IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Review and Empirical Analysis of Software Effort Estimation

MIZANUR RAHMAN[1], HASAN SARWAR[2], MD ABDUL KADER[3],TERESA GONÃĞALVES[4], and TING TIN TIN[5]

[1]School of Computer Science, Western Illinois University (e-mail: mizancse7462@gmail.com)
[2]Department of Computer Science and Engineering, United International University, Dhaka, Bangladesh (e-mail: hsarwar@cse.uiu.ac.bd)
[3]Faculty of Computing, Universiti Malaysia Pahang Al-Sultan Abdullah (e-mail: abdkader@umpsa.edu.my)
[4]Departamento de InformÃątica, Universidade de Ãl'vora (e-mail: tcg@uevora.pt)
[5]FACULTY OF DATA SCIENCE AND INFORMATION TECHNOLOGY, INTI International University (e-mail: tintin.ting@newinti.edu.my)

Corresponding author: Hasan Sarwar (e-mail: hsarwar@cse.uiu.ac.bd)

ABSTRACT

The average software company spends a huge amount of its revenue on R&D for how to deliver software on time. Accurate software effort estimation is critical for successful project planning, resource allocation, and on-time delivery within budget for sustainable software development. However, both overestimation and underestimation pose significant challenges in software development, necessitating continuous improvement in estimation techniques. This study reviews recent machine learning approaches exploited to enhance software effort estimation (SEE) accuracy, focusing on research published between 2020 and 2023. The literature review employed an approach to identify pertinent research on machine learning techniques for software estimation efforts. Additionally, comparative experiments were conducted employing five commonly used ML methods: K-Nearest Neighbor, Support Vector Machine, Random Forest, Logistic Regression, and LASSO Regression. These techniques were assessed using five widely employed accuracy metrics such as Mean Squared Error (MSE), Mean Magnitude of Relative Error (MMRE), R-squared, Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) on seven benchmark datasets (Albrecht, Desharnais, China, Kemerer, Mayazaki94, Maxwell, COCOMO). By carefully reviewing study quality, analyzing results across the literature, and rigorously evaluating experimental outcomes, clear conclusions were drawn about the most promising techniques for achieving state-of-the-art accuracy in estimating software effort. This study makes three key contributions to the field: firstly, it furnishes a thorough overview of recent machine learning research in software effort estimation (SEE); secondly, it provides data-driven guidance for researchers and practitioners to select optimal methods for accurate effort estimation; and thirdly, it demonstrates the performance of publicly available datasets through experimental analysis. Enhanced estimation supports the development of better predictive models for software project time, cost, and staffing needs. The findings aim to focus future research directions and tool development toward the most accurate machine learning approaches for modeling software development effort, costs, and delivery schedules.

INDEX TERMS Software Effort Estimation, Software Development Efforts Estimation, Linear Regression, Support Vector Machine, Random Forest, LASSO, KNN, R&D investment

I. INTRODUCTION

Accurate software effort estimation is essential in software development, as it involves predicting the level of effort and time required to successfully develop a software system within budget and on schedule. Since the 1960s, research on improving software effort prediction has been ongoing, motivated by the need for reliable results that enable effective planning and resource allocation [1]. Notwithstanding, the persistent challenge in achieving precise estimates is rooted in inherent complexities. The accuracy of estimations significantly influences planning, budgeting, scheduling, and resource management [2]. However, the initial phases of the development process offer limited information, thereby hampering precision. Moreover, the intricacies surrounding this

process have sparked extensive debate within the software engineering community. Enhancing estimation reliability and precision thus remains a crucial research pursuit. Obtaining accurate estimates is critical for project success, but the complex nature of software development complicates this. As such, the quest for proper and reputable SEE continues to be an important open research problem [3]. In light of the continuous evolution of software development methodologies and technologies, enhancing effort estimation techniques is crucial to address challenges posed by changing requirements, team dynamics, and other influencing factors [4]. A 2017 survey by the Project Management Institute highlighted the impact of inaccurate effort estimation on software project success. The survey found that 69% of software projects achieved their goals and objectives [5]. Moreover, the survey also highlighted that a substantial portion faced challenges, with 43% exceeding initial budgets, 48% experiencing delivery delays, and 15% failing due to poor effort estimates [5]. These findings underscore the significance of precision in effort prediction for effective project execution and management. As methodologies and technologies rapidly advance, estimation techniques must continuously evolve to account for modern development practices and environments [6].

Within the literature on Software Effort Estimation (SEE), techniques are categorized into three main groups: algorithmic, non-algorithmic, and machine learning models [7] [8]. These diverse approaches are crucial in enhancing the accuracy and effectiveness of effort prediction, aiding project managers in making informed decisions for improved outcomes. Algorithmic techniques utilize statistical and mathematical formulations, encompassing widely used methods like function point analysis [9], COCOMO II [10], source lines of code [11], Putnam SLIM, and use case points [12]. These models employ equations and formulas to quantitatively estimate effort. In contrast, non-algorithmic techniques rely on subjective assessments of historical data and expert judgment [13]. Planning poker [14], wideband Delphi [15], work breakdown structures, and expert estimation fall under this category. These qualitative approaches leverage insights from previous projects and specialist knowledge. With the evolution of artificial intelligence, machine learning has emerged as the third category for software effort estimation. Machine learning applies algorithms to learn from data patterns and make predictive effort estimations without explicit programming. As each technique has its strengths and limitations, utilizing a combination of complementary approaches can improve estimation reliability, precision, and accuracy - crucial factors for successful project development and delivery [16].

ML techniques like ANN, case-based reasoning, SVR, DT, Bayesian networks, and genetic algorithms offer alternative approaches to effort estimation [17] [18]. These models leverage computational learning algorithms to predict estimates based on input data patterns without explicit programming. Each estimation technique category caters to diverse software project needs with its own strengths. However, selecting appropriate models is pivotal for accurate and reliable estimation to enable robust planning and resource management. Over decades, numerous software effort estimation methods have been put forward by experts and researchers. However, determining the most effective approach remains a key challenge [19]. Recently, ensemble estimation techniques have emerged as a potential solution by combining multiple methods to mitigate individual limitations and harness their collective strengths [20]. The extensive research conducted has yielded various estimation models aiming for greater accuracy. However, there remains a lack of consensus within the research community on a single definitive best approach. This highlights the need to critically compare and evaluate methods to identify the most suitable ones for different project contexts.

The mentioned machine learning algorithms often use publicly available datasets [21] [6] [22] to compare the performance of each other algorithms. The list of datasets that authors most frequently use is in Table 1. Besides, these ML algorithms use a performance evaluation matrix, which is described in Section 2. Exploiting machine learning in software effort estimation (SEE) is a pivotal research topic, with numerous researchers over the past decade employing various techniques and contributing enhanced solutions to the software community.

The global goal 9.4 (i.e., the sustainability of industry, innovation, and infrastructure) [23] talks about how all industries and infrastructures can be ameliorated to achieve sustainability by 2030. The level of sustainability should be reached by making better use of resources and using green technologies and manufacturing methods. To target this global goal, the software development industries are investing in R&D (research and development) to continuously improve their strategies to develop sustainable software that requires less energy to run and reduces the environmental impact of computing. Effort estimation is crucial in the software development process, which helps the team to develop and deliver the software on time. According to recent research, 30% to 60% of software or IT projects fail [24]. Almost all projects don't finish within the time or budget that was planned because of the wrong estimation of resources. Hence, it justifies the requirement to put more effort in the investment for R&D of software effort estimation process. The main point of this paper is software effort estimation that makes better use of resources to meet the global goal 9.4.

In our study, we specifically focus on the last four years, taking into account recent advancements, and acknowledge the existence of pertinent review papers within this timeframe [1], [2] [16] [25]–[27]. Besides, some review papers on software effort estimation using agile methodology [28]–[30]. Also, there are some comparative analyses available in the field of SEE by using various ML on publicly available datasets [31]–[33]. While numerous review papers and comparative analyses exist, there is a notable gap in the literature where no study concurrently offers both a comparative analysis and a comprehensive review. This research en-

deavors to contribute to the body of knowledge by critically examining and evaluating various software effort estimation approaches to offer insights that can improve precision and reliability. Through comprehensive analysis, this study aims to provide valuable perspectives to help refine estimation practices, ultimately driving improved project outcomes and customer satisfaction. The key contribution of the paper is the following.

- This research aims to conduct a systematic analysis of estimation methods to offer useful insights into their relative effectiveness and applicability. The goal is to provide recommended practices for selecting appropriate techniques, and guiding project managers towards improved effort prediction to support successful project execution and delivery. This paper presents a literature review of research published between 2020 and 2023 focusing on software effort estimation.
- This study offers a comparative examination of five machine learning algorithms (KNN, LR, RF, SVM, LASSO) using seven publicly available datasets which are Albrecht, Desharnais, China, Kemerer, Mayazaki94, Maxwell, and COCOMO.
- The performance of the machine learning algorithms employed in this study is evaluated using five common performance evaluation metrics, which are MAE, MSE, RMSE, R-square, and MAPE. The performance evaluation based on diverse metrics offers a multifaceted perspective on strengths, limitations, and suitability to guide appropriate method selection aligned with project characteristics.

The subsequent sections are organized as follows: Section 2 contains a review of the last 4 years' research articles between 2020 and 2024. Section 3 outlines the methodology of the comparative study, which includes the machine learning algorithm and their parameters, and datasets. Section 4 entails the presentation and discussion of results in the context of the comparative analysis paper. The conclusion and future works are discussed in section 5.

II. LITERATURE REVIEW

In recent times, there has been a notable increase in scholarly attention directed towards the field of software effort estimation. A variety of estimation methodologies have been proposed in the academic literature to determine the required work for different projects. In the field of software effort estimation systems, researchers have shown a major preference for machine learning techniques, while algorithmic models have been largely applied in other domains.

A. LITERATURE SEARCH METHOD

The primary studies were sourced from six reputable digital libraries, namely IEEE Xplore, ACM Digital Library, Science Direct, PubMed, and Google Scholar. These libraries are widely favored and commonly used within the effort estimation community. A specific search query string was employed for each database to identify relevant studies aligned with the research questions.

B. RESEARCH QUESTIONS

Aligned with the research objectives, our focus centers on addressing the following research questions:

- Which datasets are most commonly used in the literature of SEE?
- In terms of SEE methods, which ones are most often employed?
- In SEE research, which accuracy metrics are most often employed?
- Between 2020 and 2023, how many scholarly articles were published on SEE?

C. SEARCH STRING

The search string: ["software effort estimation" OR "software cost estimation" OR "software development effort estimation"] was selected for the core subject that is dealt with in this paper.

D. CRITERIA FOR INCLUSION AND EXCLUSION

Inclusion Criteria

- Conducted a study that utilized machine learning (ML) techniques to estimate software effort.
- Every single one of the papers that are authored in the English language.
- Scholarly works that have been published in an academic journal or at a conference.
- A publication that drew on publicly available datasets.

Exclusion Criteria

- Not only was there no semantic interaction but neither the title nor the abstract had anything to do with our search query.
- There was no overlap with the research topic, and the focused aim did not conflict in the slightest with the goals of the problems addressed in the RQs.
- Research article that does not rely on datasets that are publicly accessible.
- Paper published before the year 2020.

E. RESULT OF THE REVIEW

We utilized the advanced search feature in each library, inputting the search query into the designated advanced search section. We obtained a total of 230 papers from IEEE Explore, 103 papers from Google Scholar, 120 papers from PubMed, 42 papers from ScienceDirect, and 7 papers from the ACM digital library over the past four years. We have chosen 24 papers out of a total of 502 papers, using specific criteria for inclusion and removal. In table 2, we have shown the results of our selected 24 papers from 2020 to 2023. Figure 1 shows the number of datasets that have been used in previously publicly available datasets. The x-axis shows the previously publicly available datasets, and the y-axis shows the number of times each dataset has been used.

TABLE 1: Summary of the publicly available dataset

SL No	Dataset Name	Source	Quantity of Data	Number of Features	Attribute of Output: Effort	Size (measured in units)	Ref.
1	COCOMO81	Promise	63	18	Person-months	LOC	[34]
2	Desharnais	GitHub	81	12	Person-hours	Function point	[35]
3	Maxwell	Promise	62	27	Person-hours	Funciton points	[36]
4	Nasa93	Promise	93	17	Person-months	LOC	[37]
5	China	Promise	499	16	Person-hours	Function points	[38]
6	Tukutuku		53	9	Person-months		[39]
7	Miyazaki94	Zenodo	48	8/9	Person months	'KSLOC	[40]
8	Kitchenham	SEACRAFT	145	9	Person-hours	Function Points	[41]
9	ISBSG	ISBSG	1192	13	Man-hours	Multiple	[42]
10	Albrecht	Promise	24	8	Person-Months	Function point	[43]
11	Kemerer	Zenodo	15	7	Person-months	KSLOC	[44]
12	UCP	Promise	70	17	Person-months	LOC	[45]
13	Edusoft	Github	200	7	Person-months		[46]
14	Finnish	Not found	38	9	Person-Hours	Function Points	[47]

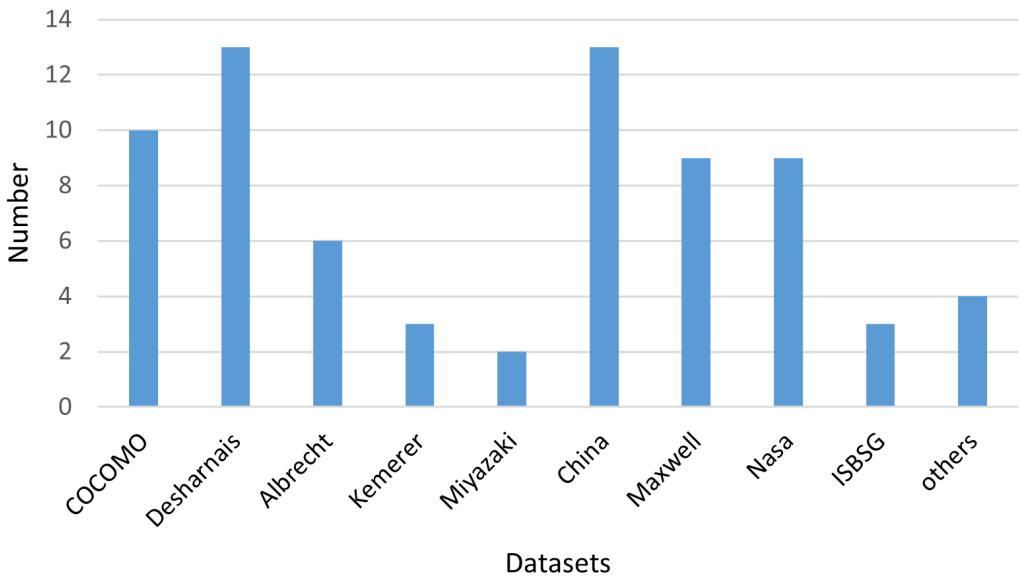


FIGURE 1: Publicly available used datasets

The review found that Desharnais and China datasets are mostly used datasets in the field of software effort estimation. Figure 2 shows the number of previous algorithms used in a study on SEE. The graph shows that the number of previous algorithms used has increased steadily over the past 4 years. Based on our findings, we can see in the figure that random forest and support vector regressor are mostly used algorithms in the field of SEE. Figure 3 shows the number of papers published on SEE from 2020 to 2023. The x-axis shows the year, and the y-axis shows the number of papers published. Based on our review, we can say that in 2022 most papers published in journals and conferences. Figure 4 depicts a performance evaluation matrix used by researchers in the field of SEE. Based on our analysis, we can say that MAE is the most used performance evaluation matrix in the field of SEE.

III. METHODOLOGY

A. MACHINE LEARNING TECHNIQUES

1) K-Nearest Neighbor

KNN is a supervised machine learning algorithm that is used to decide what to classify and what to predict. To use this non-parametric algorithm, find the k data points in the feature space that are closest to the input data point and use their labels to make predictions [68]. The KNN algorithm works by first defining a distance metric between the input data points in the feature space. The most common distance metric is the Euclidean distance, but other metrics such as the Manhattan distance and cosine distance can also be used [69]. KNN has a few hyperparameters that can be tuned to achieve optimal performance on a given dataset. These are the number of Neighbors based on the value of K, distance metric (Euclidean, Manhattan, and Minkowski), and weights (uniform and distance). It is particularly useful for problems where the decision boundaries are non-linear and where the data is low-dimensional [70].

TABLE 2: Literature review on SEE

Ref	Year	Datasets	Used Techniques	Evaluation Matrix
[48]	2023	Desharnais	ANN	MMRE, MRE and PRED
[18]	2023	Maxwell, Albrecht, NASA, Telecom, China, Kemerer, Desharnais	CBR-GA	MBRE, MAE, MIBRE
[49]	2023	Edusoft	SVR, KNN, DT	MSE, MAE, R-Square
[50]	2023	Kemerer, Albrecht, Miyazaki, China, Maxwell, COCOMO, NASA	FCNN, GWO-FC	RAE, MSE, MAE, RRSE, RMSE, R2, MdMRE
[51]	2022	Desharnais	LR, KNN, RF, SVM, XgBoost, AdaBoost	
[52]	2022	COCOCMO81, Deasharnais, Maxwell, China, and Miyazaki94	SVR ANN,	MMRE, MAR
[53]	2022	Maxwell, COCOMO81, China	RF, DTR, Ridge R, LR, LassoR,	MMRE, PRED (25),
[54]	2022	Cocomonasav1, Desharnais, COCOMONASA2	RF, LR, SVM, MP, Stacking, CART R, Vote GA-HSBA, FRSBM R, BHO-HSBA, FFA-HSBA	RMSE, MAE
[55]	2022	ISBSG	LinearSVR, GBRegr, M5P, and RFR	RMSE, MAE
[56]	2022	Maxwell, NASA93, Cocomo81, China	analogy-based SEE	MdMRE, SA, PRED, MMRE
[57]	2022	NASA93, UCP, ISBSG2021 , China	RF, SVR, Ridge Regression, KNN, Gradient Boosting Machines	MdAE, MSE, MAE
[58]	2022	CHINA, COCOMO81	KNN, SGD, DT, RFR, bagging regressor, gradient boosting regressor, Ada-boost regressor,	MSE, RMSE, MAE, and R2
[19]	2021	COCOMO81, NASA, Maxwell	SB, GWO, ACO, GA, CO, BAT, PSO	MMRE, MRE, MBRE, PRED, IBRE SA, MAE
[59]	2021	China, Albrecht, Maxwell, Desharnais	NN, Deepnet, SVM , RF	RMSE, MAE, MSE, R-Squared
[19]	2021	China, COCOMO-81, NASA	Deep-MNN, GWDNNSB	MBRE, MMRE, MRE, MI-BRE, PRED, SA, MR, MAE,
[31]	2021	China, Albrecht, Desharnais, Kitchenham, Kemerer, Maxwell, Cocomo8	SVM, RF, DT, Ridge, NN, LASSO, Deep-Net, ElisticNet, Bagging, Averaging, Stacking using RF, Boosting,	RMSE, MAE, R-squared
[60]	2021	Desharnais	LR, KNN, ML, SVM	RMSE, MAE, MSE
[61]	2020	China, Desharnais, Albrecht	GP (genetic programming)	Pred (25), MMRE,
[62]	2020	NASA(93,63,60)	FPA	MMRE
[63]	2020	Desharnais	RF, LR, Multi-layer perception	MAE, CC, RMSE, RSE, RRAE
[64]	2020	Desharnais, COCOMO81	SVM, KNN, RF, NN, and backpropagation	MMRE
[65]	2020	NASA 93	COCOMO-II	MRE, MMRE
[66]	2020	China, Desharnais, Albrecht	Genetic programming	MMRE, PRED (25)
[67]	2020	Cocomo81, Nasa93, China, Maxwell, Deasharnais, ISBSG	ABC	PRED, MMRE, SA

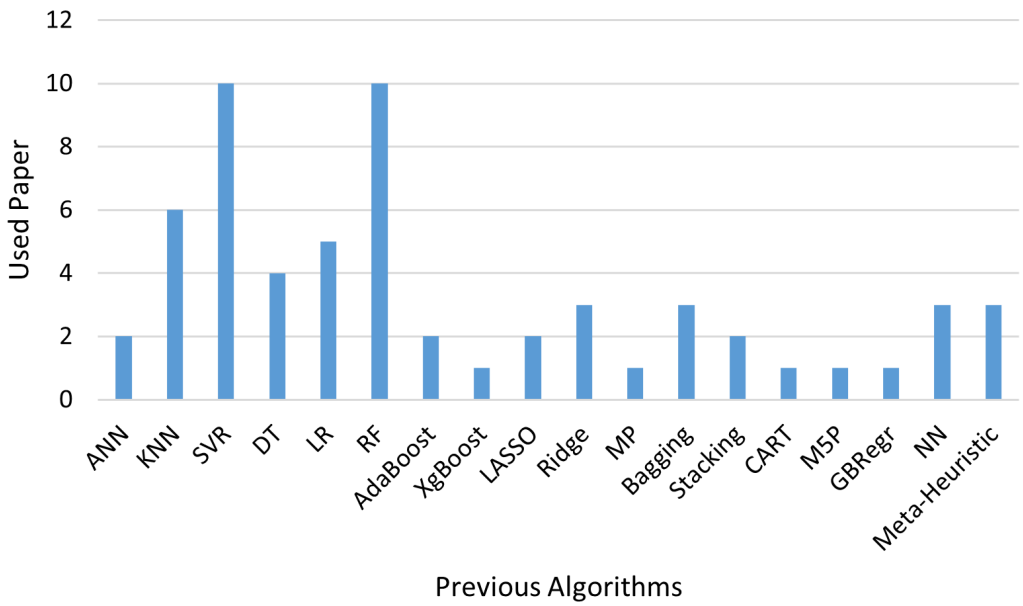


FIGURE 2: Commonly used algorithm by researchers

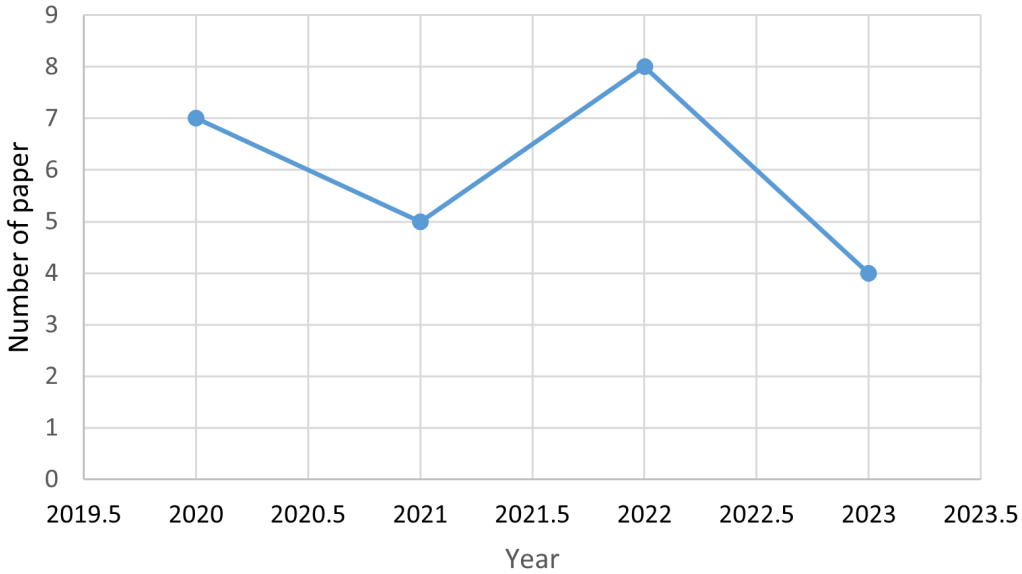


FIGURE 3: Year wise paper published by journal and conferences

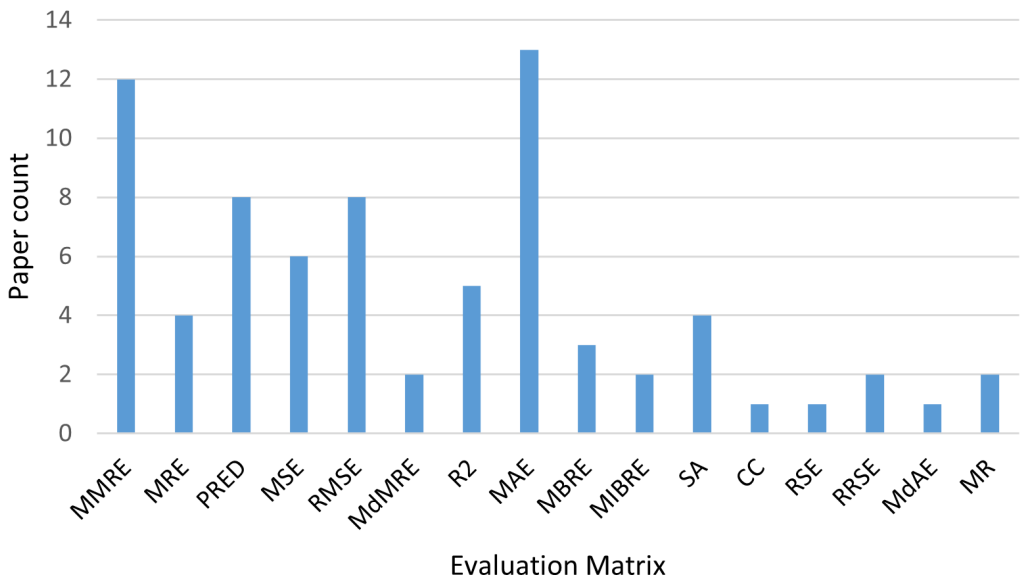


FIGURE 4: Performance evaluation matrix used by scholars

2) SVM

Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates the data into different classes. The hyperplane is defined as the decision boundary that maximizes the margin between the different classes [71]. SVM has several hyperparameters that can be tuned to achieve optimal performance on a given dataset. Some of the key hyperparameters are the Kernel function ('linear', 'poly', 'RBF', 'sigmoid', and 'precomputed'), the Regularization parameter, and the Gamma parameter (scale). It is beneficial for problems where the number of features is high relative to the number of samples, as it can handle large-dimensional datasets efficiently. SVM also works well in cases where the data is not linearly separable, as it can use nonlinear kernel functions to transform the data into a higher-dimensional space where a hyperplane can separate it [72].

3) RF

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. The algorithm works by building multiple decision trees on random

subsets of the input variables and data samples. The decision tree predictions are combined to make the final prediction [73]. During the prediction phase, each decision tree in the ensemble makes a prediction on the input data, and the predictions are combined using a majority voting scheme for classification tasks [74]. It has a few parameters like the number of trees (n_estimators: int, default=100), Maximum depth of trees (max_depth: int, default=None), Number of the input variable, and many more parameters. It is particularly useful for problems where the input variables have non-linear relationships with the target variable, and where the data contains noisy or missing values. Random Forest has been successfully applied in various domains, including finance, marketing, and bioinformatics.

4) LR

Linear regression is a statistical technique that models the connection between a dependent variable and one or more independent variables. The model assumes a linear relationship among the variables, represented by a straight line [75]. The equation of a linear regression model is $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$, where y is the dependent variable, $x_1, x_2, \ldots, x_n$ are the independent variables, and $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the regression coefficients representing the slope of the line. The goal of linear regression is to estimate the values of the regression coefficients to best fit the data and make predictions for the dependent variable based on the independent variables.

5) LASSO

LASSO (*Least Absolute Shrinkage and Selection Operator*) regression is a variant of linear regression that combines LASSO and Ridge regression [76]. It minimizes the sum of squared residuals while penalizing the absolute values of coefficients using the L1 norm (λ_1) and the squared values of coefficients using the L2 norm (λ_2). This leads to feature selection and coefficient shrinkage, making it suitable for high-dimensional datasets with correlated predictors. λ_1 and λ_2 are regularization parameters controlling the strength of the LASSO and Ridge penalties, respectively [56].

B. DATASETS DESCRIPTION

1) Albrecht Dataset (AD)

The AD consists of twenty-four software applications written in third-generation languages such as PL1, COBOL, etc. The dataset's definition consists of 6 individual number variables and one defenseless number attribute called "work hours," which represents the proper effort in 1000 hours. Database management languages were used for the remaining projects, and COBOL and PL1 were used for development. The AD is explained in full in Table 3.

2) Desharnais Dataset (DT)

The DT originated from the collection of 81 software projects obtained from Canadian software enterprises. The dataset consists of ten features, including two dependent variables (time and effort measured in 'person-hours') and eight independent variables. Regrettably, a total of four projects, out of the original 81, contained missing values. Consequently, we opted to exclude these projects from the estimating procedure, since their presence could have potentially impacted the accuracy of the results. After the data pre-processing step, a total of 77 software projects were completed. The DT is fully described in Table 4.

3) CHINA Dataset

There are nineteen attributes in the CHINA dataset used to predict software effort. In total, there are 499 distinct project instances. The descriptive statistics for the CHINA dataset are given in Table 5.

4) Kemerer Dataset

Thirteen software projects with a "man-month" unit of measurement are included in the Kemerer dataset. The projects are defined by six traits and one predictable property. Each of the six attributes is represented by two categories and four numerical attributes. The Kemerer dataset's complete description is given in Table 6.

5) Miyazaki94 Dataset

The Miyazaki94 dataset was contributed by Miyazaki. There are 48 software projects included in this compilation. There are nine traits altogether. One is an identifier, one is a decision attribute, and the remaining seven are conditional attributes. The complete description of the Miyazaki94 dataset may be found in Table 7.

6) Maxwell Dataset

The 62 projects in the Maxwell dataset, which was assembled from one of Finland's biggest commercial banks, are each characterized by 23 attributes. A detailed overview of the Maxwell dataset may be found in Table 8. The only numerical attribute is project size in function points.

7) COCOMO81 Dataset

The COCOMO'81 collection comprises 252 software projects, with the majority being scientific applications coded in Fortran [2, 10]. Every project consists of 13 qualities (see Table 9): the size of the program is quantified in KDSI (Kilo Delivered Source Instructions), while the other 12 attributes are assessed using a scale of six linguistic values: 'extremely low', 'low', 'nominal', 'high', 'very high', and 'extra high'. These 12 characteristics pertain to the software development environment, encompassing factors such as the expertise of the personnel engaged in the software project, the development methodology employed, and the constraints imposed by time and storage limitations.

C. EVALUATION MATRIX

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2 \tag{1}$$

TABLE 3: Description of Albrecht dataset [6]

SL No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	Input	the quantity of inputs a software must handle.	Integer	InquiryNumeric	40.25	36.913	7	193
2	Output	The quantity of outputs generated by software.	Integer	OutputNumeric	47.25	35.169	12	150
3	File	The quantity of files required for a program to write to or read from.	Integer		17.375	15.522	3	60
4	FPAdj	The raw function points are adjusted by the function point adjustment factor based on certain software attributes.	Double		0.989	0.135	0.75	1.2
5	Inquiry	how many queries or questions a software must respond to.	Integer		16.875	19.337	0	75
6	AdjFP	The Function Point Adjustment Factor is multiplied by the raw function points to determine the adjusted function points.	Integer	AdjfpNumeric	658.875	492.204	199	1902
7	RawFPcounts	The Function Point Metrics are used to calculate the raw function points.	Double	RawFPcounts	638.53	452.653	189.52	1902
8	Effort	Person-months are used to quantify the software development effort.	Double	Effort				

TABLE 4: Description of Desharnais dataset [26]

SL No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std dev	Min	Max
1	Project	Number of Project	Non-continuous					
2	ManagerExp	Years of project managers' experience	Non-continuous		2.531	1.644	-1.00	7.00
3	TeamExp	Years of team experience	Non-continuous		2.185	1.415	-1.00	4.00
4	YearEnd	Completion year	Non-continuous		85.741	1.222	82.00	88.00
5	Effort	assessed in person	Continuous	Effort	5046.309	4418.767	546.00	23940.0
6	Length	Duration of the Project	Continuous		11.667	7.425	1.00	39.00
7	Transaction	Quantity of transactions completed	Continuous	Transactions	182.123	144.035	9.00	886.00
8	PointsNon Adjust	Function points without adjustments	Continuous	PointsNon Adjust	304.457	180.210	73.00	1127.00
9	Entities	The quantity of entities	Continuous		122.333	84.882	7.00	387.00
10	PointsAjust	Function points for adjustments	Continuous	PointsAjust	289.235	185.761	62.00	1116.00
11	Adjustment	Factor of adjustment	Continuous		27.630	10.592	5.00	52.00
12	Language	Programming language	Categorical					

TABLE 5: China dataset description [26]

SL no	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	ID		Numerical		250	144	1	499
2	AFP	Function Points(FP) adjustments	Integer	AFP	487	1059	9	17518
3	Input	input of FP	Integer	Output	167	486	0	9404
4	Output	External output of FP	Integer	File	114	221	0	2455
5	Enquiry	FP of external output enquiry	Integer	Interface	62	105	0	952
6	File	FP of internal logical files	Integer	Added	91	210	0	2955
7	Added	FP for additional functions	Integer	NPDR_AFP	260	830	0	13580
8	Interface	Added FP to the external interface	Integer	PDR_AFP	24	85	0	1572
9	Deleted		Integer	N-Effort	12	124	0	2657
10	Changed	FP of modified functions	Integer	NPDU_UFP	85	291	0	5193
11	PDR_UFP	Delivery rate of productivity (unadjusted FP)	Double		13	14	0.4	101
12	PDR_AFP	Delivery rate of productivity (adjusted FP)	Double	Effort	12	12	0.3	83.8
13	NPDU_UFP	Delivery rate of productivity (unadjusted FP)	Double		1	1	1	4
14	NPDR_AFP	Delivery rate for normalized productivity (adjusted FP)	Double		14	15	0.4	108
15	Dev.Type		Numerical Only {0}		0	0	0	0
16	Resource	Type of team	Discrete		12	12	0.3	83.8
17	N_effort	Normalized effort	integer		4278	7071	31	54620
18	Duration	Time spent on the project overall	Integer		9	7	1	84
19	Effort	An overview of the work report	Integer		3921	6481	26	54260

TABLE 6: Description of Kemerer dataset [6]

Sl No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	ID	The project's identity.						
2	Language	The language of programming utilized in the project.	Non-continuous					
3	Duration	The project's duration expressed in months.	Numerical	Duration	14.26667	7.544787	5	31
4	Hardware	The type of hardware utilized in the project.	Non-continuous		2.333333	1.676163	1	6
5	AdjFP	Function points were modified.	Numerical	AdjFP	999.14	589.5921	99.9	2306.8
6	KSLOC	The project is expected to have thousands of source lines of code.	Numerical	KSLOC	186.5733	136.8174	39	450
7	EffortMM	calculated in person-months for effort		Efforts	219.2479	236.0554	23.2	1107.31
8	RAWFP	Function points not adjusted.	Numerical	RawFP	993.8667	597.4261	97	2284

TABLE 7: Description of Miyazaki94 dataset [26]

Sl No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	ID							
2	KSLOC	The number of lines of code in COBOL, not counting comments	Continuous	KLOC	70.792	87.5678	6.9	417.6
3	FORM	Quantity of distinct (report) forms	Non-continuous	FROM	22.38	20.55	0	91
4	SCRN	Quantity of distinct output or input screens	Non-continuous	SCRN	33.39	47.27	0	281
5	ESCRN	The total number of info points on all the screens	Non-continuous		525.60	626.058	0	3000
6	FILE	The quantity of distinct record formats	Non-continuous		34.81	53.36	2	370
7	EFILE	Total amount of data elements across all files	Non-continuous		1854.58	6398.605	57	45000
8	EFORM	The aggregate quantity of data elements across all forms	Non-continuous		460.67	396.816	0	1566
9	MM	System formed by Man-Months	Continuous	Man Month	87.475	228.7597	5.6	1586.0

MSE is another metric for measuring the error between actual values () and predicted values (). It calculates the squared differences between actual and predicted values and then takes the average of these squared differences. Squaring the errors amplifies larger errors and is commonly used in optimization problems and statistical analysis.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|Y_i - \hat{Y}_i| \tag{2}$$

MAE is a metric used to measure the average absolute difference between the actual values () and the predicted values (). It computes the absolute value of the residuals (the differences) between actual and predicted values, and then takes the average of these absolute differences. MAE is useful because it gives an idea of how far off the predictions are from the actual values, without considering the direction of the errors.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{3}$$

RMSE is a modification of MSE that calculates the square root of the average squared differences between actual and predicted values. RMSE is often preferred when you want to express the error in the same units as the original data, making it more interpretable.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=1}^{N}(Y_i - \bar{Y})^2} \tag{4}$$

R-squared is a measure of the goodness-of-fit of a regression model. It compares the variance explained by the model to the total variance in the data. A higher R-squared value (closer to 1) indicates that the model explains a larger proportion of the variance in the data.

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Y_i - \hat{Y}_i}{Y_i}\right| \times 100\% \tag{5}$$

MAPE is a metric used for forecasting accuracy. It calculates the percentage difference between actual values () and predicted values () and then takes the average of these percentages. MAPE expresses errors as a percentage of the actual values, making it easy to understand in practical terms.

D. PARAMETER VALUES

Figure 5 shows the parameter values of the employed machine learning algorithms (KNN, SVR, RF, Linear Regression, DT, LASSO) in this study.

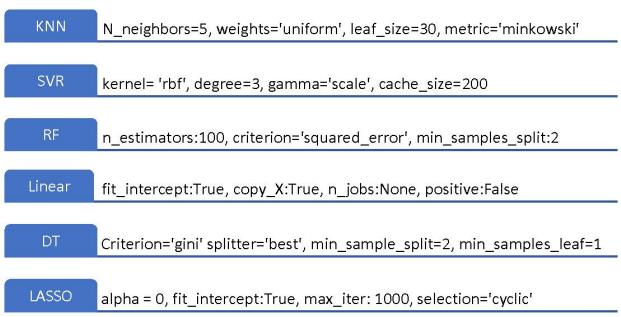


FIGURE 5: Parameter settings of the compared ML algorithms

TABLE 8: Description of Maxwell dataset [77]

Sl No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	year	The project's inception year	Continuous	Year				
2	App	The application's name that is currently being developed	Non-continuous		2.35	0.99	1	5
3	Har	The platform of hardware that the application is being created for	Non-continuous		2.61	1	1	5
4	Dba	The project's information management system	Non-continuous		1.03	0.44	0	4
5	Ifc	The technology employed for user interface	Non-continuous		1.94	0.25	1	2
6	Source	The utilized source code management system	Non-continuous	Source	1.87	0.34	1	2
7	Telonuse	Whether or not the project is developing legacy mainframe applications with IBM Telon	Binary		2.55	1.02	1	4
8	Nlan	The quantity of programming languages implemented within the undertaking	Non-continuous	Nlan	0.24	0.43	0	1
9	T01	Client involvement	Non-continuous		3.05	1	1	5
10	T02	Sufficient development environment	Non-continuous		3.05	0.71	1	5
11	T03	Employee accessibility	Non-continuous		3.03	0.89	2	5
12	T04	Utilizing standards	Non-continuous		3.19	0.70	2	5
13	T05	used Methods	Non-continuous	T05	3.05	0.71	1	5
14	T06	Utilizing tools	Non-continuous		2.90	0.69	1	4
15	T07	The logical complexity of software	Non-continuous		3.24	0.90	1	5
16	T08	specifications volatility	Non-continuous		3.81	0.96	2	5
17	T09	Standards for quality	Non-continuous	T09	4.06	0.74	2	5
18	T10	The requirements for efficiency	Non-continuous		3.61	0.89	2	5
19	T11	prerequisites for installation	Non-continuous		3.42	0.98	2	5
20	T12	Skills for staff analysis	Non-continuous		3.82	0.69	2	5
21	T13	Employee application expertise	Non-continuous		3.06	0.96	1	5
22	T14	Employee teamwork abilities	Non-continuous		3.26	1.01	1	5
23	T15	Employee teamwork abilities	Non-continuous	T15	3.34	0.75	1	5
24	Size	The project's size measured in lines of code	Continuous	Size	673.31	784.08	48	3643
25	Duration	The project's length expressed in months	Continuous	Duration	17.21	10.65	4	54
26	Effort	The entire work put into the project, measured in person-months	Continuous	Effort	8223.21	10499.9	583	63694
27	Time	The total number of person-months that were spent on the project	Non-continuous	Time	5.58	2.13	1	9

TABLE 9: Description of the COCOMO81 dataset [31]

SL No	Features	Information regarding the features	Types of Data	Selection of the feature	Mean	Std Dev	Min	Max
1	Data	Size of database	Float	Data	1.004	0.073	0.94	1.16
2	Rely	Requirements for software dependability	Float	Rely	1.036	0.193	0.75	1.4
3	Cplx	Complexity of the Product	Float		1.091	0.203	0.7	1.65
4	Time	Time limit for execution	Float	Time	1.114	0.162	1	1.66
5	Stor	primary storage limitation	Float	Stor	1.144	0.179	1	1.56
6	Virt	volatility of virtual machines	Float		1.008	0.121	0.87	1.3
7	Acap	Capability of analysts	Float	Acap	0.905	0.152	0.71	1.46
8	Turn	Turnaround time required	Float		0.972	0.081	0.87	1.15
9	Pcap	Skills of programmers	Float		0.937	0.167	0.7	1.42
10	Aexp	application-related experience	Float		0.949	0.119	0.82	1.29
11	Vexp	Work with virtual machines	Float		1.005	0.093	0.9	1.21
12	Lexp	Fluency in a programming language	Float		1.001	0.052	0.95	1.14
13	Modp	Applying state-of-the-art programming techniques	Float	Modp	1.004	0.131	0.82	1.24
14	Sced	Timeline for necessary development	Float	Sced	1.049	0.076	1	1.23
15	Tool	The application of software	Float		1.017	0.086	0.83	1.24
16	Effort	Physical exertion measured in person-months	Float	Effort	683.321	1821.582	5.9	11400
17	Loc	Lines of code	Float	Loc	77.21	168.509	1.98	1150

IV. RESULT AND DISCUSSION

Table 10 presents the performance evaluation of the Albrecht dataset using five different machine learning algorithms. The results show that the Linear Regression (LR) and LASSO models show good performance with minimal errors, exhibiting excellent predictive accuracy. In contrast, the KNN model displayed the highest errors and the lowest R-square, indicating relatively poor performance compared to the other algorithms. Figure 6 shows the result of the actual vs. predicted value using different machine learning algorithms on the Albrecht dataset.

TABLE 10: Performance evaluation of Albrecht dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
Albrecht_KNN	18.26	937.24	30.61	0.10	184.36
Albrecht_LR	0.00	0.00	0.00	1.00	0.00
Albrecht_RF	7.86	151.66	12.31	0.86	108.27
Albrecht_SVM	0.32	0.22	0.47	1.00	1.67
Albrecht_LASSO	0.04	0.00	0.05	1.00	0.47

TABLE 11: Performance evaluation of the Desharnais dataset

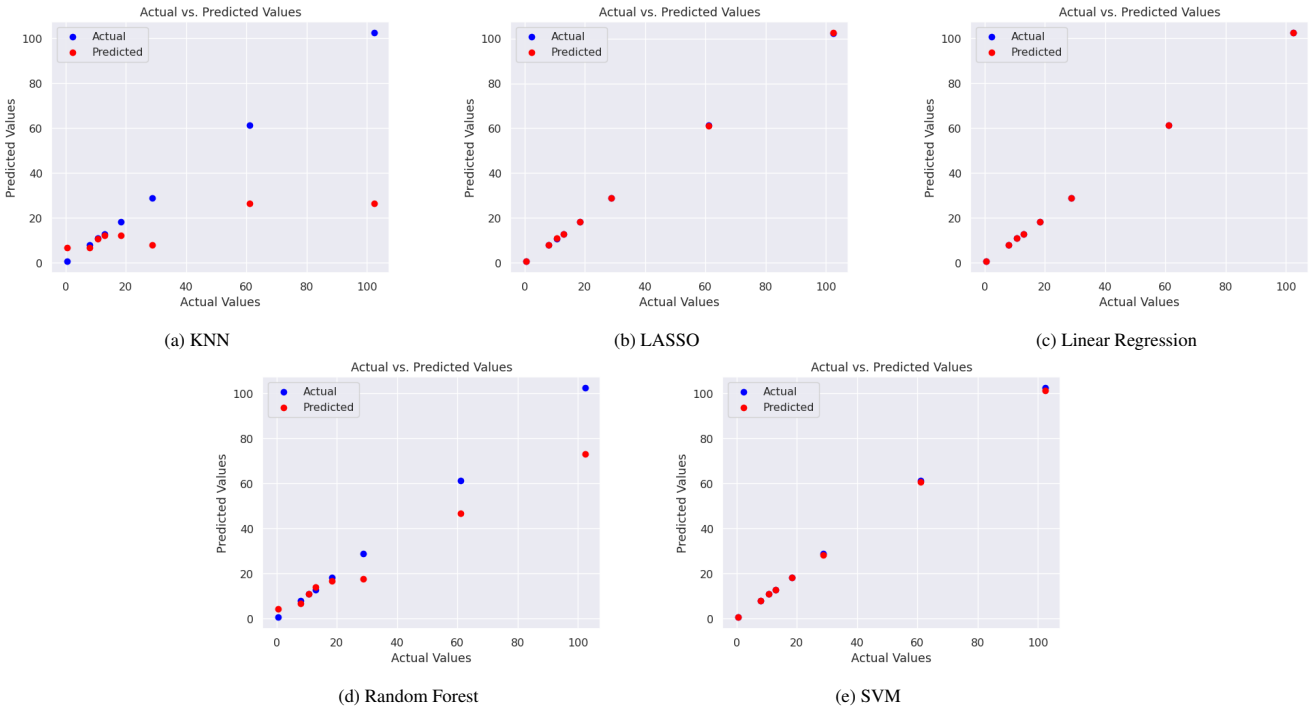
Dataset	MAE	MSE	RMSE	R-square	MAPE
Desharnais_KNN	1358.89	3254903.30	1804.14	0.71	72.85
Desharnais _LR	2124.31	7242009.89	2691.10	0.36	61.83
Desharnais_RF	2591.78	12518906.12	3538.21	-0.11	86.96
Desharnais_SVM	2125.44	7298069.06	2701.49	0.36	64.57
Desharnais_LASSO	2124.30	7241927.21	2691.08	0.36	61.83

Table 11 presents the performance evaluation of the Desharnais dataset. The results indicate that the KNN model achieved the lowest MAE and RMSE values, suggesting relatively better predictive accuracy compared to other models. However, all models exhibit lower R-squared values, indicating that the predictive performance is suboptimal, with RF even showing a negative R-square. The choice of the best model would depend on the specific objectives and trade-offs between different performance metrics in this context. Figure 7 shows the result of the actual vs predicted value using different machine learning algorithms on the Desharnais dataset.

TABLE 12: Performance evaluation of China dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
China_KNN	208.14	598766.54	773.80	773.80	6.51
China _LR	0.00	0.00	0.00	1.00	0.00
China_RF	236.25	450709.49	671.35	0.99	9.08
China _SVM	0.06	0.01	0.08	1.00	0.01
China_LASSO	0.46	2.42	1.56	1.00	0.02

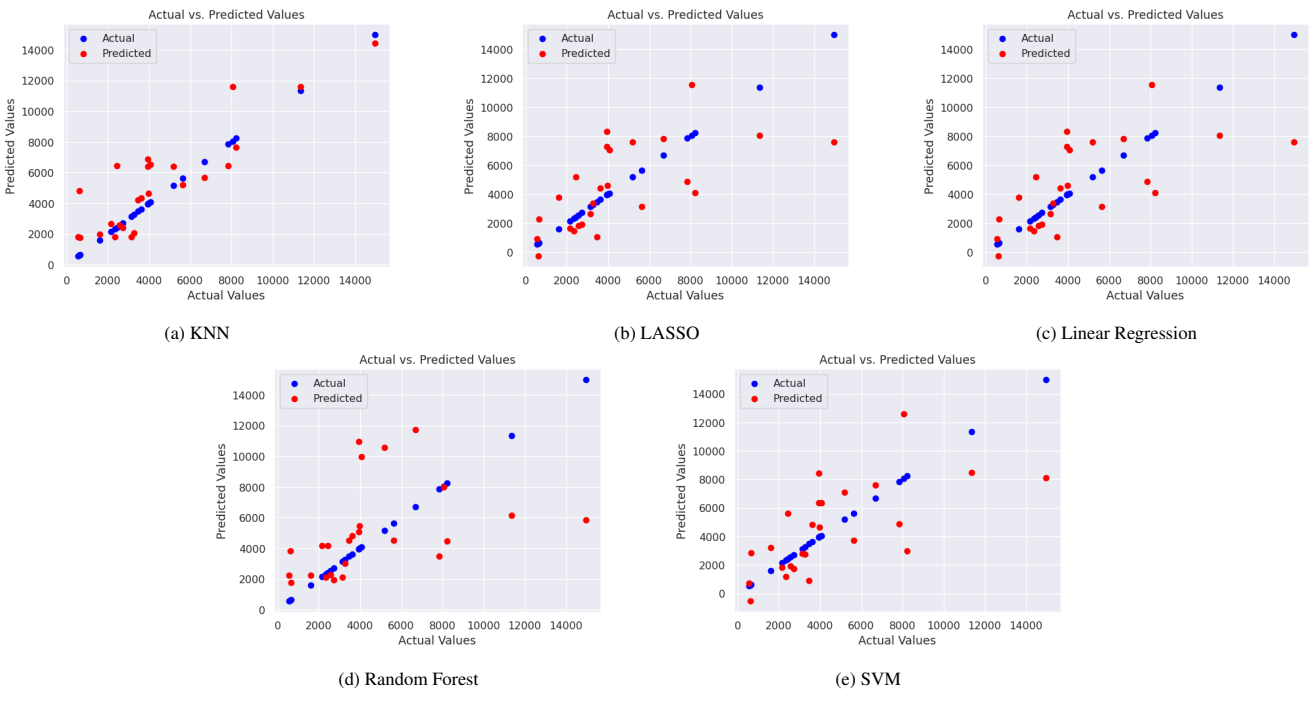
Table 12 displays the performance evaluation of the China dataset. Notably, the LR (Linear Regression) model achieved good scores in MAE, MSE, RMSE, and R-square, indicating an ideal fit to the data with no prediction errors. In contrast, the RF (Random Forest) model showed a relatively high RMSE and MAPE, suggesting that its predictions had notable deviations from the actual values. The choice of the most suitable model would depend on the specific objectives and the importance of various performance metrics in the context of the China dataset. Figure 8 shows the result of the actual vs predicted value using different machine learning algorithms on the China dataset,



(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 6: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for Albrecht Dataset

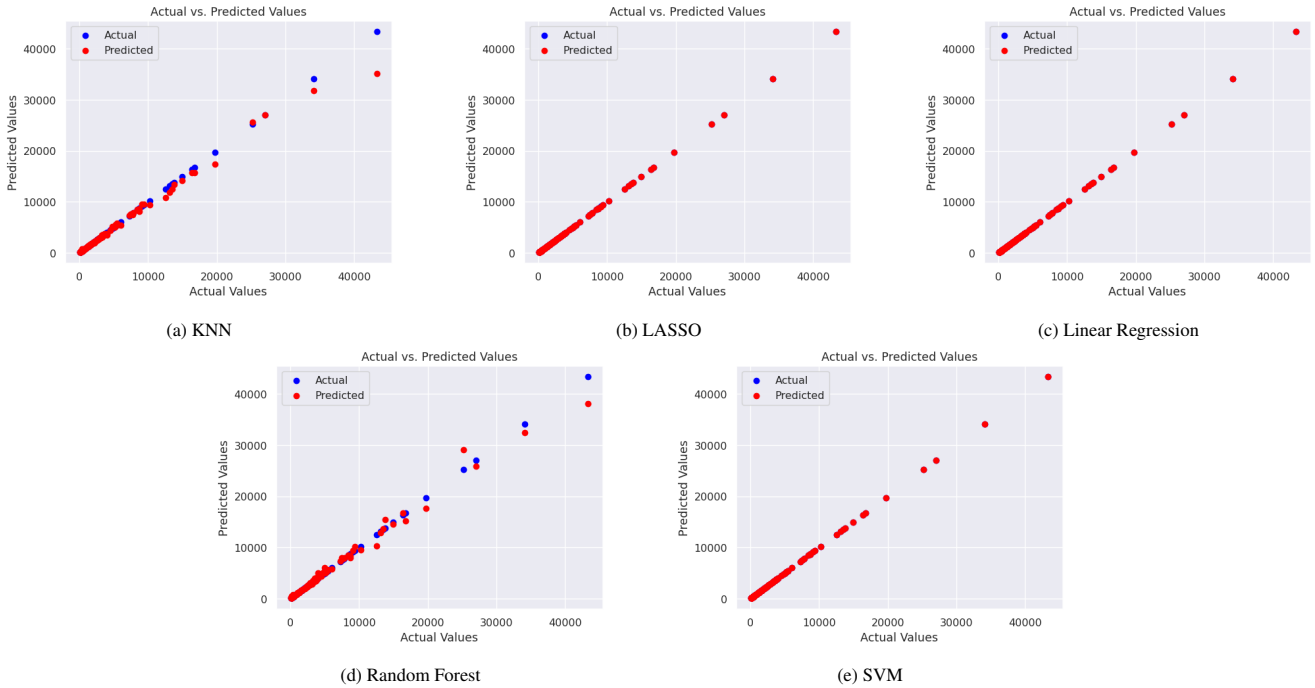


(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 7: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for Desharnais Datasets

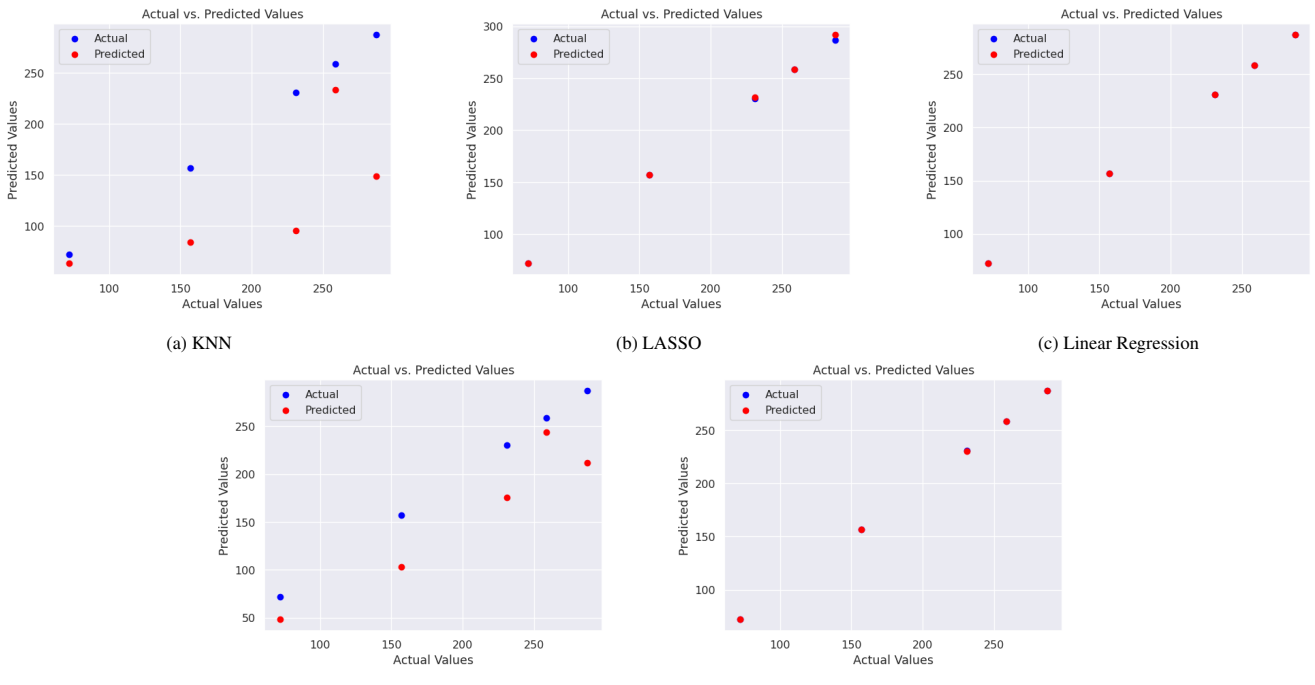
Table 13 presents the performance evaluation of the Kemerer dataset. Notably, the LR (Linear Regression) model achieved better scores in MAE, MSE, RMSE, and R-square, indicating an excellent fit to the data with no prediction errors. In contrast, the KNN model had a high RMSE and a negative R-square, suggesting that its predictions deviated significantly from the actual values. The choice of the most suitable model for the Kemerer dataset would depend on specific objectives and priorities regarding different performance metrics. Figure 9 shows the result of the actual vs predicted value using different Machine learning algorithms on the Kemerer Dataset,



(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 8: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for China Dataset

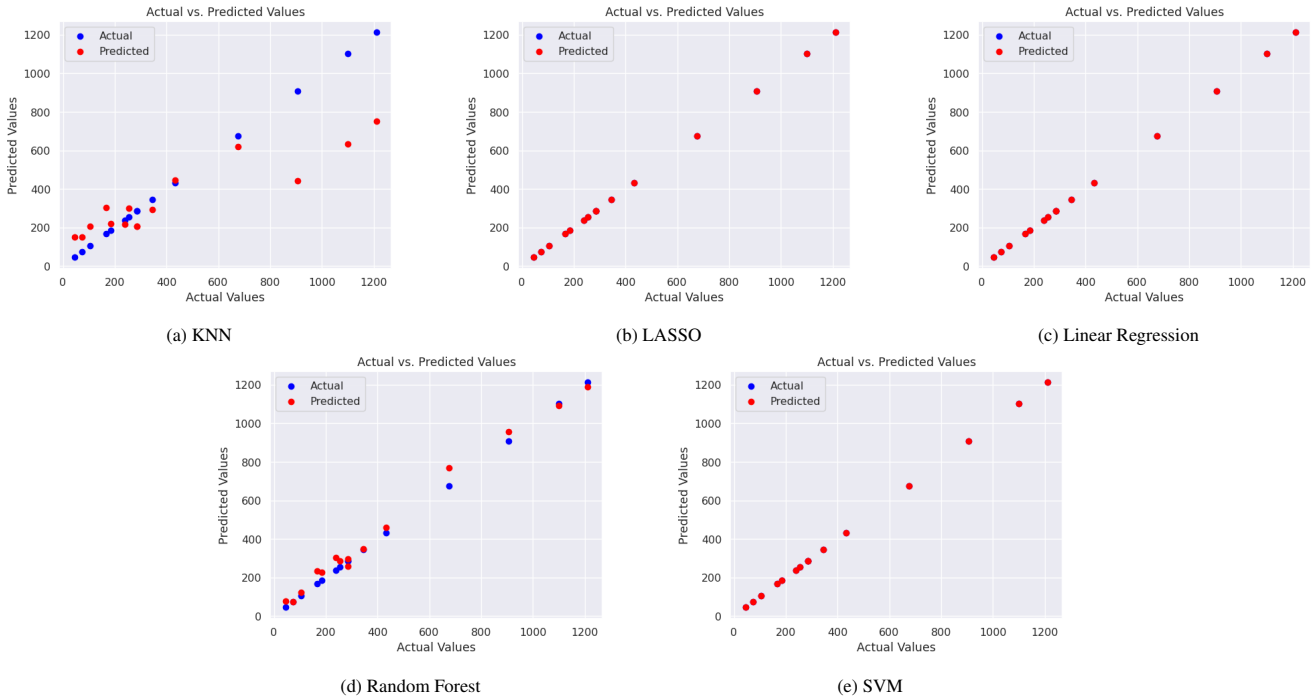


(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 9: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for Kemerer Dataset

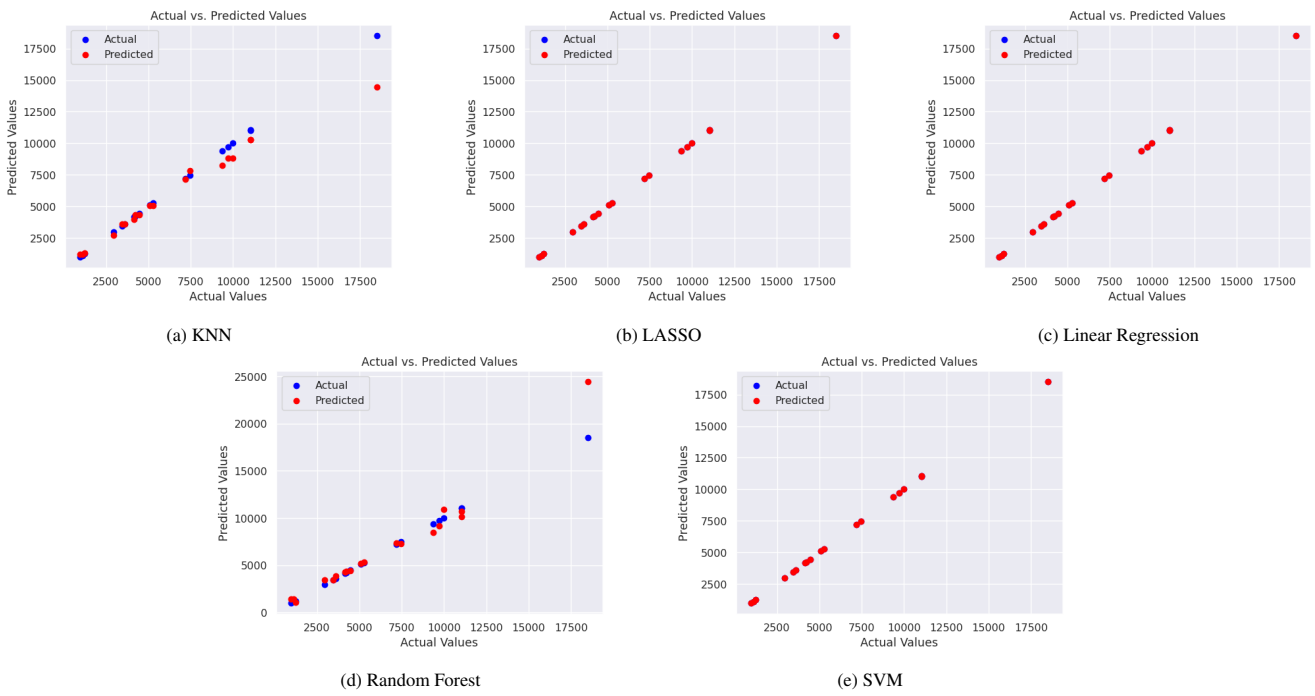
Table 14 provides a performance evaluation of the Mayazaki94 dataset. The LR (Linear Regression) model achieved good scores in MAE, MSE, RMSE, and R-square, indicating a flawless fit to the data with no prediction errors. On the other hand, the KNN model exhibited a relatively higher RMSE and lower R-square, suggesting that its predictions had notable deviations from the actual values. The choice of the most suitable model for the Mayazaki94 dataset would depend on specific objectives and priorities for different performance metrics. Figure 10 shows the result of the actual vs predicted value using different Machine learning algorithms on the Miyazaki94 dataset.



(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 10: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for miyazaki94 Dataset

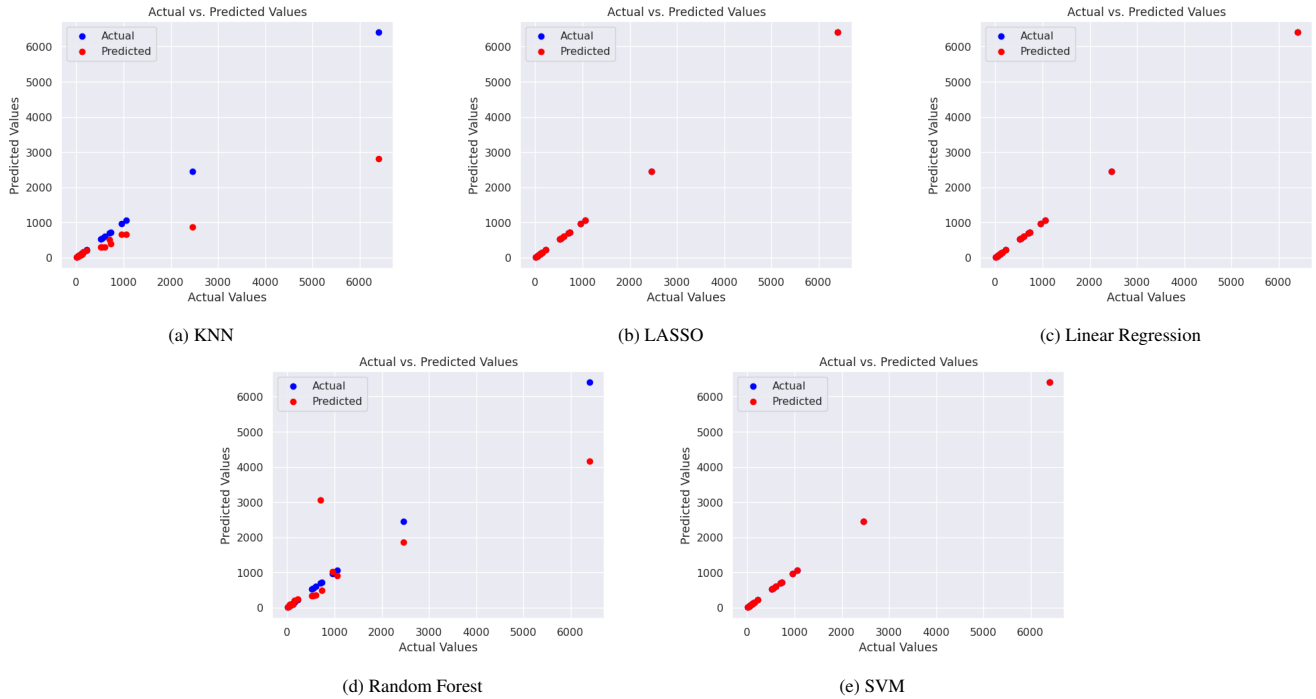


(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 11: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for Maxwell Dataset

Table 15 provides a performance evaluation of the Maxwell dataset. Notably, the LR (Linear Regression) model achieved good scores in MAE, MSE, RMSE, and R-square, indicating an excellent fit to the data with no prediction errors. In contrast, the RF (Random Forest) model showed a relatively higher RMSE and lower R-square, suggesting that its predictions had notable deviations from the actual values. The choice of the most suitable model for the Maxwell dataset would depend on specific objectives and priorities concerning different performance metrics. Figure 11 shows the result of the actual vs predicted value using different Machine learning algorithms on the Maxwell dataset,



(a) KNN (b) LASSO (c) Linear Regression

(d) Random Forest (e) SVM

FIGURE 12: Predictive Performance Comparison Across Multiple Machine Learning Algorithms for Cocomo81 Dataset

TABLE 13: Performance evaluation of the Kemerer dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
Kemerer_KNN	75.98	8650.96	93.01	-0.43	34.97
Kemerer _LR	0.00	0.00	0.00	1.00	0.00
Kemerer_RF	44.62	2480.90	49.81	0.59	24.74
Kemerer_SVM	0.05	0.00	0.07	1.00	0.05
Kemerer_LASSO	1.32	4.69	2.17	1.00	0.57

TABLE 14: Performance evaluation of Mayazaki94 dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
Miyazaki_KNN	146.36	47625.61	218.23	0.64	52.05
Miyazaki_LR	0.00	0.00	0.00	1.00	0.00
Miyazaki_RF	33.05	1756.79	41.91	0.99	15.50
Miyazaki_SVM	0.08	0.01	0.09	1.00	0.04
Miyazaki_LASSO	0.01	0.00	0.01	1.00	0.00

TABLE 15: Performance evaluation of Maxwell dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
Maxwell_KNN	558.73	1120299.02	1058.44	0.94	7.18
Maxwell_LR	0.00	0.00	0.00	1.00	0.00
Maxwell _RF	629.21	2023548.15	1422.51	0.89	10.19
Maxwell _SVM	0.05	0.00	0.06	1.00	0.00
Maxwell_LASSO	0.04	0.00	0.06	1.00	0.00

TABLE 16: Performance evaluation of Cocomo81 dataset

Dataset	MAE	MSE	RMSE	R-square	MAPE
Cocomo_KNN	380.40	842748.51	918.01	0.59	29.38
Cocomo_LR	0.00	0.00	0.00	1.00	0.00
Cocomo_RF	342.96	585815.68	765.39	0.72	50.32
Cocomo_SVM	0.09	0.01	0.09	1.00	0.16
Cocomo_LASSO	0.00	0.00	0.00	1.00	0.00

The Table 16 presents a performance evaluation of the Cocomo81 dataset. Remarkably, the LR (Linear Regression) model achieved good scores in MAE, MSE, RMSE, and R-square, indicating an ideal fit to the data with no prediction errors. In contrast, the KNN model displayed a relatively higher RMSE and lower R-square, suggesting that its predictions had notable deviations from the actual values. The choice of the most suitable model for the Cocomo81 dataset would depend on specific objectives and priorities regarding different performance metrics. Figure 12 shows the result of the actual vs predicted value using different machine learning algorithms on the cocomo81 dataset.

A. DISCUSSION

1) What are the most frequent datasets in the literature of SEE? (Research Question 1)

About fifteen distinct datasets were used in the research that were chosen. We search for datasets that are utilized in a minimum of one study. A review of the literature is displayed in table 2. One of the most widely used datasets in the SEE literature is that of China and Desharnais, which has been used in several studies.

2) What are the most frequently used SEE techniques? (Research Question 2)

SEE is the practice of projecting the amount of money, time, and resources required to complete a software development project. SEE represents a specific type of regression problem. Regression utilizes input data, including project size, complexity assessments, and past data, to predict a continuous numerical value. In this particular case, it aims to estimate the quantity of work needed. The input characteristics could comprise a range of project criteria and variables that impact

the amount of effort required for development. The following machine learning methods were used, either by themselves or in conjunction with other (ML and nonML) estimation methods, to estimate the SEE. SVM, Bayesian networks (BN), kNNs, ANNs, decision trees (DT), Genetic Programming (GP), CBR, Random forests (RF), and classification and regression trees (CART) are some examples of artificial neural networks. Our study indicates that the most commonly used ML techniques in SEE literature are SVR and RF.

3) Performance Evaluation Matrix (Research Question 3)

Evaluation metrics are utilized in regression issues to assess the performance of a model of prediction that seeks to estimate a continuous destination variable. The literature on software effort estimating has several metrics that have been developed and applied to evaluate a prediction model's accuracy. Primarily, these measures rely on prediction error, sometimes referred to as absolute error, which quantifies the difference between the anticipated and observed values [78]. Our analysis indicates that MAE (Mean Absolute Error) and MMRE (Mean Magnitude Relative Error) are the most often employed performance assessment metrics in the literature on SE), relying on historical data.

4) How many research papers are published on SEE between 2020-2023 (Research Question 4)

Our review indicates that there are a total of 24 papers within the specified period. After considering our inclusion and exclusion criteria, we have narrowed down the selection from the numerous published papers. Figure 4 presents a concise overview of the article we have chosen.

B. THREATS TO VALIDITY

An empirical study's validity may be compromised by a variety of reasons. We tried to tailor the search string to our research questions and used it to find the appropriate research for our systematic literature review. However, since some studies (rarely) failed to include crucial keywords in their study title, abstract, or keywords, it's possible that we overlooked some pertinent research. Even though we made every effort to prevent this scenario by consulting each study's bibliography to choose all the pertinent research, there's still a chance we might have overlooked some significant studies that pose a hazard. The dangers to the internal, construct, conclusion, and external validity of our study are covered in this part along with the steps that have been taken to mitigate them.

1) Internal Validity

The primary risk to the internal validity of the data under study is selection bias. Four datasets with various model-based techniques have been employed. The NASA93 dataset is LOC-based, the China datasets are FP-based, and a third dataset follows the UCP technique. The dimensions, application scope, size, and complexity of these databases all differ. Therefore, internal danger is not a significant worry for this study.

2) Construct Validity

Regarding construct validity, one potential hazard is the presence of verification bias. To mitigate this issue during the process of empirical analysis, we have employed a diverse range of impartial error and accuracy metrics. Each of these metrics addresses a unique and specific component of performance evaluation.

3) Conclusion Validity

The degree of variability of the outcomes under various experimental conditions is correlated with the validity of the conclusions. In this study, we have worked on sensitivity analysis to address this issue. Every analysis has been performed again with various cross-validation strategies. Furthermore, we used standard data splits to conduct the trials on separate datasets for each of the five ML-based SEE techniques. We have sought to exclude the potential that randomness could lead to performance enhancement.

4) External Validity

The main possible risk to external validity is associated with the generalization of results. All the experiments have been conducted using a wide range of datasets, including cross-company and within-company data. Consequently, we think that this study's findings will aid in the generalization of the results for both homogeneous and heterogeneous datasets of various sizes and domains.

V. CONCLUSION

In this paper, we contribute to the domain of software effort estimation, with a specific focus on machine learning techniques. Our research presents two key contributions. First, we conduct a comprehensive literature review investigating state-of-the-art effort estimation using machine learning, following established protocols. By analyzing primary studies, we gain valuable insights into the most effective approaches. Second, we perform a comparative evaluation of various machine learning models on seven well-known datasets, assessing accuracy using metrics like MAE, MSE, RMSE, R-square, and MAPE. Our research provides a solid foundation for further exploration into software effort estimation. The outcomes offer practical guidance for practitioners in selecting appropriate techniques for future projects, enhancing planning and resource allocation for improved processes and outcomes. In conclusion, our work significantly contributes to the field, benefitting both the research community and software development practitioners. Through meticulous investigation and comparative evaluation, this study elucidates the strengths and limitations inherent in machine learning approaches for effort estimation. The findings establish a foundation for future advancements in the field. This paper offers diverse perspectives for future investigation into agile

development effort prediction. One area of interest is studying the influence of cost factors on model accuracy. However, software effort estimation is crucial for sustainable software development Which is aligned to the global goal 9.4.

Future work could explore enhancing effort estimation precision using different machine-learning techniques and evaluation metrics. Additionally, investigating improved estimation through homogeneous or heterogeneous ensemble models combining algorithmic, expert, and machine learning approaches would be valuable. Incorporating human experts' context-specific knowledge not captured by algorithms, especially for emerging technologies or new domains, could prove beneficial. Further research can focus on utilizing combinations of complementary estimation techniques involving expert estimation to create hybrid frameworks, aiming to propose an ensemble model that harnesses algorithmic, expert, and machine learning strengths for improved accuracy. The resulting instrument could serve software firms and practitioners for early-stage estimation of new projects. In summary, this research contributes significantly to software effort prediction, benefitting researchers and practitioners. The literature analysis and comparative assessment provide diverse insights into machine learning techniques used for SEE, paving the way for future enhancements. The findings aim to guide improved effort estimation practices for successful project planning and execution.

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] 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.

[2] C. Eduardo Carbonera, K. Farias, and V. Bischoff, "Software development effort estimation: A systematic mapping study," IET Software, vol. 14, no. 4, pp. 328–344, 2020.

[3] J. A. Khan, S. U. R. Khan, J. Iqbal, and I. U. Rehman, "Empirical investigation about the factors affecting the cost estimation in global software development context," IEEE Access, vol. 9, pp. 22 274–22 294, 2021.

[4] Meharunnisa, M. Saqlain, A. M, M. Awais, and Å. Stevic, "Analysis of software effort estimation by machine learning techniques," IngÃl'nierie des systÃĺmes d information, vol. 28, pp. 1445–1457, 12 2023.

[5] S. S. Ali, J. Ren, K. Zhang, J. Wu, and C. Liu, "Heterogeneous ensemble model to optimize software effort estimation accuracy," IEEE Access, vol. 11, pp. 27 759–27 792, 2023.

[6] A. Baghel, M. Rathod, and P. Singh, "Software effort estimation using parameter tuned models," arXiv preprint arXiv:2009.01660, 2020.

[7] V. Van Hai, H. L. T. K. Nhung, Z. Prokopova, R. Silhavy, and P. Silhavy, "Toward improving the efficiency of software development effort estimation via clustering analysis," IEEE Access, vol. 10, pp. 83 249–83 264, 2022.

[8] M. Rahman, T. Goncalves, and H. Sarwar, "Review of existing datasets used for software effort estimation," International Journal of Advanced Computer Science and Applications, vol. 14, no. 7, 2023.

[9] A. Z. Abualkishik and L. Lavazza, "Ifpug function points to cosmic function points convertibility: A fine-grained statistical approach," Information and Software Technology, vol. 97, pp. 179–191, 2018.

[10] P. Musílek, W. Pedrycz, N. Sun, and G. Succi, "On the sensitivity of cocomo ii software cost estimation model," in Proceedings Eighth IEEE Symposium on Software Metrics. IEEE, 2002, pp. 13–20.

[11] B. W. Boehm, Software engineering economics. Springer, 2002.

[12] K. Sangeetha and P. Dalal, "Software sizing with use case point," Int. J. Innov. Sci. Eng. Technol, vol. 3, no. 8, pp. 146–150, 2016.

[13] P. Faria and E. Miranda, "Expert judgment in software estimation during the bid phase of a project–an exploratory survey," in 2012 Joint Conference of the 22nd International Workshop on Software Measurement and the 2012 Seventh International Conference on Software Process and Product Measurement. IEEE, 2012, pp. 126–131.

[14] J. Grenning, "Planning poker or how to avoid analysis paralysis while release planning," Hawthorn Woods: Renaissance Software Consulting, vol. 3, pp. 22–23, 2002.

[15] M. Cohn, Agile estimating and planning. Pearson Education, 2005.

[16] M. Saqlain, M. Abid, M. Awais, Ž. Stević et al., "Analysis of software effort estimation by machine learning techniques." Ingénierie des Systèmes d'Information, vol. 28, no. 6, 2023.

[17] P. Rijwani and S. Jain, "Software effort estimation development from neural networks to deep learning approaches," Journal of Cases on Information Technology (JCIT), vol. 24, no. 4, pp. 1–16, 2022.

[18] S. Hameed, Y. Elsheikh, and M. Azzeh, "An optimized case-based software project effort estimation using genetic algorithm," Information and Software Technology, vol. 153, p. 107088, 2023.

[19] M. S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, and W. Abdul, "Metaheuristic algorithms in optimizing deep neural network model for software effort estimation," Ieee Access, vol. 9, pp. 60 309–60 327, 2021.

[20] H. L. T. K. Nhung, V. Van Hai, R. Silhavy, Z. Prokopova, and P. Silhavy, "Parametric software effort estimation based on optimizing correction factors and multiple linear regression," IEEE Access, vol. 10, pp. 2963–2986, 2021.

[21] M. F. Bosu, S. G. MacDonell, and P. A. Whigham, "Analyzing the stationarity process in software effort estimation datasets," International Journal of Software Engineering and Knowledge Engineering, vol. 30, no. 11n12, pp. 1607–1640, 2020.

[22] M. F. Bosu and S. G. Macdonell, "Experience: Quality benchmarking of datasets used in software effort estimation," Journal of Data and Information Quality (JDIQ), vol. 11, no. 4, pp. 1–38, 2019.

[23] S. Saxena, "Sustainable development goal 9: Building resilient infrastructure, sustainable industrialization and fostering innovation," Dr. Sandeep Marwah, p. 21, 2019.

[24] B. Flyvbjerg and D. Gardner, How Big Things Get Done: The Surprising Factors that Determine the Fate of Every Project, from Home Renovations to Space Exploration and Everything in Between. Signal, 2023.

[25] 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.

[26] V. Tawosi, F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation: A replication study," IEEE Transactions on Software Engineering, vol. 48, no. 8, pp. 3185–3205, 2021.

[27] A. Priya Varshini and K. Anitha Kumari, "Predictive analytics approaches for software effort estimation: A review," Indian J. Sci. Technol, vol. 13, pp. 2094–2103, 2020.

[28] P. Sudarmaningtyas and R. Mohamed, "A review article on software effort estimation in agile methodology," Pertanika Journal of Science & Technology, vol. 29, no. 2, pp. 837–861, 2021.

[29] B. Alsaadi and K. Saeedi, "Data-driven effort estimation techniques of agile user stories: a systematic literature review," Artificial Intelligence Review, vol. 55, no. 7, pp. 5485–5516, 2022.

[30] M. Fernández-Diego, E. R. Méndez, F. González-Ladrón-De-Guevara, S. Abrahão, and E. Insfran, "An update on effort estimation in agile software development: A systematic literature review," IEEE Access, vol. 8, pp. 166 768–166 800, 2020.

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

[32] H. D. Carvalho, M. N. Lima, W. B. Santos, and R. A. d. A. Fagunde, "Ensemble regression models for software development effort estimation: A comparative study," arXiv preprint arXiv:2007.01719, 2020.

[33] S. Shukla and S. Kumar, "Self-adaptive ensemble-based approach for software effort estimation," in 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER). IEEE, 2023, pp. 581–592.

[34] A. Kaushik, A. Chauhan, D. Mittal, and S. Gupta, "Cocomo estimates using neural networks," International Journal of Intelligent Systems and Applications (IJISA), vol. 4, no. 9, pp. 22–28, 2012.

[35] J. Desharnais, "Desharnais dataset," https://www.kaggle.com/datasets/toniesteves/desharnais-dataset/code?datasetId=50782&sortBy=dateRun&tab=collaboration, 2018.

[36] Y.-F. Li, M. Xie, and T. Goh, "A study of mutual information based feature selection for case based reasoning in software cost estimation," Expert Systems with Applications, vol. 36, no. 3, pp. 5921–5931, 2009.

[37] T. Menzies, "nasa93," Feb. 2008, Instances: 93 Attributes: 24-15 standard COCOMO-I discrete attributes in the range Very_Low to Extra_High -7 others describing the project -one lines of code measure -one goal field being the actual effort in person months. [Online]. Available: https://doi.org/10.5281/zenodo.268419

[38] F. H. Yun, "China: Effort estimation dataset," in Zenodo, Switzerland, Tech., 2010.

[39] B. Kitchenham and E. Mendes, "A comparison of crosscompany and within-company effort estimation models for web applications," 01 2004.

[40] S. Amasaki, "miyazaki94," Feb. 2016. [Online]. Available: https://doi.org/10.5281/zenodo.268473

[41] M. Tsunoda, "kitchenham," Feb. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.268457

[42] 2021, "The international software benchmarking standards group," in Available: http://www.isbsg.org, 2021.

[43] J. W. Li, Yanfy; Keung, "Effort estimation: Albrecht," Apr. 2010. [Online]. Available: https://doi.org/10.5281/zenodo.268467

[44] J. W. Keung, "kemerer," apr 2010. [Online]. Available: https://doi.org/10.5281/zenodo.268464

[45] R. Silhavy, "Use case points benchmark dataset," https://zenodo.org/record/344959, 2017.

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

[47] B. Sigweni, M. Shepperd, and P. Forselius, "Finnish Software Effort Dataset," 3 2015. [Online]. Available: https://figshare.com/articles/dataset/Finnish_Effort_Estimation_Dataset/1334271

[48] S. Y. G. Venkatesh A Pavan and S. Manoli, "Software effort estimation based on use case reuse (back propagation)," Ijraset Journal For Research in Applied Science and Engineering Technology, vol. 11, no. 4, 2023.

[49] M. Rahman, P. P. Roy, M. Ali, T. GoncÂÿalves, and H. Sarwar, "Software effort estimation using machine learning technique," International Journal of Advanced Computer Science and Applications, vol. 14, no. 4, 2023. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2023.0140491

[50] S. Kassaymeh, M. Alweshah, M. A. Al-Betar, A. I. Hammouri, and M. A. Al-MaâĂŹaitah, "Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques," Cluster Computing, pp. 1–24, 2023.

[51] R. Shah, V. Shah, A. R. Nair, T. Vyas, S. Desai, and S. Degadwala, "Software effort estimation using machine learning algorithms," in 2022 6th International Conference on Electronics, Communication and Aerospace Technology. IEEE, 2022, pp. 1–8.

[52] S. Goyal, "Effective software effort estimation using heterogenous stacked ensemble," in 2022 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), vol. 1. IEEE, 2022, pp. 584–588.

[53] M. Jawa and S. Meena, "Software effort estimation using synthetic minority over-sampling technique for regression (smoter)," in 2022 3rd International Conference for Emerging Technology (INCET), 2022, pp. 1–6.

[54] W. Rhmann, B. Pandey, and G. A. Ansari, "Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms," Innovations in Systems and Software Engineering, pp. 1–11, 2021.

[55] Z. Sakhrawi, A. Sellami, and N. Bouassida, "Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method," Cluster Computing, vol. 25, no. 4, pp. 2779–2792, 2022.

[56] A. Kaushik, P. Kaur, N. Choudhary, and Priyanka, "Stacking regularization in analogy-based software effort estimation," Soft Computing, pp. 1–20, 2022.

[57] 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.

[58] 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.

[59] A. P. Varshini, K. A. Kumari, D. Janani, and S. Soundariya, "Comparative analysis of machine learning and deep learning algorithms for software effort estimation," in Journal of Physics: Conference Series, vol. 1767, no. 1. IOP Publishing, 2021, p. 012019.

[60] H. D. P. De Carvalho, R. Fagundes, and W. Santos, "Extreme learning machine applied to software development effort estimation," IEEE Access, vol. 9, pp. 92 676–92 687, 2021.

[61] M. K, "Estimation of effort in software projects using genetic programming," International Journal of Engineering Research and, vol. V9, 07 2020.

[62] B. Khan, R. Naseem, M. Binsawad, M. Khan, and A. Ahmad, "Software cost estimation using flower pollination algorithm," Journal of Internet Technology, vol. 21, no. 5, pp. 1243–1251, 2020.

[63] A. Singh and M. Kumar, "Comparative analysis on prediction of software effort estimation using machine learning techniques," in Proceedings of the International Conference on Innovative Computing & Communications (ICICC), 2020.

[64] P. Suresh Kumar and H. Behera, "Estimating software effort using neural network: an experimental investigation," in Computational Intelligence in Pattern Recognition: Proceedings of CIPR 2020. Springer, 2020, pp. 165–180.

[65] A. A. Fadhil and R. G. Alsarraj, "Exploring the whale optimization algorithm to enhance software project effort estimation," in 2020 6th International Engineering Conference âĂIJSustainable Technology and Development"(IEC). IEEE, 2020, pp. 146–151.

[66] M. K, "Estimation of effort in software projects using genetic programming," International Journal of Engineering Research and, vol. V9, 07 2020.

[67] M. A. Shah, D. Jawawi, M. Isa, M. Younas, A. Abdelmaboud, and F. Sholichin, "Ensembling artificial bee colony with analogy-based estimation to improve software development effort prediction," IEEE Access, vol. PP, pp. 1–1, 03 2020.

[68] H. Karna, S. Gotovac, and L. Vicković, "Data mining approach to effort modeling on agile software projects," Informatica, vol. 44, no. 2, 2020.

[69] L. Huang, T. Song, and T. Jiang, "Linear regression combined knn algorithm to identify latent defects for imbalance data of ics," Microelectronics Journal, vol. 131, p. 105641, 2023.

[70] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," SN computer science, vol. 2, no. 3, p. 160, 2021.

[71] A. Roy and S. Chakraborty, "Support vector machine in structural reliability analysis: A review," Reliability Engineering & System Safety, p. 109126, 2023.

[72] T. Kangwantrakool, K. Viriyayudhakorn, and T. Theeramunkong, "Software development effort estimation from unstructured software project description by sequence models," IEICE TRANSACTIONS on Information and Systems, vol. 103, no. 4, pp. 739–747, 2020.

[73] P. Prakash, S. Bhanu, and S. D. Bigul, "Random forest and logistic regression algorithms: A comparison of their performance," in AIP Conference Proceedings, vol. 2548, no. 1. AIP Publishing, 2023.

[74] E. Rodríguez Sánchez, E. F. Vázquez Santacruz, and H. Cervantes Maceda, "Effort and cost estimation using decision tree techniques and story points in agile software development," Mathematics, vol. 11, no. 6, p. 1477, 2023.

[75] B. Yarahmadi, S. M. Hashemianzadeh, and S. M.-R. Milani Hosseini, "Machine-learning-based predictions of imprinting quality using ensemble and non-linear regression algorithms," Scientific Reports, vol. 13, no. 1, p. 12111, 2023.

[76] M. Maabreh and G. Almasabha, "Machine learning regression algorithms for shear strength prediction of sfrc-dbs: Performance evaluation and comparisons," Arabian Journal for Science and Engineering, pp. 1–17, 2023.

[77] F. Sarro and A. Petrozziello, "Linear programming as a baseline for software effort estimation," ACM transactions on software engineering and methodology (TOSEM), vol. 27, no. 3, pp. 1–28, 2018.

[78] R. K. Gora and R. R. Sinha, "A study of evaluation measures for software effort estimation using machine learning," International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 6s, pp. 267–275, 2023.

MIZANUR RAHMAN completed his B.Sc. in Computer Science & Engineering from the United International University, Dhaka, Bangladesh. He is pursuing his Master's degree in the Faculty of Computer Science at Western Illinois University, USA. His research interests are Software Engineering, Optimization, Meta-heuristic Algorithms, Machine Learning, and Natural Language Processing.

MR. SARWAR is a Professor of Computer Science and Engineering (CSE) at United International University (UIU). He did his Ph.D. in Applied Physics, Electronic and Communication Engineering. He graduated with a CSE from the Bangladesh University of Engineering and Technology (BUET) He has a strong background in researching on the Bangla OCR, Software engineering, Telemedicine, Education Management, Education, and Plasmonics. Prof. Sarwar is also an entrepreneur and runs a software company of around 30 staff. This company develops and provides software services for the automation of higher education. At present, he is working on the prediction of ICU patient status at a Heart hospital, the application of machine learning algorithms to improve the status of Education, and the application of machine learning algorithms for better project management.

MD. ABDUL KADER is a senior lecturer in the Faculty of Computing at Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), Malaysia. He received his bachelor degree in computer science and engineering from Khulna University of Science and Technology (KUET), Bangladesh. He completed his M.Sc. degree in computer engineering from Universiti Malaysia Perils (UniMAP), Malaysia, and his PhD in computational intelligence from the Faculty of Computing at Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA), Malaysia. His research interests include soft computing, optimization algorithms, image processing, and software engineering.

TERESA GONÃĞALVES is Associate Professor at the Department of Computer Science, University of ÃĽvora, Portugal. She has a PhD in Computer Science from University of ÃĽvora and a MSc degree in Informatics Engineering from New University of Lisbon. Her domain of specialization is Machine Learning and Artificial Intelligence, with research and publications in the areas of information extraction and retrieval, evolutionary algorithms, text classification (Portuguese and English) and image mining (medicalÂăandÂăsatellite).

TING received her BSc and PhD in Computer Sciences from University of Science, Malaysia. Ting joined Gemalto as telecommunication software engineer in Singapore before joined academic industry after her PhD graduation. She has more than 12 years of lecturing, supervising projects, and research. Her research interests including big data analytics, information systems engineering, educational data mining, psycho-academic research, and software engineering. Ting received her professional certification in project management from PMI and data analytics from SAS. Currently, Ting is attached to INTI International University responsible primary in research and postgraduate supervision. At the same time, Ting serves as freelance lecturer in Monash University, Tunku Abdul Rahman University of Management and Technology, and Methodist College Kuala Lumpur..