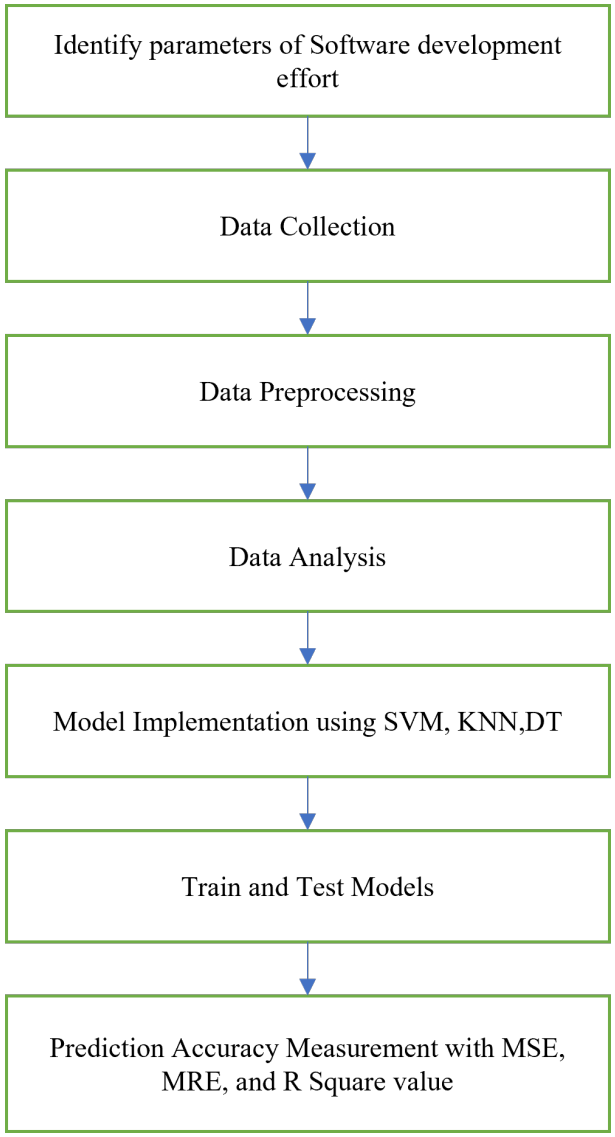

Fig. 1. Methodology of SEE.

A. Support Vector Regression (SVR)

A support vector machine (SVM) handles classification and regression issues. In machine learning, a classifier is a model built to make inferences about a class from additional features. The term "classification" refers to the act of labeling an unlabeled record with a unique value [18]. An SVM variant is an SVR. The regression issues are transformed into classification issues. A linear decision surface (known as a hyperplane) divides two segments of vectors in the SVM training process. The margin between the LDS and the vectors that are nearest to it is maximum in this separation. They are known as support vectors. The input vectors are nonlinearly mapped into a high-dimensional space when the LDS is supposed to be non-linear. In this feature space, the LDS is built using characteristics that provide maximum margin and, thus, a low generalization error for the machine. This LDS has the best generalization of all speculative hyperplanes. Using this ideal hyperplane, classification or regression models' predictive ability is improved.

$$f(\mathbf{x}) = \boldsymbol{\omega}^T \boldsymbol{\phi}(\mathbf{x}) + b \tag{1}$$

where $\mathbf{x}$ is the input vector, $\boldsymbol{\phi}(\mathbf{x})$ is the feature mapping function, $\boldsymbol{\omega}$ is the weight vector, and b is the bias term.

B. K-Nearest Neighbor Regression (KNN)

KNN, one of the most straightforward estimation techniques, was chosen for this study because of its perceived resemblance to human-based expert opinion[19]. Technically, KNN does not train a model[20] but rather uses Euclidean distance [21] to calculate distances between locations. The algorithm makes a prediction about the class to which a point belongs by gathering the nearest samples. Regression involves taking the average of the closest samples to a location to determine its value. The effort of the target project is then estimated by averaging the efforts of the k projects that are the closest analogs.

$$\text{Euclidean} = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2} \tag{2}$$

Where k is the user-defined constant, i is the number of instances x and y are the vectors of each instance

C. Decision Tree (DT)

To obtain insightful information that will help it achieve its objectives, data mining uses DT. DT is an intelligent model that looks like a binary tree with the root at the top that has been turned on its side. The decision tree model is used to turn the data into a tree structure to help with the machine learning challenge. DT serves as an example of forecasting a dependent variable using a set of predictor variables. The decision-making process in this model is analogous to other models, which facilitates understanding. Because a picture is worth a thousand words, this approach makes it simple for anyone to comprehend the essence of a complex problem by simply looking at its schematic. The method used by DT is comparable to how people make decisions. DT does have certain disadvantages, though. Contrasted with other machine learning models, the accuracy of the dataset predictions is lower. Because DTs provide a collection of if-then-else rules, they are easier to grasp and analyze when compared to other machine learning techniques like neural networks and Bayesian networks [10].

$$f(\mathbf{x}) = \sum_{j=1}^{J} c_j I\left(\mathbf{x} \in R_j\right) \tag{3}$$

where $\mathbf{x}$ is the input vector, $f(\mathbf{x})$ is the predicted output label or value, J is the number of leaf nodes in the tree, R_j

TABLE I. SETTING PARAMETERS FOR ALL TECHNIQUES TAKEN INTO CONSIDERATION

Used method	Values for the parameters
SVR	kernel= 'rbf', degree=3, gamma='scale', cache_size=200
KNN	N_neighbors=5, weights='uniform', leaf_size=30,metric='minkowski'
DT	Criterion='gini' splitter='best', min_sample_split=2, min_samples_leaf=1

TABLE II. PROPERTIES OF DIFFERENT DATASETS

Dataset name	Source	No of attributes	Output attributes
Albrecht	PROMISE	8	Person-months
China	PROMISE	16	Person-hours
Desharnais	GITHUB	12	Person-hours
Kemerer	GITHUB	7	Person-months
Maxwell	PROMISE	27	Person-hours
Kitchenham	GITHUB	9	Person-hours
Cocomo81	GITHUB	17	Person-months
UCP	GITHUB	23	Person-months
ISBSG10	GITHUB	10	Person-months
Our dataset	GITHUB	7	Person-hours

is the region of the input space that is assigned to the jth leaf node, c_j is the output value or class label assigned to the jth leaf node, and $I(\cdot)$ is the indicator function that returns 1 if the input condition is true and 0 otherwise

IV. EXPERIMENTAL DESIGN

A. Dataset

In this research, we use a real-world dataset compiled by Edusoft Consultant Ltd. to conduct an empirical evaluation of the presented models for estimating the time and effort required for software development[22]. It's worth noting that this dataset includes quite a variety of characteristics, including task history ID, project ID, Client ID, task types, task priority, task overall state, total working time in hours, etc. our dataset consist of 2000 real-time data samples.

For estimating effort, a number of databases are utilized, including China, Kemerer, Cocomo81[23], Albrecht[24], Maxwell[25], Desharnais, and Kitchenham, Nasa93, ISBSF10. Along with our dataset, Table II shows repository information for other datasets, such as the number of characteristics, source, and an output unit for each dataset[26]. We won't consider all datasets in this research for performance evaluation, but we will evaluate the performance of our dataset.

B. Dataset Pre-Processing

The data preprocessing methodology is an efficient method for estimating the amount of work that will be required [27], and it is an essential step in the process of enhancing the performance of machine learning [28]. The first stage is to eliminate irrelevant features from the dataset as stated in [29]. If unnecessary features are taken out of machine learning algorithms, they will function better. Categorical data undergoes further processing to become numerical. Each category is given its own numeric code in ordinal coding, which has the benefit of not expanding the problem space by displaying each category as a separate input [30].

After the data collection, the dataset has been pre-processed to eliminate inconsistencies, duplicates in data, or missing values, which can otherwise negatively affect a model's accuracy. The researchers have used different methods to achieve the qualitative dataset [31]. The collected dataset also had some missing values, and some missing data were eliminated by inspecting the dataset, and features were chosen according to the degree of correlation between each dataset's values. Furthermore, some of the missing information has been filled in using the moving median method with a window of the length of 10.

C. Performance Evaluation

Research has demonstrated the capabilities of software development effort estimation models using a wide range of performance indicators. Different aspects of performance are being measured and/or represented by these different metrics. Performance evaluation measures are crucial to the accuracy of performance measurements [32]. However, no single metric has gained widespread acceptance for use across all software development effort estimation model comparisons without some form of criticism. we employ more generic evaluation metrics such as Mean Absolute Error (MAE) (2), Mean Square Error (MSE) (1), and R Squared (3).

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{4}$$

where y_i is the true value of the ith data point, $\hat{y}_i$ is the predicted value for the ith data point, n is the total number of data points, and $(\cdot)^2$ denotes squaring. The MSE measures the average of the squared differences between the predicted values and the true values, which gives greater weight to larger differences than smaller ones. This means that the MSE is more sensitive to large errors than to small errors. A lower MSE indicates that the model is better at predicting the true values, while a higher MSE indicates that the model is less accurate.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5}$$

where y_i is the true value of the ith data point, $\hat{y}_i$ is the predicted value for the ith data point, n is the total number of data points, and $|\cdot|$ denotes the absolute value. The MAE is a useful metric for evaluating regression models, as it gives an idea of how far off the predictions are on average. A lower MAE indicates that the model is better at predicting the true values, while a higher MAE indicates that the model is less accurate

$$R^2 = 1 - \frac{SS_{\text{Regression}}}{Ss_{\text{Total}}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{6}$$

where y_i is the true value of the ith data point, $\hat{y}_i$ is the predicted value for the ith data point, $\bar{y}$ is the mean of the true values, and n is the total number of data points. A higher R^2 value indicates that the model explains more of the variance in the dependent variable. However, a high R^2 value does not necessarily indicate that the model is a good

fit for the data, as it may be overfitting the data or failing to capture important relationships between the independent and dependent variables.

D. Parameter Setting

In Table I, a comprehensive list of all parameter values was examined for each SDEE technique in this study.

V. Experimental Results and Discussion

In this study, machine learning algorithms have been conducted with our proposed dataset. Firstly, the preprocessed dataset has been separated into a training and a testing set. The training set has been used to train the models, and the testing set has been used to test the performance of the models with our dataset. In this experiment, we have used a total of 80% dataset for training, and the rest 20% dataset for testing the models. The parameter values that have been examined for each SDEE method in this study are all listed in Table I.

Table III shows the performance metrics (MAE, MSE, and R-Squared) for ML algorithms in predicting our dataset's data. Table III illustrates that the Decision Tree Algorithm yields more accurate outcomes with smaller MAE and MSE values. Besides, based on the MSE value second best result was given by the KNN algorithm. Finally, the third position is held by SVR based on the performance evaluation matrix. Fig. 2 shows the actual vs predicted result using SVM, Fig. 3 shows the actual vs predicted result using DT, and finally, Fig. 4 show the predicted results plot with respect to the actual data using KNN. As we proposed a new dataset in this work so we did not compare it with other datasets, we will compare the result of our dataset with other existing datasets using various ML algorithms in our next paper.

TABLE III. Comparison of all models' effectiveness in terms of MSE, MAE and R Square

Model	MSE	MAE	R Square
Decision Tree	20.54043	1.649842	-0.008089
SVM	148.4454	4.63788	-0.059185
KNN	135.3942	5.42981	0.033978

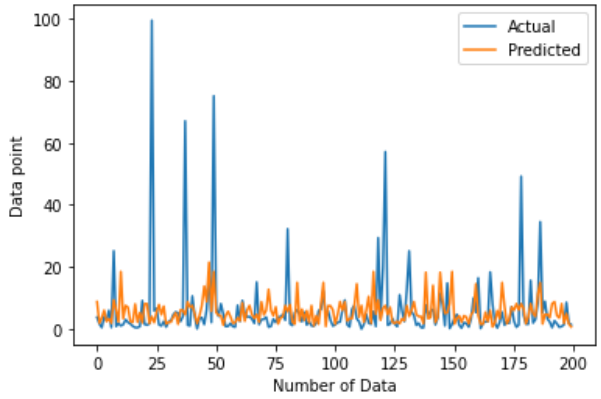

Fig. 2. SVM models' actual effort and predicted effort.

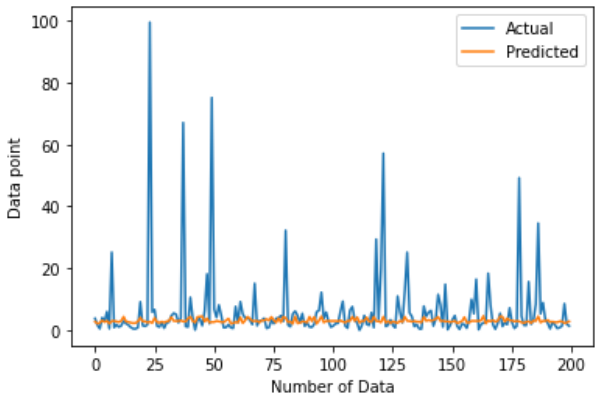

Fig. 3. DT models' actual effort and predicted effort.

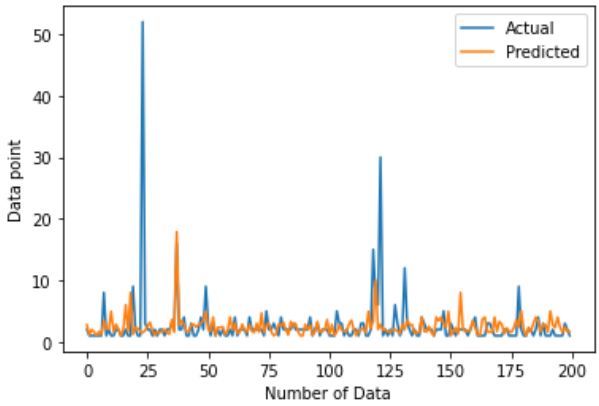

Fig. 4. KNN models' actual effort and predicted effort.

VI. Conclusion

At the beginning of software development, the project manager must take care of an extremely important step called effort estimation. To estimate effort, we used a real dataset created by Edusoft Consultant Ltd. In this study, we implemented the decision tree regressor, random forest, and KNN machine learning models. Comparative experimental results demonstrate that decision trees perform better at predicting effort than other techniques. The MAE, MSE, and R-Squared values are evaluated as evaluation metrics. In the future, we will compare the result of our dataset with other existing datasets using various ML algorithms.

Acknowledgment

The work reported in this paper is funded by the Institute for Advanced Research (IAR), United International University (UIU), Bangladesh titled: Implementation of 3D model to assess the performance improvement of a software company UIU/IAR/02/2019-20/SE/06.

References

[1] Z. R. Mohsin, "Comparative study for software effort estimation by soft computing models," *Journal of Education for Pure Science-University of Thi-Qar*, vol. 11, no. 2, pp. 108–120, 2021.

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

[3] A. Zaid, M. H. Selamat, A. Ghani, R. Atan, and K. Wei, "Issues in software cost estimation," *International Journal of Computer Science and Network Security*, vol. 8, no. 11, pp. 350–356, 2008.

[4] T. Vera, S. F. Ochoa, and D. Perovich, "Survey of software development effort estimation taxonomies," *Computer Science Department, University of Chile: Santiago, Chile*, 2017.

[5] Y. Mahmood, N. Kama, A. Azmi, A. S. Khan, and M. Ali, "Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation," *Software: Practice and Experience*, vol. 52, no. 1, pp. 39–65, 2022.

[6] A. Najm, A. Zakrani, and A. Marzak, "Systematic review study of decision trees based software development effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 11, no. 7, 2020.

[7] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *Journal of software: evolution and process*, vol. 31, no. 10, p. e2211, 2019.

[8] Y. Garg *et al.*, "Comparative analysis of machine learning techniques in effort estimation," in *2022 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COM-IT-CON)*, vol. 1. IEEE, 2022, pp. 401–405.

[9] S. S. Gautam and V. Singh, "Adaptive discretization using golden section to aid outlier detection for software development effort estimation," *IEEE Access*, vol. 10, pp. 90 369–90 387, 2022.

[10] P. V. AG and V. Varadarajan, "Estimating software development efforts using a random forest-based stacked ensemble approach," *Electronics*, vol. 10, no. 10, p. 1195, 2021.

[11] F. B. Ahmad and L. M. Ibrahim, "Software development effort estimation techniques: A survey," *development*, vol. 2, p. 23.

[12] H. Mustapha, N. Abdelwahed *et al.*, "Investigating the use of random forest in software effort estimation," *Procedia computer science*, vol. 148, pp. 343–352, 2019.

[13] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, "Software development effort estimation using regression fuzzy models," *Computational intelligence and neuroscience*, vol. 2019, 2019.

[14] S. Sharma and S. Vijayvargiya, "Applying soft computing techniques for software project effort estimation modelling," in *Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS 2019*. Springer, 2021, pp. 211–227.

[15] P. Suresh Kumar, H. Behera, J. Nayak, and B. Naik, "A pragmatic ensemble learning approach for effective software effort estimation," *Innovations in Systems and Software Engineering*, vol. 18, no. 2, pp. 283–299, 2022.

[16] T. Mahboob, S. Gull, S. Ehsan, and B. Sikandar, "Predictive approach towards software effort estimation using evolutionary support vector machine," *International journal of advanced computer science and applications*, vol. 8, no. 5, 2017.

[17] R. Marco, S. S. S. Ahmad, and S. Ahmad, "Bayesian hyperparameter optimization and ensemble learning for machine learning models on software effort estimation," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 3, 2022.

[18] O. A. Montesinos López, A. Montesinos López, and J. Crossa, "Support vector machines and support vector regression," in *Multivariate Statistical Machine Learning Methods for Genomic Prediction*. Springer, 2022, pp. 337–378.

[19] V. Resmi and S. Vijayalakshmi, "Analogy-based approaches to improve software project effort estimation accuracy," *Journal of Intelligent Systems*, vol. 29, no. 1, pp. 1468–1479, 2019.

[20] C. Albon, *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*. " O'Reilly Media, Inc.", 2018.

[21] A. C. Faul, *A concise introduction to machine learning*. CRC Press, 2019.

[22] E. C. Ltd, "Software effort estimation," https://github.com/edusoftresearch/SEE_Data, 2023.

[23] B. Boehm, *Software Engineering Economics*. Prentice Hall, 1981.

[24] Albrecht, "Software function, source lines of code, and development effort prediction: A software engineering," https://zenodo.org/record/268467#.ZB5-BHZBxPY, 1983.

[25] K. D. Maxwell, "Maxwell dataset," https://zenodo.org/record/268461#.ZBBPX3ZBzIU, 2002.

[26] J. Sayyad Shirabad and T. Menzies, "The PROMISE Repository of Software Engineering Databases." School of Information Technology and Engineering, University of Ottawa, Canada, 2005. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[27] J. Huang, Y.-F. Li, J. W. Keung, Y. T. Yu, and W. Chan, "An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the isbsg data," in *2017 IEEE International Conference on Software Quality, Reliability and Security (QRS)*. IEEE, 2017, pp. 442–449.

[28] J. Huang, Y.-F. Li, and M. Xie, "An empirical analysis of data preprocessing for machine learning-based software cost estimation," *Information and software Technology*, vol. 67, pp. 108–127, 2015.

[29] E. T. Bekar, P. Nyqvist, and A. Skoogh, "An intelligent approach for data pre-processing and analysis in predictive maintenance with an industrial case study," *Advances in Mechanical Engineering*, vol. 12, no. 5, p. 1687814020919207, 2020.

[30] E. Fitkov-Norris, S. Vahid, and C. Hand, "Evaluating the impact of categorical data encoding and scaling on neural network classification performance: The case of repeat consumption of identical cultural goods," in *Engineering Applications of Neural Networks: 13th International Conference, EANN 2012, London, UK, September 20-23, 2012. Proceedings 13*. Springer, 2012, pp. 343–352.

[31] A. Ahmed, S. Elkatatny, A. Ali, M. Abughaban, and A. Abdulraheem, "Application of artificial intelligence techniques in predicting the lost circulation zones using drilling sensors," *Journal of Sensors*, vol. 2020, 2020.

[32] A. Jadhav, M. Kaur, and F. Akter, "Evolution of software development effort and cost estimation techniques: five decades study using automated text mining approach," *Mathematical Problems in Engineering*, vol. 2022, pp. 1–17, 2022.