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.2022.Doi Number

Improved Software Effort Estimation through Machine Learning: Challenges, Applications, and Feature Importance Analysis

Panduranga Vital Terlapu[1], K. Kishore Raju[2], G Kiran Kumar[3], Jagadeeswara Rao G[4], K Kavitha[5], Shirina Samreen[6]

[1]Department of Computer Science and Engineering, Aditya Institute of Technology and Management, Tekkali, Srikakulam District, Andhra Pradesh-532 201, India
[2]Department of Information Technology, Aditya Institute of Technology and Management, Tekkali, Srikakulam District, Andhra Pradesh, India.
[3]Department of Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Hyderabad , Telangana, India.
[4]Department of Information Technology, S.R.K.R Engineering College, Bhimavaram, Andhra Pradesh, India.
[5]Department of Computer Science and Engineering, GMR Institute of Technology, GMR Nagar, Razam, Andhra Pradesh 532127 India
[6]Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Al Majmaah, Saudi Arabia

Corresponding author: Shirina Samreen (e-mail: s.samreen@mu.edu.sa).

The authors would like to thank Deanship of Scientific Research at Majmaah University for supporting this work under Project Number xxx.

ABSTRACT

Effort estimations are a crucial aspect of software development. The tasks should be completed before the start of any software project. Accurate estimations increase the chances of project success, and inaccurate information can lead to severe issues. This study systematically reviewed the literature on effort-estimating models from 2015-2024, identifying 69 relevant studies from various publications to compile information on various software work estimation models. This review aims to analyze the models proposed in the literature and their classification, the metrics used for accuracy measurement, the leading model that has been chiefly applied for effort estimation, and the benchmark datasets available. The study utilized 542 relevant articles on software development, cost, effort, prediction, estimation, and modelling techniques in the search strategy. After 194 selections, the authors chose 69 articles to understand ML applications in SEE comprehensively. The researchers used a scoring system to assess each study's responses (from 0 to 5 points) to their research questions. This helped them identify credible studies with higher scores for a comprehensive review aligned with its objectives. The data extraction process identified 91% (63) of 69 studies as either highly or somewhat relevant, demonstrating a successful search strategy for analysis. The literature review on SEE indicates a growing preference for ML-based models in 59% of selected studies. 17% of the studies chosen favor hybrid models to overcome software development challenges. We qualitatively analyzed all the literature on software effort estimation using expert judgment, formal estimation techniques, ML-based techniques, and hybrid techniques. We discovered that researchers have frequently used ML-based models to estimate software effort and are currently in the lead. This study also explores the application of feature importance and selection in machine learning models for Software Effort Estimation (SEE) using popular algorithms like support Vector Machine (SVM), AdaBoost (AB), Gradient Boost (GB), and Random Forest (RF) with six benchmark datasets like CHINA, COCOMO-NASA2, COCOMO, COCOMO81, DESHARNAIS, and KITCHENHAM. We analyze the dataset descriptions and feature importance of the dataset analysis using ML models for choosing crucial play attributes in SEE.

INDEX TERMS Accuracy Measure, Classification Models, Feature Importance, Machine Learning, Software Effort Estimation, Software Metrics.

I. INTRODUCTION

Millions of software engineers work hard on software projects in one or more categories of software applications like system software, application software, embedded software, web and mobile applications, and artificial intelligence software [1]. The development of these applications must be done systematically. Hence, several development models have been proposed in the literature. These models comprise different stages in which the software will lie during its development. Software (project) effort estimation (SEE) is an essential job during the early stages of project development, and it is used in various streams like engineering, science, agriculture, construction, accounting, etc. Project estimation is quite a challenging task, which mainly includes six constraints: time, cost, scope, resources, quality, and risk, to estimate the project accurately [2]. So, using an efficient project estimation technique is very important and plays a crucial role in project success. The incorrect estimations may lead to miserable situations like overruns of budget, delivery delays, and failure to satisfy customer requirements, thereby affecting the quality of the software [3]. Hence, estimating the software development effort is critical in software project management.

Several algorithmic and non-algorithmic models are proposed, using which we can predict the amount of effort during the software development process [4]. The three significant parts of project estimation include effort estimation, cost estimation, and resource estimation. The main uncertainty in estimation comes from the input given to the software and the objectives of the company that is creating or using this software [5]. In the past, algorithmic models were applied for effort estimation using constant values supplied to a single formula, and further, they had some mathematical computation derived from historical projects. Hence, these are not dynamic and may not fit into the current technology trends in software engineering. Jorgenson and Shepperd have done a systematic literature review to study various software estimation techniques [6]. Mobile applications are also developing rapidly, so the existing classical software estimation models may need to be directly suitable. Since portable application implementation differs from traditional software implementation in technology, size, cost, etc., we can fit the existing classical software estimation models by extending them to mobile applications [7]. Some of the challenges in the software development effort estimation are the availability of historical data, uncertainty in the client's requirements, improper WBS (work breakdown structure), dependencies of the project, and complicated, mega-large projects.

This study evaluates software effort estimation models, their effectiveness in predicting software development efforts, and their application in real-world scenarios, focusing on expert assessment, machine learning, and feature importance. The main objectives of this research

- Classify and evaluate the different software effort estimation models identified through a systematic literature review.
- Assess the accuracy metrics used in these models to determine their effectiveness in predicting software development efforts.
- Identify the most frequently applied models for effort estimation and discuss their application in real-world scenarios.
- Review benchmark datasets commonly used in software effort estimation research, such as China, COCOMO-NASA2, COCOMO, COCOMO81, DESHNAIS, and Kitchenham.
- Conduct a qualitative analysis of the literature surrounding software effort estimation, focusing on expert judgment, formal estimation techniques, machine learning (ML)-based techniques, and hybrid approaches.
- Highlight the prominence of ML-based models in current research and their effectiveness in software effort estimation.
- Investigate the role of feature importance and selection in enhancing the performance of ML models for Software Effort Estimation (SEE).

Understanding the importance of features in ML helps data scientists identify influential variables in predictive models, improve model interpretation, and understand the underlying mechanisms. It improves model accuracy and generalization, improving decision-making in domains like SEE, Health Care, Design, etc. Feature importance analysis fosters transparency and trust in ML models, enhancing their applicability and impact in real-world scenarios. Feature importance in SEE datasets like China, COCOMO-NASA2, COCOMO, COCOMO81, DESHNAIS, and Kitchenham is crucial for accurate predictive modelling. It reveals which variables influence software development efforts, ranging from project size to complexity metrics and resource allocation. ML algorithms can prioritize these variables, improving model performance and reliability. This approach aids in optimizing resource allocation, budgeting, and project planning, contributing to more efficient software development processes.

The existing research on software effort estimation reveals several gaps within the field. Despite the extensive research on machine learning and fuzzy logic techniques [8], more than one method has consistently proven to be the most effective for accurate prediction [9]. Some studies have concentrated on integrating various approaches, such as Logarithmic Fuzzy Preference Programming (LFPP) and Least Squares Support Vector Machines (LSSVM) [10]. Additionally, researchers have suggested ensemble ML techniques like bagging and voting for software effort estimation [11]. Additionally, some studies [12-14] focus on identifying new software traits and developing regression models for analysis. Most existing estimation

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

techniques are non-deterministic, which means that their results can vary. Some methods may also be computationally expensive. The study is required to improve the accuracy and effectiveness of estimating methods.

Most existing estimation approaches are non-deterministic, meaning they can produce varying results. Additionally, some methods can be computationally expensive. Research is needed to improve the accuracy and efficiency of estimation techniques. While there's a lot of research on traditional methods, Agile software development requires more exploration. Specifically, there's a need for a better understanding of how to use size metrics and cost drivers in Agile settings. Moreover, the datasets need to focus more on feature interpretation. A gap exists in understanding the underlying relationships between those features and the estimated effort. The importance of features might differ based on project type (e.g., mobile app vs. enterprise software). Studies should use object-type variations when adequately analyzing features. The estimation of the quality of effort depends highly on the data used to train the models. Research is needed to improve data collection and ensure quality for better estimation.

We identified some of gaps in existing literature that are a) Limited Focus on Feature Interpretation: Studies often report "important" features without explaining their significance. b) Lack of Generalizability: Feature importance varies based on specific datasets and contexts. c) Actionable Insights for Project Management: The knowledge of critical features must be translated into actionable insights for project managers. d) Integration with Traditional Techniques: Focus on ML models and overlook the potential benefits of combining feature importance analysis with traditional estimation techniques. e)Addressing Specific Project Types: Studies may need to consider project type variations adequately when analyzing the importance of features.

The remain of this article is structured as follows:

- Section II (REVIEW METHOD) This section presents a structured review method using Kitchenham and Charters' structured approach to evaluate software effort estimation (SEE) models, explore their classification, popular preferences, ML model impact, evaluation metrics, and feature importance for benchmark datasets. The systematic search strategy, selection process, quality assessment criteria, and data extraction methodology provide a comprehensive analysis.
- Section III (PRELIMINARIES) This section provides an in-depth understanding of software development, focusing on process models and planning. It discusses various estimation evaluation metrics like MAE and MdAE, which extend a comprehensive framework for evaluating the accuracy and performance of estimation models. These metrics enhance the credibility and reliability of software development planning, ensuring the accuracy of estimation models.
- Section IV (OVERVIEW OF SELECTED STUDIES) This section explores software effort estimation (SEE) models, highlighting their evolution from the 1960s to the present. It covers parametric and non-algorithmic approaches and the integration of machine learning techniques. The section also provides a classification of SEE models, highlighting their applications and effectiveness in software effort estimation.
- Section V (SUMMARY OF FINDINGS) This section summarizes the findings of a literature review on software effort estimation (SEE) and highlights machine learning (ML) models as the most popular, accounting for 58% of studies. ML models improve prediction accuracy and overcome challenges in software development, with benchmark datasets crucial for accurate estimation.
- Section VI (DISCUSSION) discusses the current state of Software Effort Estimation (SEE), highlighting the growing use of machine learning models and the significance of feature importance analysis. It suggests future research directions, including hybrid approaches, explainable AI methods, and domain-specific models for improved reliability.
- Section VII (THREATS TO VALIDITY) This section critically analyzes potential biases and limitations that could impact the reliability and generalizability of the study, including threats to internal, external, and construct validity.
- Section VIII (CONCLUSION AND FUTURE SCOPE) The conclusion and future scope section provides a comprehensive overview of effort estimation models, highlighting the dominance of ML-based models, the importance of benchmark datasets, and the need for further research in AI methods, ensemble learning, and hybrid approaches.

II. REVIEW METHOD

Our review method follows Kitchenham and Charters' 2007 guidelines [15], a structured approach with various stages. This framework helps navigate literature synthesis complexities, ensuring rigour and consistency. By adhering to established protocols, we extract valuable insights from selected literature, providing a robust foundation for systematic review and analysis.

A. RESEARCH QUESTIONS (RQ)

The literature review presented in this paper intended to answer the research questions given in Table I. This paper's literature review explores metrics for evaluating SEE model accuracy, classifies SEE models by their nature, and identifies popular models and benchmark datasets for SEE. It aims to contribute to a comprehensive understanding of the state-of-the-art in SEE research.

Fig. 1 shows the systematic literature review process. Straightforward research questions determined the study's scope and objectives. At the same time, a comprehensive search strategy was developed, incorporating specific terms and literature resources to ensure complete coverage of relevant studies [16]. The study utilized an extraction form to systematically extract data from selected studies, enabling robust data synthesis and quality assessments.

TABLE I.
RESEARCH INQUIRY: KEY QUESTIONS FOR REVIEW.

ID	Research (Queries) Questions	Motivation
RQ1	What are the current approaches and their classification based on their nature in SEE?	This question focuses to identify the present approaches and the classification of SEE models by considering their work nature.
RQ2	Which category of models has been popularly preferred for SEE in recent times?	To review the popular models for SEE based on the previous studies.
RQ3	Is ML models really contributing to enhancing the SEE?	Aims to explore the recent improvements in SEE using ML models.
RQ4	What are the metrics used to evaluate the SEE model's accuracy?	This question aims to know the model's performance evaluation methods and accuracy measures.
RQ5	Identifying feature importance for benchmark datasets available for SEE?	Aims to explain the important features from benchmark datasets available.

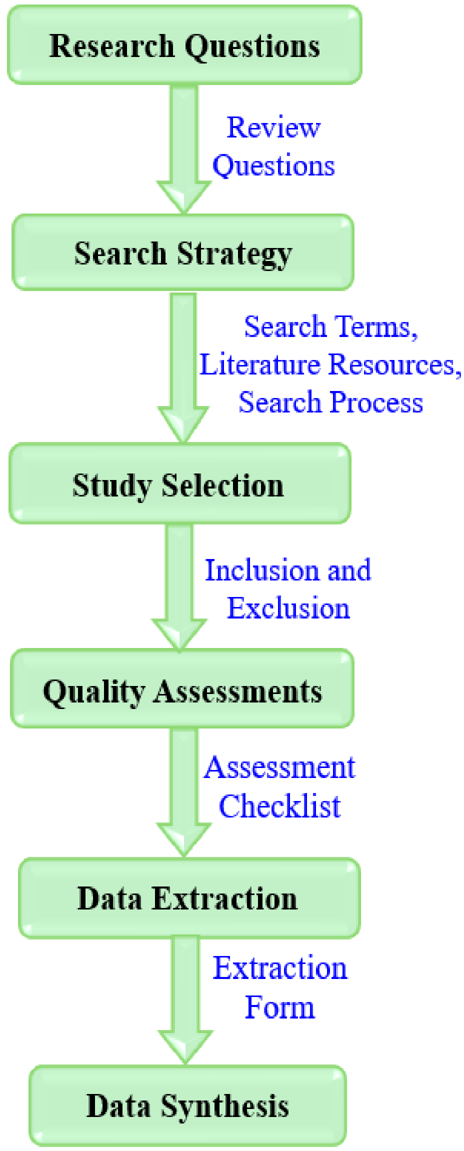


FIGURE 1. Systematic literature review process

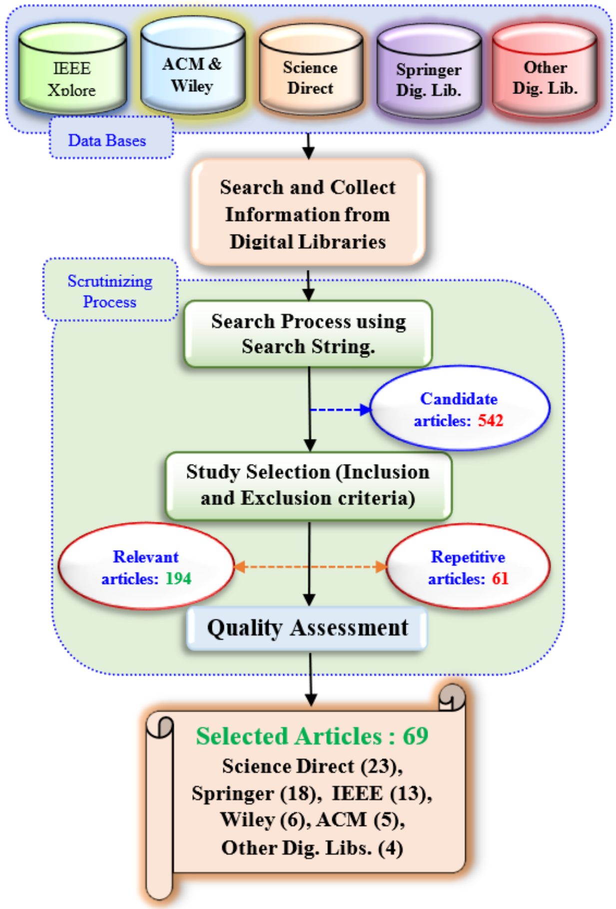


FIGURE 2. The Process of Study Selection

B. SEARCH STRATEGY

The search terms are used to form the query string (QS) to get the relevant studies that help us answer RQs. The query string used is represented below.

(Project OR Software OR application OR system) AND development AND (cost OR effort OR size) AND (predict OR estimate OR assess OR calculate OR measure) AND (model

OR method OR algorithm OR procedure OR technique) AND (taxonomy OR category OR approach). This review used electronic databases like IEEE Xplore, ScienceDirect, ACM Digital Library, Springer, Elsevier, Wiley, and Scopus articles to examine recent advancements and trends in e-commerce. The manual search process was meticulous, using a formulated query string (QS) to retrieve relevant articles. The review spanned from 2015 to 2024. This study used a comprehensive query string to retrieve relevant studies to address research questions. The string includes terms related to project, software, application, system development, cost, effort, size, prediction, estimation, assessment, calculation, measurement, model, method, algorithm, procedure, technique, and taxonomy. This strategic combination ensures a focused search across various dimensions of the subject matter, facilitating the retrieval of diverse literature relevant to the study's objectives.

C. STUDY SELECTION

This study explores software effort estimation models, their strengths and weaknesses, the impact of data sources on accuracy, the use of machine learning algorithms, challenges and limitations of data mining techniques, and recent trends in research. It also discusses the importance of analyzing data mining techniques and identifying emerging technologies or methodologies for improving accuracy and efficiency. Fig. 2 shows the detailed study selection process. In this study, the selection process involved a systematic search across major digital libraries, identifying 521 candidate articles. After rigorous inclusion and exclusion criteria, 198 relevant articles were selected. The quality assessment further refined the selection, yielding 68 selected articles. Science Direct contributed 27 articles, followed by Springer, IEEE Xplore, Wiley, ACM Digital Library, Scopus, and Google Scholar. This coverage ensured a comprehensive understanding of ML applications and challenges in SEE.

D. QUALITY ASSESSMENT

The papers were evaluated using the following quality assessment (QA) checklist suggested in Table II by Wen et al. [17] after applying inclusion and exclusion criteria. The quality assessment stage involved evaluating papers using a checklist which included criteria like study objectives clarity, estimation model definition, supporting evidence, model accuracy measurement, study age, and comparison with existing work. Each paper was scored based on these criteria to ensure reliability and rigor in the selection process.

TABLE II.
CHECKLIST FOR ASSESSING QUALITY

ID	QA Questions	Score
QA1	Are the objective of the study is clearly stated?	Y/N/P
QA2	Are the estimation models being well defined?	Y/N/P
QA3	Are the studies supported by evidence?	Y/N/P
QA4	Is the model accuracy measured?	Y/N/P
QA5	Is the publication date of the article documented?	Y/N/P
QA6	Is the proposed model(s) compared with existing work?	Y/N/P

Each question in the checklist is scaled with 3-points, i.e., yes (Y) indicates 1 point, no (N) indicates 0 points, and partial (P) indicates 0.5. The highest score attained is 6. We have set the acceptable score for each study at greater than 3 (50% of the total); the studies that fail to reach an acceptable score are eliminated from the list of selected studies. As a result, 80 studies were unlisted from the selected studies. Finally, the list of selected studies with acceptable scores concluded with 68 studies.

E. DATA EXTRACTION

Table IV shows the list of selected studies and how they address our research questions in our review. Each study may or may not directly or indirectly address all the research questions. We have analyzed the overall quality of the studies in Table IV. We have taken a scale of 0 to 5 points and assigned 1 point for every research question if it is addressed in that study. Hence, the higher-value studies can be considered credible ones. The scores achieved by all the studies corresponding to our research questions are shown in Table III. Fig. 3 shows the score attained by each study in the selected studies.

TABLE III.
SCORE LEVEL ATTAINED BY THE STUDIES

Score	No. of papers	Attain (%)
4	**25** (s8, s12, s18, s20, s21, s24, s26, s28, s29, s38, s40, s45, s46, s49, s51, s53, s55, s57, s59, s61, s62, s64, s65, s66, s68)	36%
3	**38** (s1, s2, s3, s4, s5, s6, s7, s9, s10, s11, s13, s14, s17, s22, s23, s25, s30, s31, s32, s33, s34, s36, s37, s39, s41, s42, s43, s44, s48, s50, s52, s54, s56, s58, s60, s63, s67, s69)	55%
<=2	**6** (s15, s16, s19, s27, s35, s47)	9%

F. DATA SYNTHESIS

The extracted data needs to be tabulated and synthesized as it is required to collect the proof needed to answer the research questions discussed. Since our study involves different types of research questions, we used a narrative synthesis approach, which improved the presentation of our findings and data distribution using some visualization tools like charts and tables.

TABLE IV.
RESEARCH QUESTIONS ADDRESSED BY THE SELECTED STUDIES

Study	Ref	RQ1	RQ2	RQ3	RQ4	RQ5
s1	124	✓	✓		✓	
s2	64		✓	✓	✓	
s3	65	✓	✓		✓	

s4	125		✓	✓	✓	
s5	126		✓	✓	✓	
s6	127		✓	✓	✓	
s7	123		✓	✓	✓	
s8	128		✓	✓	✓	✓
s9	129		✓	✓	✓	
s10	130		✓	✓	✓	
s11	122		✓	✓	✓	
s12	66	✓	✓	✓	✓	
s13	67		✓		✓	✓
s14	68	✓	✓		✓	
s15	69		✓		✓	
s16	70		✓			
s17	71	✓	✓		✓	
s18	72		✓	✓	✓	✓
s19	73		✓		✓	
s20	74		✓	✓	✓	✓
s21	75		✓	✓	✓	✓
s22	76	✓	✓		✓	
s23	77		✓	✓	✓	
s24	78	✓	✓	✓	✓	
s25	79		✓	✓	✓	
s26	80	✓	✓	✓	✓	
s27	81			✓	✓	
s28	82		✓	✓	✓	✓
s29	83	✓	✓	✓	✓	
s30	52		✓	✓		✓
s31	84	✓		✓	✓	
s32	85		✓		✓	✓
s33	86	✓	✓		✓	
s34	87	✓	✓		✓	
s35	88		✓		✓	
s36	89		✓	✓	✓	
s37	90	✓		✓	✓	
s38	91	✓	✓	✓	✓	
s39	92		✓	✓	✓	
s40	93	✓	✓	✓	✓	
s41	94		✓		✓	✓
s42	95	✓	✓	✓		
s43	96	✓	✓		✓	
s44	97	✓		✓	✓	
s45	98	✓	✓		✓	✓
s46	99		✓	✓	✓	✓
s47	100	✓			✓	
s48	101		✓	✓	✓	
s49	102	✓	✓		✓	✓
s50	103	✓	✓		✓	
s51	61	✓	✓	✓	✓	
s52	105	✓		✓	✓	
s53	106	✓	✓	✓	✓	
s54	107		✓		✓	✓
s55	108	✓	✓	✓	✓	
s56	109		✓		✓	✓
s57	110		✓	✓	✓	✓
s58	111		✓		✓	✓
s59	112	✓	✓	✓	✓	
s60	113	✓	✓		✓	
s61	114	✓		✓	✓	✓
s62	115	✓	✓	✓	✓	
s63	116	✓		✓	✓	
s64	117	✓	✓	✓	✓	
s65	118	✓	✓		✓	✓
s66	119	✓	✓	✓	✓	
s67	120	✓	✓		✓	
s68	121	✓	✓		✓	✓
S69	143	✓	✓		✓	

G. UNDERSTANDING FEATURE IMPORTANCE PROCESS OF DATASETS IN SEE

The process of understanding feature importance in SEE involves a systematic journey from data collection and analysis to model training, evaluation, and visualization. The SEE data sets collection process includes robust datasets from China, COCOMO-NASA2, COCOMO, COCOMO81, DESHNAIS, and Kitchenham, followed by comprehensive data analysis, feature importance, and visualization. This involves a thorough exploration of the datasets, employing statistical techniques to gain deeper insights into features and targets. Patterns, trends, and potential outliers are identified, laying the groundwork for subsequent modelling.

The core of the process lies in determining feature importance, which is achieved by leveraging classification models that are carefully chosen to suit the dataset characteristics. The model's performance is evaluated using classification accuracy as a key metric, providing a quantitative assessment of its predictive power. The focus then shifts to uncovering the importance of individual features, calculating feature importance scores, which quantify the contribution of each feature to the model's predictive performance. Visualizing feature importance helps stakeholders understand the critical factors influencing SEE, empowering informed decision-making and resource allocation.

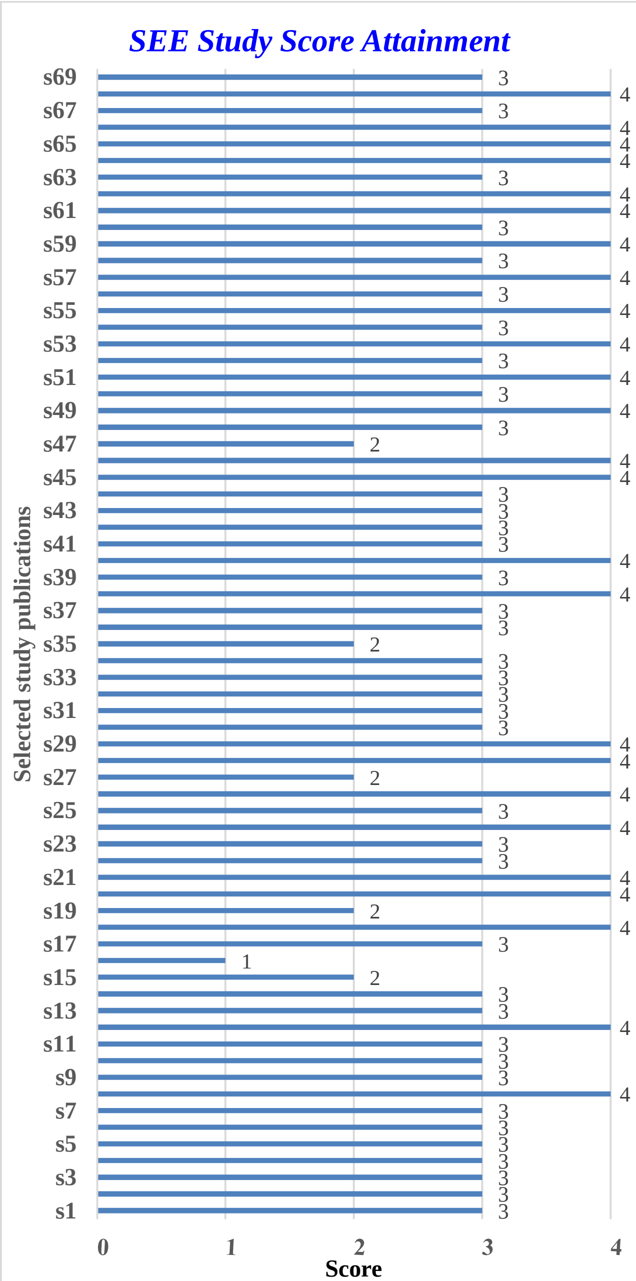


FIGURE 3. Score Attained by Each Study

H. DATA SETS DESCRIPTIONS OF SEE

The Descriptions of SEE benchmark datasets, including China, COCOMO-NASA2, COCOMO, COCOMO81, DESHNAIS, and Kitchenham, offer insights into software development, NASA projects, historical data, industrial software projects, and a benchmark for evaluating SEE techniques.

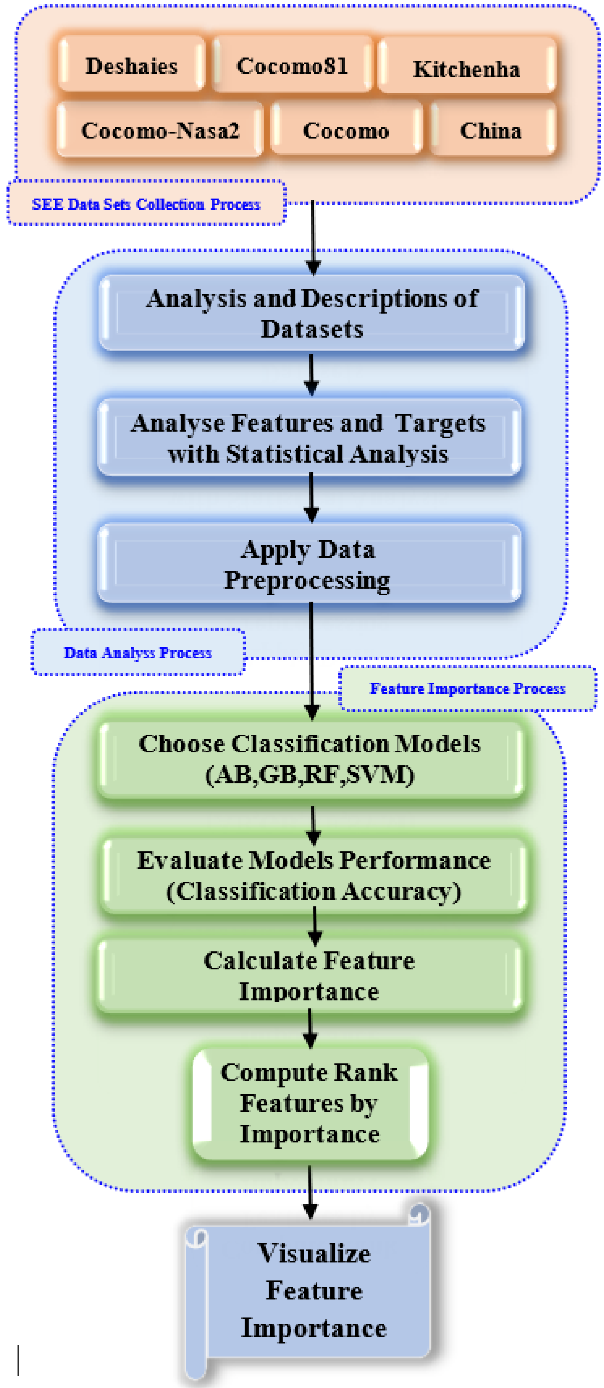


FIGURE 4. SEE Benchmark Dataset Analysis and Features Importance Process

CHINA Benchmark SEE Dataset: The "CHINA" (Table V) dataset is a tabular format with 499 rows representing individual data instances and 19 columns representing different features. It includes features like input, PDR_UFP, file, output, added, interface, ID, NPDU_UFP, deleted, inquiry, NPDR_AFP, changed, AFP, and PDR_AFP. The dataset provides a comprehensive view of attributes associated with software development projects, including sizes,

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

complexities, resource allocations, and effort requirements. This information is valuable for analyzing and predicting SES using ML models.

TABLE V.
CHINA DATASET DESCRIPTION

Feture No.(Feature)	Data Type	Min-Max	Mean±(Std)
F01(ID)		1-499	±(0)
F02(AFP)		9-17518	486.86±(1059.17)
F03(Input)		0-9404	167.1±(486.34)
F04(Output)		0-2455	113.6±(221.27)
F05(Enquiry)		0-952	61.6±(105.42)
F06(File)		0-2955	91.23±(210.27)
F07(Interface)		0-1572	24.23±(85.04)
F08(Added)		0-13580	360.35±(829.84)
F09(Changed)		0-5193	85.06±(290.86)
F10(Deleted)		0-2657	12.35±(124.22)
F11(PDR_AFP)	Quant itative	0.3-83.8	11.77±(12.11)
F12(PDR_UFP)		0.3-96.6	12.08±(12.82)
F13(NPDR_AFP)		0.4-101	13.27±(14.01)
F14(NPDU_UFP)		0.4-108.3	13.63±(14.84)
F15(Resource)		1-4.0	1.46±(0.82)
F16(Dev.Type)		0-0	0±(0)
F17(Duration)		1-84.0	8.72±(7.35)
F18(N_effort)		31-54620	4277.64±(7071.25)
C01(Effort) (**Class in Ranges**)		26-54620	3921.05±(6480.86)

COCOMONASA2 Benchmark SEE Dataset: The "COCOMONASA2" (Table VI) dataset analyses and predicts software development effort and cost using the COCOMO methodology. It consists of 93 data instances and 24 features, including reliability, mode, year, storage constraints, project name, tool support, center, turnaround time, programmer capability, equivalent physical lines of code, language experience, virtual constraints, analyst capability, application experience, complexity, data processing, schedule constraint, cat2, modern programming practices, record number, forg, virtual machine experience, and time. The dataset provides a detailed overview of project characteristics, development mode, required capabilities, and constraints, making it valuable.

TABLE VI.
COCOMO-NASA2 DATASET DESCRIPTION

Feature No. (Feature)	Type	Values
F01(recordnumber)	Continuous	real
F02(projectname)		{de, spl, slp, erb, hst, gal, Y, X}
F03(cat2)	{Discrete}	14 Distant Values Discrete values
F04(forg)		{f, g}
F05(center)		{1, 2, 3, 4, 5, 6}
F06(year)	Continuous	real
F07(mode)		{embedded, organic, semidetached}
F08(rely)		{l, n, h, vh, vl, xh}
F09(data)		{vh, vl, n, h, xh, l}
F10(cplx)		{l, h, vh, vl, xh, n}
F11(time)	{Discrete}	{n, vh, vl, xh, h, l}
F12(stor)		{n, h, vl, vh, l, xh}
F13(virt)		{vh, n, h, xh, vl, l}
F14(turn)		{xh, n, vh, h, l, vl}
F15(acap)		{vl, l, h, vh, n, xh}
F16(aexp)		{vl, vh, l, h, n, xh}
F17(pcap)		{n, l, vh, vl, h, xh}
F18(vexp)		{vh, l, h, xh, vl, n}
F19(lexp)		{h, vl, l, xh, vh, n}
F20(modp)		{l, vh, xh, h, vl, n}
F21(tool)		{l, n, h, vh, vl, xh}
F22(sced)	Continuous	real
C01(equivphyskloc)	Target	Real(Grouping 1-7)

COCOMO Benchmark SEE Dataset: The "COCOMO" (Table VII) dataset is a comprehensive resource that includes 60 data instances with 17 essential attributes for estimating software development effort using the COCOMO methodology. The dataset provides project planning, resource allocation, and precise estimation of software development effort by analyzing project-specific characteristics and constraints, including software reliability, database size, product complexity, time, storage constraints, and turnaround time.

TABLE VII.
COCOMO DATASET DESCRIPTION

F#(Feature)	Type	Value
FO1(RELY)		{High, Low, Nominal, Very_High}
FO2(DATA)		{Nominal, High, Low}
FO3(CPLX)		{Nominal, High, Very_Low, Low}
FO4(TIME)		{Very_High, High, Nominal}
FO5(STOR)		{Nominal, Very_High, High}
FO6(VIRT)		{Nominal, High, Very_High, Very_Low, Low}
FO7(TURN)		{High, Very_High, Nominal}
FO8(ACAP)		{Nominal, Very_High, High}
FO9(AEXP)	{Discrete}	{Nominal, Very_High, High, Extra_High}
FO10(PCAP)		{Nominal, Very_High, High, Extra_High}
FO11(VEXP)		{Nominal, High, Low}
FO12(LEXP)		{Low, Nominal, High}
FO13(MODP)		{Very_High, High, Nominal, Extra_High, Low}
FO14(TOOL)		{Low, Nominal, High}
FO15(SCED)		{Low, Nominal, High}
FO16(LOC)	Numeric	Real
CO17(Act-Effort)	Numeric	Real

COCOMO81 Benchmark SEE Dataset: The "COCOMO81" (Table VIII) dataset consists of 16features (Fe) and one class attribute with primarily numeric data types and one target variable, "actual." Numeric features like "rely," "data," and "cplx" have varying values, indicating different levels of importance. Attributes like "loc" have a wide range, indicating potential outliers or significant variability. The target variable "actual" represents real values, ranging from 5.9 to 11400, indicating the dataset's diversity and complexity.

TABLE VIII.
COCOMO81DATASET DESCRIPTION

Feture No. (F_Name)	Data Type	Min-Max	Mean(±Std)
F01(rely)		0.75-1.4	1.04±(0.19)
F02(data)		0.94-1.16	1±(0.07)
F03(cplx)	{Numeric}	0.7-1.65	1.09±(0.2)
F04(time)		1-1.66	1.11±(0.16)
F05(stor)		1-1.56	1.14±(0.18)
F06(virt)		0.87-1.3	1.01±(0.12)

F07(turn)		0.87-1.15	0.97±(0.08)
F08(acap)		0.71-1.46	0.91±(0.15)
F09(aexp)		0.82-1.29	0.95±(0.12)
F10(pcap)		0.7-1.42	0.94±(0.17)
F11(vexp)		0.9-1.21	1.01±(0.09)
F12(lexp)		0.95-1.14	1±(0.05)
F13(modp)		0.82-1.24	1±(0.13)
F14(tool)		0.83-1.24	1.02±(0.09)
F15(sced)		1-1.23	1.05±(0.08)
F16(loc)		1.98-1150	77.21±(168.51)
F17(actual)	**Target(Groups)**	5.9-11400	683.32±(1821.58)

DESHARNAIS Benchmark SEE Dataset: The "DESHARNAIS" (Table IX) The dataset includes 12 features related to project management and software development, including numeric data like "TeamExp" and "Effort", representing years of experience and person-hours of effort, and categorical data like "Language", indicating different programming languages. It also provides: Insights into project timelines and scope. Offering a comprehensive view of project characteristics like team expertise. Effort allocation. Language preferences.

TABLE IX.
DESHARNAIS DATASET DESCRIPTION

Feature (#)	Feature Name	Data Type & Values
F01	Project	numeric (% proj if.)
F02	Team_Exp	{numeric (years)}
F03	Manager_Exp	
F04	Year_End	{Numeric}
F05	Length	
F06	Effort	
F07	Transactions	{Numeric}
F08	Entities	
F09	PointsAdjust	
F10	Envergure	
F11	PointsNonAjust	
F12	Language(1,2,3)	Integer 1,2,3

KITCHENHAM Benchmark SEE Dataset: The "KITCHENHAM" (Table X) SES dataset, consisting of 145 data instances and 4 features, estimates software effort. It includes project duration, adjusted function points, first estimate, and actual effort. The first estimate ranges from 121 to 79870, while the target effort is 3113.12. This dataset offers valuable insights into software development effort estimation.

TABLE X.
KITCHENHAM DATASET DESCRIPTION

F##Feature Name	Data Type	mean ± std	Min-Max
FO1(Actual.duration)	Continuous	206.45±134.09	37-946
FO2(Adjusted.function .points)		527.67±1521.99	15.36-18137.48
FO3(First.estimate)		2855.97±6789.29	121-79870
FO4(Actual.effort) (Target)		3113.12±9598.01	219-113930

III. PRELIMINARIES

This section discusses the preliminary concepts in software application development, like the software process model and planning.

A. SOFTWARE PROCESS MODEL

For software engineering projects, a process model provides a road map. It concerns the flow of all activities, and actions, the work products, and the organization of the work to be done. Organizations will have their own specific process. But these individual models generally follow the abstract process models [18]. There are a lot of process models available, but the generic process model contains the following phases of software development.

- Communication - initiating the project and collecting requirements.
- Planning - estimating, setting schedules, and monitoring progress.
- Modelling - analysing and designing.
- Construction - coding and testing.
- Deployment - delivering, providing support, and receiving feedback.

B. SOFTWARE DEVELOPMENT PLANNING

Software project (SPP) planning provides a framework for managers to estimate resources, cost, effort, and schedule. Along with these, it also creates the project scope and analyses the risk. SEE has been identified as an important criterion for the past few decades [19]. It is proven to be a crucial measure since inaccurate predictions may lead to severe problems like overestimation and underestimation.

1. RQ4: Software Estimation Evaluation Metrics

Here we will discuss some of the popular evaluation metrics for SEE, which will determine the accuracy and performance of various estimation models.

a. Mean Absolute Error (MAE)

It is a measure of errors between paired observations, i.e., actual value and the measured value, exhibiting the same phenomenon [20]. The MAE is calculated by using Eq. 1.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - x| \quad (1)$$

where, n = the number of errors, x_i is the measured value, x is the actual value.

b. Median Absolute Error (MdAE)

It is calculated by taking the median of all absolute differences between the actual value and the measured value [21]. The MdAE is calculated by using Eq. 2.

$$\text{MdAE} = \text{median}\ (|x_i - x|) \quad (2)$$

where, x_i is the measured value, x is the actual value.

c. Magnitude of Relative Error (MRE)

Relative error (RE) is the ratio of the absolute error (difference between the actual value (A) and the measured value or estimated value (E)) of a measurement to the actual measurement [22]. The same is calculated by using Eq. 3.

$$\text{RE} = \frac{E-A}{A} \quad (3)$$

MRE is an absolute value of RE. The MRE is calculated by using Eq. 4.

$$\text{MRE} = \left|\frac{E-A}{A}\right| \quad (4)$$

d. Mean Magnitude of Relative Error (MMRE)

MRE is not reliable when the items are skewed. MMRE is the mean of the magnitude relative error. MMRE is calculated by using Eq. 5.

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n} MRE_i \quad (5)$$

where, n = the number of errors.

e. Median Magnitude of relative error (MdMRE)

MdMRE is the median MRE. MdMRE is calculated by using Eq. 6.

$$\text{MdMRE} = \text{median (MRE)} \quad (6)$$

f. Root Mean Squared Error (RMSE)

RMSE is the square root of the average of squared difference taken from each actual and predicted value [23]. RMSE is calculated by using Eq. 7.

$$\text{RMSE} = \sqrt{\frac{\sum_{j=1}^{k}(x_j - x)^2}{k}} \quad (7)$$

where, k = the number of errors, x_j is the predicted value, x is the actual value.

g. Standard Deviation (SD)

SD is a measure of a set value, in which it finds the amount of variation or deviation relative to its mean value [24]. SD is calculated by using Eq. 8.

$$\text{SD} = \sqrt{\frac{\sum_{j=1}^{k}(x_j - \bar{x})^2}{k-1}} \quad (8)$$

where, k = the number of errors, x_j is the predicted value, $\bar{x}$ is the mean value, which is given as,

$$\bar{x} = \sum_{j=1}^{k}\frac{x_j}{k} \quad (9)$$

h. Relative Standard Deviation (RSD)

RSD is also called the coefficient of variation. RSD is a measure of the dispersion of a set of values scattered around the mean. It is calculated in percent, obtained by multiplying SD by 100 and dividing the product by mean value. RSD is calculated by using Eq.10.

$$\text{RSD} = \frac{SD*100}{\bar{x}} \quad (10)$$

where SD = standard deviation, $\bar{x}$ is the mean value. The higher value of RSD indicates that the values are widely spread from their mean. The lower value of RSD indicates that the values are closer to their mean.

i. Logarithmic Standard Deviation (LSD)

LSD is a measure obtained by normalizing the SD of errors to a logarithmic scale. SD is calculated by using Eq. 11.

$$\text{LSD} = \sqrt{\frac{\sum_{i=1}^{n}\left(e_i - \left(-\frac{s^2}{2}\right)\right)^2}{n-1}} \quad (11)$$

where, n = the number of errors,
$e_i = \ln(Actual_i) - \ln(predict_i)$,
s = estimator of the variance of e_i

As proposed by Foss et al. [25], the mean and variance of errors of a model are taken by logarithms and will be equal to -s2/2, s2respectively, if they exhibit a normal distribution.

j. PRED(p)

The PRED(p) effort estimation measure is a quantitative assessment of model performance, providing insight into the accuracy of predicted values within a specified percentage range. Derived from the Mean Relative Error (MRE) methodology [26], it enhances credibility and reliability in evaluating estimation accuracy.

IV. OVERVIEW OF SELECTED STUDIES

A. RQ1: SEE MODELS

In the 1960s, Farr and Nelson [27-28] studied the problems during the project effort estimation. Most of the early estimating models were mathematically drawn from other areas or based on regression analysis. These models are called 'Formal Software Estimation Models'. Since 1906, a huge number of approaches have evolved, like classification and regression trees (CART), fuzzy logic modelling, parametric estimation models, etc. Out of these, parametric estimation models like COCOMO, SEER-SEM, and SLIM were the most popular estimation models during the 1980s and even today. During the 1990s, new approaches like function points, use case points [29], and size-based estimation [30] evolved based on functionality and size. With COCOMO II's release in 2000, the parametric models were updated with new data. Since then, many strategies and models have been evolving with the help of advancements in technology, like models integrating ML techniques, neural networks, etc. [31-32].

1.Classification of SEE models

In the literature, several SEE models and their categories were proposed by past researchers, discussed in the following section. From the history and literature knowledge, we are classifying the SEE models as shown in Fig. 5.

1.1 Expert Judgement (EJ)

In these types of models, the effort estimation depends on the estimator's expertise based on historical data and the same kinds of projects done in the past. Jorgenson et al. (2003) suggested that EJ is the dominant strategy in SEE [6]. These models are very subjective and cannot be reused. Since the estimation documentation frequently contains phrases such as "I believe that..." or "I feel that..." a lack of analytical presentation is going to be a drawback of these models.

a. Wideband Delphi (WD)

WD is a consensus-based effort estimation model. It was developed during the 1960s, derived from the Delphi method [33]. This model generates an estimation from a team of expert members, selected by the project manager. The estimation process is given in the following steps [34].

i. Choose the team: An estimation team of 3 to 7 members will be selected.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

ii. Kick-off meeting: The moderator leads the team to brainstorm and collect the assumptions, determines the estimating units and creates a Work Breakdown Structure (WBS).
iii. Individual Preparation: For each task in WBS, initial estimates will be generated by each team member individually.
iv. Estimation session: The moderator iterates the sequence of steps to get consensus on the estimates. During every iteration, the team resolves the issues and updates their estimation. The iterations will be continued until no team member wants to modify his or her estimate or all members agree on the range.
v. Assemble tasks: The project manager collects the individual estimates from the team members and finally prepares the assumptions, task list, and estimates.
vi. Review results: The project manager reviews the final task list with team members.

B. FORMAL ESTIMATION MODELS (FEM)

These models have been a very popular category in literature to date [35]. These models were implemented based on the parametric model approach and the size-based estimation approach. The models implemented using the parametric model approach are also termed 'algorithmic models' or 'parametric models. The models implemented using a size-based estimation approach are also termed 'non-algorithmic models' or 'non-parametric models. The major cost drivers for these models are the size of the software and kilo lines of code (KLOC).

1. ALGORITHMIC MODELS

a. Constructive Cost Model (COCOMO)

COCOMO classifies projects based on characteristics like team size, experience, and development approach, offering cost estimation capabilities throughout the development lifecycle. COCOMO is a procedural model based on lines of code, developed by Barry Boehm in 1981, used to estimate software project effort, time, cost, size, and quality.

- An **organic** project involves a small team with extensive problem knowledge and prior experience in solving it.
- A **semi-detached** software project involves a team of both experienced and inexperienced members working on project development.
- A software project is considered **embedded** in development due to its complexity and creativity, necessitating a large team with experienced members and creativity.

Different COCOMO models have been developed for **cost estimation** at different project stages, varying in accuracy and suitable for various tasks.

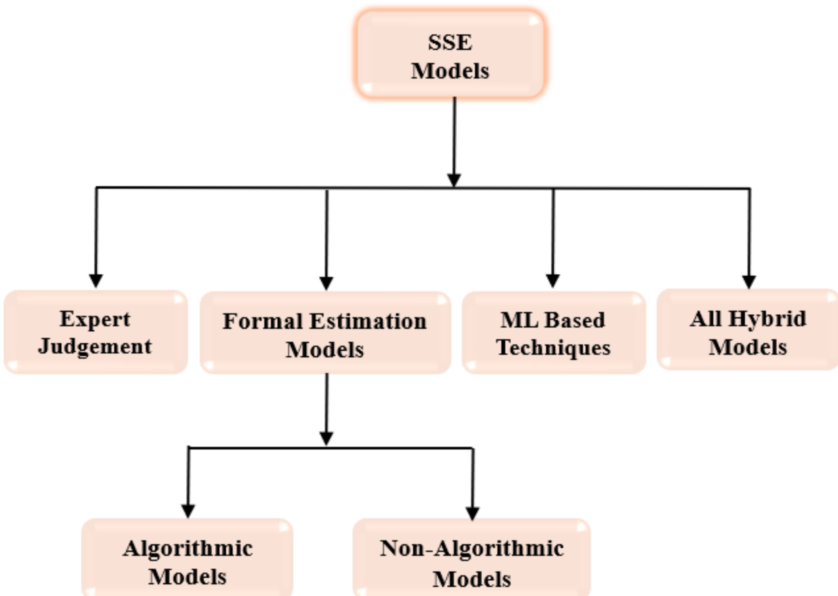


FIGURE 5. **Classification of SEE Models**

TABLE XI.
CONSTANT VALUES FOR PROJECT TYPES

Project Type	a	b	c	d
Organic: Natural	2.4	1.05	2.5	0.38
Semi-Detached: Duplex	3.0	1.12	2.5	0.35

Embedded: Integrated	3.6	1.20	2.5	0.32

TABLE XII.
LIST OF DRIVERS AND RANGES

Cost driver Attributes	Very low	Low	Nominal	High	Very high
Product Features:					
Software Requirements:	0.75	0.88	1	1.15	1.4
Database Scope:	N/A	0.94	1	1.08	1.16
Project Complexity:	0.7	0.85	1	1.15	1.3
Hardware Specifications:					
Performance Constraints:	N/A	N/A	1	1.11	1.3
Memory Limitations:	N/A	N/A	1	1.06	1.21
Virtualization Environment:	N/A	0.87	1	1.15	1.3
Turnaround Time Requirement:	N/A	0.94	1	1.07	1.15
Team Skills:					
Analytical Skills:	1.46	1.19	1	0.86	0.71
Domain Expertise:	1.29	1.13	1	0.91	0.82
Technical Skills:	1.42	1.17	1	0.86	0.7
Virtualization Expertise:	1.21	1.1	1	0.9	N/A
Programming Proficiency:	1.14	1.07	1	0.95	N/A
Project Details:					
Development Methodology:	1.24	1.1	1	0.91	0.82
Development Tools:	1.24	1.1	1	0.91	0.83
Development Timeline:	1.23	1.08	1	1.04	1.1

i) Basic COCOMO

The method accurately predicts project parameters size, including workload and resources, and calculates effort using specific equations, referencing Equations 12 and 13.

$$\text{Effort} = a * (\text{KLOC})^{b}\text{PM} \quad (12)$$

$$T = c * (\text{effort})^{d}\ \text{Months} \quad (13)$$

The COCOMO formula calculates the effort needed to develop a software product by considering its size, project type, and development time. It uses specific constants based on project types and refers to the total person-months needed for software development, aiding project managers in better planning and resource allocation. Table XI displays the values of constants a, b, c, and d for different project types, which are crucial parameters in project calculations and equations.

ii) Intermediate COCOMO

The software development environment is not considered in the basic COCOMO. To address this issue, Boehm added a set of 15 cost drivers to the intermediate COCOMO to improve the accuracy of the estimated effort. Table XII contains a list of 15 cost drivers. The cost drivers have up to five degrees of rating: Very Low, Low, Nominal, High, and Very High. The intermediate COCOMO equation is given by Eq. 14 and 15.

$$\text{Effort}=a * (\text{KLOC})^{b} * \text{EAF} \quad (14)$$

$$T=c * (E)^{d} \quad (15)$$

where, E = Total effort required for the project in Man-Months (MM).

T = Total time required for project development in Months (M).

KLOC = Kilo lines of code.

a, b, c, d = The constant parameters for the software project.

EAF = Effort Adjustment Factor

EAF can be calculated by multiplying the different cost driver's parameter values.

iii) Detailed COCOMO

The primary and intermediate COCOMO treat software product development as a single entity, while the comprehensive COCOMO uses multiple effort multipliers for each cost driver. The software is divided into various modules, and the total effort is calculated by adding up all estimated efforts. The comprehensive COCOMO lists six phases: planning and requirements, system structure, complete structure, module code and test, integration and test, and cost-constructive model. The effort is determined based on program estimates and specific cost drivers at every stage of the software lifecycle.

b. COCOMO II

COCOMO II, unlike the COCOMO'81 model, considers various factors like software development approaches and reuse strategies. The software development model uses a three-level approach, with the early prototyping level for projects with high reusability [36], and the effort estimation equation is provided by Eq. 16.

$$\text{PM} = (\text{NOP} \times (1 - R/100)) / P \quad (16)$$

were, PM - person months,

NOP - number of object points,

R - percentage of reuse,

P – productivity

The second level is the early design level, in which estimates are made after requirements are concluded. The third level is the post architecture level, in which the estimations are adjusted when requirements are volatile, and rework is required. This model was developed in 1995 and published in 2000. It has introduced 17 cost drivers, listed in Table XIII, for better SEE.

TABLE XIII.
COST DRIVERS FOR COCOMO-II

Product Attributes	
RELY	Reliability Requirements: This emphasizes the desired level of system dependability and uptime.
CPLX	Module Complexity: This focuses on the intricacy and interconnectedness of individual system components.
DOCU	Documentation Scope: This clarifies the extent and detail required for system documentation.
DATA	Database Size: This refers to the volume of data the system needs to store and manage.
RUSE	Reusability Target: This specifies the desired proportion of code or components that can be reused in future projects.
Computer attributes	
TIME	Execution time constraints
PVOL	Volatility of development platform
STOR	Memory constraints
Personnel attributes	
ACAP	Capability of project analysts
PCON	Personnel continuity
PEXP	Programmer experience in project domain
PCAP	Programmer capability
AEXP	Analyst experience in project domain
LTEX	Language and tool experience
Project attributes	
TOOL	Use of software tools
SCED	Development schedule compression
SITE	Extent of multi-site working and quality of site communications

c. SEER-SEM

Galorath developed an application called SEER, a software estimation model that includes the effort, staffing, cost, schedule, and risk associated with the software. It incorporates the latest techniques, such as reusable models, effective sizing metrics, schedule analysis, individual employee effort, monthly effort estimation, project-specific staffing issues, trade-offs between reliability and effort, and maintenance of the software. The SEER model allows us to enter our values for knowledge bases like target platforms, development standards, and methods used [37]. The SEER software model will accept as little or as much data as is available. Fig. 6 shows the inputs and outputs from the SEER software model.

d. SLIM

SLIM is an algorithmic estimation model that was proposed by Putnam et al. [38]. It makes use of the Norden-Rayleigh function, which is mostly used for complex projects. The SLIM model uses past project data for estimation and also considers the KLOC, other project parameters, and attributes. Putnam has derived the software equation from his observation of the Rayleigh distribution, given as

$$L = C_k * K^{1/3} * t_d^{4/3} \qquad (17)$$

where, K is the total effort (in person months) in product development, and td is the time required for developing the product. L is the product estimate in KLOC. C_k is the technology constant. The value of C_k for a task can be computed from the historical data of the organization developing it.

3. NON-ALGORITHMIC MODELS

a. Function Point Analysis (FPA)

In 1979, Allan Albrecht first proposed this model [39], in which he believes that the function point is one of the measurements to exhibit the software product's functionality to a user. Function points (FPs) are used to measure the function size of the software. The objective of this model is to measure and provide the functional size of the software to the customer, client, and any other stakeholder in the product. The other objectives of FP are: (i) development is technology-independent; (ii) FPs are easy to apply; (iii) FPs are taken from requirement specifications; and (iv) they are valid for the clients [40]. The FPs of software will be found by counting the number of types of functions used in the software. The various FP attributes used in the software are shown in Table XIV below.

The below five attributes are also called information domain characteristics. The FPs are determined by multiplying the unadjusted functional points (UFPs) by the value adjustment factor (VAF), as given in Eq. 18. The International Function Point User Group (IFPUG) manual [41] explains how to calculate UFPs and VAFs.

$$FPs = UFPs * VAF \qquad (18)$$

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

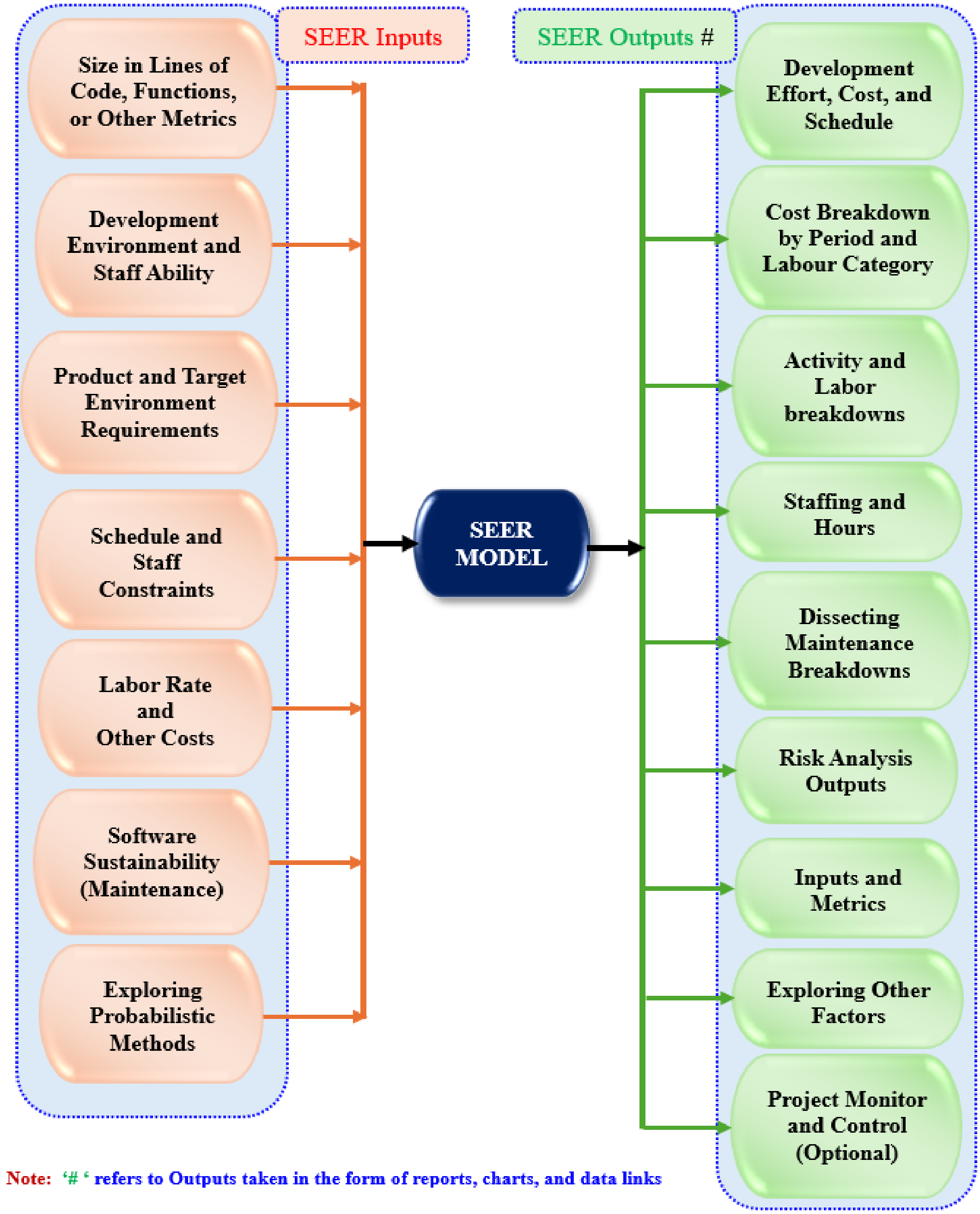


FIGURE 6. General SEER Model

TABLE XIV.
TYPES OF FP_ATTRIBUTES

S.No	Attribute	Description	Example
1.	External Inputs (EI)	Capturing inputs from the user	Tables, input screen
2.	External Outputs (EO)	Sending data to an external user or system.	Output screens, reports
3.	Inquiries (EQ)	Querying for relevant information from internal/external database	Prompts interrupts
4.	Internal Logic File (ILF)	The collection of data is present in the system.	Directories, databases
5.	External Logic File (ELF)	The collection of data from external resources or applications.	Shared databases and routines

The UFPs are calculated by summing all attributes (shown in Table 3), with their level of complexity, such as low, average, and high. The 14 General System Characteristics (GSC) are used to evaluate VAF, and they are given in Eq. 19.

$$\text{VAF} = 0.65 + \frac{(\sum_{i=1}^{14} C_i)}{100} \quad (19)$$

where, C_i = degree of influence for each GSC.

The final FPs can be calculated by substituting the VAF value obtained by using Eq.19, into Eq. 18.

b. Use Case Point (UCP)

UCP is a SEE technique used by Karner et al. in 1993. The UCP model is based on the system requirements, which are represented in the use case model [42]. It is inspired by the FP model but modified according to object-oriented systems and use case-based system requirements. The number of UCPs is the function of the following four factors:

i. Unadjusted Use Case Weight (UUCW): This factor contributes to estimating size of the software that is being developed. It is evaluated based on the number of simple, average, and complex use cases of the system. UUCW is calculated by using Eq. 20.

UUCW = (SUC * 5) + (AUC * 10) + (CUC * 15) (20)

where, SUC = number of simple use cases, AUC = number of average use cases, CUC = number of complex use cases. The type of use case and corresponding weights are given in Table XV.

TABLE XV.
USE CASE TYPES AND WEIGHTS.

Use case type	No. of transactions	Weight
Simple	1 to 3	5
Average	4 to 7	10
Complex	8 or more	15

ii. Unadjusted Actor Weight (UAW): It is calculated based on the number of simple, average, and complex actors for the system. UUCW is calculated by using Eq. 21.

UAW = (SA* 1) + (AA * 2) + (CA* 3) (21)

where, SA = simple actor, AA = average actor, CA = complex actor. The type of actors and corresponding weights are given in Table XVI.

TABLE XVI.
ACTOR TYPES AND WEIGHTS

Actor type	Description	Weight
Simple	Any external system interacting via a well-defined API	1
Average	Any external system interacting via a communication protocol	2
Complex	Humans interacting with the system using GUI	3

iii. Technical Complexity Factor (TCF): This factor accounts for the technical considerations of the system. It varies how complex the system is to be constructed. There are 13 factors which make an impact of TCF on UCPs of a project, and the corresponding weights are given in Table XVII. The TCF is computed by using the following Eq. 22 [43].

$$\text{TCF} = C_1 + C_2 \sum_{i=1}^{13} (F_i * W_i) \quad (22)$$

where, C_1 = 0.6, C_2 = 0.01. The factor, F_i is rated on a scale of 0, 1, 2, 3, 4, and 5. The irrelevant factor is indicated with 0, and the relevant factor is by 5.

TABLE XVII.
FACTORS CONTRIBUTING TO THE COMPLEXITY.

F_i	Factors contributing to complexity	W_i
F_1	Distributed systems	2
F_2	objectives for the response or throughput of the application.	1
F_3	User effectiveness (on-line)	1
F_4	Complex internal processing	1
F_5	Reusability, that the code must be adaptable to different applications.	1
F_6	Ease of installation.	0.5
F_7	Usability and ease of operation	0.5
F_8	Portability	2
F_9	Changeability	1
F_{10}	Concurrency	1
F_{11}	Special security features	1
F_{12}	Give third parties direct access.	1
F_{13}	Facilities for special user training	1

iv. Environmental Complexity Factor (ECF): This factor accounts for the environmental considerations of the system. It helps us tell how efficient the project is. There are 8 factors that have an impact on the ECF of a project, and the corresponding weights are given in Table XVIII. Finally, the UCP can be calculated by using Eq. 23.

UCP = (UUCW + UAW) * TCF * ECF (23)

TABLE XVIII.
FACTORS CONTRIBUTING TO EFFICIENCY.

F_i	Factors contributing to efficiency	W_i
F_1	Familiar with the project model that is used	1.5
F_2	Part time workers	-1
F_3	Capability of analyst	0.5
F_4	Application expertise	0.5
F_5	Object-oriented expertise	1
F_6	Motivation	1
F_7	Difficult programming language	-1
F_8	Stable requirements	2

4. MACHINE LEARNING (ML) BASED TECHNIQUES

With the advent and improvement of algorithms in artificial intelligence, ML techniques like supervised and unsupervised algorithms, fuzzy logic, neural networks, genetic algorithms, etc. are integrated with the popular existing models, with which researchers are trying to increase the estimating model's accuracy [44-45]. The ML models will be trained using the huge historical data to give better predictions than the non-ML models. The application of ML techniques in SE has shown great improvement in the results of quality predictions, project management, etc. Hence, SEE models are

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

also integrated with ML techniques to get better performance over stand-alone non-ML models. Several studies have been continued by researchers to improve the SEE [46-47]. Some of the popular ML-based techniques that have been used are regression, Bayesian networks, neural networks (NN), and support vector machines (SVM).

a. Regression

Regression is a statistical approach that is used to predict a continuous dependent variable, usually denoted by the Y-axis, from several independent variables, usually denoted by the X-axis. Legendre [48] and Gauss [49] are two of the people who first worked on regression models. Regression models are very old and powerful techniques, using which developers and project managers use historical data from past projects to build regression models. There are many types of regression analysis, which include simple linear regression (SLR), multiple linear regression (MLR), and logistic regression (LR) [50]. The simple linear regression equation is given by Eq. 24 when the data is normally distributed.

$$y = a * x + b \qquad (24)$$

Where, a and b are constants, is an independent variable, and is a dependent variable. The SLR allows us to predict one dependent variable based on one independent variable. Whereas MLR allows us to predict one dependent variable based on more than one independent variable. The equation for the MLR is shown in Eq. 25.

$$y = \sum_{i=1}^{n}(a_i * x_i) + b \qquad (25)$$

Where, a_i and b are constants, x_i is the i^{th} independent variable, y is a dependent variable, and n is several independent variables. The LR is applied to discrete data, which means the possible values for the dependent variable are only two.

b. Bayesian Networks

A Bayesian network (BN) is a probabilistic graphical model that represents the conditional dependencies of the variables in the network. The pair BN (G, P) is typically used to implement the BN, where P is a collection of probability distributions for all the variables in the network and G is a directed acyclic graph (DAG) [65]. In G, each edge refers to a conditional dependency, and each node refers to a unique random variable. Fig. 7 shows the sample BN on the random variable sets X, Y, and Z. From the figure, we can say that both X and Z are conditionally dependent on Y. It can be represented as P(X|Y) or P(Z|Y). Here, Y is unaffected by the variables X and Z. So, the joint probability P(X,Y,Z), summarized by the model, is calculated as shown in Eq. 26.

$$P(X,Y,Z) = P(X|Y) * P(Z|Y) * P(Y) \qquad (26)$$

If A = $\{X_1, X_2, \ldots, X_n\}$ is the set of random variables in BN, then joint probability distribution, P(A) is a product of conditional probability distributions for each variable given its parents, and P(A)is calculated by using Eq. 27.

$$P(A) = \prod_{i=1}^{n} P\left(x_i \mid pa(x_i)\right) \qquad (27)$$

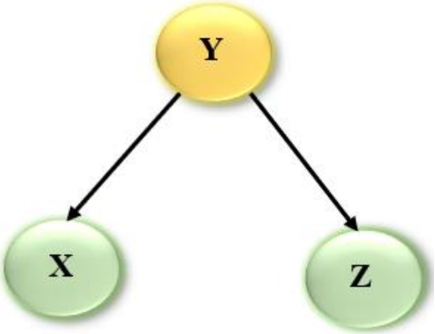


FIGURE 7. Sample Bayesian Network

where, $pa(x_i)$ is the set of parents of x_i or the set of variables which are having a direct edge to x_i.The BN gives a better assurance on the accuracy of the SEE in agile software development [51].

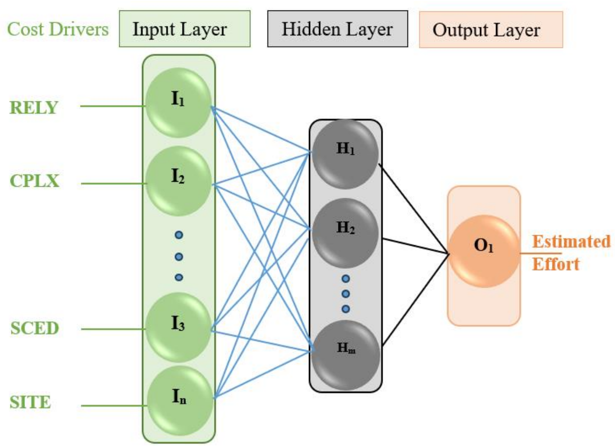


FIGURE 8. Neural Network Architecture for SEE.

c. Neural Networks

A neural network (NN) is a collection of biological neurons. The artificial NN (ANN), naturally inspired by the biological neurons, forms a neural network by organizing these neurons into several layer formats [52-54]. The advancement in ANN models makes them popular, and they are used to solve AI problems.

ANNs are very useful in modelling complex non-linear relationships in the data. In general, NN is composed of input and output layers, in between these there is a hidden layer, which consists of a set of neurons whose purpose is to add some weights (w) to the input coming from the input layer. Each neuron sums all these weighted inputs and determines the neuron output concerning a threshold value [54]. Fig. 8 shows the neural network architecture for SEE. The general form of NN is the feed-forward neural network, in which data travels in one direction from the input to the output layer. The common network models that have been used for SEE are, Back-propagation NN, Radial basis function (RBF) NN, Recurrent NN, Neuro-Fuzzy networks [55][95].

d. Support Vector Machine

Support vector machine (SVM) is a supervised ML model that is used to analyze the data for regression and classification [56-57]. This model was developed by Cortes et al. [58-59] in 1963. In SVM, when there are an N number of features that are plotted on the N-dimensional space, it tries to find a hyperplane that differentiates the data points of the two classes. These hyperplanes are called decision boundaries, which help us classify data points. SVMs can perform linear classification when the given set of data points is linearly separable. Fig. 9 shows a linear hyperplane in a two-dimensional feature space that separates data points belonging to classes A and B. In 1992, Boser et al. [60] created a non-linear SVM classifier by applying the kernel trick. SVM has been significantly performing well for SEE [61-62]. Along with these, there are many ML techniques like decision trees, instance-based models (e.g., K-nearest neighbour, etc.), and ensemble learning models (e.g., Random Forest, Ada Boost, Gradient Boosting, etc.). Genetic algorithms (GA) are also proven to outperform in predicting software effort.

5. Hybrid Models

The parametric or non-parametric algorithms are combined with ML techniques to give a hybrid model, which sometimes proves to be a better model for SEE in terms of accuracy [63][89]. Table xix shows the comparison of SEE models concerning their strengths and weaknesses.

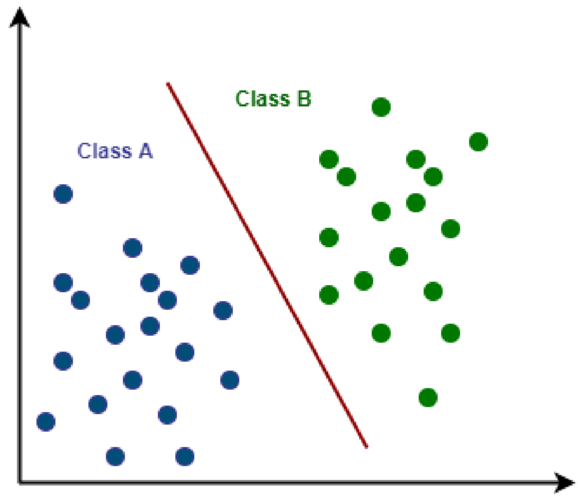


FIGURE 9. **Hyperplane In 2D Feature Space.**

TABLE XIX.
SEE MODEL'S STRENGTHS AND WEAKNESS.

Sl. No.	Category / Sub-category	Model Name	Strengths	Weakness
1	FEM / Algorithmic	COCOMO	- Simple and takes less time and effort. Also suitable for large projects	- Need large historical data of past projects. - May not fit in the current day software trends
2	FEM / Algorithmic	SEER-SEM	- Can handle different types of application configurations and environments.	- More than 50 input parameters related to a project, which increases the complexity of the model. - Difficulty in identifying non-linear relationships between input parameters and the output.
3	FEM / Algorithmic	SLIM	- Provides a set of software development management tools that support the entire software life cycle. - Can forecast software reliability.	- Best for large projects. - Sensitive to the project size. - It assumes a water-fall model, so it cannot map with the other process models.
4	FEM / non-algorithmic	FPA	- Standardized and consistent method. - Doesn't depend on the tools and language used.	- Low prediction accuracy. - Consumes more time to learn, thus may result in being costly.
5	FEM / non-algorithmic	UCP	- Can be automated, hence saves a lot of estimating time. - Gives accurate measure on size.	- Estimation can't be concluded until all use cases are found.
6	EJ	WD	- Experience from the past project maps to the proposed project. - Can assess the difference between past and future programs.	- The biases in the expert and insufficient experience may create difficulties.
7	ML techniques	Regression	- Very simple to implement. Requires less time and computational power.	- Prone to under-fitting. - Sensitive to outliers.
8	ML techniques	BN	- Easily extensible than other ML techniques. - Visualizes the knowledge in DAG, using which information can be interpreted easily.	- Requires a lot of effort to construct a network. - Poor performance with high dimensional data.
9	ML techniques	NN	- Can capture the non-linear relationships in the data. - Availability of pre-designed models.	- Overfitting problem. - Requires huge memory, computation power, especially when working with images.
10	ML techniques	SVM	- Works well when there is a clear margin of differentiation between classes. - Effective in high dimensional space.	- Not suitable for large data. - Prone to noisy data. - No probabilistic illustration of the result.

V. SUMMARY OF FINDINGS

The findings of this literature review on SEE from 81 chosen studies are presented here, and some of the findings address our research questions as well. Tables XX and XXI elaborates on the details of various SEE models and evaluation metrics employed in our selected studies.

A. RQ2: Which category of models is popularly preferred for SEE in recent times?

From the observation of selected studies, ML-based models are leading in effort estimation. Fig. 10 shows the percentage share of all the SEE models taken from our studies. Fig. 11 shows the distribution of ML-based models from our studies over the years 2015–2024. It shows the distribution of selected studies over the year of publication.

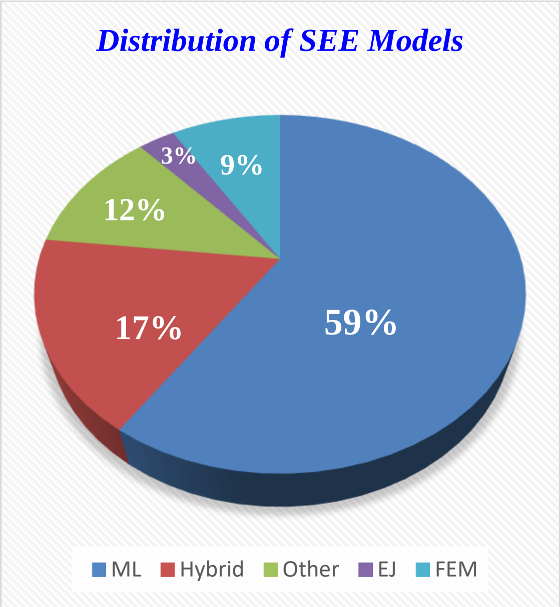


FIGURE 10. Distribution Of SEE Models.

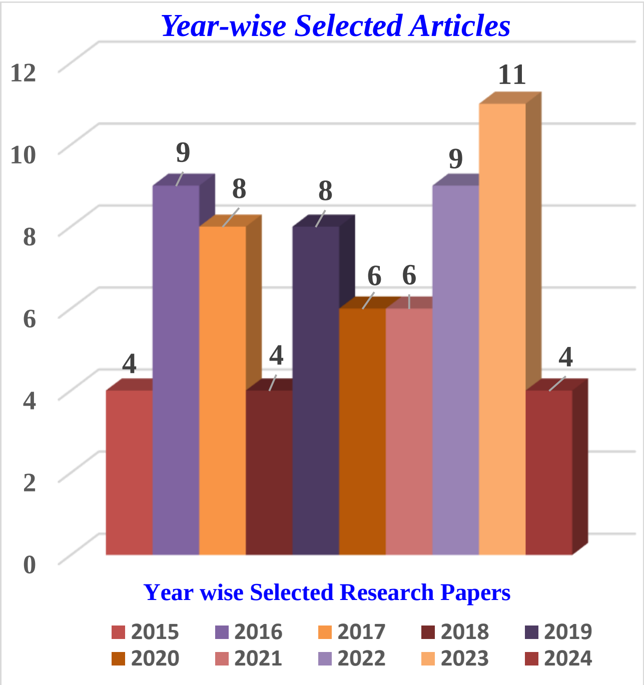


FIGURE 11. Distribution of the Studies Over Publication Year

From Fig. 11, we observe that for the past five years, the average number of studies using ML-based models has been increasing consistently. In total, ML-based models occupied first place with 59% of selected studies, followed by hybrid models (17%), formal estimation models and other models with equal shares (12%), and expert judgement with 3%.

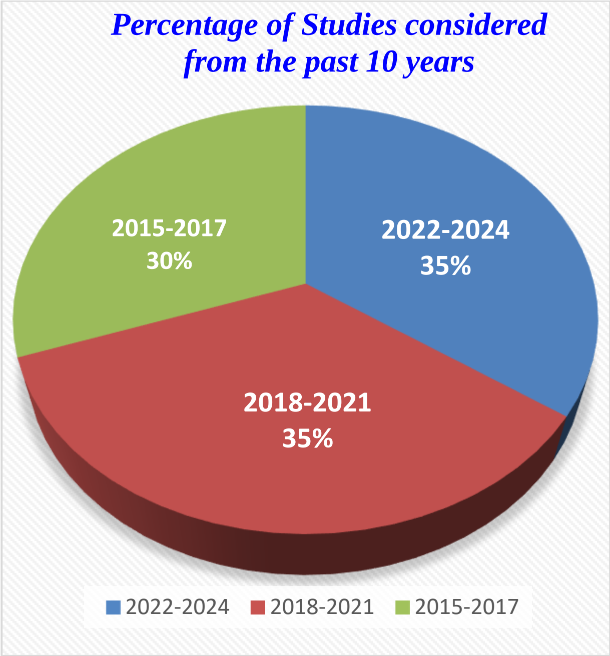


FIGURE 12. Percentage of Studies considered from the past 10 years

The percentage of studies taken into consideration during the last ten years is displayed in Fig. 12. The other models' category studies include analogy-based, case-based, re-sampling, and search-based techniques. Fig.13 shows the frequency of datasets used in the literature.

B. RQ3: Is ML models really contributing in enhancing the SEE?

ML models are significantly enhancing *SEE* by improving prediction accuracy and overcoming challenges in the software development field. Researchers are exploring various ML techniques [125-127] to identify the most accurate estimation methods. Feature selection algorithms play a crucial role in enhancing accuracy by selecting relevant features and eliminating redundant ones [134]. Here's how ML contributes:

Data-driven insights: ML algorithms can analyze historical project data, uncovering patterns that influence effort. This leads to more objective estimates compared to traditional, expert-based methods.
Improved accuracy: By considering a wider range of factors, ML models can potentially generate more accurate effort predictions [101]. This can help avoid underestimation

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

(leading to missed deadlines) or overestimation (resulting in wasted resources).

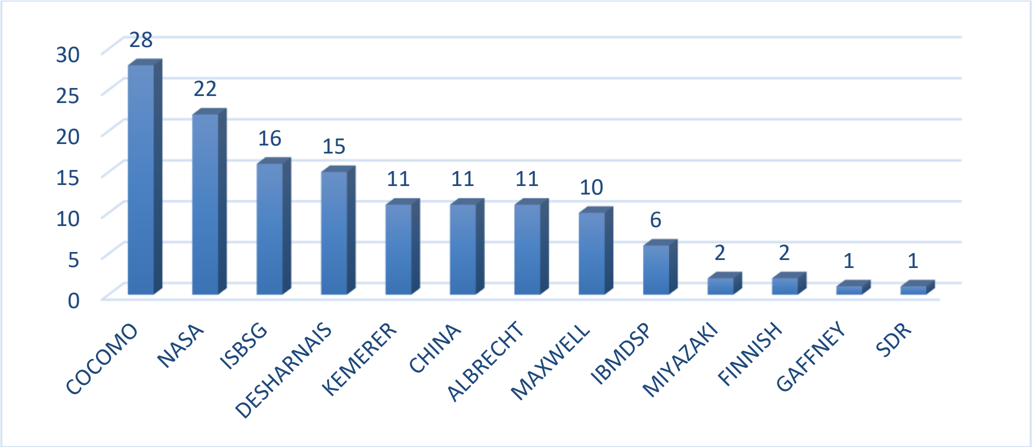


FIGURE 13. **Datasets Used by Studies in Our Review.**

Adaptability: Machine learning models can continuously learn and improve as they are exposed to new data. This allows them to adapt to evolving project landscapes and development methodologies [129].

Researchers and practitioners are exploring various machine learning techniques to improve estimation models, such as random forest, k-nearest neighbor regression, support vector regression, and decision trees. These techniques aid in selecting relevant features, eliminating redundant ones, and predicting effort accurately, leading to better project cost, quality, and time management. Lalitha et al. [110] has proposed a methodology using machine learning techniques, specifically Gaussian Process Regression (GPR), was proposed to estimate the duration of software development projects, particularly in Agile environments. The utilization of machine learning algorithms allows for tailored strategies, agile methodology adoption, and data-driven predictive models, enabling teams to make informed decisions based on historical project data. Ensembling regressor models through voting estimators further enhances prediction rates and reduces bias, bridging the gap between actual and predicted effort for future projects [122]. Apart from ML many researchers have explored neural networks for SEE. Hybrid methods, combining artificial neural network models, deep learning models, and higher-order Neural Network models, were found to yield better results in software effort estimation [132-133].

However**,** some challenges remain:

Data quality: The accuracy of ML models heavily relies on the quality and completeness of training data. Inconsistent or biased data can lead to flawed estimations.

Model interpretability: Understanding how an ML model arrives at its estimations can be difficult. This lack of transparency can be a hurdle for stakeholders who need to reason about the predictions.

Context dependency: ML models might struggle to generalize well if not carefully tailored to specific project contexts and development environments.

C. RQ5: Feature importance for benchmark datasets available for SEE.

It is very important to have historical data for software product development. However, this data is valuable, and its quality surely affects the accuracy and efficiency of the effort estimation models [139]. Most of our selected studies have used more than one dataset obtained from different repositories. About 13 different public datasets and 8 private datasets have been used in our selected studies. The COCOMO dataset is mostly used in our review; it has been employed in 28 (34%) different studies. The second widely used dataset is NASA, employed in 22 (approx. 26%) studies; ISBSG has been used in 16 (approx. 20%); and Desharnais has been used in 15 (18%). Kemerer, China, and Albrecht have been recorded equally in 11 (13%), and Maxwell has been used in 10 (12%). IBMDSP has been used in 6 (7%). Miyazaki and Finnish were used equally in two studies. Gaffney and Sdr were used in one study each. Fig. 20 summarizes the usage of datasets in our review. Six studies used their private datasets, like 214 industrial and 26 educational projects, 148 IT projects from Korean IT service vendors [138], 33 real-world software projects [131], a database of 160 tasks from real agile projects [s30], 21 agile software development-based projects taken from 6 different software houses [s24], and 512 stories taken from 24 agile software development projects during 2015 to 2017 [s22]. Fig. 13 shows the bar chart representation of the datasets used by studies in our review.

D. SEE WITH ML AND HYBRID MODELS IMPORTANT BACKGROUND WORKS

The research explores SEE by integrating ML and Hybrid models [135]. It examines applications and challenges, focusing on ML techniques for improved estimation accuracy. The study also explores the role of features extracted from SEE datasets in enhancing estimation models. The aim is to provide insights into advancing SEE methodologies and addressing the complexities of software development effort estimation.

1. SEE WITH ML MODELS BACKGROUND WORKS

Table XX comprehensively analyses SEE concepts and ML models. It highlights various studies utilizing datasets and ML methodologies to improve estimation accuracy. Key contributions include Bayesian Network models, artificial neural networks, gradient-boosting regressors, and ensemble-based models. Optimizers like swarm intelligence, back propagation, and genetic algorithms are used to fine-tune model performance. Evaluation metrics like MAE, MSE, and MMRE are used to assess model efficacy.

TABLE XX.
VARIOUS RESEARCH SEE WORKS WITH DIFFERENT ML MODELS ON VARIOUS BENCHMARK DATASETS

Ref	Dataset	ML Method / Contribution	Optimizer	Evaluation Metric	Year
[106]	70 projects from three repositories	KNN, RF, SVR, LR, MLP, GB, and DT	-	MAE, RMSE, MBRE, MIBRE (mean inverted balance relative error), MdMRE, and PRED.	2023
[105]	China, Maxwell	Random forest	-	MAE, MRE, R^2, and accuracy	2023
[108]	PROMISE data repository	case-based reasoning (CBR)	GA	MAE, Mean Balanced Relative Error (MBRE)	2023
[112]	Strongstep and Fraunhofer AICOS	Ensemble model	-	MAE, RMSE, R2, and MMER	2023
[124]	China and Maxwell	random forest	-	accuracy, R^2 value, relative error, and mean absolute error	2023
[123]	dataset created by a software company called Edusoft Consulted LTD.	k-nearest neighbor regression, support vector regression, and decision trees	-	mean absolute error (MAE), mean squared error (MSE), and R-squared error.	2023
[10]	COCOMO and NASA	Logarithmic Fuzzy Preference Programming (LFPP) and Least Squares Support Vector Machines Machine (LSSVM)	-	MMRE, RMSE	2023
[64]	Real world dataset	artificial neural networks (ANNs) and case-based reasoning (CBR)	-	CA	2022
[65]	Real world dataset	swarm intelligence and functional link neural networks		CA	2022
[128]	Real world dataset	radial basis function neural network (RBFN) and functional link artificial neural network (FLANN)	WOA	CA	2022
[92]	China, Maxwell and COCOMO81	Synthetic Minority Over-Sampling Technique for Regression	-	MMRE, PRED (25)	2022
[67]	COCOMO81, CHINA	Ensemble approach	-	MAE, MSE, RMSE, and R^2	2021
[68]	CHINA	LR, SVR, ANN, RF, DT	-	MAE, MER, MdMRE, MMRE and PRED (25)	2021
[70]	Albrecht, China, Nasa93, Cocomo81, and Maxwell	Output layer self-connection recurrent neural networks (OLSRNN) with kernel fuzzy c-means clustering (KFCM)	-	MdMRE, PRED (25)	2020
[72]	COCOMO81, Desharnais	KNN, SVM, RF, and back propagation algorithm using feed forward neural network	-	MMRE	2020
[73]	COCOMO, NASA, Desharnais, Maxwell	PSO		MAE, MER, MdMRE, MMRE and PRED (25)	2020
[74]	Desharnais	prediction model built using MLP architecture with back propagation algorithm	_	MMRE	2020
[75]	ISBSG	regression fuzzy logic	_	MAE, MIBRE, MBRE, SA, and effect size (Δ)	2019
[76]	24 agile software development initiatives represented by 503 stories	ensemble-based model	_	MAE, MBE, RMSE	2019
[77]	MAXWELL, CHINA, COCOMO81, and NASA93	hybrid model of metaheuristic algorithm and wavelet neural network (WNN)	_	MRE, MMRE, Pred(l), and MdMRE	2019
[78]	21 ASD-based projects from 6 different software houses	Elman neural network and ANN-feedforward back-propagation neural network.	-	MSE, MMRE, and Pred(l)	2019
[79]	SBSG, IBMDSP, and China	Spiking Neural Networks	-	RMSE, MMRE	2019

[107]	cocomo, isbsg, tukutuku, abrecht, desharnais, kemerer, miyazaki, china	Support vector regression	-	pred(0.25), MMRE, MdMRE and MdMRE	2019
[81]	International Software Benchmarking Standards Group (ISBSG)	Support Vector Regression	-	MRE, MAR, MdAR	2018
[82]	International Software Benchmarking Standards Group (ISBSG)	SVM, MLP and GLM (General Linear Models)	-	MRE, MMRE, Pred(l), and MER	2018
[52]	database of 160 tasks from real agile projects	Bayesian Network model	-	MMRE, PRED(k), MAE, RMSE, RAE and RRSE	2018
[84]	ISBSG R12 Dataset	genetic algorithm-based framework	-	Spearman's rank correlation, MMRE, MdMRE, MMAR, SA, and Pred25	2017
[86]	Albrecht, Cocomo, Desharnais, Kemerer, Kotengray and Nasa	Multilayer dilation-erosion-linear perceptron (MDELP)	-	MMRE and PRED25	2017
[90]	COCOMO 81, NASA 93and COCOMO_SDR	Artificial Bee Colony-Trained FLANN Model	ABC	MRE, MdMRE, and MMRE	2016
[91]	COCOMO81, China, Maxwell and NASA93	FLANN with IFCM	-	MMRE, MdMRE, PRED (25)	2016
[93]	COCOMO	Multi Layered Feed Forward Neural Network	-	MSE, MMRE	2016
[97]	ISBSG	MLP, RBFNN	-	Pred(l), absolute residuals, and a Friedman statistical test	2015
[98]	ISBSG	Comparison of statistical regression and neural networks	-	MMRE, MBRE, MIBRE	2015
[99]	ISBSG, CSBSG	BREM	-	MRE, PRED (25)	2015

Rahman et al. (2023) [123] emphasized the importance of software engineering effort estimation in project management and the use of ML techniques for improved accuracy. It recommends algorithms like k-nearest neighbor regression, support vector regression, and decision trees for improved prediction evaluation. The study uses a dataset from Edusoft Consulted LTD and evaluates their effectiveness using performance measures like MAE, MSE, and R square error. Decision trees show superior performance in effort prediction. Neural network and genetic algorithm techniques were utilized for estimation, achieving high accuracy with R2 values based on story points and lines of code [114]. Najm et al. (2023) [136] explored interpretability in ML models for software effort prediction, focusing on the optimal additive cluster-based fuzzy regression trees (Opt-ACFRT) ensemble model. It uses cross-validation and global and local interpretability methods to improve trust and comprehension among software professionals. The findings show the effectiveness of these techniques in providing understandable estimates, enabling confident decision-making. The study (Jadhav et al. (2022)) [124] analysed SDECE techniques over the past 50 years using an automated text-mining framework. It reveals the evolution of SDECE techniques, with artificial neural networks, fuzzy logic, and regression emerging as prominent methods. Validation against previous review works confirms the consistency of results, while detailed bibliometric analysis enriches understanding of research patterns. This study is valuable for developing new models for cost and effort estimations in software engineering. The study (Swandari et al. (2023)) [137] examined the development of SDEE and its importance in project success. It identifies five key research topics from 2018 to 2022: ML approaches, algorithmic techniques, expert judgment, dataset analysis, and evaluation metrics. The ML approach is the most discussed topic across 27 journals, offering valuable insights for researchers to improve SDEE practices.

2. HYBRID MODELS IMPORTANT BACKGROUND WORKS

Table XXI analyses hybrid and other SEE models, highlighting essential background works in the field. It highlights various datasets, methods, optimizers, evaluation metrics, and model types. Hybrid models combine techniques like fuzzy inference systems, search-based algorithms, and neural networks to improve estimation accuracy. Standard evaluation metrics include Mean Relative Error, Mean Magnitude of Relative Error, Prediction Accuracy, and statistical measures. Model types include fuzzy inference systems, convolutional neural networks, and regression models. Recent works have focused on enhancing traditional models like COCOMO and COCOMO II using optimization algorithms and neural network architectures.

E. RQ5: FEATURE IMPORTANCE OF RELATED SEE BENCHMARK DATASETS USING ML MODELS

In this section, the research uses ML models to analyze the feature importance of related SEE benchmark datasets. It aims to identify the most influential features for accurately predicting software development efforts. It provides valuable insights into critical factors influencing estimation and facilitating the development of more effective estimation models.

TABLE XXI.
VARIOUS RESEARCH SEE WORKS WITH DIFFERENT HYBRID AND OTHER MODELS ON VARIOUS BENCHMARK DATASETS

Ref	Dataset	Method / Contribution	Optimizer	Evaluation Metric	Model Type	Year
[109]	Albrecht, Desharnais, and Maxwell	Analogy based	-	Kruskal–Wallis test	other	2024
[66]	Mendeley	Amplified COCOMO II	-	expert judgment and MRE	FEM	2021
[69]	Albrecht, China, Nasa93, Cocomo81, and Maxwell	hybrid search-based algorithms (firefly algorithm, genetic algorithm, and black hole optimization)	FF and Block Hole	CA	Other	2021
[71]	NASA 93	COCOMO II model	WOA	MMRE	FEM	2020
[80]	52 projects from State Gird Ltd., China	BiLSTM-CRF	-	Accuracy	Hybrid	2019
[83]	COCOMO 81, NASA	Cuckoo search-based hybrid models	Cuckoo search	MMRE, PRED(k)	Hybrid	2018
[85]	Real World	regression model for the Use Case Points	-	R2, MSE, SSE, RMSE, SSE 10-Fold p -value	Hybrid	2017
[87]	Albrecht, China, Cocomo-sdr, Finnish, Cocomo-81, Desharnais, Kemerer, Maxwell, Nasa93 (c1, c2, and c5)	Case-based solution adapting technique	-	SD,MdAR,MAR, RSD, and LSD	Other	2017
[119]	NASA93	Harmony Search Algorithm	-	MRE, MMRE	Hybrid	2017
[88]	Albrecht, COCOMO, Desharnais and NASA	random forest technique based on use case points	-	MMRE, MdMRE, PRED	Hybrid	2016
[94]	NASA	Optimizing Basic COCOMO Model using Simplified Genetic Algorithm	-	MD, MMRE, VAF and RMS	Hybrid	2016
[96]	COCOMO 81 and COCOMONASA	Multi-objective Genetic Algorithm (MOGA)	-	MMRE, PRED	Hybrid	2016
[100]	dataset with 448 software projects	analogy-based estimation through localization, and selective classification.	-	MRE, MMRE, PRED(25)	Other	2015
[102]	Nasa	Optimizing estimation Models using Firefly Algorithm	FF	RMSE, MAE, MMRE, VAF, and R2	Hybrid	2015

E. RQ5: FEATURE IMPORTANCE OF RELATED SEE BENCHMARK DATASETS USING ML MODELS

In this section, the research uses ML models to analyze the feature importance of related SEE benchmark datasets. It aims to identify the most influential features for accurately predicting software development efforts. It provides valuable insights into critical factors influencing estimation and facilitating the development of more effective estimation models.

1. FEATURE IMPORTANCE OF CHINA SEE BENCHMARK DATASET USING ML MODELS

The dataset file "CHINA" contains 499 data instances, each with 19 features. The dataset is related to software development, with features like AFP, input, output, inquiry, file, interface, and more. The "classes" feature suggests that this dataset may be used for classification tasks, with each row representing a data instance and each column representing a specific feature. Table XXII presents the ranked features of the software product evolution dataset China using SVM, GB, RF, and AB classification algorithms. It provides insights into their importance in predicting evolutionary patterns, aiding in informed decision-making in software product development and evolution.

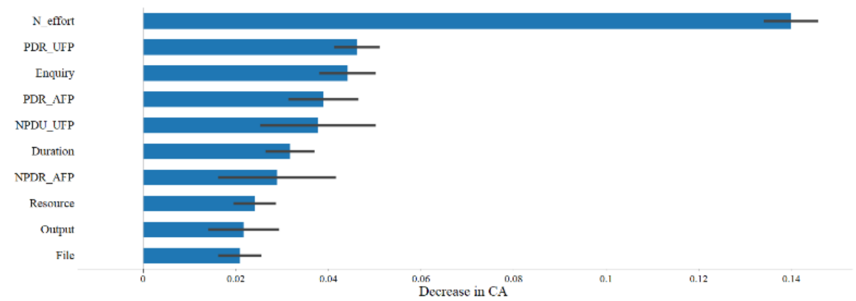
The SVM model with parameters like SVM type, C=1.0, ε=0.1, kernel: RBF, numerical tolerance: 0.001, iteration limit: 100, identified feature importance ranked as F18 (Duration), F12 (PDR_UFP), and F05 (Enquiry), with F18 having the highest mean importance of 0.1399, 0.0461, and 0.0441, respectively. The gradient boosting model, using sci-kit-learn, was evaluated for feature importance. The highest mean importance was given to Feature F18 (Duration), with a SD of ±0.0059. Other features, such as F14 (NPDU_UFP), F01 (ID), and F02 (AFP), had lower mean importance values. Other features, like F17 (Dev. Type) and F08 (Resource), had minimal impact on model predictions. Several features, like F03 (Input), F04 (Output), and F05 (Enquiry), had negligible influence on the model's performance. The model parameters ensured the replicability and effectiveness of the feature importance evaluation process. The Random Forest model's feature importance evaluation revealed that Feature F18 (Duration) has the highest mean importance of 0.7876, indicating its strong influence on model predictions. Feature F02 (AFP) and Feature F08 (Resource) also significantly contribute to the model. Feature F17 (Dev. Type) and NPDR_AFP also show notable mean importance values. Despite the unlimited number of considered features, moderate mean importance values of Feature F13 (PDR_UFP) and Feature F03 (Input)

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

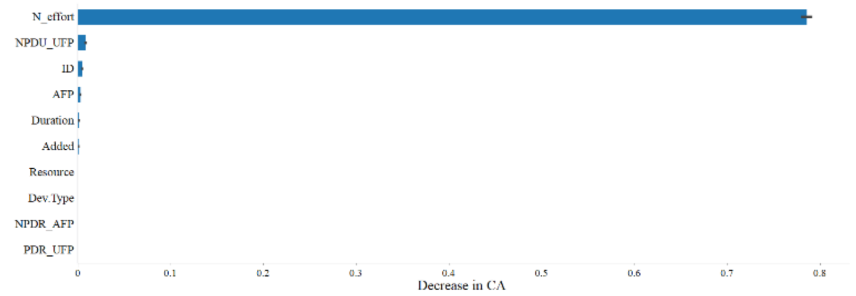
contribute to the model's predictive performance. The feature importance evaluation with the AdaBoost model uses a base estimator of decision trees and ten estimators. Feature F18 (duration) had the highest mean importance, indicating its significant influence on model predictions. Other features, like AFP and Resource, also showed substantial contributions. Despite the algorithm's SAMME and exponential loss, features like PDR_UFP and input showed moderate mean importance, contributing to the model's predictive performance. The AdaBoost model ensured robust feature importance evaluation, with minimal or negligible importance assigned to features like files and interfaces.

TABLE XXII.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION CHINA DATA SET WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

SVM		Gradient Boost		RF Tree		Adaboost	
Feature Rank (F #)	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**
R01 (F18)	0.1399±0.0059	R01 (F18)	0.7856±0.0059	R01 (F18)	0.5647±0.0107	R01 (F18)	0.7876±0.0113
R02 (F12)	0.0461±0.0049	R02 (F14)	0.0084±0.0015	R02 (F08)	0.0445±0.0056	R02 (F02)	0.0473±0.0069
R03 (F05)	0.0441±0.0061	R03 (F01)	0.0048±0.001	R03 (F12)	0.0437±0.0043	R03 (F08)	0.0381±0.0079
R04 (F11)	0.0389±0.0076	R04 (F02)	0.0028±0.001	R04 (F02)	0.0321±0.0028	R04 (F17)	0.0325±0.0041
R05 (F14)	0.0377±0.0125	R05 (F17)	0.0012±0.001	R05 (F04)	0.0184±0.0046	R05 (F15)	0.0148±0.0033
R06 (F17)	0.0317±0.0053	R06 (F08)	0.0008±0.001	R06 (F11)	0.0156±0.0029	R06 (F03)	0.0136±0.0015
R07 (F13)	0.0289±0.0127	R07 (F03)	0±0	R07 (F03)	0.0132±0.002	R07 (F13)	0.0136±0.0041
R08 (F15)	0.024±0.0046	R08 (F04)	0±0	R08 (F17)	0.0068±0.001	R08 (F01)	0.0112±0.0037
R09 (F04)	0.0216±0.0076	R09 (F05)	0±0	R09 (F05)	0.0064±0.0027	R09 (F04)	0.0112±0.001
R10 (F06)	0.0208±0.0047	R10 (F06)	0±0	R10 (F06)	0.006±0.0013	R10 (F12)	0.0088±0.002
R11 (F09)	0.0128±0.007	R11 (F07)	0±0	R11 (F13)	0.006±0.0022	R11 (F14)	0.0072±0.0027
R12 (F07)	0.0104±0.0027	R12 (F09)	0±0	R12 (F09)	0.0052±0.002	R12 (F05)	0.0028±0.001
R13 (F01)	0.0084±0.0023	R13 (F10)	0±0	R13 (F07)	0.0032±0.0016	R13 (F06)	0±0
R14 (F08)	0.008±0.0049	R14 (F11)	0±0	R14 (F01)	0.002±0.0013	R14 (F07)	0±0
R15 (F10)	0.0056±0.002	R15 (F12)	0±0	R15 (F15)	0.0012±0.001	R15 (F09)	0±0
R16 (F02)	0.0044±0.0056	R16 (F13)	0±0	R16 (F14)	0.0008±0.001	R16 (F10)	0±0
R17 (F16)	0±0	R17 (F15)	0±0	R17 (F10)	0±0	R17 (F11)	0±0
R18 (F03)	-0.0068±0.0067	R18 (F16)	0±0	R18 (F16)	0±0	R18 (F16)	0±0



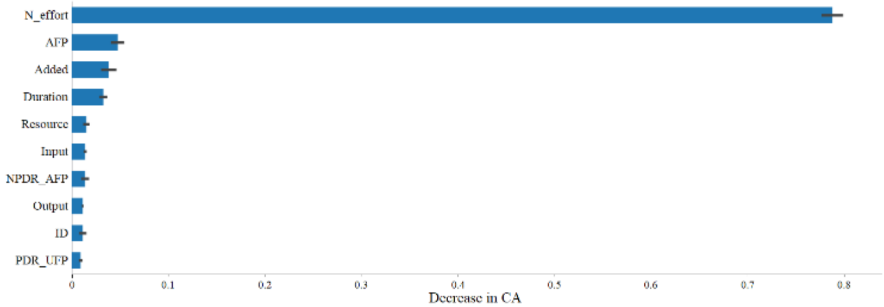
(A) SVM classifier



(B) Gradient Boosting Classifier

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

(C) Random Forest Classifier



(D) AdaBoost Classifier

FIGURE 14. Ranked Feature of the software product evolution COCOMO data set with SVM, GB, RF and AB Classification Algorithms

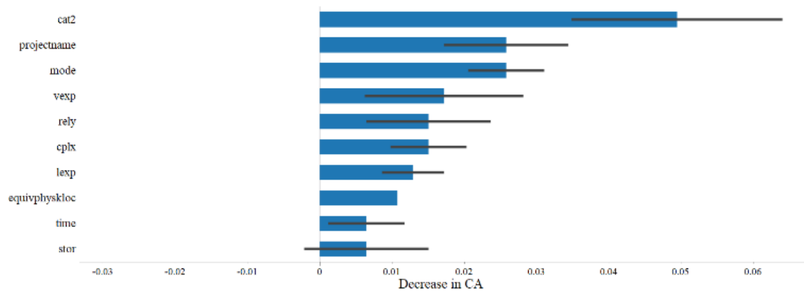
2. FEATURE IMPORTANCE OF COCOMO_NASA2 SEE BENCHMARK DATASET USING ML MODELS

There are 24 features in the "COCOMO_NASA2 " dataset that show how software products have changed over time. These features are ranked by how important they are for classification algorithms like SVM, GB, RF, and AB. This gives us an idea of how predictive the dataset is. The SVM model's attribute rankings place "cat2" first, then "projectname" and "mode," all of which make significant contributions to classification. "sced" appears least impactful, with a negative mean, suggesting minimal relevance in the model. These rankings provide valuable guidance for feature selection and model optimization in software product evolution analysis. The Gradient Boosting model, using sci-kit-learn, prioritizes "equivphyskloc" (F23) for classification, with a significant mean of 0.6151 and SD of 0.0315. Other features like "record number" and "time" also show importance, but some have a mean importance score of 0, indicating minimal relevance. These rankings provide valuable insights for feature selection and model refinement in software product evolution analysis shown in Table XXIII. The Random Forest model's feature "equivphyskloc" has a significant role in classification, with a mean importance of 0.3484. Other features, like "acap" and "cplx," have lower importance scores. However, "tool" and "virt" have negative importance scores, suggesting minimal predictive contribution. These rankings guide feature prioritization and model optimization in software product evolution analysis. The AdaBoost model's most influential feature is "equivphyskloc," with a mean importance of 0.6796. Other significant features include "projectname" and "time." Feature importance rankings provide insights for feature selection and refinement in software product evolution analysis, aiding decision-making processes.

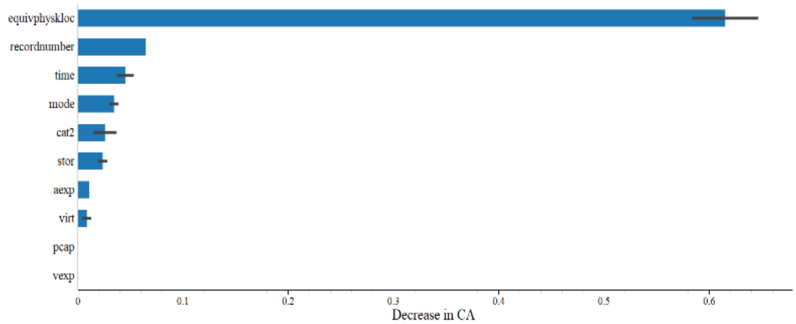
TABLE XXIII.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION "COCOMO_NASA2 DATA SET" WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

SVM		Gradient Boost		RF Tree		Adaboost	
Feature Rank (F #)	Mean ± Std	Feature Rank (F #)	Mean ± Std	Feature Rank (F #)	Mean ± Std	Feature Rank (F #)	Mean ± Std
R01 (F03)	0.0495±0.0146	R01 (F23)	0.6151±0.0315	R01 (F23)	0.3484±0.0241	R01 (F23)	0.6796±0.0275
R02 (F02)	0.0258±0.0086	R02 (F01)	0.0645±0	R02 (F08)	0.0602±0.0199	R02 (F01)	0.1183±0.0226
R03 (F07)	0.0258±0.0053	R03 (F11)	0.0452±0.008	R03 (F10)	0.028±0.011	R03 (F11)	0.0925±0.0086
R04 (F18)	0.0172±0.011	R04 (F07)	0.0344±0.0043	R04 (F17)	0.0215±0.0096	R04 (F15)	0.086±0.0254
R05 (F08)	0.0151±0.0086	R05 (F03)	0.0258±0.011	R05 (F04)	0.0172±0.0086	R05 (F18)	0.0839±0.0322
R06 (F10)	0.0151±0.0053	R06 (F12)	0.0237±0.0043	R06 (F01)	0.0151±0.0161	R06 (F08)	0.0817±0.0199
R07 (F19)	0.0129±0.0043	R07 (F16)	0.0108±0	R07 (F05)	0.0151±0.0086	R07 (F12)	0.0409±0.008
R08 (F23)	0.0108±0	R08 (F13)	0.0086±0.0043	R08 (F02)	0.0108±0.0068	R08 (F22)	0.0409±0.0158
R09 (F11)	0.0065±0.0053	R09 (F02)	0±0	R09 (F07)	0.0108±0.0068	R09 (F09)	0.0344±0.0125
R10 (F12)	0.0065±0.0086	R10 (F04)	0±0	R10 (F12)	0.0108±0.0068	R10 (F03)	0.0301±0.008
R11 (F13)	0.0043±0.0053	R11 (F05)	0±0	R11 (F13)	0.0086±0.008	R11 (F05)	0.028±0.0146

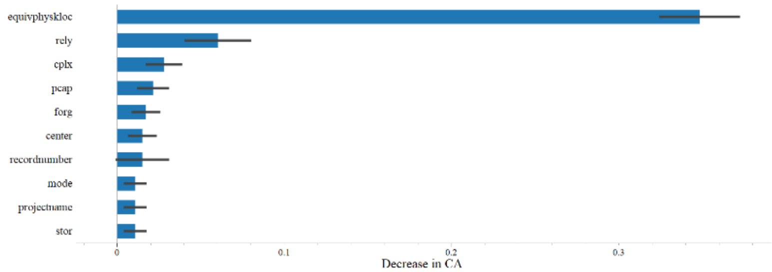
R12 (F16)	0.0043±0.0053	R12 (F06)	0±0	R12 (F11)	0.0022±0.008	R12 (F06)	0.0258±0.0086
R13 (F17)	0.0043±0.0053	R13 (F08)	0±0	R13 (F14)	0.0022±0.0043	R13 (F10)	0.0172±0.0086
R14 (F05)	0.0022±0.0043	R14 (F09)	0±0	R14 (F15)	0±0	R14 (F21)	0.0172±0.0086
R15 (F04)	0±0	R15 (F10)	0±0	R15 (F16)	0±0	R15 (F02)	0±0
R16 (F06)	0±0.0068	R16 (F14)	0±0	R16 (F18)	0±0.0152	R16 (F04)	0±0
R17 (F14)	-0.0043±0.0053	R17 (F15)	0±0	R17 (F19)	0±0.0096	R17 (F07)	0±0
R18 (F01)	-0.0065±0.0086	R18 (F17)	0±0	R18 (F20)	0±0.0068	R18 (F13)	0±0
R19 (F09)	-0.0065±0.0086	R19 (F18)	0±0	R19 (F21)	-0.0043±0.0086	R19 (F14)	0±0
R20 (F21)	-0.0065±0.011	R20 (F19)	0±0	R20 (F03)	-0.0086±0.008	R20 (F16)	0±0
R21 (F20)	-0.0129±0.0143	R21 (F20)	0±0	R21 (F22)	-0.0108±0.0118	R21 (F17)	0±0
R22 (F15)	-0.0151±0.0086	R22 (F21)	0±0	R22 (F09)	-0.0129±0.0043	R22 (F19)	0±0
R23 (F22)	-0.0237±0.008	R23 (F22)	0±0	R23 (F06)	-0.0151±0.0086	R23 (F20)	0±0



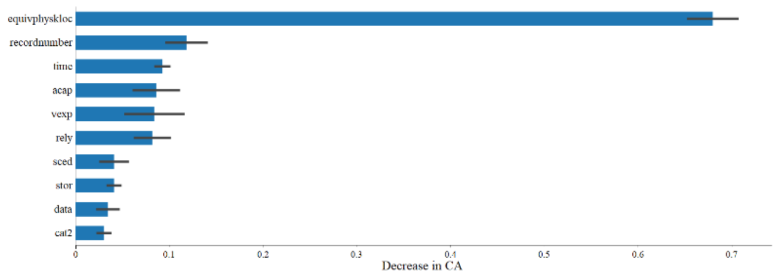
(A) SVM classifier



(B) Gradient Boosting Classifier



(C) Random Forest Classifier



(D) AdaBoost Classifier

FIGURE 15. **Ranked Feature of the software product evolution COCOMO data set with SVM, GB, RF and AB Classification Algorithms**

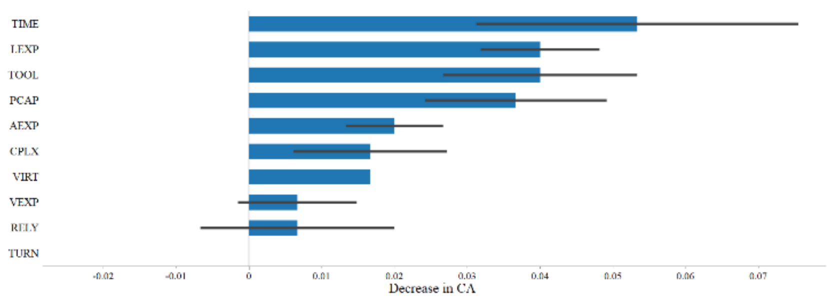
3. FEATURE IMPORTANCE OF COCOMO SEE BENCHMARK DATASET USING ML MODELS

The "COCOMO" dataset's SVM model reveals that key feature attributes, including RELY, LEXP, TOOL, PCAP, AEXP, and CPLX, significantly contribute to the model's predictive performance. RELY ranks highest, while LEXP is related to estimating software development efforts. TOOL influences project estimations, while PCAP is significant. AEXP is moderately important, and CPLX is lower but still significant, shaping SEEs. These attributes are crucial in predicting software development efforts. Table XXIVpresents the ranked feature importance of software product evolution datasets analyzed with SVM, Gradient Boosting, and Random Forest classification algorithms. Features such as RELY, LOC, and SCED exhibit notable importance across different algorithms, providing insights into their predictive power for estimating software development efforts.

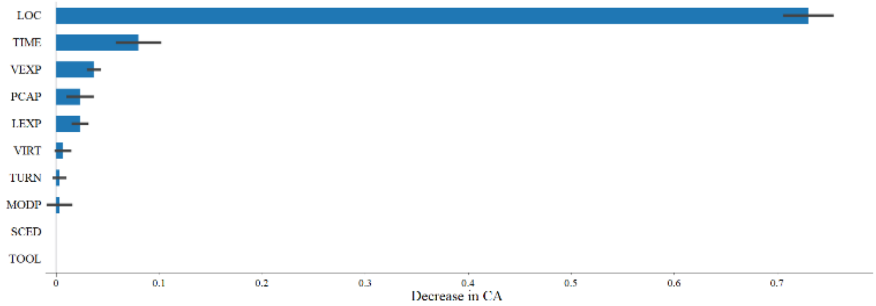
The Gradient Boosting model, Random Forest model, and AdaBoost algorithm are all important in estimating software effort. Lines of code, development time, and application experience are the most significant factors, with lines of code having the highest mean importance score of 0.73 ± 0.0245. Development time has a moderate importance score of 0.08 ± 0.0221, while application experience contributes to predictive accuracy. Programmer capability, programming language experience, and virtual machine volatility also play a role in the estimation model. The Random Forest model has a higher mean importance score of 0.3467 ± 0.0194, indicating its impact on project timelines. The use of software tools also significantly impacts project outcomes. Database size is moderate importance, and application experience and programming capability contribute to predictive accuracy. The AdaBoost algorithm, a tree base estimator with ten estimators, has the highest importance score of LOC, followed by time, AEXP, Lexp, and PCAP.

TABLE XXIV.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION "COCOMO DATASET" WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

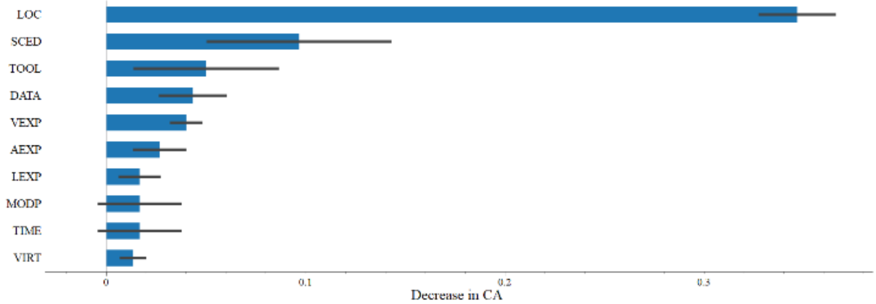
SVM		Gradient Boost		RF Tree		Adaboost	
Feature Rank (F #)	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**
R01 (F04)	0.0533±0.0221	R01 (F16)	0.73±0.0245	R01 (F16)	0.3467±0.0194	R01 (F16)	0.7767±0.0309
R02 (F12)	0.04±0.0082	R02 (F04)	0.08±0.0221	R02 (F15)	0.0967±0.0464	R02 (F04)	0.0933±0.0226
R03 (F14)	0.04±0.0133	R03 (F11)	0.0367±0.0067	R03 (F14)	0.05±0.0365	R03 (F11)	0.0667±0.0211
R04 (F10)	0.0367±0.0125	R04 (F10)	0.0233±0.0133	R04 (F02)	0.0433±0.017	R04 (F12)	0.02±0.0067
R05 (F09)	0.02±0.0067	R05 (F12)	0.0233±0.0082	R05 (F11)	0.04±0.0082	R05 (F09)	0.0167±0
R06 (F03)	0.0167±0.0105	R06 (F06)	0.0067±0.0082	R06 (F09)	0.0267±0.0133	R06 (F10)	0.0167±0.0149
R07 (F06)	0.0167±0	R07 (F07)	0.0033±0.0067	R07 (F04)	0.0167±0.0211	R07 (F01)	0±0
R08 (F01)	0.0067±0.0133	R08 (F13)	0.0033±0.0125	R08 (F12)	0.0167±0.0105	R08 (F02)	0±0
R09 (F11)	0.0067±0.0082	R09 (F01)	0±0	R09 (F13)	0.0167±0.0211	R09 (F03)	0±0
R10 (F07)	0±0	R10 (F02)	0±0	R10 (F06)	0.0133±0.0067	R10 (F05)	0±0
R11 (F05)	-0.0033±0.0067	R11 (F05)	0±0	R11 (F08)	0.0133±0.0125	R11 (F06)	0±0
R12 (F13)	-0.0033±0.0067	R12 (F08)	0±0	R12 (F05)	0.01±0.0133	R12 (F07)	0±0
R13 (F16)	-0.0033±0.0067	R13 (F09)	0±0	R13 (F07)	0.0067±0.0133	R13 (F08)	0±0
R14 (F02)	-0.0067±0.0082	R14 (F14)	0±0	R14 (F03)	0.0033±0.0194	R14 (F13)	0±0
R15 (F15)	-0.0067±0.0082	R15 (F15)	0±0	R15 (F01)	-0.0067±0.0226	R15 (F14)	0±0
R16 (F08)	-0.01±0.017	R16 (F03)	-0.0033±0.0067	R16 (F10)	-0.02±0.0067	R16 (F15)	0±0



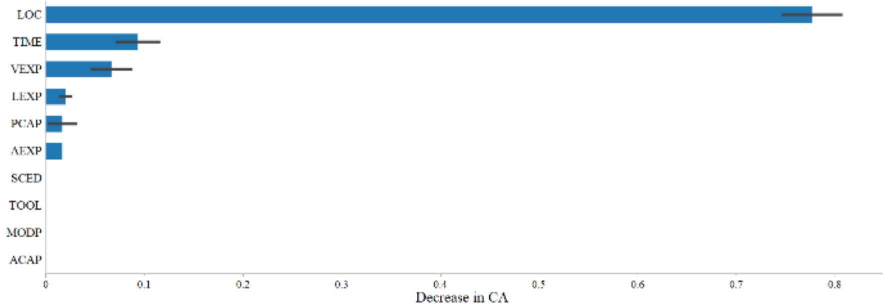
(A) SVM classifier



(B) Gradient Boosting Classifier



(C) Random Forest Classifier



(D) AdaBoost Classifier

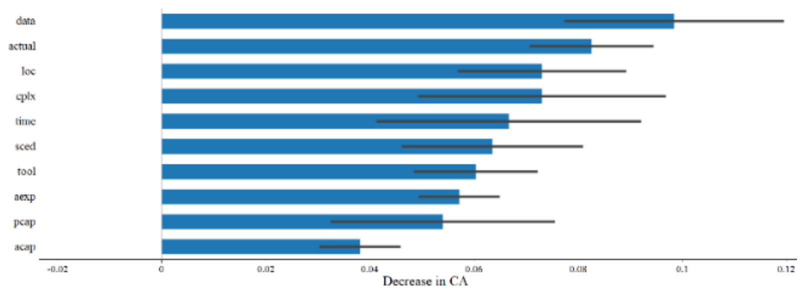
FIGURE 16. Ranked Feature of the software product evolution COCOMO data set with SVM, GB, RF and AB Classification Algorithms

4. FEATURE IMPORTANCE OF COCOMO81 SEE BENCHMARK DATASET USING ML MODELS

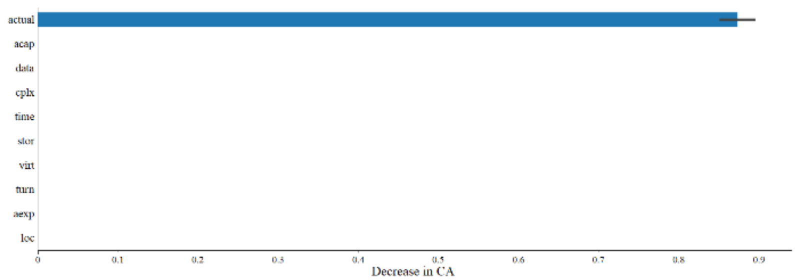
The "**COCOMO81**" dataset contains 63 instances with 17 features and one class, including reliability, complexity, and development time. A Support Vector Machine (SVM) model was trained on the dataset, and its feature ranking revealed that attributes related to data reliability, actual effort, and complexity are the top predictors. These attributes are crucial in predicting software development efforts, providing valuable insights for project planning and resource allocation. The Gradient Boosting model, developed using sci-kit-learn, uses 100 trees with a 0.1 learning rate and a maximum tree depth of 3. It ensures balanced complexity and is replicable for consistent results across runs. To prevent overfitting, nodes are stopped from splitting when they contain a maximum of two instances. The model's feature ranking analysis shows that attribute F17 is the most influential predictor, with a mean of 0.873 and an SD of 0.0224. The remaining features (F01 to F16) are negligible, indicating F17's significant contribution to the model's predictive power shown in Table XXV. The Random Forest model, with ten trees and unlimited features per tree, is non-replicable due to randomness. Tree depth and node splitting are unconstrained, but nodes can't split further when they contain five instances. The model's feature ranking analysis shows the attribute F17 is the most important, with a mean value of 0.4317 and an SD of 0.0431. Other attributes, F16, F02, and F06, also contribute to the model's predictive accuracy. The AdaBoost model uses decision trees as base estimators and consists of 10 estimators. The same is used for classification, and exponential loss is used for regression tasks. The attribute of the highest importance, F17, is found in the model, with a mean value of 0.873 and an SD of 0.0224. The remaining features (F01 to F16) have negligible importance, indicating that F17 significantly contributes to the model's predictive power.

TABLE XXV.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION "COCOMO81" DATA SET WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

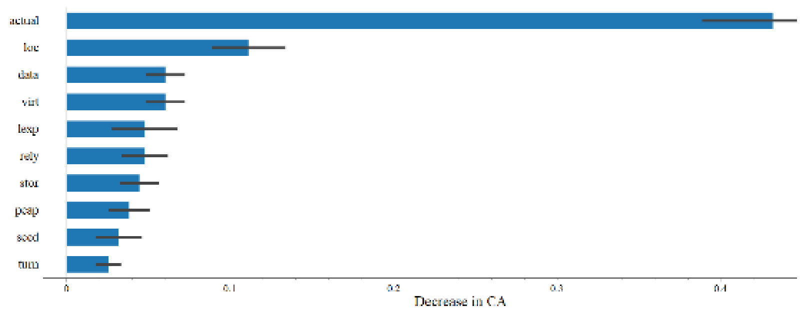
SVM		Gradient Boost		RF Tree		Adaboost	
Feature Rank (F #)	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**
R01 (F02)	0.0984±0.0211	R01 (F17)	0.873±0.0224	R01 (F17)	0.4317±0.0431	R01 (F17)	0.873±0.0224
R02 (F17)	0.0825±0.0119	R02 (F01)	0±0	R02 (F16)	0.1111±0.0224	R02 (F01)	0±0
R03 (F03)	0.073±0.0238	R03 (F02)	0±0	R03 (F02)	0.0603±0.0119	R03 (F02)	0±0
R04 (F16)	0.073±0.0162	R04 (F03)	0±0	R04 (F06)	0.0603±0.0119	R04 (F03)	0±0
R05 (F04)	0.0667±0.0254	R05 (F04)	0±0	R05 (F01)	0.0476±0.0142	R05 (F04)	0±0
R06 (F15)	0.0635±0.0174	R06 (F05)	0±0	R06 (F12)	0.0476±0.0201	R06 (F05)	0±0
R07 (F14)	0.0603±0.0119	R07 (F06)	0±0	R07 (F05)	0.0444±0.0119	R07 (F06)	0±0
R08 (F09)	0.0571±0.0078	R08 (F07)	0±0	R08 (F10)	0.0381±0.0127	R08 (F07)	0±0
R09 (F10)	0.054±0.0215	R09 (F08)	0±0	R09 (F15)	0.0317±0.0142	R09 (F08)	0±0
R10 (F06)	0.0381±0.0215	R10 (F09)	0±0	R10 (F07)	0.0254±0.0078	R10 (F09)	0±0
R11 (F08)	0.0381±0.0078	R11 (F10)	0±0	R11 (F08)	0.019±0.0119	R11 (F10)	0±0
R12 (F13)	0.0317±0.0362	R12 (F11)	0±0	R12 (F09)	0.0159±0.0174	R12 (F11)	0±0
R13 (F07)	0.0254±0.0238	R13 (F12)	0±0	R13 (F13)	0.0127±0.0063	R13 (F12)	0±0
R14 (F11)	0.0159±0.0201	R14 (F13)	0±0	R14 (F03)	0.0095±0.0078	R14 (F13)	0±0
R15 (F12)	0.0159±0.0142	R15 (F14)	0±0	R15 (F11)	0.0063±0.0078	R15 (F14)	0±0
R16 (F05)	0.0095±0.0127	R16 (F15)	0±0	R16 (F14)	0.0063±0.0127	R16 (F15)	0±0
R17 (F01)	0±0.0224	R17 (F16)	0±0	R17 (F04)	-0.0063±0.0078	R17 (F16)	0±0



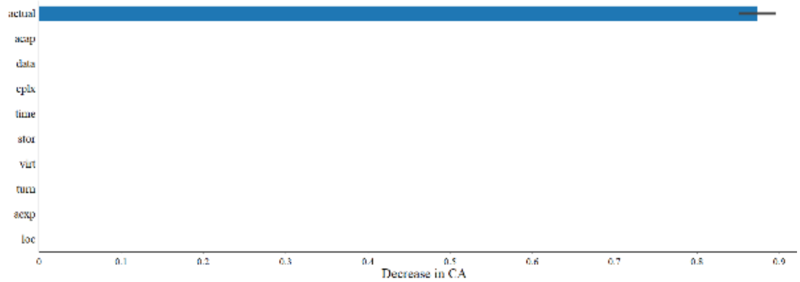
(A) SVM classifier



(B) Gradient Boosting Classifier



(C) Random Forest Classifier



(D) AdaBoost Classifier

FIGURE 17. **Ranked Feature of the software product evolution COCOMO data set with SVM, GB, RF and AB Classification Algorithms**

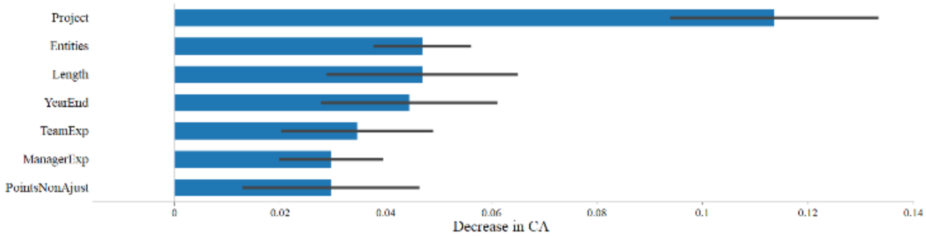
5. FEATURE IMPORTANCE OF DESHARNAIS SEE BENCHMARK DATASET USING ML MODELS

The “DESHARNAIS” dataset, consisting of 81 data instances and 12 attributes, analyses software product evolution. It uses SVM, Gradient Boosting, Random Forest, and AdaBoost classification algorithms to rank feature importance of project characteristics and effort-related metrics, providing insights into their predictive power for software development effort estimation shown in Table XXVI. An SVM model was used to rank the features, with Project, Length, and ManagerExp being the top-ranked. These features significantly influence language evolution within software projects. TeamExp, on the other hand, has a negative influence, suggesting lesser relevance in predicting language evolution.

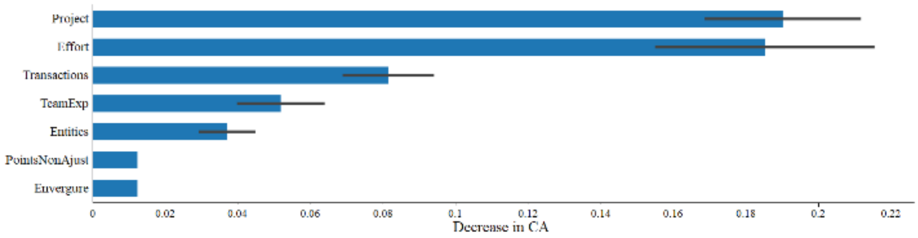
The analysis of the DESHARNAIS software product evolution dataset reveals the importance of various attributes in predicting software product evolution. The top-ranked feature is "rely," followed by "virt" and "turn." Other notable features include "data," "acap," and "pcap." However, "next" has a negative mean importance, suggesting it may not significantly contribute to the classification process. The Random Forest model on the DESHARNAIS dataset reveals that project is the most influential attribute, followed by effort and manager exp. However, "TeamExp" has a negative mean importance, suggesting limited relevance. These rankings provide valuable insights into critical factors driving software product evolution, aiding in informed decision-making for future development endeavors. The AdaBoost model ranks key attributes influencing software product evolution, with "Project" being the most crucial feature.

TABLE XXVI.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION “DESHARNAIS” DATA SET WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

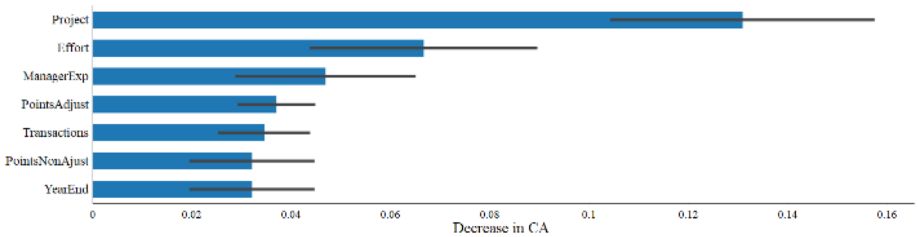
SVM		Gradient Boost		RF Tree		Ada-boost	
Feature Rank (F #)	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**
R01(F01)	0.1136±0.0198	R01(F01)	0.1136±0.0198	R01(F01)	0.1136±0.0198	R01(F01)	0.1136±0.0198
R02(F05)	0.0469±0.0181	R02(F06)	0.0469±0.0181	R02(F06)	0.0469±0.0181	R02(F06)	0.0469±0.0181
R03(F08)	0.0469±0.0092	R03(F07)	0.0469±0.0092	R03(F03)	0.0469±0.0092	R03(F02)	0.0469±0.0092
R04(F04)	0.0444±0.0167	R04(F02)	0.0444±0.0167	R04(F09)	0.0444±0.0167	R04(F11)	0.0444±0.0167
R05(F02)	0.0346±0.0144	R05(F08)	0.0346±0.0144	R05(F07)	0.0346±0.0144	R05(F03)	0.0346±0.0144
R06(F03)	0.0296±0.0099	R06(F10)	0.0296±0.0099	R06(F04)	0.0296±0.0099	R06(F05)	0.0296±0.0099
R07(F11)	0.0296±0.0167	R07(F11)	0.0296±0.0167	R07(F11)	0.0296±0.0167	R07(F08)	0.0296±0.0167
R08(F07)	0.0272±0.0144	R08(F03)	0.0272±0.0144	R08(F10)	0.0272±0.0144	R08(F10)	0.0272±0.0144
R09(F09)	0.0247±0.0234	R09(F04)	0.0247±0.0234	R09(F05)	0.0247±0.0234	R09(F04)	0.0247±0.0234
R10(F06)	0.0148±0.0164	R10(F05)	0.0148±0.0164	R10(F08)	0.0148±0.0164	R10(F07)	0.0148±0.0164
R11(F10)	-0.0049±0.0099	R11(F09)	-0.0049±0.0099	R11(F02)	-0.0049±0.0099	R11(F09)	-0.0049±0.0099



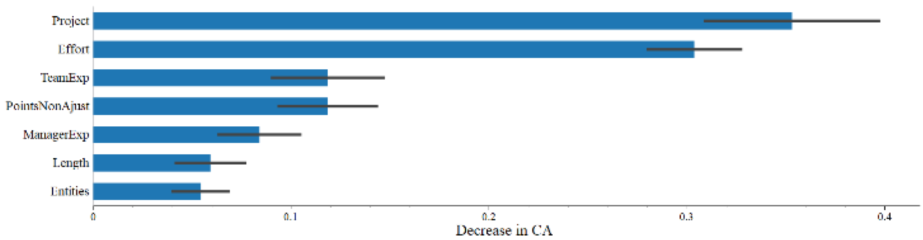
(A) SVM classifier



(B) Gradient Boosting Classifier



(C) Random Forest Classifier



(D) AdaBoost Classifier

FIGURE 18. **Ranked Feature of The Software Product Evolution COCOMO Data Set With SVM, GB, RF And AB Classification Algorithms**

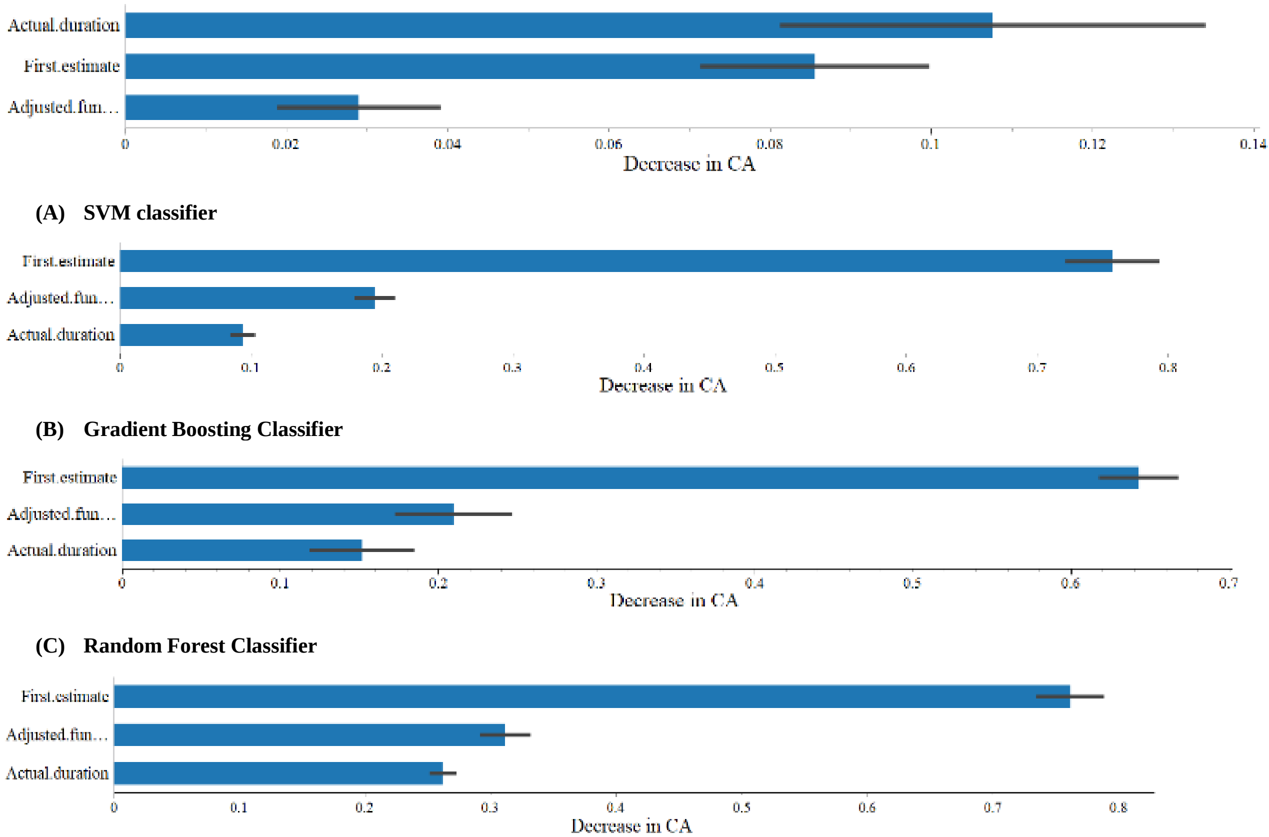
6. FEATURE IMPORTANCE OF KITCHENHAM SEE BENCHMARK DATASET USING ML MODELS

The **"KITCHENHAM "** dataset, consisting of 145 instances, has four features: Actual duration, Adjusted function points, First estimate, and Class. The SVM analysis reveals that Feature F01 (Actual duration) has the highest mean importance, significantly impacting model predictions. Feature F03 (First estimate) follows closely, contributing significantly to the model's performance. Feature F02 (Adjusted function points) ranks third, suggesting a lesser influence on prediction outcomes. The Gradient Boosting analysis of the dataset, using sci-kit-learn with 100 trees, 0.1 learning rate, and three maximum tree depths, reveals that Feature F03 (First estimate) is the most crucial feature, with a mean importance of 0.7572 and an SD of ±0.0358, significantly influencing model predictions shown in Table XXVII. Feature F02 (Adjusted function points) follows with a mean importance of 0.1945 and an SD of ±0.0154, while Feature F01 (Actual duration) ranks third with a lower mean value and SD. The Random Forest analysis of a dataset reveals that Feature F03 (First estimate) is the most essential feature, with a mean importance of 0.6428. Feature F02 (Adjusted function points) contributes significantly to the model's predictive performance, while Feature F01 (Actual duration) ranks third in importance, indicating its relevance in predicting the target variable. The AdaBoost analysis of the dataset, using a base estimator of decision trees and the SAMME algorithm for classification, reveals that Feature F03 (First estimate) is the most crucial feature, with a mean importance of 0.7614. Feature F02 (Adjusted function points) is also significant, with a mean significance of 0.3117. Feature F01 (Actual duration) is also substantial.

Table XXVII.
DETERMINATION OF RANKED FEATURE OF THE SOFTWARE PRODUCT EVOLUTION "KITCHENHAM" DATA SET WITH SVM, GB, RF AND AB CLASSIFICATION ALGORITHMS

SVM		Gradient Boost		RF Tree		Adaboost	
Feature Rank (F #)	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**	**Feature Rank (F #)**	**Mean ± Std**
R01 (F01)	0.1076±0.0264	R01 (F03)	0.7572±0.0358	R01 (F03)	0.6428±0.0252	R01 (F03)	0.7614±0.0271
R02 (F03)	0.0855±0.0142	R02 (F02)	0.1945±0.0154	R02 (F02)	0.2097±0.0372	R02 (F02)	0.3117±0.0202
R03 (F02)	0.029±0.0101	R03 (F01)	0.0938±0.0094	R03 (F01)	0.1517±0.0329	R03 (F01)	0.2621±0.0107

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771



(A) SVM classifier

(B) Gradient Boosting Classifier

(C) Random Forest Classifier

(D) AdaBoost Classifier

FIGURE 19. **Ranked Feature of the software product evolution COCOMO data set with SVM, GB, RF and AB Classification Algorithms**

7. ANALYSIS, OUTCOMES AND SUMMARY

Table XXII shows feature ranking results for the China software product evolution dataset using SVM, Gradient Boosting, Random Forest, and AdaBoost algorithms. Key features like R01, R02, and R03 consistently ranked as top predictors, indicating robustness. Key features like R01, R02, and R03 consistently ranked as top predictors, indicating robustness. Table XXIII (COCOMO_NASA2 Dataset) Feature R01 (F03) consistently ranks high across all algorithms, indicating its importance in predicting software product evolution. The importance rankings of features like R01 (F23) and R02 (F01) show significant differences, indicating algorithm-dependent variations in feature relevance. Feature R01 (F16) significantly influences the COCOMO dataset's (Table XXIV) software product evolution prediction, with variations across algorithms highlighting the need for optimal analysis. Feature R01 (F04) and R02 (F12) maintain significant ranks across different algorithms. For the dataset COCOMO81 (Table XXV), The top-ranked feature (R01) has the highest mean value among all algorithms, while the most crucial feature (F17) varies across the board. SVM ranks F02 highest, while Gradient Boost, RF Tree, and AdaBoost consider F17 the most important. Feature R01 (F01) consistently ranks highest across all algorithms, which shows that it has a significant impact on how software products change over time for the "DESHARNAIS" dataset (Table XXVI). Other features like R02, R03, and R04 also show notable ranks across different algorithms. Feature R01 (F01) consistently ranks highest across all algorithms, indicating its significant influence on software product evolution for the "KITCHENHAM" dataset (Table XXVII). Feature R02 (F03) and R03 (F02) also show notable ranks across different algorithms.

The literature survey explores Software Effort Estimation (SEE) methodologies, including Expert Judgement (EJ), Formal Estimation Models (FEM), and Machine Learning (ML). EJ offers a quick, straightforward approach, while FEMs provide a structured framework. Algorithmic models offer efficient calculations, while non-algorithmic models handle project specifics. The optimal SEE method depends on project characteristics and resource availability.

VI. DISCUSSIONS

The review discusses the current state of SEE, highlighting the growing use of ML models and the importance of feature importance analysis. Future research should explore hybrid approaches, explainable AI methods, and domain-specific models to improve reliability and robustness. We started our work by framing four research questions. Later, we selected

82 studies from a vast collection of different digital libraries. From these studies, we tried to answer the following research questions:1) The metrics for a SEE model's accuracy measure.2) Classification of SEE models. 3) Models that have been popularly preferred for SEE in recent times. 4) Benchmark datasets available for SEE. In recent studies, the accuracy of SEE models was predicted using ML ensemble techniques, with better results [77]. This study analyses SEE models and their evaluation metrics based on 68 studies. It reveals SEE models' diverse nature and functionalities and the importance of robust evaluation methodologies. The paper highlights the rise of ML-based models in SEE, indicating a shift towards data-driven estimation practices. From our study, ML-based models are in first place with a 54% share in selected studies compared to other techniques. The ML-based models, which include artificial neural networks, were also used and are proven to be leading techniques in effort estimation [65-66]. Studies are using several optimization techniques like particle swarm optimization [105] [108–109], firefly optimization [102], artificial bee colony [90], cuckoo search [83], whale optimization algorithm [4] [69], and neural network models [71]. Wen et al. (2012) [17] conducted a comprehensive analysis of ML-based software development (SDEE) effort estimation models over two decades. They identified 84 studies and found that eight ML techniques have been used in SDEE models, demonstrating close-to-acceptable estimation accuracy. However, their industrial application remains limited, highlighting the need for more efforts and incentives. Baskeles et al.'s study (2007) [19] proposes an ML model to estimate software development effort accurately, a crucial aspect for project success within budget and schedule constraints. The study evaluates these models on public datasets and real-world data from Turkish software organizations, emphasizing adaptability in model selection. Mahmood et al. (2022) [64] conducted a systematic literature review on SEE accuracy, revealing that ML techniques outperform solo techniques, especially in ensemble effort estimation, providing valuable insights for researchers and practitioners to improve SEE practices.

The study explores the integration of ML and Hybrid models for SEE, focusing on their applications and challenges (Table XX and Table XXI). It highlights using Bayesian Network models, artificial neural networks, and gradient-boosting regressors to improve estimation accuracy. Optimizers like swarm intelligence and genetic algorithms refine model performance, while evaluation metrics like MAE and MSE assess model efficacy. The study also highlights the integration of hybrid models, which combine techniques like fuzzy inference systems and search-based algorithms to enhance estimation accuracy. The analysis covers various datasets, methods, optimizers, and evaluation metrics, providing valuable insights for future research and development in SEE. The datasets, focusing on Effort Estimation in Software Engineering, include project characteristics, team characteristics, and historical data. It aids in predicting the human resources needed for software development projects. The dataset can be used for research and analysis tasks like comparison, factor identification, model development, and predictive modelling. However, potential challenges include missing data, data quality issues, and overfitting models. The dataset can evaluate estimation techniques, identify factors influencing software development efforts, and develop ML models for informed resource allocation. The research also examines the feature importance of SEE benchmark datasets using ML models. It aims to identify critical factors influencing software development efforts and develop more effective estimation models. The analysis reveals the significance of various features in predicting software product evolution patterns using algorithms like SVM, GB, RF, and AB. The rankings guide feature selection and model refinement, enhancing estimation accuracy. The DESHARNAIS SEE benchmark dataset highlights the importance of project characteristics and effort-related metrics in predicting language evolution within software projects. The study emphasizes the critical role of feature importance analysis in SEE, guiding the development of more robust estimation models. Table XXVIII summarizes different studies on software effort estimation, including the datasets, models, results, and limitations identified. It highlights the varied methodologies and difficulties in achieving accurate and generalizable predictions.

In this, we also provide proposed model with comparative analysis. This systematic literature review analyzes software effort estimation models, uses ML algorithms for feature importance analysis, and compares expert judgment, formal estimation techniques, and ML-based and hybrid approaches.

TABLE XXVIII.
SUMMARY OF STUDIES ON SOFTWARE EFFORT ESTIMATION DATASETS, MODEL ANALYSES, RESULTS, AND IDENTIFIED LIMITATIONS

Study & Ref	Description & Datasets	Model Analysis	Result Analysis	Limitations & Gaps
Ramacharan et al. (2016) [140]	Investigated effort estimation for GSD projects using calibrated parametric models (COCOMO II, SLIM, and scheduling-based).	Employed calibration techniques on existing parametric models.	Calibrated models improved accuracy for GSD projects compared to uncalibrated models. Scheduling-based model showed better performance.	Limited by the scope of the datasets used. more research is needed to test them on different datasets and environments.

Hosni et al. (2017) [141]	Investigated the use of Random Subspace ensembles with Classical Analogy for software effort estimation. Used seven datasets (six from PROMISE, one from ISBSG).	Applied Classical Analogy as base estimator and Random Subspace for ensemble creation. Used two combination rules (average and median).	Ensembles outperformed solo Classical Analogy in most cases. Median rule generally showed better performance than average rule.	Lacks baseline comparison with other ML techniques. Limited evaluation metrics (SA). No statistical significance testing.
Sarro et al. (2018) [116]	Proposed a heterogeneous ensemble effort estimation model using dynamic selection of regressors. Real world Dataset is used.	Combined multiple regression models with classifiers for dynamic selection.	Ensemble model outperformed individual regressors.	The DES method is good but takes a long time and needs lots of data to work well.
Amaral et al. (2019) [142]	The study uses XGBoost, a ML technique, to estimate software effort. It analyses four datasets: Cocomo81, Desharnais, Kemerer, and Maxwell.	Applied XGBoost with different cross-validation techniques (Leave-One-Out, 15-fold, 10-fold, 5-fold).	XGBoost achieved promising results across datasets, with high PRED values. Performance varied across datasets and cross-validation methods.	Lacks baseline models, and in-depth feature engineering, statistical significance testing.
Rahman et al. (2019) [115]	Compared RBFNN, ELM, and Decision Tree for software effort estimation based on software size categories (small, medium, large). Used an unspecified dataset.	Categorized software into three size groups and applied respective algorithms.	Decision Tree performed best for small and medium-sized software, while ELM excelled for large-sized software.	Limited to three algorithms and software size categories. The studies conceive on a single ML algorithm limits its generalizability to different software project sizes.
Villalobos-Arias et al. (2020)	Evaluated the effectiveness of Random Search (RS) for hyperparameter tuning in Support Vector Regression (SVR) for software effort estimation.	Compared RS-tuned SVR with grid search (GS)-tuned SVR, ridge regression, and non-tuned models.	RS-tuned SVR achieved similar performance to GS-tuned SVR and improved prediction stability.	Limited to SVR algorithm and four datasets from ISBSG repository. Lacks details on data preprocessing and specific hyperparameter settings.
Rankovic et al. (2021) [120]	Investigated the use of ANN architectures optimized by Taguchi method for software effort estimation. Used COCOMO81, COCOMO2000, and Kemerer datasets.	Applied two ANN architectures and Taguchi's orthogonal arrays for hyperparameter tuning.	ANN models outperformed traditional parametric models (COCOMO2000) of MMRE.	Validation on more datasets needed for generalization Explore different ANN architectures, consider cost-effect functions, investigate orthogonal vector plans for error minimization
Khan et al. (2021) [117]	Investigated the use of metaheuristic algorithms (GWO, SB) for optimizing DNN parameters in software effort estimation. Used NASA, COCOMO81, and Maxwell datasets.	Applied GWO and SB for optimizing DNN initial weights and learning rates. Compared with other metaheuristic algorithms.	GWO outperformed other metaheuristics of accuracy. GWDNNSB model showed better performance than traditional models.	Lacks comprehensive comparison of DNN architectures. Limited evaluation metrics. Only looked at GWO and SB, not other methods.
Fávero et al. (2022) [143]	Investigated the use of pre-trained embedding models (context-less and contextualized) for software effort estimation based on requirements texts. Real world Dataset is used.	Applied fine-tuned pre-trained models as input to a deep learning architecture with linear output.	Contextualized models, especially BERT, showed promising results with low MAE and standard deviation.	Limited scope with focus on pre-trained embeddings and deep learning. Limited dataset size and potential bias. Lack of comparison with other ML techniques
Ali et al. (2023) [144]	Proposed a heterogeneous ensemble model combining UCP, EJ, and ANN for software effort estimation. Used ISBSG dataset and industrial case studies.	Applied linear combination rule for ensemble combination.	Ensemble model outperformed standalone models (UCP, EJ, ANN) of accuracy.	Choosing the best way to combine different models is hard. The method might not work for all problems. We don't know how well it works with different data or model settings.
Jadhav et al. (2023) [145]	Proposed an Omni-Ensemble Learning (OEL) approach combining static and dynamic ensemble selection for software effort estimation. Used Finnish and Maxwell datasets.	Applied OEL with multiple ML models.	OEL outperformed individual ML models of various evaluation metrics.	High computational cost and complex interpretation. Limited dataset evaluation. Data dependency: The model's performance depends on the quality and relevance of the data used.
Alsheikh et al. (2023) [130]	Investigated the use of Grey Wolf Optimization (GWO) for optimizing COCOMO model parameters. Used NASA18 dataset.	Compared GWO with other metaheuristic algorithms (ZOA, MFO, PDO, WSO) for parameter tuning.	GWO outperformed other algorithms in terms of various evaluation metrics.	The focus on a single model (GWO).Limited dataset evaluation.
Hameed et al. (2023) [121]	Investigated the use of GA to optimize CBR parameters for software effort estimation. Used unspecified datasets (potentially PROMISE).	Combined CBR with GA for parameter tuning (k-nearest neighbours).	CBR-GA model outperformed conventional CBR in terms of accuracy.	Limited scope with focus on CBR and GA. Lack of comprehensive performance evaluation.
Mustafa et al. (2024) [113]	Investigated the use of Random Forest for early-stage software effort estimation using the SEERA dataset.	Employed oversampling and feature selection techniques to improve model performance.	Random Forest achieved better accuracy than random guessing.	Limited to Random Forest algorithm. Lack of comparison with other ML techniques.

Kassaymeh et al. (2024) [111]	Investigated the use of GWO to optimize FCNN parameters for software effort estimation. Real World Dataset is used for this study.	Employed GWO for FCNN parameter optimization. Compared with other metaheuristic algorithms.	GWO-FC outperformed other methods in terms of accuracy.	The GWO-FC model works well but might only work well with certain types of data.
Sharma et al. (2024) [103]	Investigated the use of MI-APNIA for optimizing COCOMO model parameters. Compared with other metaheuristic algorithms. Used Real world Dataset	Employed MI-APNIA for COCOMO parameter tuning.	MI-APNIA outperformed other metaheuristic algorithms and basic COCOMO model in terms of MMRE.	Limited scope with a focus on the COCOMO model and MI-APNIA. The MI-APNIA algorithm is good at predicting how long software will take to build, but not all time for all data.
Proposed Method	**CHINA, COCOMO-NASA2, COCOMO, COCOMO81, DESHARNAIS, KITCHENHAM**	**SVM, AdaBoost, Gradient Boost, Random Forest**	**Feature Rank (F #) and Mean ± Std for feature importance**	**Feature importance analysis, ML model comparison**

The study explores the current state of SEE using ML models and analyses the importance of features. It highlights the shift towards data-driven estimation practices and the effectiveness of ensemble techniques. The study also explores the integration of ML and hybrid models, highlighting the use of Bayesian Networks and artificial neural networks for improved estimation accuracy.

VII. THREATS TO VALIDITY

This section critically analyzes potential biases and limitations that could impact the reliability and generalizability of the study, including threats to internal, external, and construct validity.

A. Internal Validity Threats

The study's findings' accuracy and reliability can be significantly impacted by internal validity threats in the review process, such as selection bias, data extraction inconsistencies, and publication bias. **Selection Bias:** The review process may have introduced bias, especially if inclusion criteria favour certain research types. This could lead to overrepresenting specific methodologies or outcomes and potentially bias the results. **Data Extraction (Quality and Heterogeneity) Bias:** The review's studies may differ in data quality, methodology, and context. These differences could affect the comparability of the results. Inconsistency in data extraction from selected studies can introduce bias in analysing and interpreting results and conclusions. Additionally, the subjective judgments influencing the scoring system could lead to inconsistent evaluations and potentially biased interpretations of the data. Therefore, careful standardized scoring systems are crucial for internal validity. **Publication and Researcher Bias:** Publication bias occurs when the focus is on published studies, potentially excluding unpublished or negative results. This bias can lead to overestimating the effectiveness of ML-based effort estimation. Subjective judgments may influence the study's relevance and quality scoring system. This can lead to inconsistent evaluations. If not standardized, it might also result in biased interpretations.

B. External Validity Threats

External validity threats, including limited generalizability, dataset scope, and contextual influence, may limit the review's applicability to diverse software development scenarios and evolving technological landscapes. The research might only apply to some kinds of software projects. The findings might only be accurate in the present as technology changes. **Generalizability:** The review's findings may be limited in applicability to other datasets or real-world projects due to a heavy reliance on specific benchmark datasets like COCOMO and DESHARNAIS. The study's findings may not apply to various software development contexts, project sizes, or domains due to the heterogeneity of the included studies. **Limited Dataset Scope and Technological Advancements:** The six benchmark datasets may not capture the full diversity of software development scenarios. This could lead to overestimating the applicability of ML models in different contexts. The review's findings may be limited due to the rapid advancements in ML techniques. **Contextual Factors:** The impact of organizational culture and project management practices on SEE might be noticed. Other contextual factors may also need to be adequately considered. This could affect the accuracy of the effort estimation.

C. Construct Validity

Construct Validity ensures consistent evaluation across studies by aligning accuracy metrics and operational definitions with software effort estimation in ML models. **Accuracy Metrics Definitions and Operationalization of Variables:** The definition and measurement of software effort can vary across studies, making it difficult to compare results. ML techniques used in SEE can also differ further. Evaluating ML models for SEE solely based on accuracy metrics may not provide a complete understanding of their performance. It is crucial to consider additional factors like interpretability and robustness when assessing the effectiveness of these models. **Feature Importance Interpretation:** The analysis of feature importance in ML is affected by the methodologies used. Different algorithms may output different conclusions about which features are essential. Inconsistent results in feature selection can arise due to variations in algorithmic approaches. The need for clear definitions for important terms and concepts can make analysing and combining research results challenging.

VIII. CONCLUSION AND FUTURE SCOPE

This article is a state-of-the-art review of various effort estimation models. This paper focuses on qualitative analysis of all the literature in SEE using expert judgment, formal estimation techniques, ML-based techniques, and hybrid techniques. This comprehensive analysis examines 69 studies published from 2015 to 2024, emphasizing contemporary methodologies and categorizations within the SEE field. The article discusses the prevalent SEE models, the contribution of machine learning models to SEE improvement, the metrics utilized to assess the accuracy of SEE models, and the significance of benchmark datasets. Most studies have used the COCOMO dataset, and the review emphasizes the need for additional research in machine learning, ensemble learning, and hybrid approaches to effort estimation. Furthermore, the investigation digs into novel optimization techniques and sophisticated neural network models. The article presents a variety of viewpoints that merit additional investigation in this field. The study analyzed 521 articles on software effort estimation (SEE) and found a significant shift towards machine learning models, with 59% of studies favouring this approach. Hybrid models were explored in 16% of studies, indicating an ongoing search for better solutions. This study examines the use of ML models like SVM, AdaBoost, Gradient Boost, and Random Forest in estimating software effort using six benchmark datasets. It highlights the significance of feature selection in SEE and contributes to improving software effort estimation practices by highlighting crucial attributes in SEE.

Future Work: AI methods like LIME and SHAP can improve transparency and stakeholder trust in machine learning models. Tailored models for specific industries or project types can enhance accuracy. Hybrid approaches can create robust estimation methodologies. Improving benchmark datasets, studying ensemble techniques, and exploring advanced optimization algorithms can improve estimation accuracy. Understanding model transferability across projects is crucial for real-world applications. Continuous improvement in datasets, ensemble techniques, and optimization algorithms will enhance ML model effectiveness. However, challenges persist in achieving precise estimates, highlighting the need for further exploration and innovation in software effort estimation methodologies.

ACKNOWLEDGMENT

The authors would like to thank Deanship of Scientific Research at Majmaah University for supporting this work under Project Number xxxx-xxx

REFERENCES

[1] Pressman, R.S., 2005. Software engineering: a practitioner's approach. Palgrave macmillan. https://www.mlsu.ac.in/econtents/16_EBOOK-7th_ed_software_engineering_a_practitioners_approach_by_roger_s._pressman_.pdf
[2] Rak, K., Car, Ž. and Lovrek, I., 2019. Effort estimation model for software development projects based on use case reuse. Journal of Software: Evolution and Process, 31(2), p.e2119. https://doi.org/10.1002/smr.2119
[3] Jorgensen, M. and Grimstad, S., 2010. The impact of irrelevant and misleading information on software development effort estimates: A randomized controlled field experiment. IEEE Transactions on Software Engineering, 37(5), pp.695-707, doi: 10.1109/TSE.2010.78
[4] Zaid, A., Selamat, M.H., Ghani, A.A.A., Atan, R. and Wei, K.T., 2008. Issues in software cost estimation. IJCSNS Int J of Computer Science and Network Security, 8(11), pp.350-356. http://paper.ijcsns.org/07_book/200811/20081150.pdf
[5] Kaushik, A., Verma, S., Singh, H.J. and Chhabra, G., 2017. Software cost optimization integrating fuzzy system and COA-Cuckoo optimization algorithm. International Journal of System Assurance Engineering and Management, 8(2), pp.1461-1471. https://doi.org/10.1007/s13198-017-0615-7
[6] Jorgensen, M. and Shepperd, M., 2006. A systematic review of software development cost estimation studies. IEEE Transactions on software engineering, 33(1), pp.33-53.doi: 10.1109/TSE.2007.256943
[7] Shahwaiz, S.A., Malik, A.A. and Sabahat, N., 2016, December. A parametric effort estimation model for mobile apps. In 2016 19th International Multi-Topic Conference (INMIC) (pp. 1-6). IEEE. doi: https://doi.org/10.1109/INMIC.2016.7840114
[8] Ruju, Shah., Vrunda, K., Shah., Anuja, Nair., Tarjni, Vyas., Shivani, Desai., Sheshang, Degadwala. (2022). Software Effort Estimation using Machine Learning Algorithms. doi: https://doi.org/10.1109/ICECA55336.2022.10009346
[9] Rashid, J., Kanwal, S., Wasif Nisar, M., Kim, J., & Hussain, A. (2022). An Artificial Neural Network-Based Model for Effective Software Development Effort Estimation. Computer Systems Science and Engineering, 44(2). https://doi.org/10.32604/csse.2023.026018
[10] Purwanto, A., & Manik, L. P. (2023). Software Effort Estimation Using Logarithmic Fuzzy Preference Programming and Least Squares Support Vector Machines. Scientific Journal of Informatics, 10(1), 1-12. doi: https://doi.org/10.15294/sji.v10i1.39865
[11] Beesetti, K. K., Bilgaiyan, S., & Mishra, B. S. P. (2023). Software Effort Estimation through Ensembling of Base Models in Machine Learning using a Voting Estimator. International Journal of Advanced Computer Science and Applications, 14(2). DOI: https://doi.org/10.14569/IJACSA.2023.0140222
[12] Jørgensen, M. (2023). Improved measurement of software development effort estimation bias. Information and Software Technology, 157, 107157. DOI: https://doi.org/10.1016/j.infsof.2023.107157
[13] Rashid, C. H., Shafi, I., Ahmad, J., Thompson, E. B., Vergara, M. M., Diez, I. D. L. T., & Ashraf, I. (2023). Software Cost and Effort Estimation: Current Approaches and Future Trends. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3312716
[14] Ritu, & Bhambri, P. (2023). Software effort estimation with machine learning–A systematic literature review. Agile software development: Trends, challenges and applications, 291-308. DOI: https://doi.org/10.1002/9781119896838.ch15
[15] Kitchenham, B. and Charters, S., 2007. Guidelines for performing systematic literature reviews in software engineering EBSE Technical Report EBSE-2007-01. Keele, Staffs, and Durham, UK. https://legacyfileshare.elsevier.com/promis_misc/525444systematicreviewsguide.pdf
[16] Huang, J., Li, Y.F. and Xie, M., 2015. An empirical analysis of data preprocessing for machine learning-based software cost estimation. Information and software Technology, 67, pp.108-127. DOI: https://doi.org/10.1016/j.infsof.2015.07.004
[17] Wen, J., Li, S., Lin, Z., Hu, Y. and Huang, C., 2012. Systematic literature review of machine learning based software development effort estimation models. Information and Software Technology, 54(1), pp.41-59.DOI: https://doi.org/10.1016/j.infsof.2011.09.002
[18] Sommerville, I., 1996. Software process models. ACM computing surveys (CSUR), 28(1), pp.269-271. DOI: https://doi.org/10.1145/234313.234420
[19] Baskeles, B., Turhan, B. and Bener, A., 2007, November. Software effort estimation using machine learning methods. In 2007 22nd

international symposium on computer and information sciences (pp. 1-6). IEEE. doi: https://doi.org/10.1109/ISCIS.2007.4456863

[20] Willmott, Cort J.; Matsuura, Kenji (December 19, 2005). "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance". Climate Research. 30: 79–82. doi: https://doi.org/10.3354/cr030079

[21] Rousseeuw, P. J.; Croux, C. (1993). "Alternatives to the median absolute deviation". Journal of the American Statistical Association. 88 (424): 1273–1283. doi: https://doi.org/10.1080/01621459.1993.10476408

[22] Golub, Gene; Charles F. Van Loan (1996). Matrix Computations – Third Edition. Baltimore: The Johns Hopkins University Press. p. 53. ISBN 0-8018-5413-X. https://pages.stat.wisc.edu/~bwu62/771/golub1996.pdf

[23] Willmott, Cort; Matsuura, Kenji (2006). "On the use of dimensioned measures of error to evaluate the performance of spatial interpolators". International Journal of Geographical Information Science. 20: 89–102. doi: https://doi.org/10.1080/13658810500286976

[24] Bland, J.M.; Altman, D.G. (1996). "Statistics notes: measurement error". BMJ. 312 (7047): 1654. doi:10.1136/bmj.312.7047.1654. https://www.bmj.com/content/312/7047/1654

[25] Foss T, Stensrud E, Kitchenham B, Myrtveit I (2003) A simulation study of the model evaluation criterionmmre. IEEE Trans SoftwEng 29(11):985–995. doi: https://doi.org/10.1109/TSE.2003.1245300

[26] Stensrud E, Foss T, Kitchenham B, Myrtveit I. A further empirical investigation of the relationship between MRE and project size. EmpirSoftw Eng. 2003;8(2):139-161. DOI: https://doi.org/10.1023/A:1023010612345

[27] Farr, L. and Zagorski, H.J., 1964. Factors that Affect the Cost of Computer Programming. Volume Ii. a Quantitative Analysis. SYSTEM DEVELOPMENT CORP SANTA MONICA CA. https://dl.acm.org/doi/pdf/10.1145/1142620.1142625

[28] Nelson, E.A., 1967. Management handbook for the estimation of computer programming costs. SYSTEM DEVELOPMENT CORP SANTA MONICA CA. https://apps.dtic.mil/sti/tr/pdf/AD0648750.pdf

[29] Anda, B., Angelvik, E. and Ribu, K., 2002, December. Improving estimation practices by applying use case models. In International Conference on Product Focused Software Process Improvement (pp. 383-397). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-36209-6_32

[30] Bundschuh, M., & Dekkers, C. (2008). Estimation of Data Warehouses, Web-Based Applications: Software Reuse and Redevelopment. The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement, 433-452. https://doi.org/10.1007/978-3-540-68188-5_16

[31] Grimstad, S. and Jørgensen, M., 2006, September. A framework for the analysis of software cost estimation accuracy. In Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering (pp. 58-65). https://doi.org/10.1007/3-540-36209-6_32

[32] Sinha, R.R., Gora, R.K. (2021). Software Effort Estimation Using Machine Learning Techniques. In: Goar, V., Kuri, M., Kumar, R., Senjyu, T. (eds) Advances in Information Communication Technology and Computing. Lecture Notes in Networks and Systems, vol 135. Springer, Singapore. https://doi.org/10.1007/978-981-15-5421-6_8

[33] Brown, B.B., 1968. Delphi process: a methodology used for the elicitation of opinions of experts. Rand Corp Santa Monica CA. https://apps.dtic.mil/sti/citations/AD0675981.

[34] Stellman, A. (2022, June 24). Building better software. Building Better Software. https://www.stellman-greene.com/

[35] Wieczorek, I., 2002. Improved software cost estimation–a robust and interpretable modelling method and a comprehensive empirical investigation. Empirical Software Engineering, 7(2), pp.177-180. https://publica.fraunhofer.de/handle/publica/274334

[36] https://www.rose-hulman.edu/class/csse/csse372/201410/SlidePDFs/session12.pdfRose-Hulman Institute of Technology. (n.d.). https://www.rose-hulman.edu/

[37] Galorath Inc. 2005. SEER-SEM Software Estimation, Planning and Project Control Training Manual. https://galorath.com/cost-estimation/seer-sem-software/

[38] Putnam, L. H. and Myers, W. 1992. Measures for Excellence. Prentice Hall, Englewood Cliffs, NJ

[39] Albrecht, A.J. and Gaffney, J.E., 1983. Software function, source lines of code, and development effort prediction: a software science validation. IEEE transactions on software engineering, (6), pp.639-648.doi: https://doi.org/10.1109/TSE.1983.235271

[40] Longstreet, D., 2004. Function points analysis training course. SoftwareMetrics. com, October.

[41] Karner, G., 1993. Resource estimation for objectory projects. Objective Systems SF AB, 17, pp.1-9.

[42] Shah, J., Kama, N. and Ismail, S.A., 2018, May. An Empirical Study with Function Point Analysis for Software Development Phase Method. In Proceedings of the 7th International Conference on Software and Information Engineering (pp. 7-11). https://doi.org/10.1145/3220267.3220268

[43] Nagar, C. (2011). Software efforts estimation using Use Case Point approach by increasing technical complexity and experience factors. *complexity*, *3*, 1-05.

[44] Jagadeeswara Rao, G., Siva Prasad, A., Sai Srinivas, S., Sivaparvathi, K., & Panda, N. (2022). Data Classification by Ensemble Methods in Machine Learning. In Advances in Intelligent Computing and Communication: Proceedings of ICAC 2021 (pp. 127-135). Singapore: Springer Nature Singapore. DOI: https://doi.org/10.1007/978-981-19-0825-5_13

[45] Kumar, P.S. and Behera, H.S., 2020. Role of soft computing techniques in software effort estimation: an analytical study. In Computational Intelligence in Pattern Recognition (pp. 807-831). Springer, Singapore. https://doi.org/10.1007/978-981-13-9042-5_70

[46] de BarcelosTronto, I.F., da Silva, J.D.S. and Sant'Anna, N., 2008. An investigation of artificial neural networks based prediction systems in software project management. Journal of Systems and Software, 81(3), pp.356-367. https://doi.org/10.1016/j.jss.2007.05.011

[47] 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," in IEEE Access, vol. 9, pp. 60309-60327, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3072380

[48] Legendre, A.M., 1806. Nouvelles méthodes pour la détermination des orbites des comètes; par AM Legendre... chez Firmin Didot, libraire pour lewmathematiques, la marine, l'architecture, et les editions stereotypes, rue de Thionville.

[49] Gauss, CF, 1809. Theory of the motion of celestial bodies in conic sections sun by Karl Friedrich Gauss. Friedrich expense. Perthes et IH Besser.

[50] Bou Nassif, Ali, "Software Size and Effort Estimation from Use Case Diagrams Using Regression and Soft Computing Models" (2012). Electronic Thesis and Dissertation Repository. 547. https://ir.lib.uwo.ca/etd/547

[51] Radu, L.D., 2019. Effort Prediction in Agile Software Development with Bayesian Networks. In ICSOFT (pp. 238-245). Doi: https://doi.org/10.5220/0007842802380245

[52] Dragicevic, S., Celar, S. and Turic, M., 2017. Bayesian network model for task effort estimation in agile software development. Journal of systems and software, 127, pp.109-119. DOI: https://doi.org/10.1016/j.jss.2017.01.027

[53] Vital, T. P. (2023). Intellectual Gestational Diabetes Diagnosis System Using MLP-Whale Optimization Algorithm including Statistical Analysis. International Journal of Computing and Digital Systems, 14(1), 1-1. DOI: http://dx.doi.org/10.12785/ijcds/140138

[54] Terlapu, P. V., Sadi, R. P. R., Pondreti, R. K., & Tippana, C. R. (2021). Intelligent Identification of Liver Diseases Based on Incremental Hidden Layer Neurons ANN Model. International Journal of Computing and Digital Systems. DOI: https://dx.doi.org/10.12785/ijcds/110183

[55] Finnie, G.R., Wittig, G.E. and Desharnais, J.M., 1997. A comparison of software effort estimation techniques: Using function points with neural networks, case-based reasoning and regression models. Journal of systems and software, 39(3), pp.281-289. https://doi.org/10.1016/S0164-1212(97)00055-1

[56] Chirra, S.M.R. and Reza, H., 2019. A survey on software cost estimation techniques. Journal of Software Engineering and Applications, 12(06), p.226. https://doi.org/10.4236/jsea.2019.126014

[57] Heiat, A., 2002. Comparison of artificial neural network and regression models for estimating software development effort. Information and software Technology, 44(15), pp.911-922. https://doi.org/10.1016/S0950-5849(02)00128-3

[58] Terlapu, P. V., Gedela, S. B., Gangu, V. K., & Pemula, R. (2022). Intelligent diagnosis system of hepatitis C virus: A probabilistic neural network-based approach. International Journal of Imaging Systems and Technology, 32(6), 2107-2136.
DOI: https://doi.org/10.1002/ima.22746

[59] Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297. https://doi.org/10.1007/BF00994018

[60] Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. InProceedings of the fifth annual workshop on Computational learning theory 1992 Jul 1 (pp. 144-152). https://doi.org/10.1145/130385.130401

[61] Sinha, B.K., Sinhal, D. and Verma, B., 2013. A software measurement using artificial neural network and support vector machine. International Journal of Software Engineering & Applications (IJSEA), 4(4). Available at SSRN: https://ssrn.com/abstract=3328443

[62] P. S. Rao, K. K. Reddi and R. U. Rani, "Optimization of neural network for software effort estimation," 2017 International Conference on Algorithms, Methodology, Models and Applications in Emerging Technologies (ICAMMAET), Chennai, India, 2017, pp. 1-7, doi: https://doi.org/10.1109/ICAMMAET.2017.8186696

[63] Moharreri, K., Sapre, A.V., Ramanathan, J. and Ramnath, R., 2016, June. Cost-effective supervised learning models for software effort estimation in agile environments. In 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC) (Vol. 2, pp. 135-140). IEEE.
doi: https://doi.org/10.1109/COMPSAC.2016.85

[64] Mahmood, Y, Kama, N, Azmi, A, Khan, AS, Ali, M. Software effort estimation accuracy prediction of machine learning techniques: A systematic performance evaluation. Softw: PractExper. 2022; 52(1): 39– 65. https://doi.org/10.1002/spe.3009

[65] Benala, T.R., Dehuri, S. (2022). Tuning Functional Link Artificial Neural Network for Software Development Effort Estimation. In: , et al. Innovations in Intelligent Computing and Communication. ICIICC 2022. Communications in Computer and Information Science, vol 1737. Springer, Cham. https://doi.org/10.1007/978-3-031-23233-6_5.

[66] Khan, J.A., Khan, S.U.R., Khan, T.A. and Khan, I.U.R., 2021. An Amplified COCOMO-II based Cost Estimation Model in Global Software Development Context. IEEE Access.doi: 10.1109/ACCESS.2021.3089870

[67] Kumar, P.S., Behera, H.S., Nayak, J. and Naik, B., 2021. A pragmatic ensemble learning approach for effective software effort estimation. Innovations in Systems and Software Engineering, pp.1-17. https://doi.org/10.1007/s11334-020-00379-y

[68] Sharma, S. and Vijayvargiya, S., 2021. Applying soft computing techniques for software project effort estimation modelling. In Nanoelectronics, Circuits and Communication Systems (pp. 211-227). Springer, Singapore. https://doi.org/10.1007/978-981-15-7486-3_21

[69] Rhmann, W., Pandey, B. and Ansari, G.A., 2021. Software effort estimation using ensemble of hybrid search-based algorithms based on metaheuristic algorithms. Innovations in Systems and Software Engineering, pp.1-11. https://doi.org/10.1007/s11334-020-00377-0

[70] Resmi, V. and Vijayalakshmi, S., 2020. Kernel Fuzzy Clustering With Output Layer Self-Connection Recurrent Neural Networks for Software Cost Estimation. Journal of Circuits, Systems and Computers, 29(06),
p.2050091.https://doi.org/10.1142/S0218126620500917

[71] Fadhil, A.A. and Alsarraj, R.G., 2020, February. Exploring the whale optimization algorithm to enhance software project effort estimation. In 2020 6th International Engineering Conference "Sustainable Technology and Development"(IEC) (pp. 146-151). IEEE. doi: 10.1109/IEC49899.2020.9122918

[72] Suresh Kumar, P. and Behera, H.S., 2020. Estimating software effort using neural network: An experimental investigation. In Computational Intelligence in Pattern Recognition (pp. 165-180). Springer, Singapore. https://doi.org/10.1007/978-981-15-2449-3_14

[73] Reddy, D.K.K. and S Behera, H., 2020. Software effort estimation using particle swarm optimization: Advances and challenges. Computational Intelligence in Pattern Recognition, pp.243-258. https://doi.org/10.1007/978-981-15-2449-3_20

[74] Goyal, S. and Bhatia, P.K., 2020. Feature selection technique for effective software effort estimation using multi-layer perceptrons. In Proceedings of ICETIT 2019 (pp. 183-194). Springer, Cham. https://doi.org/10.1007/978-3-030-30577-2_15

[75] Nassif, A.B., Azzeh, M., Idri, A. and Abran, A., 2019. Software development effort estimation using regression fuzzy models. Computational intelligence and neuroscience, 2019.https://doi.org/10.1155/2019/8367214

[76] Malgonde, O. and Chari, K., 2019. An ensemble-based model for predicting agile software development effort. Empirical Software Engineering, 24(2), pp.1017-1055.https://doi.org/10.1007/s10664-018-9647-0

[77] Kaushik, A. and Singal, N., 2019. A hybrid model of wavelet neural network and metaheuristic algorithm for software development effort estimation. International Journal of Information Technology, pp.1-10. https://doi.org/10.1007/s41870-019-00339-1

[78] Bilgaiyan, S., Mishra, S. and Das, M., 2019. Effort estimation in agile software development using experimental validation of neural network models. International Journal of Information Technology, 11(3), pp.569-573. https://doi.org/10.1007/s41870-018-0131-2

[79] Venkataiah, V., Mohanty, R. and Nagaratna, M., 2019. Prediction of software cost estimation using spiking neural networks. In Smart Intelligent Computing and Applications (pp. 101-112). Springer, Singapore. https://doi.org/10.1007/978-981-13-1927-3_11

[80] Qin, M., Shen, L., Zhang, D. and Zhao, L., 2019, December. Deep Learning Model for Function Point Based Software Cost Estimation-An Industry Case Study. In 2019 International Conference on Intelligent Computing, Automation and Systems (ICICAS) (pp. 768-772). IEEE. doi: 10.1109/ICICAS48597.2019.00165

[81] García-Floriano, A., López-Martín, C., Yáñez-Márquez, C. and Abran, A., 2018. Support vector regression for predicting software enhancement effort. Information and Software Technology, 97, pp.99-109.https://doi.org/10.1016/j.infsof.2018.01.003

[82] Pospieszny, P., Czarnacka-Chrobot, B. and Kobylinski, A., 2018. An effective approach for software project effort and duration estimation with machine learning algorithms. Journal of Systems and Software, 137, pp.184-196.https://doi.org/10.1016/j.jss.2017.11.066

[83] Kumari, S. and Pushkar, S., 2018. Cuckoo search based hybrid models for improving the accuracy of software effort estimation. Microsystem Technologies, 24(12), pp.4767-4774. https://doi.org/10.1007/s00542-018-3871-9

[84] Murillo-Morera, J., Quesada-López, C., Castro-Herrera, C. and Jenkins, M., 2017. A genetic algorithm-based framework for software effort prediction. Journal of software engineering research and development, 5(1), pp.1-33. https://doi.org/10.1186/s40411-017-0037-x

[85] Silhavy, R., Silhavy, P. and Prokopova, Z., 2017. Analysis and selection of a regression model for the Use Case Points method using a stepwise approach. Journal of Systems and Software, 125, pp.1-14. https://doi.org/10.1016/j.jss.2016.11.029

[86] Araujo, R.D.A., Oliveira, A.L. and Meira, S., 2017. A class of hybrid multilayer perceptrons for software development effort estimation problems. Expert Systems with Applications, 90, pp.1-12. https://doi.org/10.1016/j.eswa.2017.07.050

[87] Phannachitta, P., Keung, J., Monden, A. and Matsumoto, K., 2017. A stability assessment of solution adaptation techniques for analogy-based software effort estimation. Empirical Software Engineering, 22(1), pp.474-504. https://doi.org/10.1007/s10664-016-9434-8

[88] Satapathy, S.M., Acharya, B.P. and Rath, S.K., 2016. Early stage software effort estimation using random forest technique based on use case points. IET Software, 10(1), pp.10-17.
https://doi.org/10.1049/iet-sen.2014.0122

[89] Kumar, L. and Rath, S.K., 2016. Hybrid functional link artificial neural network approach for predicting maintainability of object-oriented software. Journal of Systems and Software, 121, pp.170-190. https://doi.org/10.1016/j.jss.2016.01.003

[90] Wani, Z.H. and Quadri, S.M.K., 2016. Artificial Bee Colony-Trained Functional Link Artificial Neural Network Model for Software Cost Estimation. In Proceedings of Fifth International Conference on Soft

Computing for Problem Solving (pp. 729-741). Springer, Singapore. DOI: https://doi.org/10.1007/978-981-10-0451-3_65

[91] Kaushik, A., Soni, A.K. and Soni, R., 2016. An improved functional link artificial neural networks with intuitionistic fuzzy clustering for software cost estimation. International Journal of System Assurance Engineering and Management, 7(1), pp.50-61. DOI: https://doi.org/10.1007/978-981-10-0451-3_65

[92] M. Jawa and S. Meena, "Software Effort Estimation Using Synthetic Minority Over-Sampling Technique for Regression (SMOTER)," *2022 3rd International Conference for Emerging Technology (INCET)*, Belgaum, India, 2022, pp. 1-6, doi: 10.1109/INCET54531.2022.9824043.

[93] Rijwani, P. and Jain, S., 2016. Enhanced software effort estimation using multi layered feed forward artificial neural network technique. Procedia Computer Science, 89, pp.307-312.https://doi.org/10.1016/j.procs.2016.06.073

[94] Sachan, R.K., Nigam, A., Singh, A., Singh, S., Choudhary, M., Tiwari, A. and Kushwaha, D.S., 2016. Optimizing basic COCOMO model using simplified genetic algorithm. Procedia Computer Science, 89, pp.492-498.https://doi.org/10.1016/j.procs.2016.06.107

[95] Aljahdali, S., Sheta, A.F. and Debnath, N.C., 2015, November. Estimating software effort and function point using regression, Support Vector Machine and Artificial Neural Networks models. In 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA) (pp. 1-8). IEEE.

[96] Kumari, S. and Pushkar, S., 2016. A framework for analogy-based software cost estimation using multi-objective genetic algorithm. In Proceedings of the world congress on engineering and computer Science (Vol. 1).doi: 10.1109/AICCSA.2015.7507149.

[97] López-Martín, C. and Abran, A., 2015. Neural networks for predicting the duration of new software projects. Journal of Systems and Software, 101, pp.127-135.https://doi.org/10.1016/j.jss.2014.12.002

[98] López-Martín, C., 2015. Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. Applied Soft Computing, 27, pp.434-449.https://doi.org/10.1016/j.asoc.2014.10.033

[99] Zhang, W., Yang, Y. and Wang, Q., 2015. Using Bayesian regression and EM algorithm with missing handling for software effort prediction. Information and software technology, 58, pp.58-70. https://doi.org/10.1016/j.infsof.2014.10.005

[100] Bardsiri, V.K. and Khatibi, E., 2015. Insightful analogy-based software development effort estimation through selective classification and localization. Innovations in Systems and Software Engineering, 11(1), pp.25-38. https://doi.org/10.1007/s11334-014-0242-2

[101] Natarajan, R., Balachandran, K. (2022). Ensemble Model of Machine Learning for Integrating Risk in Software Effort Estimation. In: Saraswat, M., Sharma, H., Balachandran, K., Kim, J.H., Bansal, J.C. (eds) Congress on Intelligent Systems. Lecture Notes on Data Engineering and Communications Technologies, vol 114. Springer, Singapore. https://doi.org/10.1007/978-981-16-9416-5_46

[102] Ghatasheh, N., Faris, H., Aljarah, I. and Al-Sayyed, R.M., 2019. Optimizing software effort estimation models using firefly algorithm. arXiv preprint arXiv:1903.02079. https://doi.org/10.48550/arXiv.1903.02079

[103] Sharma, A., & Rajpoot, D. S. (2024). A memetic approach for optimizing software effort estimation using anti-predatory NIA. International Journal of Information Technology, 16(2), 641-649. https://doi.org/10.1007/s41870-023-01652-6

[104] Azzeh, M., Elsheikh, Y. and Alseid, M., 2017. An optimized analogy-based project effort estimation. arXiv preprint arXiv:1703.04563. https://doi.org/10.48550/arXiv.1703.04563

[105] Jadhav, A., & Kumar Shandilya, S. (2023). Towards effective feature selection in estimating software effort using machine learning. Journal of Software: Evolution and Process, e2588

[106] Nhung HLTK, Van Hai V, Silhavy P, Prokopova Z, Silhavy R. Incorporating statistical and machine learning techniques into the optimization of correction factors for software development effort estimation. J Softw Evol Proc. 2023;e2611. doi:10. 1002/smr.2611

[107] Zakrani, A., Hain, M., & Idri, A. (2019). Improving software development effort estimating using support vector regression and feature selection. *IAES International Journal of Artificial Intelligence*, *8*(4), 399.

[108] Hameed, S., Elsheikh, Y., &Azzeh, M. (2023). An optimized case-based software project effort estimation using genetic algorithm. Information and Software Technology, 153, 107088.

[109] Pal, N., Yadav, M. P., & Yadav, D. K. (2024). Appropriate number of analogues in analogy-based software effort estimation using quality datasets. Cluster Computing, 27(1), 531-546

[110] R., Lalitha., P., N., Sreelekha. (2023). A methodology to analyse and estimate software development process using machine learning techniques. International Journal of Software Engineering and Knowledge Engineering, doi: 10.1142/s021819402350016x

[111] Kassaymeh, S., Alweshah, M., Al-Betar, M. A., Hammouri, A. I., & Al-Ma'aitah, M. A. (2024). Software effort estimation modeling and fully connected artificial neural network optimization using soft computing techniques. Cluster Computing, 27(1), 737-760. https://doi.org/10.1007/s10586-023-03979-y

[112] Sousa, A. O., Veloso, D. T., Gonçalves, H. M., Faria, J. P., Mendes-Moreira, J., Graça, R., ... & Henriques, P. C. (2023). Applying Machine Learning to Estimate the Effort and Duration of Individual Tasks in Software Projects. *IEEE Access*.

[113] Mustafa, E. I., & Osman, R. (2024). A random forest model for early-stage software effort estimation for the SEERA dataset. Information and Software Technology, 169, 107413. DOI: https://doi.org/10.1016/j.infsof.2024.107413.

[114] Amrita, Sharma., Neha, Chaudhary. (2022). Analysis of Software Effort Estimation Based on Story Point and Lines of Code using Machine Learning. International Journal of Computing and Digital Systems, 12(1):131-140. doi: 10.12785/ijcds/1201012

[115] Rahman, M. T., & Islam, M. M. (2019, June). A comparison of machine learning algorithms to estimate effort in varying sized software. In 2019 IEEE Region 10 Symposium (TENSYMP) (pp. 137-142). IEEE. DOI: https://doi.org/10.1109/TENSYMP46218.2019.8971150

[116] Sarro, F., & Petrozziello, A. (2018). Linear programming as a baseline for software effort estimation. ACM transactions on software engineering and methodology (TOSEM), 27(3), 1-28. https://doi.org/10.1016/j.jss.2021.110904

[117] Khan, M. S., Jabeen, F., Ghouzali, S., Rehman, Z., Naz, S., & Abdul, W. (2021). Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. Ieee Access, 9, 60309-60327.DOI: https://doi.org/10.1109/ACCESS.2021.3072380

[118] Villalobos-Arias, L., Quesada-López, C., Guevara-Coto, J., Martínez, A., & Jenkins, M. (2020, November). Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation. In Proceedings of the 16th ACM international conference on predictive models and data analytics in software engineering (pp. 31-40). https://doi.org/10.1145/3416508.3417121

[119] Puspaningrum, A., & Sarno, R. (2017). A hybrid cuckoo optimization and harmony search algorithm for software cost estimation. *Procedia Computer Science*, *124*, 461-469.

[120] Rankovic, N., Rankovic, D., Ivanovic, M., & Lazic, L. (2021). A new approach to software effort estimation using different artificial neural network architectures and Taguchi orthogonal arrays. IEEE access, 9, 26926-26936. DOI: https://doi.org/10.1109/ACCESS.2021.3057807

[121] Hameed, S., Elsheikh, Y., & Azzeh, M. (2023). An optimized case-based software project effort estimation using genetic algorithm. Information and Software Technology, 153, 107088. https://doi.org/10.1016/j.infsof.2022.107088

[122] Brar, P., & Nandal, D. (2022, July). A systematic literature review of machine learning techniques for software effort estimation models. In 2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT) (pp. 494-499). IEEE. DOI: https://doi.org/10.1109/CCiCT56684.2022.00093

[123] Rahman, M., Roy, P. P., Ali, M., Gonc, T., & Sarwar, H. (2023). Software Effort Estimation using Machine Learning Technique. International Journal of Advanced Computer Science and Applications, 14(4). https://doi.org/10.14569/IJACSA.2023.0140491

[124] Hameed, S., Elsheikh, Y., & Azzeh, M. (2023). An optimized case-based software project effort estimation using genetic algorithm. Information and Software Technology, 153, 107088. DOI: https://doi.org/10.1016/j.infsof.2022.107088

[125] Victor, Emanuel, De, Atocha, Uc, Cetina. (2023). Recent Advances in Software Effort Estimation using Machine Learning. arXiv.org, abs/2303.03482 doi: 10.48550/arXiv.2303.03482
[126] Sakib, S. M. N. (2022). Software Effort Estimation for Improved Decision Making. Cambridge Open Engage. doi:10.33774/coe-2022-bt2d9 This content is a preprint and has not been peer-reviewed. Doi: https://doi.org/10.33774/coe-2022-bt2d9
[127] Song, L., &Minku, L. L. (2023). Artificial intelligence in software project management. In Optimising the software development process with artificial intelligence (pp. 19-65). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-19-9948-2_2
[128] Assefa, Y., Berhanu, F., Tilahun, A., &Alemneh, E. (2022, November). Software Effort Estimation using Machine learning Algorithm. In 2022 International Conference on Information and Communication Technology for Development for Africa (ICT4DA) (pp. 163-168). IEEE. DOI: 10.1109/ICT4DA56482.2022.9971209
[129] Ahmad, F. B., & Ibrahim, L. M. (2022, August). Software effort estimation Based on long short-term memory and stacked long short term memory. In 2022 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM) (pp. 165-170). IEEE. DOI: https://doi.org/10.1109/ICCITM56309.2022.10031794
[130] Alsheikh, N. M., & Munassar, N. M. (2023). Improving software effort estimation models using grey wolf optimization algorithm. IEEE Access. DOI: https://doi.org/10.1109/ACCESS.2017
[131] Huang, S.J. and Chiu, N.H., 2006. Optimization of analogy weights by genetic algorithm for software effort estimation. Information and software technology, 48(11), pp.1034-1045. https://doi.org/10.1016/j.infsof.2005.12.020
[132] Poonam, Rijwani., Sonal, Jain. (2022). Software Effort Estimation Development From Neural Networks to Deep Learning Approaches. Journal of Cases on Information Technology, doi: https://doi.org/10.4018/jcit.296715
[133] Ramya, P., Sai Mokshith, M., Abdul Rahman, M., Nithin Sai, N. (2023). Software Development Estimation Cost Using ANN. In: Ogudo, K.A., Saha, S.K., Bhattacharyya, D. (eds) Smart Technologies in Data Science and Communication. Lecture Notes in Networks and Systems, vol 558. Springer, Singapore. https://doi.org/10.1007/978-981-19-6880-8_23
[134] Rao, K. E., Terlapu, P. R. V., Naidu, P. A., Kumar, T. R., &Pydi, B. M. (2024). Feature importance for software development effort estimation using multi-level ensemble approaches. Bulletin of Electrical Engineering and Informatics, 13(2), 1090-1102. DOI: https://doi.org/10.11591/eei.v13i2.5531
[135] Hameed, S., Elsheikh, Y., &Azzeh, M. (2023). An optimized case-based software project effort estimation using genetic algorithm. Information and Software Technology, 153, 107088. Doi: https://doi.org/10.1016/j.infsof.2022.107088
[136] Najm, A., &Zakrani, A. (2023, December). On the Interpretability of a Tree-based Ensemble for Predicting Software Effort. In 2023 7th IEEE Congress on Information Science and Technology (CiSt) (pp. 129-134). IEEE.
DOI: https://doi.org/10.1109/CiSt56084.2023.10410020
[137] Swandari, Y., Ferdiana, R., &Permanasari, A. E. (2023, September). Research Trends in Software Development Effort Estimation. In 2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI) (pp. 625-630). IEEE. DOI: https://doi.org/10.1109/EECSI59885.2023.10295716.
[138] Benala, T.R., Dehuri, S., Satapathy, S.C. and Raghavi, C.S., 2011, December. Genetic algorithm for optimizing neural network-based software cost estimation. In International Conference on Swarm, Evolutionary, and Memetic Computing (pp. 233-239). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-27172-4_29
[139] Hsu, C.J. and Huang, C.Y., 2011. Comparison of weighted grey relational analysis for software effort estimation. Software Quality Journal, 19(1), pp.165-200. https://doi.org/10.1007/s11219-010-9110-y
[140] Ramacharan, S., & Rao, K. V. G. (2016, March). Software effort estimation of GSD projects using calibrated parametric estimation models. Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (pp. 1-8). https://doi.org/10.1109/ACCESS.2021.3089870
[141] Hosni, M., & Idri, A. (2017, April). Software effort estimation using classical analogy ensembles based on random subspace. In Proceedings of the Symposium on Applied Computing (pp. 1251-1258). https://dl.acm.org/doi/10.1002/smr.2343
[142] Amaral, W., Rivero, L., Junior, G. B., & Viana, D. (2019, October). Using machine learning technique for effort estimation in software development. In Proceedings of the XVIII Brazilian Symposium on Software Quality (pp. 240-245).
DOI: https://doi.org/10.1145/3364641.3364670
[143] Fávero, E. M. D. B., Casanova, D., & Pimentel, A. R. (2022). SE3M: A model for software effort estimation using pre-trained embedding models. Information and Software Technology, 147, 106886. DOI: https://doi.org/10.1016/j.infsof.2022.106886
[144] Ali, S. S., Ren, J., Zhang, K., Wu, J., & Liu, C. (2023). Heterogeneous ensemble model to optimize software effort estimation accuracy. IEEE Access, 11, 27759-27792.
https://doi.org/10.1109/ACCESS.2023.3256533
[145] Jadhav, A., Shandilya, S. K., Izonin, I., & Gregus, M. (2023). Effective Software Effort Estimation enabling Digital Transformation. IEEE Access. https://doi.org/10.1109/ACCESS.2023.3293432

DR. PANDURANGA VITAL TERLAPU earned his Bachelor of Science in Computer Science from Andhra University in Andhra Pradesh, India, in 1995. He obtained his Master of Computer Application from the same institution in 1998. He pursued further studies, completing his M. Tech in Computer Science and Engineering from Acharya Nagarjuna University, Andhra Pradesh, India. Dr. Terlapu continued his academic journey by achieving a PhD in Computer Science and Engineering from GITAM University, Andhra Pradesh, India. With an impressive career spanning 24 years in teaching and 18 years in research, Dr. Terlapu presently serves as a Professor in the Department of Computer Science and Engineering at Aditya Institute of Technology and Management (AITAM), India. Dr. Terlapu holds esteemed memberships in professional organizations such as the Association for Computing Machinery (ACM), as well as Lifetime Memberships from the International Computer Science and Engineering Society (ICSES), USA, and the Indian Society for Technical Education (ISTE), New Delhi, India. Additionally, he maintains affiliations with five other reputable associations. His scholarly contributions are significant, boasting over 68 research papers published in distinguished international journals, including those indexed in SCI and SCOPUS, along with conferences hosted by IEEE, Springer, and Elsevier, among others, and accessible online. Dr. Terlapu is also actively reviewing submissions for prominent journals affiliated with Springer, Elsevier, and IEEE databases. Dr. Terlapu, a renowned expert in knowledge dissemination, has published two books and filed three patents, including one granted by South Africa, showcasing his innovative thinking and significant impact in the field. Dr. Terlapu research interests encompass a broad spectrum within the field of computer science, including Machine Learning, Image Processing, Deep Learning, Data Mining, Big Data Analytics, IoT and Computational Intelligence, Voice Analysis, Software Engineering. His dedication and expertise contribute substantially to the advancement of these areas of study.

This article has been accepted for publication in IEEE Access. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2024.3457771

DR. KALIDINDI KISHORE RAJU received his M.Tech. Degree in Information Technology from J.N.T University-Kakinada and a Ph.D. in Data Mining and Image Processing in the Department of Computer Science and Engineering from J.N.T University-Kakinada. He is an Associate Professor in the Department of Information Technology, S.R.K.R Engineering College, Bhimavaram, AP, India. He published 14 research papers in international journals, two textbooks, and four patents. He has 19 years of teaching experience in Information Technology. His research interests include Data Mining, Pattern Recognition, Machine Learning and Digital Image Processing.

Dr. G Kiran Kumar received his PhD from Acharya Nagarjuna University, Andhra Pradesh, India and an M. Tech from Osmania University, Hyderabad, India. He works as an Associate Professor at Chaitanya Bharathi Institute of Technology, Hyderabad. He has more than 20 years of teaching experience. He published more than 20 publications in international journals and conferences of repute. His research interests include Data Mining, Machine Learning, Deep Learning, and Image processing.

MR. JAGADEESWARARAOG is dedicated to pursuing his PhD in the Computer Science and Systems Engineering Department at Andhra University, Visakhapatnam. He completed his master's in information technology from JNTU, Kakinada, in 2010. With a combined experience of 14 years in teaching and nine years in research, he holds the position of Assistant Professor in the Department of Information Technology at Aditya Institute of Technology and Management (AITAM), India. His research interests span Machine Learning, Deep Learning, Data Mining, and Bioinformatics, where he actively contributes to advancing these fields.

DR. K.KAVITHA is a Sr. Asst. Professor in the department of Information Technology at Aditya Institute of Technology and Management, Tekkali, India. She received Ph.D. degree in Computer Science and Engineering with specialization in Artificial Intelligence from Acharya Nagarjuna University, Guntur, India. Her research interests include Artificial Intelligence, Machine Learning and Deep Learning. Dr. K. Kavitha has published more than 15 research papers in various international journals.

DR. SHIRINA SAMREEN is an Associate Professor in the Department of Computer Science, College of Computer and Information Sciences, Majmaah University, Kingdom of Saudi Arabia. She received her Ph.D in Computer Science and Engineering from JNTUH, India in 2016. She received best paper awards twice for her research papers at IEEE ICCIC, a renowned international conference. She is a reviewer for IEEE Access, IEEE Wireless Communication Magazine, Journal of Engineering and Applied Sciences (JEAS), and numerous other peer-reviewed International Journals. She has over 40 research papers to her credit in various IEEE International conferences and SCI/Scopus indexed Journals. Currently, her research interests include Cyber Security, Applied machine learning, and Deep learning.