

Software Testing Effort Estimation Based on Machine Learning Techniques: Single and Ensemble Methods

Mohamed Hosni¹ , Ibtissam Medarhri²  and Juan Manuel Carrillo de Gea³ 

¹*MOSI Research Team, LM2S3, ENSAM, Moulay Ismail University of Meknes, Morocco*

²*MMCS Research Team, LMAID, ENSMR-Rabat, Morocco*

¹*Department of Informatics and Systems, Faculty of Computer Science, University of Murcia, Spain*

Keywords: Software Testing, Software Testing Effort, Machine Learning, Ensemble Method, ISBSG.


Abstract: Delivering an accurate estimation of the effort required for software system development is crucial for the success of any software project. However, the software development lifecycle (SDLC) involves multiple activities, such as software design, software build, and software testing, among others. Software testing (ST) holds significant importance in the SDLC as it directly impacts software quality. Typically, the effort required for the testing phase is estimated as a percentage of the overall predicted SDLC effort, typically ranging between 10% and 60%. However, this approach poses risks as it hinders proper resource allocation by managers. Despite the importance of this issue, there is limited research available on estimating ST effort. This paper aims to address this concern by proposing four machine learning (ML) techniques and a heterogeneous ensemble to predict the effort required for ST activities. The ML techniques employed include K-nearest neighbor (KNN), Support Vector Regression, Multilayer Perceptron Neural Networks, and decision trees. The dataset used in this study was obtained from a well-known repository. Various unbiased performance indicators were utilized to evaluate the predictive capabilities of the proposed techniques. The overall results indicate that the KNN technique outperforms the other ML techniques, and the proposed ensemble showed superior performance accuracy compared to the remaining ML techniques.


1 INTRODUCTION


The software development life cycle (SDLC) encompasses a comprehensive range of activities that cover multiple aspects of a software project. These activities include strategic planning, thorough requirements specification, meticulous analysis and design, precise programming, rigorous testing, seamless integration, smooth deployment, and various other supportive tasks. Together, they form a cohesive framework for the successful development and implementation of high-quality software systems (Radliński, 2023). Ensuring precise estimation of the effort needed to accomplish each of these activities is crucial for the overall success of the project (Charette, 2005). Despite the majority of research in the literature focusing on proposing automated techniques for accurate effort estimation in software development (Hosni et al.,

2019a; Azzeh and Nassif, 2013), there has been relatively limited research conducted specifically on predicting the effort required to complete a specific activity in the SDLC, such as testing, even though it is a significant and challenging area. Therefore, this research work attempts to propose a software testing effort estimation technique based on machine learning methods.

Recently, a systematic literature review (SLR) was conducted on the use of ML in software testing (Ajlou et al., 2024). This work systematically analyzes 40 studies published between 2018 and 2024, exploring various ML methods, including supervised, unsupervised, reinforcement, and hybrid approaches in software testing. It highlights ML's significant role in automating test case generation, prioritization, and fault detection, but also identifies a critical gap in the area of software test effort prediction—an important element for effective resource management, cost estimation, and project scheduling. Despite its importance, the review reveals that few studies specifically address this area, underscoring the urgent need for

^a  <https://orcid.org/0000-0001-7336-4276>

^b  <https://orcid.org/0009-0003-0052-8702>

^c  <https://orcid.org/0000-0002-3320-622X>

further research on ML-based models to improve test effort predictions and enhance overall software testing efficiency.

Software testing holds significant importance in the SDLC as it serves to identify defects, errors, and inconsistencies within a software system (López-Martín, 2022). The primary objective of this important phase is to execute software components or systems to uncover bugs, verify adherence to specified requirements, and ensure the overall quality of the software product. By conducting comprehensive testing, developers can detect and rectify any flaws, ensuring that the software meets the desired standards and functions optimally (Radliński, 2023). The testing process plays a critical role in enhancing the reliability, performance, and user experience of the software, contributing to the success of the overall development project.

Software testing activities play a vital role in evaluating the functionality of software and determining the extent to which it meets stakeholders' expectations. Essentially, this phase ensures the software's desired quality. In terms of time and cost, software testing holds significant importance within the SDLC. Researchers have made several efforts to estimate the effort required for conducting testing activities (Radliński, 2023). Typically, the effort needed to test a software system is measured in person-hours (López-Martín, 2022). During the planning phase of a project, the overall effort required for the SDLC is estimated, and a certain percentage is allocated to account for software testing activities. However, accurately predicting the effort necessary for testing poses challenges due to the considerable variability in the percentage allocation for testing critical software components. This percentage can vary widely, from 10% to 60% or even higher (López-Martín, 2022). Thus, accurately estimating the effort required for testing remains a complex task.

ML techniques have been widely employed for over three decades to estimate software development effort with a higher degree of accuracy (Hosni and Idri, 2018). These techniques utilize historical data from completed projects to uncover complex relationships between various software factors and the effort required to develop a software system (Ali and Gravino, 2019; Wen et al., 2012). This enables ML models to generate more accurate predictions, overcoming the limitations of traditional software estimation techniques, such as parametric methods. Unlike traditional approaches, ML techniques can capture non-linear relationships between the target variable (i.e., effort) and the independent variables. This flexibility makes ML models well-suited for provid-

ing reliable estimations, which in turn assist project managers in making informed decisions regarding resource allocation and effectively monitoring overall project progress.

In Software Development Effort Estimation (SDEE), researchers have extensively explored a novel approach known as ensemble effort estimation (EEE) (Hosni et al., 2019b; Idri et al., 2016; d. A. Cabral et al., 2023). This technique involves combining multiple ML techniques into a single ensemble model, utilizing a combination rule to generate predictions. The EEE approach has demonstrated superior accuracy compared to using a single ML technique. Extensive literature reports consistently indicate that EEE outperforms individual ensemble members in most cases, highlighting the effectiveness of the ensemble approach in improving the accuracy of SDEE.

In this paper, our objective is to explore the application of well-established ML techniques in SDEE specifically for estimating the effort required in software testing activities. We have selected four widely used ML techniques: k-nearest neighbor (KNN), Support Vector Regression (SVR), Multilayer Perceptron (MLP) Neural Networks, and decision trees (DTs). Additionally, we propose an ensemble model that combines these four ML techniques. To obtain the final estimation from the ensemble, three combiners are employed: average, median, and inverse ranked weighted mean.

To conduct our study, we utilized a historical dataset obtained from the International Software Benchmarking and Standards Group (ISBSG) database, Release 12. In this research work, we address three research questions (RQs):

- **(RQ1). Among the four ML techniques used, which one generates the most accurate results?**
- **(RQ2). Is there any evidence that the proposed ensemble method performs better than the individual ML techniques?**
- **(RQ3). What are the main features that impact software testing effort (STE) among the input features used for the ML techniques?**

The main features of this paper are as follows:

- Utilizing four well-known ML techniques for estimating software testing effort (STE).
- Employing an ensemble method for estimating STE.
- Evaluating the predictive capabilities of these STE techniques using unbiased performance measures.

- Identifying the most significant features that impact the estimation of STE.

The organization of the remaining parts of this paper is as follows: Section 2 presents a comprehensive analysis of previous studies. Section 3 provides the list of the ML techniques employed in this research. Section 4 outlines the methodology implemented, including the materials utilized. Section 5 discusses the significant findings derived from the study. Lastly, the concluding section summarizes the paper and proposes future research directions.

2 RELATED WORK

This section presents some related work conducted in the literature of STE estimation and defines the EEE approach.

López-Martín (López-Martín, 2022) carried out an empirical study to explore the use of ML techniques for predicting software testing effort (STE) in the software development lifecycle (SDLC). The research examined five ML models—case-based reasoning, artificial neural networks (ANN), support vector regression (SVR), genetic programming, and decision trees (DTs)—to assess their accuracy in estimating software development effort (SDEE). The models were trained and evaluated using datasets from the ISBSG, which were chosen based on factors such as data quality, development type, platform, programming language, and resource level. The findings revealed that support vector regression (SVR) provided the most accurate predictions, particularly when evaluated using mean absolute error (MAE).

Labidi et al. (Labidi and Sakhrawi, 2023) conducted an empirical study aimed at predicting software testing effort (STE) using ensemble methods. The proposed approach combined three machine learning techniques: ANN, SVR, and DTs, with each model optimized through grid search. The ISBSG dataset was employed after a preprocessing step for empirical evaluation. Results indicated that the ensemble model outperformed the individual ML techniques based on performance metrics such as root mean square error (RMSE), R-squared, and MAE. However, the study lacks specific details about the dataset used for training and testing, only mentioning that 17 features were used as inputs for the predictive models. To the best of the authors' knowledge, this study, along with another, represents the limited research exploring ML techniques for predicting software testing effort.

In the last decade, there has been significant investigation into the ensemble approach in the context

of SDEE. This approach involves predicting the effort needed to develop a software system by using multiple estimators. Ensembles can be categorized into two types (Azzeh et al., 2015; Elish et al., 2013): homogeneous and heterogeneous. Homogeneous ensembles combine at least two variants of the same estimation technique or combine one estimation technique with a meta-learner such as Bagging, Boosting, or Random Subspace. Heterogeneous ensembles, on the other hand, involve combining at least two different techniques. A review conducted by Idri et al. (Idri et al., 2016) identified 16 SDEE techniques that have been used to construct EEE techniques. The review revealed that the homogeneous type of ensemble was the most frequently investigated. In terms of combiners, the review identified 20 different combiners that were adopted to merge the individual estimates provided by the ensemble members. It was found that linear rules were the most commonly used type of combiner.

3 MACHINE LEARNING

Four ML techniques were employed in this study: KNN (Altman, 1992), MLP (Simon, 1999), SVR (Simon, 1999), and DT (Jeffery et al., 2001), besides an heterogeneous ensemble consisting of the four ML techniques using three combiners: average, median, and inverse ranked weighted mean.

4 EMPIRICAL DESIGN

This section outlines the experimental design adopted to conduct the experiments presented in this paper. It begins by specifying the performance metrics and statistical tests used to assess the accuracy of the proposed predictive models. Next, it details the use of the grid search hyperparameter optimization technique to fine-tune the parameter settings of the predictive models. It then provides information on the dataset chosen for empirical analysis. Finally, it describes the methodology employed for building the predictive models.

4.1 Performance Metrics and Statistical Test

To evaluate the accuracy of the proposed techniques, we employed a set of eight widely used performance criteria commonly found in the SDEE literature. These criteria include Mean Absolute Error (MAE),

Mean Balanced Relative Error (MBRE), Mean Inverted Balanced Relative Error (MIBRE), along with their respective median values, Logarithmic Standard Deviation (LSD), and Prediction at 25% (Pred(25)) (Miyazaki, 1991; Minku and Yao, 2013; Foss et al., 2003).

Additionally, to determine whether the investigated STEE techniques outperformed random guessing, we utilized standardized accuracy (SA) and effect size as additional evaluation measures (Shepperd and MacDonell, 2012). The mathematical formulas for these performance indicators are provided in Equations (1)-(8).

$$AE_i = |e_i - \hat{e}_i| \quad (1)$$

$$Pred(0.25) = \frac{100}{n} \sum_{i=1}^n \begin{cases} 1 & \text{if } \frac{AE_i}{e_i} \leq 0.25 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$MAE = \frac{1}{n} \sum_{i=1}^n AE_i \quad (3)$$

$$MBRE = \frac{1}{n} \sum_{i=1}^n \frac{AE_i}{\min(e_i, \hat{e}_i)} \quad (4)$$

$$MIBRE = \frac{1}{n} \sum_{i=1}^n \frac{AE_i}{\max(e_i, \hat{e}_i)} \quad (5)$$

$$LSD = \sqrt{\frac{\sum_{i=1}^n (\lambda_i + \frac{s^2}{2})^2}{n-1}} \quad (6)$$

$$SA = 1 - \frac{MAE_{p_i}}{MAE_{p_0}} \quad (7)$$

$$\Delta = \frac{MAE_{p_i} - MAE_{p_0}}{S_{p_0}} \quad (8)$$

where:

- e_i and \hat{e}_i denote the actual and predicted effort, respectively, for the i^{th} project.
- The average mean absolute error from numerous random guessing trials is represented by \overline{MAE}_{p_0} . It is computed by randomly sampling (with equal probability) from the remaining $n-1$ cases and setting $\hat{e}_i = e_r$, where r is a randomly selected value from 1 to n , excluding i . This randomization method is robust as it does not rely on any assumptions about the population.
- The mean of absolute errors for a given prediction technique i , denoted as MAE_{p_i} , corresponds to the standard deviation of the sample derived from the random guessing approach.
- λ_i is determined by taking the natural logarithm of e_i and subtracting the natural logarithm of \hat{e}_i .
- The term s^2 is used as an estimator of the residual variance associated with λ_i .

The predictive models were built using the Leave-One-Out Cross-Validation (LOOCV) technique.

To assess the statistical significance of the proposed technique based on AE, the Scott-Knott (SK) test was employed. The SK test is a statistical method used to compare and rank different approaches or techniques based on their performance metrics. It helps determine whether there are significant differences in performance between the evaluated approaches.

4.2 Hyperparameters Optimization

Several papers in the SDEE literature have discussed hyperparameter settings in detail (Song et al., 2013; Hosni et al., 2018; Hosni, 2023). These studies have highlighted the importance of optimization techniques in enhancing the accuracy of predictive models. It has been observed that the performance of ML techniques in SDEE can vary significantly across different datasets. Consequently, using the same parameter settings for a given technique may result in an incorrect assessment of its predictive capability. To address this issue, we employ the grid search optimization method to determine the optimal parameter values for the selected models. Table 1 presents the predefined search space, specifying the range of optimal parameter values for each ML technique.

4.3 Datasets

The predictive analysis conducted in this paper utilized the dataset from the International Software Benchmarking Standards Group (ISBSG). This comprehensive dataset includes over 6,000 projects and more than 120 features covering aspects such as project size, effort, schedule, development type, and application environment. Prior to building the machine learning models, the dataset undergoes preprocessing. This process starts with selecting software projects with high data quality, adhering to the guidelines established by the ISBSG group. The selection criteria are based on the standards outlined in (Hosni et al., 2019a; Labidi and Sakhrawi, 2023).

Afterwards, we selected attributes that, according to the authors' knowledge, have a clear influence on the STE. As a result, we selected nine numerical features along with the target variable 'Effort Test'. The input features used for our predictive models are listed in Table 2. It is worth noting that any data rows with missing values were removed from the dataset.

Table 1: Range of parameters values for each ML technique.

Technique	Search space
KNN	'n_neighbors': [1,11], 'weights': ['uniform', 'distance'], 'metric': ['euclidean', 'manhattan', 'cityblock', 'minkowski']
SVR	'kernel': ['rbf', 'poly'], 'C': [5, 10, 20, 30, 40, 50, 100], 'epsilon': [0.0001, 0.001, 0.01, 0.1], 'degree': [2, 3, 4, 5, 6], 'gamma': [0.0001, 0.001, 0.01, 0.1]
MLP	'hidden_layer_sizes': [(8,), (8,16), (8, 16, 32), (8,16,32,64)], 'activation': ['relu', 'tanh', 'identity', 'logistic'], 'solver': ['adam', 'lbfgs', 'sgd'], 'learning_rate': ['constant', 'adaptive', 'invscaling'],
DT	'criterion': ['squared_error', 'friedman_mse', 'absolute_error', 'poisson'], 'max_depth': [None] + [1, number of feature space], 'max_features': [None, 'sqrt', 'log2']

Table 2: Selected features.

Feature	Importance score
Enquiry count	0.131424
File count	0.12467
Output count	0.121829
Adjusted function points	0.12108
Input count	0.120946
Max team size	0.120207
Interface count	0.103466
Value adjustment factor	0.083748
User base - locations	0.07263
Effort test	-

4.4 Evaluation Methodology

This subsection outlines the experimental design employed to develop and evaluate the proposed STE techniques in this paper.

- **Step 1:** Four ML algorithms: KNN, SVR, MLP and DT, were trained and optimized using grid search with 10-fold cross-validation to identify the best hyperparameters.
- **Step 2:** Optimal hyperparameter values were selected for each model based on the lowest Mean Absolute Error (MAE).
- **Step 3:** The models were then retrained using the identified optimal parameters and evaluated using LOOCV.
- **Step 4:** The validity of the optimized models was assessed through Standardized Accuracy (SA) and effect size analysis, comparing their performance against the 5% quantile of random guessing.

- **Step 5:** Performance was measured using a comprehensive set of indicators: Mean Absolute Error (MAE), Median Absolute Error (MdAE), Mean Inverted Balanced Relative Error (MIBRE), Median Inverted Balanced Relative Error (MdIBRE), Mean Balanced Relative Error (MBRE), Median Balanced Relative Error (MdBRE), Logarithmic Standard Deviation (LSD), and Prediction at 25% (Pred(25)).
- **Step 6:** A heterogeneous ensemble was created by integrating the four models using three combination methods: average (AVR), median (MED), and inverse rank-weighted mean (IRWM).
- **Step 7:** The ensemble's performance was evaluated using the same metrics outlined in Step 5.
- **Step 8:** The software effort estimation methods were ranked using the Borda count voting system, considering all eight performance metrics.
- **Step 9:** The Scott-Knott statistical test was applied to group the estimation techniques into statistically similar categories based on AE, identifying those with comparable predictive performance.

5 EMPIRICAL RESULTS

This section presents the empirical findings derived from the experiment conducted in this paper. The experiments were executed using various tools, with Python and its associated libraries being used to run the experiments. Additionally, the R programming language was utilized to perform the SK test.

5.1 Single Techniques Assessment

In this phase, the first step involves identifying the optimal parameters that yield improved estimates for each individual technique. To achieve this, multiple rounds of preliminary experiments were conducted using the grid search optimization technique. The hyperparameters were varied within the range values specified in Table 1 for the four selected ML techniques: KNN, SVR, MLP, and DT. The evaluation was performed using the 10-fold cross-validation technique. The objective function targeted for minimization was the MAE criterion. The rationale behind selecting MAE is its unbiased nature as a performance measure.

Subsequently, we constructed our predictive models using the optimal parameters identified in the previous step, employing the LOOCV technique for validation. This approach was selected for its ability to provide low bias and high variance estimates, enhancing the replicability of the study.

We then evaluated the reasonability of our STE techniques by comparing them to a baseline estimator suggested by Shepperd and MacDonell (Shepperd and MacDonell, 2012), which constructs an estimator through multiple runs of random guessing.

The evaluation was carried out using the Standardized Accuracy (SA) metric and effect size (Δ), as proposed by the authors. As shown in Table 3, all four ML techniques significantly outperformed random guessing, showing substantial improvement with effect sizes greater than 0.8 ($\Delta > 0.8$). Notably, all techniques exceeded the 5% quantile of random guessing. Among the techniques, KNN ranked highest in both SA and effect size improvement, while SVR ranked lowest.

Table 3: SA and effect size value of the constructed techniques.

Technique	SA	Delta
	$SA_{5\%} = 0.2061$	
KNN	0.981245	-7.134
SVR	0.384077	-2.79237
MLP	0.548538	-3.98806
DT	0.554524	-4.03159

We then assessed the accuracy of the four ML techniques using the eight chosen performance metrics. The evaluation results are summarized in Table 4.

The KNN technique demonstrated the highest accuracy among the four ML techniques used in this study, consistently ranking first across all eight performance metrics. DT and MLP followed, frequently

alternating between second and third positions across several indicators. SVR consistently ranked lowest across all performance measures.

These results suggest that the proposed approach provides satisfactory accuracy, with KNN standing out as the most effective technique for estimating STE among those evaluated.

5.2 Ensemble Methods

This step involves constructing the proposed heterogeneous ensemble using the four ML techniques. The ensemble produces the final estimation through three combiners: AVR, MED, and IRWM based on the MAE. This approach is grounded in SDEE literature, which indicates that ensembles typically achieve higher accuracy than individual estimation techniques.

Performance metrics of the constructed ensemble, based on the eight selected indicators, are presented in Table 5. The ensemble with the IRWM combiner (EIRWM) consistently outperformed the others, ranking first across all performance metrics. The ensembles with AVR (EAVR) and MED (EMED) combiners ranked second and third, respectively. The consistent rankings of the ensemble techniques across all performance indicators demonstrate their reliable and stable accuracy.

5.3 STE Techniques Comparison

In this step, we ranked all the proposed techniques using the eight accuracy measures. The final ranking was determined through the Borda count voting system, which considers all eight performance metrics. This approach was chosen because the accuracy of a technique can depend on the selected performance indicators, potentially leading to conflicting results as different metrics may produce varying rankings for each technique (Myrtveit et al., 2005; Mittas and Angelis, 2013). Table 6 presents the final rankings obtained through the Borda count system. As shown, the KNN technique achieved the top position, followed by the three heterogeneous ensembles, with SVR ranked last.

To validate these results, we conducted the SK statistical test to identify techniques with statistically similar predictive capabilities. The SK test was performed based on the AE of the proposed techniques. Table 6 shows the content of clusters identified by the SK test.

The first cluster contained only the KNN technique, while the second cluster included the proposed ensemble methods. The last cluster was comprised

Table 4: Performance metrics for the four ML techniques.

Technique	MAE	MdAE	MBRE	MdBRE	MIBRE	MdIBRE	PRED	LSD
KNN	17.66399	0	0.168228	0	0.019629	0	95.55556	0.320259
SVR	580.0858	336.8311	112995.8	1.602058	0.523648	0.615689	17.77778	3.649184
MLP	425.1941	313.952	106435.1	0.753829	0.456234	0.429819	24.44444	3.235486
DT	419.5556	225	75112.29	0.836956	0.453273	0.455621	24.44444	3.418981

Table 5: Accuracy performance of the ensemble methods.

Technique	MAE	MdAE	MBRE	MdBRE	MIBRE	MdIBRE	PRED	LSD
EAVR	306.2841	190.1564	73635.61	0.555469	0.374774	0.3571069	35.55556	3.103686
EMED	333.5098	207.3907	90773.51	0.710584	0.394308	0.4154042	35.55556	3.236895
EIRWM	244.7196	156.9293	55120.18	0.439571	0.327887	0.3053485	42.22222	2.971071

Table 6: Rank obtained by Borda Count Voting System, and identified Clusters.

Rank	Models	Cluster
1	KNN	1
2	EIRWM	2
3	EAVR	2
4	EMED	2
5	DT	3
6	MLP	3
8	SVR	4

solely of the SVR technique. Notably, the clusters identified by the SK test correspond closely with the rankings obtained through the Borda count method. This confirms that the KNN technique remains statistically the most superior, while the three proposed ensemble methods consistently outperform the other individual techniques.

5.4 Features Importance

An important aspect of our investigation was assessing feature importance in explaining the target variable, Effort Test. We employed the **ExtraTreesClassifier**, which uses multiple decision trees to evaluate and rank the significance of features within the dataset.

Table 2 shows the importance scores for each feature used in our predictive models. The results confirm that all features contribute to the target variable, aligning with our manual feature selection process on the original ISBSG dataset. Notably, the ISBSG dataset contains over 100 features, suggesting that incorporating additional relevant features could enhance the predictive models' accuracy.

It is important to note that there is currently no literature specifically addressing which software features are most effective for predicting software testing activities. Therefore, a more comprehensive analysis

is required to identify the most impactful features for this purpose.

6 CONCLUSIONS AND FURTHER WORK

This empirical study explored the effectiveness of ML techniques in estimating the effort required for software testing activities within the SDLC. Four ML techniques and three heterogeneous ensembles were examined, with hyperparameters optimized using grid search. The evaluation employed the Leave-One-Out Cross-Validation (LOOCV) technique and eight unbiased performance metrics. The key findings related to each research question are summarized below:

- **(RQ1).** The KNN technique consistently outperformed the other three ML techniques across all eight performance metrics.
- **(RQ2).** Results indicated that the ensemble methods did surpass the accuracy of the individual techniques (SVR, DT, and MLP) and show less performance than KNN. This conclusion was supported by the SK test.
- **(RQ3).** All features used in training the ML techniques were identified as important; however, integrating additional features could further enhance the models' predictive capabilities.

Ongoing research is focused on exploring alternative ensemble methods, particularly homogeneous ensembles, which were not covered in this study. Efforts are also underway to improve the selection of ensemble components. Additionally, acquiring more relevant datasets for STE is a key priority, as this will contribute to the development of more robust and accurate STE models.

REFERENCES

- Ajorloo, S., Jamarani, A., Kashfi, M., Kashani, M. H., and Najafzadeh, A. (2024). A systematic review of machine learning methods in software testing. *Applied Soft Computing*, page 111805.
- Ali, A. and Gravino, C. (2019). A systematic literature review of software effort prediction using machine learning methods. *J. Softw. Evol. Process*, 31(10):1–25.
- Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *Am. Stat.*, 46(3):175–185.
- Azzeh, M. and Nassif, A. B. (2013). Fuzzy model tree for early effort estimation. In *2013 12th International Conference on Machine Learning and Applications*, pages 117–121.
- Azzeh, M., Nassif, A. B., and Minku, L. L. (2015). An empirical evaluation of ensemble adjustment methods for analogy-based effort estimation. *J. Syst. Softw.*, 103:36–52.
- Charette, R. N. (2005). Why software fails? *IEEE Spectr.*, 42(9):42–49.
- d. A. Cabral, J. T. H., Oliveira, A. L. I., and da Silva, F. Q. B. (2023). Ensemble effort estimation: An updated and extended systematic literature review. *J. Syst. Softw.*, 195:111542.
- Elish, M. O., Helmy, T., and Hussain, M. I. (2013). Empirical study of homogeneous and heterogeneous ensemble models for software development effort estimation. *Math. Probl. Eng.*, 2013.
- Foss, T., Stensrud, E., Kitchenham, B., and Myrtveit, I. (2003). A simulation study of the model evaluation criterion mmre. *IEEE Trans. Softw. Eng.*, 29(11):985–995.
- Hosni, M. (2023). Encoding techniques for handling categorical data in machine learning-based software development effort estimation. In *KDIR*, pages 460–467.
- Hosni, M. and Idri, A. (2018). Software development effort estimation using feature selection techniques. In *Frontiers in Artificial Intelligence and Applications*, pages 439–452.
- Hosni, M., Idri, A., and Abran, A. (2019a). Evaluating filter fuzzy analogy homogenous ensembles for software development effort estimation. *J. Softw. Evol. Process*, 31(2).
- Hosni, M., Idri, A., and Abran, A. (2019b). Improved effort estimation of heterogeneous ensembles using filter feature selection. In *ICSOFT 2018 - Proceedings of the 13th International Conference on Software Technologies*, pages 405–412. SciTePress.
- Hosni, M., Idri, A., Abran, A., and Nassif, A. B. (2018). On the value of parameter tuning in heterogeneous ensembles effort estimation. *Soft Comput.*, 22(18):5977–6010.
- Idri, A., Hosni, M., and Abran, A. (2016). Systematic mapping study of ensemble effort estimation. In *Proceedings of the 11th International Conference on Evaluation of Novel Software Approaches to Software Engineering*, pages 132–139.
- Jeffery, R., Ruhe, M., and Wiczorek, I. (2001). Using public domain metrics to estimate software development effort. In *Seventh International Software Metrics Symposium. METRICS 2001*, pages 16–27.
- Labidi, T. and Sakhravi, Z. (2023). On the value of parameter tuning in stacking ensemble model for software regression test effort estimation. *J. Supercomput.*, page 0123456789.
- López-Martín, C. (2022). Machine learning techniques for software testing effort prediction. *Softw. Qual. J.*, 30(1):65–100.
- Minku, L. L. and Yao, X. (2013). An analysis of multi-objective evolutionary algorithms for training ensemble models based on different performance measures in software effort estimation. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering - PROMISE '13*, pages 1–10.
- Mittas, N. and Angelis, L. (2013). Ranking and clustering software cost estimation models through a multiple comparisons algorithm. *IEEE Trans. Softw. Eng.*, 39(4):537–551.
- Miyazaki, Y. (1991). Method to estimate parameter values in software prediction models. *Inf. Softw. Technol.*, 33(3):239–243.
- Myrtveit, I., Stensrud, E., and Shepperd, M. (2005). Reliability and validity in comparative studies of software prediction models. *IEEE Trans. Softw. Eng.*, 31(5):380–391.
- Radliński, Ł. (2023). The impact of data quality on software testing effort prediction. *Electron.*, 12(7).
- Shepperd, M. and MacDonell, S. (2012). Evaluating prediction systems in software project estimation. *Inf. Softw. Technol.*, 54(8):820–827.
- Simon, H. (1999). *Neural networks: a comprehensive foundation*. MacMillan Publishing Company, 2nd edition.
- Song, L., Minku, L. L., and Yao, X. (2013). The impact of parameter tuning on software effort estimation using learning machines. In *Proceedings of the 9th International Conference on Predictive Models in Software Engineering*.
- Wen, J., Li, S., Lin, Z., Hu, Y., and Huang, C. (2012). Systematic literature review of machine learning based software development effort estimation models. *Inf. Softw. Technol.*, 54(1):41–59.