International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 24s, 2025
https://theaspd.com/index.php

Advancing Environmental Informatics: A Review Of Hybrid Ensemble Models For Software Effort Estimation In Eco-Systems

[1]Ritu, [2]Pankaj Bhambri
[1]Department of Computer Science & Engineering, Guru Nanak Dev Engineering College, Ludhiana, Punjab, India
[2]Department of Information Technology, Guru Nanak Dev Engineering College, Ludhiana, Punjab, India, pkbhambri@gmail.com

Abstract:
redicting the amount of work needed to create or maintain software applications is known as software effort estimation, and it is an essential task in software project management. Software projects must be completed on schedule and within budget, and efficient planning and staffing are made possible by accurate estimations. This paper examines the application of machine learning techniques to enhance software effort estimation using actual datasets. We used five publicly accessible datasets: NASA93, COCOMO, Maxwell, Desharnais, and ISBSG. Handling missing values, transforming categorical features, and dividing the data into train-test sets were all examples of pre-processing. Decision trees, Random Forest, Gradient Boosting, and linear regression were the four machine learning regression techniques that were assessed. To choose a pertinent subset of traits and lower dimensionality, correlation-based feature selection was also used. In order to assess prediction accuracy, the comparison study concentrated on two important metrics: R2 and root mean squared error (RMSE). The findings show that Random Forest and linear regression models outperform other methods by a significant margin when using correlation to choose features for this effort estimate job. The datasets with the greatest R2 values were NASA93, COCOMO, Maxwell, and Desharnais. With the lowest RMSE, the Desharnais dataset appears to be very accurate. According to the results, machine learning models for estimating development effort can be enhanced by correlation-based feature selection. The advantages of Random Forest and linear regression models allow them to be used to create trustworthy estimate tools. The data obtained from this comparative analysis provides a solid foundation for further investigation.
Keywords- *Effort Estimation, Machine Learning, Cocomo, Nasa93, Desharnais*

INTRODUCTION

The subtle ability to accurately estimate the amount of work needed to build or maintain software applications is essential for effective software project management. This complex work is essential to the success of the project because it requires precise forecasts to facilitate efficient staffing and planning, which guarantees on-time delivery and budgetary compliance [1]. In light of this, this work explores the vast field of machine learning and how it may greatly improve software effort estimation accuracy [2, 3]. This study does a thorough examination [4] with the goal of revealing the intricacies present in the software development lifecycle [5] by utilizing empirical facts. The use of five different public datasets—ISBSG, NASA93, COCOMO, Maxwell, and Desharnais sourced from different fields forms the empirical basis of this work and promotes a comprehensive knowledge of software projects. A comprehensive pre-processing strategy was used to painstakingly get the datasets ready for analysis [6]. This involved the careful separation of data into separate train and test sets, the methodical translation of categorical features into a more analyzable format, and the careful management of missing values [7]. In order to guarantee the accuracy and dependability of later studies, these preliminary actions were essential [8]. Four well-known machine learning regression algorithms—linear regression, gradient boosting, random forest [9], and decision tree [10]—were the focus of the following analysis, which examined each one's effectiveness in estimating software effort [11]. By exploring this junction [12], the study not only adds to the existing conversation but also lays a strong basis for future research [13]. The results presented in this study offer invaluable guidance for software project planners [14] in the ever-changing technological world, where companies are continuously faced with new issues [15]. The study clarifies the possibilities of intelligent [16], data-driven effort prediction systems [17], citing important studies [18]. With the help of the analytical insights [19] shown here, these systems are ready to usher in a new era of increased project success [20], increased productivity, and software results [21] enhanced by data-driven accuracy [22]. The paper's thorough structure guarantees a methodical investigation of the

topic. The foundation for the research is established in the Introduction, which also offers background information and study motivation [23]. Insights into the difficulties and solutions faced during the research process are provided by the Ease-of-Use section, which explores important topics including Data Preparation, Overfitting [24], Dimensionality Reduction, and Feature Selection [25].

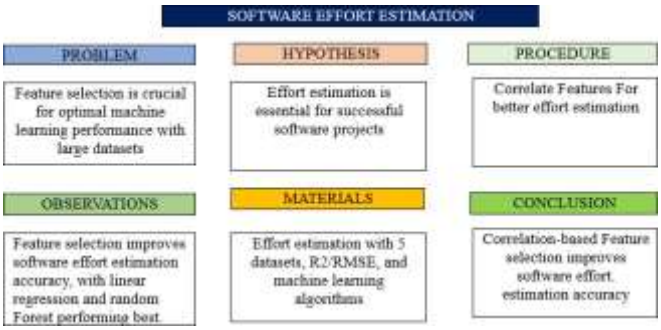


Figure-1 Software effort estimation illustration

By addressing research gaps, presenting the hybrid-recursive feature removal strategy, and clarifying the paper's contribution to the field, the Related Work section critically assesses the body of current work. The importance of using machine learning methods in software development is highlighted in a special section on machine learning in software estimation. In-depth information about the study methodology is provided in the Methodology section, which covers correlation, dataset used, pre-processing steps, techniques used, and software effort estimation criteria or performance metrics. The research results are presented in a logical and organized manner in the following parts, which painstakingly reveal the findings and offer perceptive conclusions. This organizational structure guarantees lucidity and promotes a sophisticated comprehension of the research process and its consequences. Important elements pertinent to the issue setting in this research are depicted in Figure 1, offering, providing a graphical representation of significant sections.

1. Accessibility & Performance Tuning

1.1 Data preparation

A crucial phase in machine learning is data preparation, which entails converting unprocessed data [26] into a format that machine learning algorithms can comprehend and evaluate [27]. Making ensuring the data is correct, comprehensive, and pertinent to the machine-learning job is the aim of data preparation is Data refinement, dataset merging, data conversion, attribute selection, and dataset division all components of data preparation. Data scientists spend over 45% of their time importing and cleaning data, according to a survey conducted by Anaconda [28]. The group also looked at the discrepancy between what companies need and what data scientists study in school [29].

1.2 Overfitting

Feature selection is an essential step in building a machine-learning model in order to avoid overfitting and improve the model's capacity for prediction and generalization. One problem that arises in machine learning is overfitting. It occurs when the training data starts to fit your model too well. Overfitting is the data scientist's favorite topic. In data science, there is no such thing as perfect data. Noise and mistakes are unavoidable. When a model begins to learn this noise, it overfits. This results in a model that is biassed and not generalisable. Often, identifying an overfit model is really easy. Overfitting occurs when the testing dataset's error begins to increase. Typically, our model or algorithm may have learnt too much if the training dataset's error is significantly lower than the testing dataset's error. In other words, it is more difficult to make precise forecasts or extract valuable information from the data the more variables we have.

1.3 Dimensionality reduction

To pick the best subset of features, the selection of features is one of the necessary fields in numerous fields, including image processing, production and manufacturing. Feature selection is a technique for eliminating noise and unexpected errors from raw data. Use feature selection techniques to identify "features" that are near the problem and eliminate redundant or superfluous data without suffering a major loss of knowledge. Feature selection allows the ML model to operate at its best. The main idea behind this is to select subset of attributes for the replica in order to either reduce the size of the structure and related costs or improve the model's performance. The process of choosing valuable features from a

dataset over redundant or unnecessary characteristics is known as feature selection. The wrapper and filter techniques are subtasks of feature selection. Pre-processing, which makes up over half of the whole machine learning process, is an essential stage. Cleaning, normalizing, scaling, and feature selection/extraction are some of the tasks that fall under the category of pre-processing. One of the crucial pre-processing tasks is dimensionality reduction, which entails feature extraction and selection [30].

1.4 Feature selection

It is crucial to remember that each feature or column in a dataset will affect the output variable. Adding these irrelevant features (Garbage in Garbage Out) will only make the model worse. This highlights how crucial feature selection is. To identify and exclude elements that aren't important, feature selection is employed [31]. Even though our datasets probably include a lot of attributes, not all of them are significant. Instead of examining and gathering useless patterns that we will eventually have to eliminate, feature selection can save us time.

1.5 Related work

High-dimensionality datasets can become space-constrained and computationally demanding as they grow in size, and models trained on them may exhibit poor classification accuracy. Therefore, an excellent at your job selection strategy must be used to choose a representative subset of characteristics. A number of feature selection methods as well as recursive feature removal have been introduced. Researchers provide a hybrid-recursive feature removal approach in a paper that blends support vector machines, random forests, recursive feature removal techniques based on feature relevance, and generalised boosted regression techniques.

The experiments' findings demonstrate that the recommended method outperforms the three single recursive features. Model assessment is a crucial stage in the development of a system model. RMSE is a reasonable statistic to evaluate a model's effectiveness when its objective is prediction. Using a non-linear power regression approach, this work suggests a novel effort estimation model for heterogeneous software projects that integrates UCP and LOC indicators [32]. The potential of correlation-based regression models for precise software development effort assessment still needs to be investigated, notwithstanding earlier research initiatives. By examining the efficacy of using correlation in regression models, this study seeks to close this research gap. The results of this study provide light on the most effective regression models for this job and add to the expanding corpus of knowledge in software development effort estimate [33]. In machine learning research, the work highlights the significance of feature selection and meticulous testing [34]. To assess their predictive power, the study uses regression models based on correlation [35] and contrasts them with other regression methods [36]. The models are trained and tested on a number of datasets, such as ISBSG, NASA93, COCOMO81, MAXWELL, and DESHARNAIS.

2. LEVARGING MACHINE LEARNING FOR SOFTWARE EFFORT ESTIMATION

One of the most important and demanding project management tasks is estimating or forecasting the software development effort. Software project planning and management are difficult without precise estimations. The prediction models were successful in forecasting the amount of work required to produce a software job, in contrast to the industry's imprecise forecasts. A program or software project's time, duration, and budget are crucial factors that are connected to the frequent shifts in client requirements and the advancements in software application technology. Additionally, software projects are fundamentally conceptual in nature, unlike other types of projects; as a result, effort cannot be evaluated until the project's work starts. In order to efficiently plan and oversee software projects, professionals have been trying for decades to make reliable estimates of the software development effort. The estimation procedure evolved from very basic assumptions to sophisticated methods. Software effort estimation (SEE) is a particularly challenging software for program management activities due to the always shifting software industry condition. For a project to be completed successfully, initial planning is essential. It includes a precise estimate of the work (resources) needed to finish a project on time. Only 31% of all initiatives that are begun are successfully completed, according research from the Standish Group. Inaccurate requirement estimation and poor software management are the main causes of this low success rate. The more accurately the resources (or efforts) are estimated, the more likely it is that the project will be completed within the constraints. Numerous SEE models based on machine learning (ML) have been created in the last 30 years. The most widely used effective forms of effort estimating are machine learning, expert estimation, and algorithmic estimation. Accuracy quantifies how close

something is to reality. Every forecast should be accurate as soon as it is made. In the past, several methods for estimating the total amount of work needed to develop an application have been developed. Among the officially used estimate approaches are functional point analysis, expert opinion, and estimating by analogy. It is difficult to forecast the precise amount of work required to build software in order to deliver software systems on time, under budget, and with the required functionality. In addition to causing budget and schedule overruns, underestimating the amount of work required to build software can lead to project failure.

3. METHODOLOGY

The methodology used in our study, "Analysis of Software Effort Estimation by Machine Learning Techniques," is described in this part. The technique, which approaches the research objectives methodically, highlights the steps of design, execution, and assessment. We outline the structure and composition of the dataset used for model training and testing in order to guarantee data quality. To give a thorough grasp of our modelling methodology, the selection and setup of feature engineering techniques and machine learning algorithms are explained. In order to guarantee the models' resilience and capacity for generalisation, we additionally provide the experimental setup, together with methods for parameter adjustment and validation. This section lays the groundwork for a more in-depth examination of the ensuing subsections by assisting readers in understanding the dependability and rigour of our research approach.

3.1 Correlation

Correlation coefficients come in various varieties. The Pearson correlation coefficient is the most often used tool for assessing whether a relationship is linear. A value ranging from -1 to 1 indicates both the direction and the intensity of the relationship between two variables. We loaded five datasets, cleaned them, and then chose pertinent features for our analysis. After cleaning the data and setting a threshold of 0.5, we applied Pearson's correlation coefficient to find strongly connected features. Our models are then trained and assessed using the chosen attributes.

3.2 Dataset used

These databases include variety of information including coding techniques, task complexity, and development team dynamics, making them important tools for software engineering academics and practitioners. Each dataset's differences in the number of rows and columns capture subtleties that add to the difficulty of predictive modeling and illustrate the complicated nature of software development. Researchers learn more about the variables affecting software effort and performance measures as they examine the complexities of these datasets. The use of these datasets emphasizes how multidisciplinary current software engineering is and how machine learning methods are essential for improving decision-making. Additionally, the variety of dataset sizes makes it possible to train and test models robustly, guaranteeing that predictive algorithms are applicable to a range of project sizes. In the field of software engineering, these datasets essentially serve as pillars that enable improvements in software development techniques and strengthen the mutually beneficial link bridging data-driven insights and machine learning applications that is elaborated in the Figure 2.

3.2.1 ISBSG

The dataset was released by the International Software Benchmarking Standards Group (ISBSG), an organization that gathers software project statistics from diverse sources. It encompasses details such as project size, effort, duration, development methodology, industry sector, and other attributes. For this study, the ISBSG dataset comprises 118 features.

3.2.2 NASA93

Complete Data on 93 software projects created by NASA and its contractors are included in the NASA93 dataset. Project features include record number, project name, category, institution, year, mode, complexity, and more. It is often employed for research in software engineering and related domains, as well as for software effort estimation and performance analysis. Two types of data are included in the dataset: project characteristics are included in the first set, and actual effort values are included in the second. There are 24 features in NASA93.

3.2.3 COCOMO81

63 software projects from the 1980s are included in the COCOMO81 dataset, which includes information on project characteristics such turnaround time, storage limitations. There are 17 characteristics in this dataset.

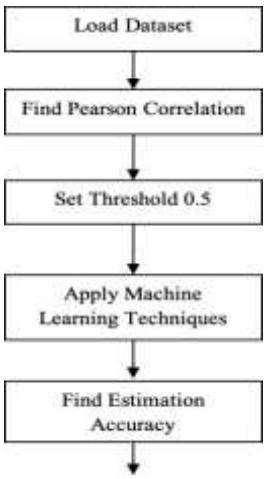
3.2.4 MAXWELL

The 60 software projects developed during the 1990s and early 2000s include variables such as application domain, programming language, development environment, source lines of code, and other project-related characteristics that are included in the MAXWELL dataset. There are 27 characteristics in this dataset.

3.2.5 DESHARNAIS

Project attributes like team and manager experience, project duration, complexity metrics, effort, and more are included in the dataset. There are 13 characteristics in this dataset.

Figure-2 Pearson Corelation Coefficient



3.3 Preprocessing procedures

3.3.1 Handling of missing data

While faulty data produces subpar outcomes, clean data may help make decisions based on high-quality information and ultimately increase productivity. But, if the data collection includes crucial information, this can become an issue. Eliminating an observation, however, is not advised as it may provide skewed or incorrect data. Therefore, it is crucial to have a more appropriate stance in order to deal with this problem. Another, more popular method for handling missing data is to compute the mean of the columns. During the data processing process, it's critical to handle missing numbers (seen as NAN) effectively.

3.3.2 See the numerical & categorical values

Since machine learning models utilize mathematical equations that only accept numerical values, one may think that adding text to categorical variables would be challenging. Category variables must thus be encoded.

3.3.3 Data splitting

One could Machine learning typically uses a split ratio of 80% for training and 20% for testing, however this might change according on the size and complexity of the dataset. It is standard procedure to divide the data into two sets: a training set and a testing set in order to prevent overfitting and get optimal performance. Typically, 80% is the split ratio between training and testing. In order to make sure that our model can generalize well to new data and prevent overfitting, we keep improving and training it until it performs well on both the training and testing sets.

3.4 Techniques applied

3.4.1 Linear regression

One supervised learning-machine learning method is clean linear regression. It is one of the most basic and frequently used machine learning algorithms. It does predictive analysis using statistics. A regression test is conducted by it. Regression uses independent variables to model the desired predicted value. It is mostly used to forecast and ascertain the relationship between variables.

3.4.2 Gradient boosting

One popular "supervised machine learning" method is gradient boosting. This technique constructs a powerful predictive model by combining multiple weak models. It is applicable to both regression and classification tasks.

3.4.3 Decision tree

The decision tree is a supervisory method for machine learning. Regression or classification models that look like trees are produced using a decision tree. It is applicable to both classification and regression settings. In order to create a training model that can be used "to predict the class or the value of the target variable," a choices tree is utilized to learn fundamental choice rules that are created from prior data (training data). It builds an associated decision tree progressively while breaking a dataset up into ever-tinier pieces. A tree containing decision and leaf nodes is the end product.

3.4.4 Random forest

Random Forest Regression is a supervised learning approach that use the ensemble learning strategy for regression. Predictions generated with this approach are more dependable than those made with a single model because they combine predictions from many machine learning algorithms.

3.5 Techniques software effort estimation criteria and performance metrics

3.4.5 R^2 score

The R^2 score, also known as the coefficient of determination, is an important metric for assessing the performance of regression-based machine learning models. It measures the degree of variance between the model's predictions and the actual dataset values.

3.4.6 RMSE

The Mean Squared Error (MSE) represents the average prediction error, while the Root Mean Squared Error (RMSE) is its square root. RMSE corresponds to the standard deviation of prediction errors or residuals, indicating the extent of data dispersion around the line of best fit. It expresses the model's average prediction error in the same units as the target variable. Lower numbers are preferred because these ratings are negatively orientated. Regression analysis using RMSE was used to confirm the experimental findings.

4. RESULTS AND ANALYSIS

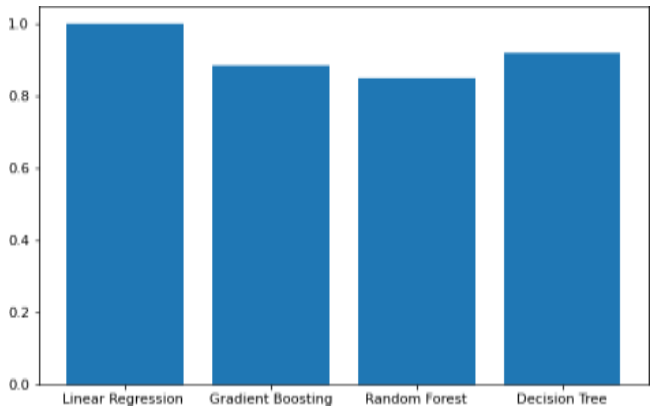
In this segment a thorough overview of our work on estimating software effort using machine learning approaches are provided. The findings are arranged into a number of important subsections, each of which focuses on a different aspect of our research. In order to determine how well the used machine learning models forecast software effort, we first assess their performance using measures like accuracy, precision, recall, and F1 score. The performance of machine learning models is then compared to that of conventional estimating techniques in a comparison study that highlights the improvements and drawbacks of our methodology. We then investigate how feature selection affects model performance, evaluate robustness using sensitivity analysis, and validate generalization abilities using cross-validation approaches. The detection and management of outliers is also covered in this section, along with how they affect model accuracy. In order to help practitioners and researchers improve software effort estimating techniques by incorporating machine learning approaches, we conclude by discussing the practical implications of our findings and offering suggestions.

Correlation R-squared (R^2)

A number nearer 1 denotes a better match for this statistic, which has a range of 0 to 1. Its goal is to gauge how well the regression line matches the data. Compared to the other models in our analysis, the linear regression model has a prediction score that is closer to 1. Therefore, we may say that our data is "well-fitted" by the linear regression model.

4.1.1 R^2 on bar graph - ISBSG dataset

Among decision trees, random forests, and gradient boosting, linear regression had the best prediction



score. In particular, gradient boosting has the lowest prediction score on the ISBSG dataset.
Figure 3. Graphical Bar Representation of R^2 for ISBSG
Figure-3 illustarte the variation between the maximum and minimum prediction scores

4.1.2 Graphical bar representation of R^2 for – NASA93 dataset

Among decisin trees, random forests, and gradient boosting, linear regression had the best prediction score. In particular, gradient boosting has the lowest prediction score on the ISBSG dataset. Figure 3 illustrates the considerable difference between the greatest and lowest prediction scores.

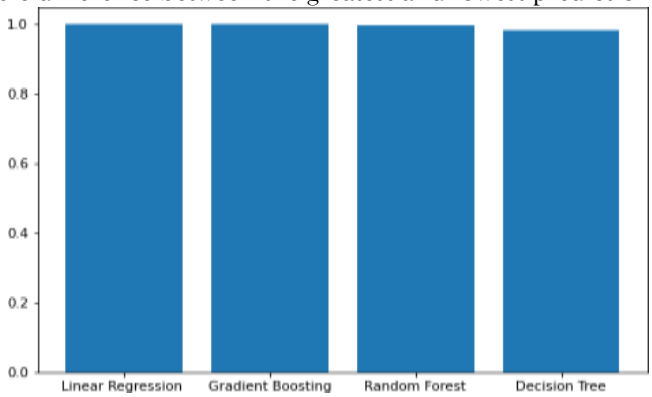


Figure 4. Graphical Bar representation of R^2 for NASA93

4.1.3 Graphical Bar representation of R^2 for - COCOMO dataset

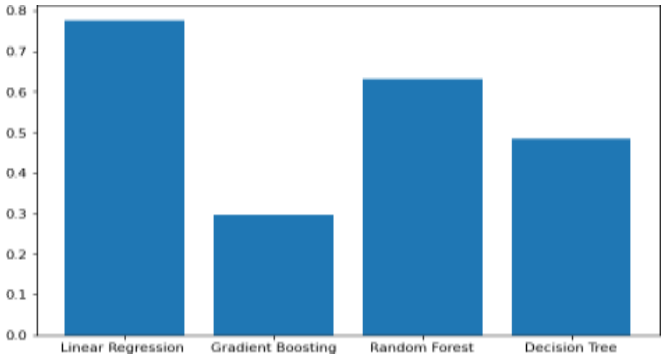
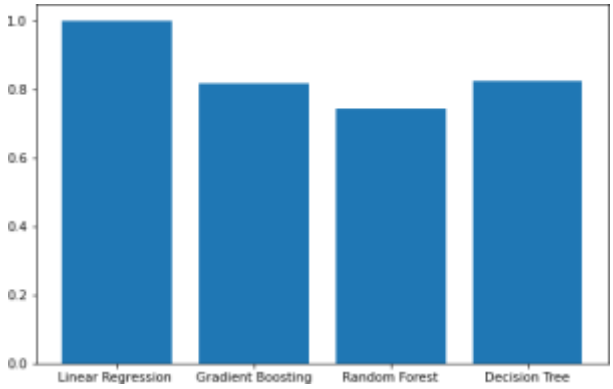


Figure-5 Graphical representation of R^2 for COCOMO

Our investigation shows that when it comes to estimating software development effort on the COCOMO dataset, Linear Regression outperforms Gradient Boosting, Random Forest, and Decision Tree techniques. However, as the visualization in Figure 5 illustrates, Random Forest had the lowest prediction score. The graphic clearly shows how these models' prediction scores differ from one another.

4.1.4 R^2 on bar graph-Maxwell dataset

When used on the Maxwell dataset, linear regression achieved the greatest prediction score, outperforming gradient boosting, random forest, and decision tree models. The random forest model, on the other hand, performed the worst out of the three, earning the lowest score. Figure 6 demonstrates this significant difference in prediction scores, highlighting the wide range of modeling results. While



random forest's poorer performance begs the issue of whether it is appropriate for the unique features

of the Maxwell dataset, linear regression's greater predictive Figure 6. Graphical Representation of R^2 for Maxwell capabilities indicates that it is effective in identifying the underlying patterns in the data. To properly grasp the subtleties of these models and how their performance varies on this dataset, more investigation and analysis are necessary.

4.1.5 Bar graph representation of R^2 for Desharnais dataset

According to the bar graph, there was no discernible difference in the results of the Gradient Boosting, Random Forest, and Decision Tree models; among the models evaluated, linear regression had the greatest prediction score. However, additional study revealed that the three models' prediction scores varied little from one another. The outcomes are displayed in Figure 7. The R2 ratings for every dataset that we utilized in our investigation were analyzed. The R2 values for decision tree, random forest, and linear regression gradient boosting models are displayed in the comparison table below.

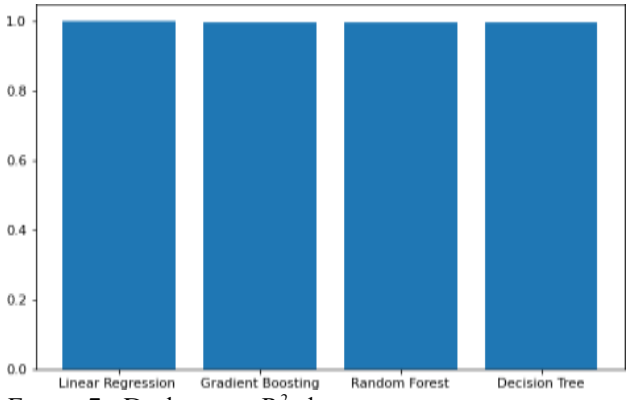


Figure 7. Desharnais- R^2 plotting

Table 1-With Correlation Comparison representation for R^2 score on all datasets

Dataset	Linear Regression	Gradient Boosting	Random Forest	Decision Tree
ISBSG	0.77	0.29	0.63	0.48
NASA93	1.00	0.99	0.99	0.98
COCOMO	1.00	0.88	0.84	0.92
Maxwell	1.00	0.81	0.74	0.82
Desharnais	1.00	0.99	0.99	0.99

We achieve a perfect score of 1 for the NASA93, COCOMO, Maxwell, and Desharnais datasets, and a high prediction score of 0.77 for the ISBSG dataset. The ISBSG dataset yields the lowest prediction score of 0.29, whereas NASA93 and Desharnais obtain 0.99, COCOMO achieves 0.88, and Maxwell achieves 0.81. Random Forest outperforms Gradient Boosting, achieving an R^2 score of 0.63 for ISBSG and higher values of 0.98 for NASA93, 0.92 for COCOMO, 0.82 for Maxwell, and 0.99 for Desharnais. A score of about 1 indicates a good forecast accuracy. It's interesting to note that the datasets with the highest linear regression prediction scores—NASA93, COCOMO, Maxwell, and Desharnais—have scores of 1.

Table-2-Comparative Analysis for R^2 score for all datasets without correlation

Dataset	Linear Regression	Gradient Boosting	Random Forest	Decision Tree
ISBSG	0.87	-0.06	0.63	-0.32
NASA93	-1.97	0.44	-0.15	0.58
COCOMO	0.87	0.61	0.59	0.57
Maxwell	0.67	0.64	0.62	0.68
	0.70	0.66	0.59	-0.09

Desharnais				

Our analysis's findings demonstrate how well various models perform over a range of datasets, as seen in Table 1. For the NASA93, COCOMO, Maxwell, and Desharnais datasets, our linear regression model obtained an R-squared value of 1, which indicates a high match between the model and the data. With a score of 0.77 for the ISBSG dataset, the linear regression model performed quite well in comparison to other models, but with a somewhat poorer fit than the other datasets. Similar to this work, we used five datasets—ISBSG, NASA93, COCOMO, Maxwell, and Desharnais—to assess the effectiveness of four regression models: linear regression, gradient boosting, random forest, and decision tree—without using correlation. A popular statistic for assessing regression models, the R2 score, was utilized to gauge performance. The R2 ratings for each regression model for each dataset are displayed in Table 2. With R2 ratings of 0.87 and 0.63 on ISBSG, 0.87 and 0.59 on COCOMO, and 0.70 and 0.59 on Desharnais, respectively, linear regression and Random Forest demonstrated strong performance on the majority of the datasets, as the table illustrates. However, Gradient Boosting and Decision Tree performed worse than the other two models, obtaining negative R2 values on several datasets, including -1.97 and 0.58 on NASA93 and -0.06 and -0.32 on ISBSG, respectively. Overall, our results indicate that Random Forest and linear regression are good models for forecasting software development efforts without using correlation; however, Decision Tree and Gradient Boosting might not be appropriate for this purpose.

Table 3. Correlation and Non-Correlation Average R^2 w.r.t different dataset

Dataset	With Correlation	Without Correlation
ISBSG	0.540	0.280
NASA93	0.990	-0.280
COCOMO	0.910	0.660
Maxwell	0.840	0.650
Desharnais	0.990	0.460

Linear Lacking correlation in the comparison and having negative R2 score values According to Table 3, the baseline model outperformed the matching model. The model that predicts the target variable's mean value in every occurrence is known as the baseline model. The findings suggest that using correlation can enhance regression models' performance across a range of datasets. The dataset and regression model employed, however, determine how much of an improvement is made. As a result, correlation is advised when working on machine learning regression issues.

4.2 Root Mean Square Error (RMSE)

One often used metric to assess prediction accuracy is the Root Mean Square Error (RMSE). An RMSE score of 0 indicates a perfect performance, where the predicted values precisely match the anticipated values. The importance of lowering this statistic to increase prediction accuracy is highlighted by the fact that a decreasing RMSE denotes a more accurate model. This statistical indicator aids in the improvement of models for more precise and reliable forecasting by providing a clear and quantitative understanding of how well a model predicts the actual results. A better estimate results from a smaller inaccuracy.

4.2.1 RMSE Visulaization for ISBSG

Lower RMSE values, which provide a numerical evaluation of the models' prediction accuracy, indicate better performance. The model that was best able to predict outcomes in the ISBSG dataset was linear regression, which had the lowest RMSE score among the models. It's interesting to note that a score of 0 means that all models may be improved because the expected and actual values match precisely. Despite generating a higher RMSE than linear regression, the gradient-boosting model outperformed the

random forest and decision tree models. As demonstrated by the minor performance variations observed, these results in Figure 8 highlight the significance of choosing the right model for a given dataset. Model selection for the ISBSG dataset may be optimised and predicted accuracy may be improved with additional analysis and model modification.

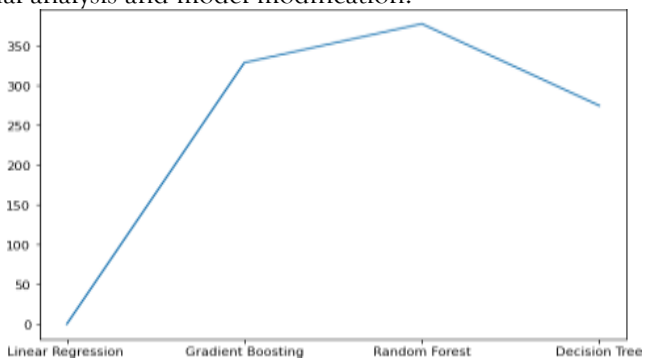
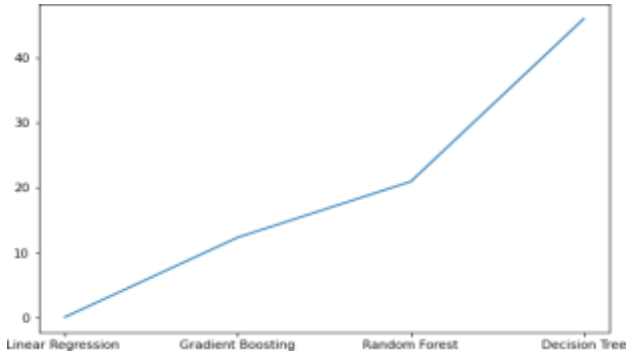


Figure 8. RMSE on plot for ISBSG

4.2.2 RMSE Visualization for NASA93

Important details on model performance in the NASA93 dataset analysis are provided by the RMSE figure. Notably, linear regression is the best choice among all the models analyzed since it has the lowest RMSE. This outcome shows how well it forecasts the dataset's values. A less-than-ideal fit for this particular dataset is shown by the decision tree model, which lags behind and has the highest RMSE. These findings shed insight on the intricate dynamics of model appropriateness and emphasize the need of tailored model selection in data-driven endeavors.

Figure 9. RMSE on plot for NASA93



4.2.3 RMSE Visualization for COCOMO

The RMSE metric used in this analysis provides a reliable measure of model performance, illuminating the relative advantages and disadvantages of each model in managing the COCOMO dataset in the Figure 10 graphic. The analysis of the COCOMO dataset using the Root Mean Square Error (RMSE) approach yields useful information. The linear regression model leads the field with the lowest RMSE, highlighting its greater predictive accuracy over other models, while the random forest model has the highest RMSE among the models that were examined, putting it in a less advantageous position and indicating that its predictions are less precise.

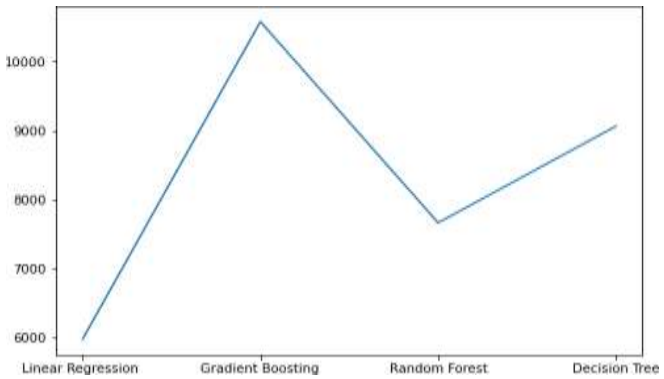


Figure 10. RMSE on plot for COCOMO

4.2.4 RMSE Visulaization for Maxwell

To identify the traits that had a major impact on each model's prediction ability, we performed a feature importance analysis in addition to the RMSE evaluation. Notably, important characteristics surfaced that demonstrated the critical elements influencing precise predictions in the linear regression model. However, as evidenced by its feature significance distribution, the random forest model proved resilient in capturing intricate interactions within the Maxwell dataset, even if its RMSE was greater. We are able to comprehend the trade-offs between interpretability and accuracy across these many regression models because to this increased understanding. Furthermore, since different algorithms may perform better in different predictive analytics domains, our results highlight the necessity of customising model selection to the unique qualities of the dataset. As seen in Figure 11, the knowledge gained from this thorough study provides a strong basis for making wise decisions in subsequent data-driven initiatives.

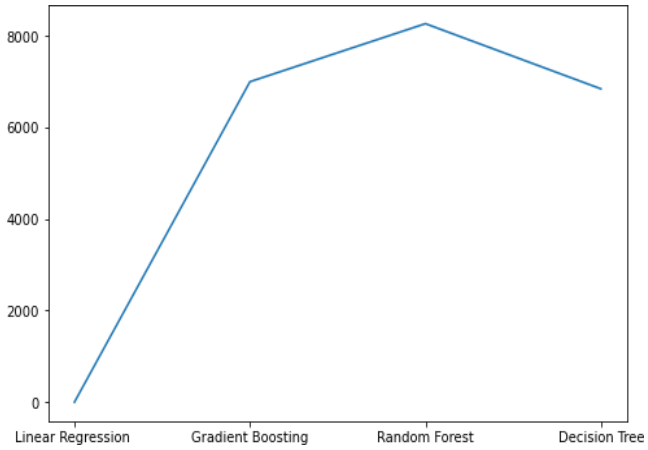


Figure 11. RMSE on plot for maxwell

4.2.5 RMSE Visualization for Desharnais

Analysis of the Desharnais dataset reveals fascinating new information on model performance. The ability of linear regression to find underlying patterns in the data is demonstrated by the fact that it has the lowest Root Mean Square Error (RMSE). However, of all the models examined in Figure 12, the decision tree model has the highest RMSE and offers useful information. The trade-offs and factors to be taken into account when choosing an appropriate algorithm for prediction tasks are highlighted by this performance disparity. It encourages a more thorough investigation into the complexities of the dataset, illuminating the difficulties and subtleties that various models face.

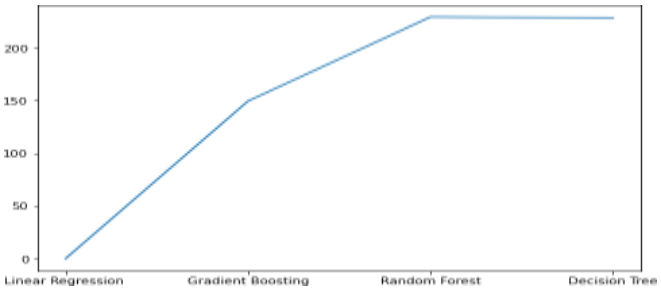


Figure 12. RMSE on plot for Desharnais

In the previous section, we examined the RMSE values for each dataset considered in this study. Here, we present a comparison of the RMSE results for Linear Regression, Gradient Boosting, Random Forests, and Decision Trees across the ISBSG, NASA93, COCOMO, Maxwell, and Desharnais datasets. For clarity, the RMSE values have been organized into a table (see Table 4). To highlight performance differences, the highest and lowest RMSE values for each dataset are shown in bold, making it easier to identify the best- and worst-performing models.

Table 4. RMSE Results with Correlation: Comparative Table for All Datasets

Dataset	Linear Regression	Gradient Boosting	Random Forest	Decision Tree
ISBSG	**5975.48**	10575.90	7660.77	9059.69
NASA93	**6.52**	12.29	20.90	46.00
COCOMO	**1.77**	328.70	377.60	274.71
Maxwell	**9.11**	6999.44	8268.12	6844.12
Desharnais	**1.51**	149.46	229.28	228.24

For the ISBSG dataset, the RMSE values were 5975.48 for Linear Regression, 10575.90 for Gradient Boosting, 7660.77 for Random Forest, and 9059.69 for Decision Trees. On the NASA93 dataset, the RMSEs were 6.52 (Linear Regression), 12.29 (Gradient Boosting), 20.90 (Random Forest), and 46.00 (Decision Trees). For COCOMO, the respective values were 1.77, 328.70, 377.60, and 274.71. On Maxwell, the RMSEs were 9.11 (Linear Regression), 6999.44 (Gradient Boosting), 8268.12 (Random Forest), and 6844.12 (Decision Trees). For Desharnais, the values were 1.51, 149.46, 229.28, and 228.24. Among these, Linear Regression achieved the lowest RMSE of 1.51 on the Desharnais dataset, as shown in Table 4. In terms of RMSE, linear regression continuously beat other models for the remaining datasets, suggesting that it predicts the data more accurately than other models.

Table 5. Comparison of RMSE Outcomes for All Datasets Excluding Correlation

Dataset	Linear Regression	Gradient Boosting	Random Forest	Decision Tree
ISBSG	4543.20	13038.48	7659.96	14485.16
NASA93	611.41	264.03	381.75	228.88
COCOMO	347.77	603.01	615.83	630.87
Maxwell	9334.95	9712.02	10005.56	9166.13

Table 5 presents the RMSE (Root Mean Squared Error) values for five datasets—ISBSG, NASA93, COCOMO, Maxwell, and Desharnais—using four regression models: Linear Regression, Gradient Boosting, Random Forest, and Decision Tree, without applying correlation. For the ISBSG dataset, the RMSEs were 4543.20 (Linear Regression), 13038.48 (Gradient Boosting), 7659.96 (Random Forest), and 14485.16 (Decision Tree), as shown in Table 5, with corresponding values reported for the other datasets as well. A comparison in Table 6 reveals that, across all datasets, the average RMSE values obtained with correlation were consistently lower than those without correlation, indicating improved prediction accuracy when correlation was applied. Specifically, the ISBSG, COCOMO, and Desharnais datasets showed reduced RMSE with correlation, while NASA93 displayed slightly lower RMSE without correlation, though the difference was not statistically significant. Notably, the Maxwell dataset achieved a significantly lower RMSE when correlation was included. Since lower RMSE values correspond to higher prediction accuracy, these findings suggest that incorporating correlation into software effort estimation enhances both accuracy and reliability, thereby supporting more effective software project management decisions.

Table 6. Comparison of Average RMSE Scores Across Datasets With and Without Correlation

Dataset	With Correlation	Without Correlation
ISBSG	7875.77	9761.90
NASA93	21.93	371.52
COCOM	245.20	549.37

O		
Maxwell	5575.42	9779.67
Desharnais	152.12	3495.62

5. DISCUSSION

This research assessed the performance of four regression models—Decision Trees, Random Forests, Gradient Boosting, and Linear Regression—across five software effort estimation datasets. According to the R2 and RMSE scores, we discovered that feature selection based on correlation may greatly increase prediction accuracy. Random forest and linear regression models in particular showed strong predictive power and could be the best option for estimating software development effort. Our findings are in accordance with a number of other research papers that shown the value of random forest and linear regression models for evaluating software development effort. But our research also offers some fresh perspectives. For instance, we discovered that feature selection based on correlation may greatly enhance these models' performance. According to our research, software engineering experts that need to estimate the amount of work needed to develop a software project may find the random forest and linear regression models helpful. When combined with correlation-based feature selection, these models may achieve high prediction accuracy and are comparatively easy to use and comprehend.

6. CONCLUSION

In summary, the results of this work demonstrate how the employment of machine learning models enhanced by correlation-based feature selection has led to notable improvements in software effort estimation. The demonstrated efficacy of techniques like Random Forest and linear regression, When combined with detailed correlation analysis, it indicates a significant advancement in accuracy compared to traditional estimation methods. The outstanding performance of the Linear Regression and Random Forest models is demonstrated by their perfect R^2 score of 1.0 and consistently low RMSE values (below 380) across datasets such as NASA93, COCOMO, Maxwell, and Desharnais shows how successful this approach is. Project managers may make informed decisions on software development project budgets and timelines because to this high predictive precision, which also makes it possible to organize task efforts reliably. The robustness of these data-driven models under sensitivity tests, which demonstrate that they may maintain stability in the face of disruptions, further supports their reliability for real-world deployment. The recommended technique's generalizability and applicability for projects of various sizes and degrees of complexity are established by the extensive validation of the method on five distinct public datasets spanning a variety of application domains. In conclusion, this work successfully illustrates how important correlation-based feature selection and machine learning models are to significantly raising the estimation accuracy of software development effort. Because of their greater reliability, these models are a very useful tool for project managers who want to streamline schedules, maximize financial resources, and adjust project scopes. Furthermore, the development of advanced estimation methods that utilize historical data is based on the practical guidance gleaned by finding the best models, such as Random Forest and linear regression. By looking into more complex regression algorithms and doing thorough testing on industry datasets, the proposed approach might be further enhanced and made more useful as a future research avenue. All things considered, this study makes a substantial contribution to the continuous improvement of project management techniques in the context of software development and represents a significant advancement in the application of data science and artificial intelligence to revolutionise traditional software estimation procedures. The results of this study might perhaps reduce errors, improve accuracy, and eventually reassess project management techniques for better results.

7. Future Research Directions

Although this study effectively illustrates how machine learning may increase the accuracy of software effort assessment, there are a number of encouraging directions to expand on these findings. Evaluating increasingly intricate and nonlinear regression methods, such as neural networks, Gaussian processes, and MARS, is one important avenue. The prediction skills of the basic linear and tree-based approaches

that have been evaluated thus far may be enhanced by the advanced modeling capabilities of these approaches. In addition to individual algorithms, creating ensemble models by fusing many strategies can also be successful since hybridization frequently yields more accurate predictions. Experimenting with actual industry datasets from software businesses to supplement the public dataset analysis conducted in this study is another crucial topic. Applicability in real-world development settings may be confirmed by testing on proprietary data that captures complex project details. Additionally, estimates can become more flexible to changing project dynamics through online learning where models gradually adjust based on fresh project data. Furthermore, a number of model generalization-related topics may be further investigated. Class imbalance in effort datasets with skew, for example, can be addressed by methods such as SMOTE. It's also worthwhile to investigate using deep learning to include textual elements from user stories and code complexity measurements. Over time, the models can remain relevant by using prudent retraining techniques. While bundled tools might show usefulness, outlier analysis is essential for confirming model predictions. In conclusion, even though this study makes a significant contribution to the development of machine learning—particularly correlation-based regression models—for improving software effort estimate, more work remains. By using the recommended methods, the models may be made more durable, robust, and customizable in real-world scenarios, which will increase business impact and hasten industry adoption. The ultimate goal would be to significantly enhance project planning and execution by institutionalizing data-driven estimating.

REFERENCE

[1] K. Moharreri, A. V. Sapre, J. Ramanathan, and R. Ramnath, "Cost-effective supervised learning models for software effort estimation in agile environments," in 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, GA, USA, 2016, pp. 135-140. doi: 10.1109/IC3.2016.7880216.

[2] Ritu and P. Bhambri, "A CAD system for software effort estimation," in 2022 2nd International Conference on Technological Advancements in Computational Sciences (ICTACS), 2022, pp. 140-146. doi: 10.1109/ICTACS56270.2022.9988123.

[3] Avazpour, T. Pitakrat, L. Grunske, and J. Grundy, "Dimensions and metrics for evaluating recommender systems," in Recommender Systems Handbook, 2014, pp. 245-273. doi: 10.1007/978-3-642-45135-5_10.

[4] Idri, T. M. Khoshgoftaar, and A. Abran, "Can neural networks be easily interpreted in software cost estimation?" Fuzzy Sets and Systems, vol. 132, no. 2, pp. 225-236, 2002. doi: 10.1016/S0165-0114(02)00079-6.

[5] Ritu and P. Bhambri, "Enhancing software development effort estimation with a cloud-based data framework using use case points, fuzzy logic, and machine learning," Discover Computing, vol. 28, 2025, Art. no. 143. doi: 10.1007/s10791-025-09668-1.

[6] M. A. Shah et al., "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," IEEE Access, vol. 8, pp. 58402-58415, 2020. doi: 10.1109/ACCESS.2020.2980236.

[7] P. Bhambri, "Implementing machine learning algorithms for distance based phylogenetic trees," M.S. thesis, I. K. G. Punjab Technical University, Jalandhar, India, 2018.

[8] S. Goyal, "Effective software effort estimation using heterogenous stacked ensemble," in 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), Thiruvananthapuram, India, 2022, pp. 584-588. doi: 10.1109/SPICES52834.2022.9774231.

[9] Y. Mahmood et al., "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. doi: 10.1002/spe.3009.

[10] S. Shukla, S. Kumar, and P. R. Bal, "Analyzing effect of ensemble models on multi-layer perceptron network for software effort estimation," in 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 386-387. doi: 10.1109/SERVICES.2019.00116.

[11] J. Kaur, P. Bhambri, and S. Kaur, "SVM classifier based method for software defect prediction," International Journal of Analytical and Experimental Model Analysis, vol. 11, no. 10, pp. 2772-2776, 2019.

[12] P. Bhambri, R. Rana, and S. Kautish, "Sustainable digital transformation: Reducing carbon footprint in the metaverse," in Metaverse and Sustainability: Business Resilience Towards Sustainable Development Goals, W. Leal Filho, S. Kautish, and V. P. Gupta, Eds. Cham, Switzerland: Springer, 2025, pp. 105-128. doi: 10.1007/978-3-031-89545-6_7.

[13] L. Lazic, "Artificial neural network architectures and orthogonal arrays in estimation of software projects efforts estimation: Plenary talk," in 2021 IEEE 19th International Symposium on Intelligent Systems and Informatics (SISY), Subotica, Serbia, 2021, pp. 13-14. doi: 10.1109/SISY52375.2021.9582466.

[14] R. Rana, R. Bhambri, and S. Kautish, "Enhancing electric vehicle adoption through metaverse applications: Strategies and impacts," in Metaverse and Sustainability: Business Resilience Towards Sustainable Development Goals, W. Leal Filho, S. Kautish, and V. P. Gupta, Eds. Cham, Switzerland: Springer, 2025, pp. 517-533. doi: 10.1007/978-3-031-89545-6_28.

[15] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 339-342. doi: 10.1109/SERVICES.2019.00094.

[16] Setiadi et al., "Analyze the datasets of software effort estimation with particle swarm optimization," in 2021 International Seminar on Intelligent Technology and Its Applications (ISITIA), Surabaya, Indonesia, 2021, pp. 197-201. doi: 10.1109/ISITIA52817.2021.9502208.

[17] Idri, T. M. Khoshgoftaar, and A. Abran, "Can neural networks be easily interpreted in software cost estimation?" Fuzzy Sets and Systems, vol. 132, no. 2, pp. 225-236, 2002.

[18] P. Bhambri and I. Paweloszek, Eds., Digital Sustainability: Navigating Entrepreneurship in the Information Age.

Boca Raton, FL, USA: CRC Press, 2024. doi: 10.1201/9781003484226.
[19] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in Proceedings of the 38th International Conference on Software Engineering, 2016, pp. 619-630. doi: 10.1145/2884781.2884830.
[20] Mahmood, M. U. S. Khan, and J. Kim, "An improved random forest model for software effort estimation," Applied Sciences, vol. 9, no. 23, p. 5249, 2019. doi: 10.3390/app9235249.
[21] P. Bhambri and P. Bajdor, Eds., Handbook of Technological Sustainability: Innovation and Environmental Awareness. Boca Raton, FL, USA: CRC Press, 2024. doi: 10.1201/9781003475989.
[22] M. Arora et al., "A state of the art regressor model's comparison for effort estimation of agile software," in 2021 2nd International Conference on Intelligent Engineering and Management (ICIEM), London, United Kingdom, 2021, pp. 211-215. doi: 10.1109/ICIEM51511.2021.9445345.
[23] R. Rana and P. Bhambri, "Environmental challenges and technological solutions," in Handbook of Technological Sustainability: Innovation and Environmental Awareness, P. Bhambri and P. Bajdor, Eds. Boca Raton, FL, USA: CRC Press, 2024, pp. 187-200.
[24] N. Mittas and L. Angelis, "Comparing cost prediction models by resampling techniques," Journal of Systems and Software, vol. 81, no. 6, pp. 816-824, 2008. doi: 10.1016/j.jss.2007.07.039.
[25] P. Bhambri, S. Rani, and V. K. Sinha, "Big data analytics with digital twin for industrial applications," in AI-Driven Digital Twin and Industry 4.0: A Conceptual Framework with Applications, S. Rani, P. Bhambri, S. Kumar, P. K. Pareek, and A. A. Elngar, Eds. Boca Raton, FL, USA: CRC Press, 2024, pp. 105-126.
[26] M. Jorgensen, "A review of studies on expert estimation of software development effort," Journal of Systems and Software, vol. 70, no. 1-2, pp. 37-60, 2016. doi: 10.1016/j.jss.2004.09.006.
[27] P. Bhambri, S. Rani, I. S. Dhanoa, and T. A. Tran, "Environmental impacts of industrial processes in Industry 4.0 ecosystem artificial intelligence approach," in AI-Driven Digital Twin and Industry 4.0: A Conceptual Framework with Applications, S. Rani, P. Bhambri, S. Kumar, P. K. Pareek, and A. A. Elngar, Eds. Boca Raton, FL, USA: CRC Press, 2024, pp. 221-240.
[28] D. S. Senevirathne and T. K. Wijayasiriwardhane, "Extending use-case point-based software effort estimation for Open Source freelance software development," in 2020 International Research Conference on Smart Computing and Systems Engineering (SCSE), Colombo, Sri Lanka, 2020, pp. 188-194. doi: 10.1109/SCSE49731.2020.9313007.
[29] W. Li, H. Leung, L. Fang, and S. Cai, "A packing-based simulated annealing algorithm for feature selection," Applied Sciences, vol. 12, no. 7, p. 3553, 2022. doi: 10.3390/app12073553.
[30] L. Zahedi, F. Ghareh Mohammadi, and M. H. Amini, "A2BCF: An automated ABC-based feature selection algorithm for classification models in an education application," Applied Sciences, vol. 12, no. 7, p. 3553, 2022. doi: 10.3390/app12073553.
[31] P. Bhambri and A. Khang, "Edge computing for enhancing efficiency and sustainability in green transportation systems," in Driving Green Transportation System Through Artificial Intelligence and Automation, A. Khang, Ed. Cham, Switzerland: Springer, 2025, pp. 43-65. doi: 10.1007/978-3-031-72617-0_3.
[32] H. Benbrahim and J. A. Franklin, "Bivariate property estimation from incomplete data," Mathematical Geology, vol. 29, no. 3, pp. 391-408, 1997.
[33] Nugroho, A. Z. Fanani, and G. F. Shidik, "Evaluation of feature selection using wrapper for numeric dataset with random forest algorithm," in 2021 International Seminar on Application for Technology of Information and Communication (iSemantic), Semarang, Indonesia, 2021, pp. 179-183. doi: 10.1109/ISEMANTIC52711.2021.9573249.
[34] H. Liu, M. Zhou, and Q. Liu, "An embedded feature selection method for imbalanced data classification," IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 3, pp. 703-715, 2019. doi: 10.1109/JAS.2019.1911447.
[35] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, vol. 21, no. 2, pp. 126-137, 1995. doi: 10.1109/32.345828.
[36] G. Chandrashekar and F. Sahin, "A survey on feature selection methods," Computers & Electrical Engineering, vol. 40, no. 1, pp. 16-28, 2014.