Journal of Theoretical and Applied Information Technology
31st January 2019. Vol.97. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A SYSTEMATIC LITERATURE REVIEW ON METHODS FOR SOFTWARE EFFORT ESTIMATION

[1]ROBERT MARCO, [2]NANNA SURYANA, [3]SHARIFAH SAKINAH SYED AHMAD
[1]Department of Information Technology, University of Amikom Yogyakarta, Yogyakarta, Indonesia
[1,2,3]University Teknikal Malaysia Melaka, Melaka, Malaysia
E-mail: [1]robertmarco@amikom.ac.id, [2]nsuryana@utem.edu.my,[3]sakinah@utem.edu.my

ABSTRACT

There have been many researchers who proposed research in an effort to develop the field of improving accuracy in the Software Effort Estimation (SEE). Collected results from a series of studies selected in the Software Effort Estimation, which was published in the period 2000-2017, using systematic mapping and review procedures. The purpose of this review is to provide a classification of study areas of SEE related to publication channels, research approaches, types of contributions, techniques used in combination. To analyze: 1) The precise estimation of SEE techniques; 2) Accuracy of the SEE model estimate compared with other models; 3) A favorable outcome context for the use of the SEE model; and 4) The impact of other techniques into the SEE model by combining models and implementation for models and tools. We have identified 74 major studies that are relevant to the purpose of this study. After investigating, we found that eight types of techniques were used in the Software effort estimation model. that techniques used for SEE usually produce acceptable estimation accuracy, and the facts are more accurate.

Keywords: *Systematic Literature Review, Software Effort Estimation, Datasets, Methods, Validation.*

1. INTRODUCTION

The complexity of software development projects, making estimation of development efforts is something that must be taken seriously into the early stages of the project. Although many models of software development effort estimations have been proposed over the past decade, the accuracy is not satisfactory. According to Jørgensen et al (2007), the major deviations between actual and estimated efforts do not necessarily reflect poor estimation skills. Therefore, it requires knowledge of the level of uncertainty estimation [1]. Improved accuracy in enhancing adaptability and flexibility to deal with the complexity and uncertainty that exist in the field of software development effort estimations [2]. SEE play an important role in controlling software costs, reducing software risk, and ensuring software quality [3]. Both over and underestimation efforts can cause problems for the company, while low estimates can result in poor quality software projects, pending or unfinished [4]. Effort estimation as the main factor to accurately estimate the model [5].

Various estimation methods have been proposed to improve the accuracy of estimates, so based on a comprehensive review, these estimation methods can be classified in types: expert judgment; regression-based methods; parametric models; case-based reasoning (CBR) method (Analogy based Estimation); dynamics-based models; and composite methods [6]; machine learning methods [7][8]; and algorithmic method [9].

Many researchers have proposed several techniques to improve accuracy for SEE. Many studies have tried to modify it new models using machine learning to improve accuracy in SEE [10][11][12][13]. Using a random sampling technique to assess the method [14], Based feature selection [15][16][17][18][19], by using bagging algorithm [20][21], or parameter optimization used classifiers [22][23][24]. Some prediction techniques have been suggested but none have proved consistently successful in predicting software development efforts [7].

Classifications for embedded software development projects based on whether the amount of effort is an outlier, classifications for embedded software development projects using an Artificial Neural Network (ANN) and Support Vector Machine (SVM) [25][26]. The machine learning technique parameters used for regression using Support Vector Regression (SVR) had the best performance [16][15]. Regression using SVM perform well [27]. Genetic algorithm (GA) with SVM can find the best parameters [28]. The most widely used method is NN, followed by Model Tree, Classification and Regression Trees (CART), and

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GA [29]. SVM and Nearest Neighbor Approach (kNN) [30]. Combination of Analogy Based Estimation (ABE) and Particle Swarm Optimization (PSO) algorithm [31]. There are software projects using NASA datasets showing that SVR significantly outperforms Radial Basis Functions Neural Networks (RBFNs) and linear regression [32]. Adaptive Regression (AR) techniques to produce better results when managing problems with complex connections and there are distortions with high noise levels [33]. Although classifiers based approaches have been introduced, they still have potential problems to provide accurate and stable effort estimates. So software development using classification algorithms to produce more reliable and accuracy development is still needed in this field of research.

Attribute noise, incomplete, and inconsistent in the software measurement dataset lowers the performance of machine-learning classifiers [34]. Data quality will decrease when used on heterogeneous and inconsistent datasets [35]. Irrelevant and inconsistent project effects on downhill estimates by designing frameworks, where all projects are clustered [36]. Implying that the effort of any not normally distributed dataset will pose a challenge to develop an accurate method [37]. The feature selection it functions to reduce the dimensions of the feature space, removes data that is excessive, irrelevant, or noise, to speed up data mining algorithms and improve data quality [38][39]. Datasets with relevant features that can lead to an increase in the accuracy of their estimates [39]. Feature selection are often implied to explore the effects of attributes that are irrelevant to classifier system performance [40]. The data can also greatly affect the predictive accuracy of the Machine learning model [41]. So it is necessary to prepare data in the process of building machine learning models, where data is preprocessed through selection, cleaning, reduction, transformation and feature selection [42][41][34].

That the level of accuracy in SEE is highly dependent on the parameter values of the method. In addition, the selection of input features may also have an important influence on the estimation accuracy [16]. Estimation by analogy is one of the machine learning techniques that predicts the software effort based on the premise that the more similar features the software project description [43]. SEE are optimization issues so that they can also be solved with Meta-heuristic algorithms. There are more than one algorithm available today to find the optimal solution for a particular problem [5]. GA as one of the feature selection models and improve the classification performance of the classifier [27][44][45][15]. SVM to get the optimal feature section and parameters must occur simultaneously [46]. Feature subset selection algorithm based on fuzzy logic can be optimized for SEE [47]. Fuzzy and NN provide objective estimates [48]. Adaptive neuro-fuzzy inference system (ANFIS) models are more efficient and stable in terms of reduced errors during training [49]; and able to provide good estimation capabilities [50]. Fuzzy Analogy ensembles achieve better performance across all datasets and no evidence concerning the best combiner [51]. Expertise judgment and Machine Learning methods with the assumption that this method is widely used by researchers and with accurate results [52]. Estimates of development efforts are a challenging issue that must be taken seriously at the early stages of the project. Inadequate information and uncertain requirements are the main reasons behind unreliable forecasts in this area. Although many models of effort estimation have been proposed over the past decade, the accuracy level is not quite satisfactory. Aims to make the connection between the problem of software effort estimation. Due to the uncertainty, complexity and lack of information in SEE, using the optimization algorithm can be the right choice to address this problem. The SEE have used many smart methods to improve estimation.

SEE is one of the methods used to make development efforts on software projects. SEE is the most important part in the early stages of software development, this is done to reduce cost and time losses. Many researchers have developed a model on SEE in improving accuracy, but researchers rarely carry out empirical evidence in the SEE field. The aims of this Systematic Literature Review (SLR), as a search strategy designed to find out the study relevant to the research question. This stage involves both determining the search terms and selection of literature sources, which are necessary for the subsequent search process [7]. The need to evaluate how some researchers conducted a systematic mapping process and identified from the systematic maps and existing SLR guidelines [53]. As a result, in this study, it collected the results of a series of selected studies on SEE, published in the 2000-2017 period, using systematic mapping and review procedures. this study aims to provide a classification of SEE field studies related to: publication channels, research approaches, contribution types, techniques used in combination.

This paper is arranged as follows. In section 2, contains an explanation of the research methodology. Section 3, it is used to answer research questions. While, the final section will summarize the overall results of the study.

2. METHOD

A systematic mapping study is a type of Systematic Literature Review (SLR) aimed at collecting and classifying research related to a particular topic [53]. This study has been conducted as a review of systematic literature based on the initial guidelines proposed by Kitchenham (2007). This review aims to assess systematic literature review (secondary studies), so that this study is categorized as a tertiary literature study [54]. The use of this procedure is motivated by the quality and accuracy of the methodology proposed [54]. Steps in how to work on the systematic literature review carried out below.

In this SLR, will propose 7 stages, In the first stage, we propose a series of research questions based on the SLR Objectives. The second stage, directing research questions, a search strategy designed to find out the study relevant to the research question. Then, In the third stage, define the criteria of research selection to identify relevant studies that can really contribute to answering research questions. Furthermore, relevant studies undergo a quality assessment process in which we design a number of quality checklists to facilitate assessment. The two remaining stages involve data extraction and data synthesis. At the data extraction stage, to design the data extraction form and then refine it by data extraction. Finally, at the synthesis stage of the data, we determined the appropriate methodology for synthesizing the data retrieved based on the data types and research questions. The review protocol is essential for an SLR. In Section 2.1 below, 2.2, 2.3, 2.4, 2.5, 2.6,2.7 will present the details of the review protocol. At the end of this session, will analyze the threat to the validity of the review protocol.

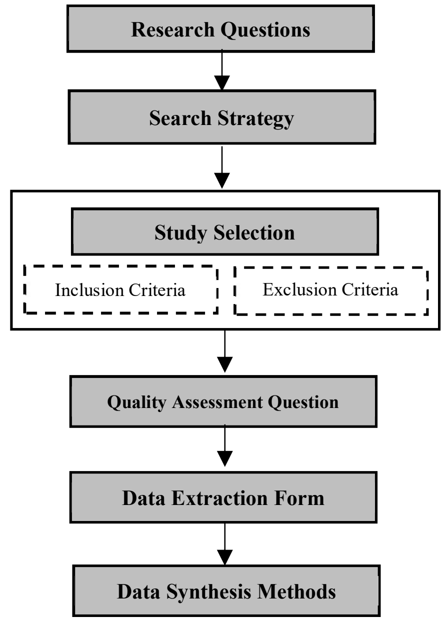

Figure 1: Mapping and Review Process

2.1. Research questions

The aims of this study, will describe the four Research Questions (RQ). A series of questions reviewing the different types of studies inside review that defines the question for systematic review technically does not involve four components, but five: Population, Intervention, Comparison, Outcome, Context (PICOC) [55].

1. Population (P): Software development project.
2. Intervention (I): Method of estimation /technique/metric size/dataset.
3. Comparison (C): No comparison intervention
4. Outcome (O): Accuracy of method/methodology of effort estimation.
5. Context (C): Any possible study, during empirical studies in the context of SEE will be considered.

The purpose of the research was proposed to describe seven research questions (RQ).

Table 1: Research Questions on Literature Review

ID	Research Questions	Motivation
RQ1	What are the types of research topic trends chosen by researchers in the field of SEE?	Identification of research topics that are trends in the field of SEE
RQ2	What types of datasets are most widely used in the field of SEE?	Identify the type of dataset that is most widely used in the field of SEE
RQ3	What method is most often used for SEE?	Identify the method most often used for SEE

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

RQ4	What types of validation and evaluation are used to measure the accuracy of the overall estimates of the model in the field of SEE?	Identify the types of validation and evaluation used to measure the accuracy of the overall estimate of the model in the field of SEE

2.2. Search strategy

After determining the research question, devise a strategy to define the search string and apply this search string to a set of selected digital libraries to extract all relevant documents, develop search procedures, and identify the primary study. The following is a list of digital databases that are used to search for relevant journals:

1. ACM Digital Library
2. IEEE eXplore
3. ScienceDirect
4. Springer
5. Google Scholar

To avoid bias of the researcher, we use the following procedure to determine the search string used in this study [56][57]:

1. Analyze questions and identify key words in terms of population, intervention, results and context
2. Identify key requirements relevant to the mapping questions and reviews listed.
3. Search all synonyms and spelling variations of the main term, if any.
4. Connect the main requirements of the population, intervention, results and context by using Boolean AND, to retrieve records containing all the requirements.
5. Use the Boolean operator OR to join the same term, to retrieve records that contain any (or all) requirements.

Steps one through five are performed by the author, As a result, obtain the following search string:

(Software* OR System OR Application OR Development* OR Web) AND (Effort* OR Cost OR Resource) AND (Estimate* OR Predict* OR Forecest OR Classification*).

2.3. Study Selection

To identify relevant studies it answers research questions based on titles, abstracts, and keywords. each candidate document identified at the initial search stage is evaluated, using inclusion and exclusion criteria, used to determine whether it must be accepted or rejected. If this decision can not be made using the title and/or abstract only, the full paper has been reviewed. The inclusion criteria and exclusion criteria are linked using the OR Boolean operator.

Inclusion criteria:

1. Use of software effort estimation techniques for software development estimation, and compare the performance of these techniques with other software effort estimation techniques.
2. The use of hybrid models that combine analogies with other techniques (eg GA, SVM or NN) to SEE.
3. Comparison of two or more software effort estimation techniques
4. Apply quality assessment criteria (defined in the next section) to the relevant paper so that to select a paper of acceptable quality, which is ultimately used for data extraction.
5. Define the following inclusion and exclusion criteria, which has been refined through pilot selection.
6. Do study selection by reading the title, abstract, or full text of the paper

Exclusion criteria:

1. Duplicate publication of the same study.
2. Estimated maintenance effort or testing effort.
3. Estimated size of software or time without effort estimates.
4. The topic of study is the accuracy of software development projects.
5. Review research will be excluded.

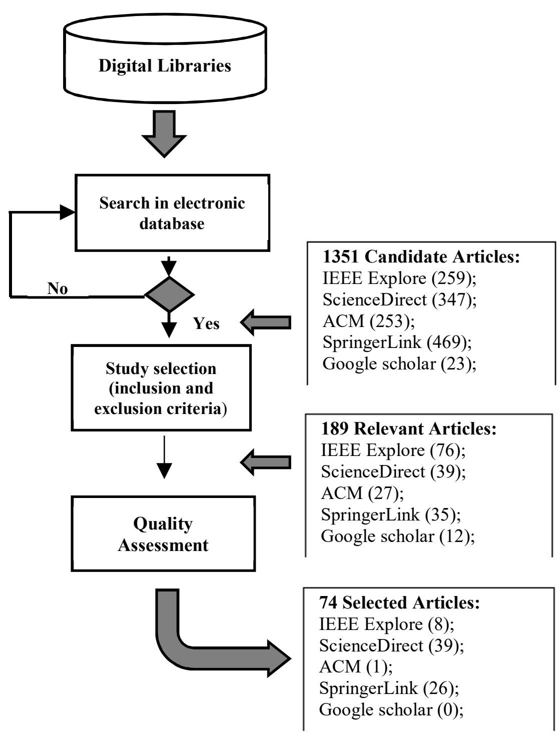

Figure 2: Study Selection

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2.4. Study Quality Assessment

Assessment of the quality of the study as a mapping study of data synthesis, to improve the research and strength of the conclusions described. By collecting evidence from selected studies to answer some of the research questions that have been proposed. The data taken in this review include as quantitative and qualitative data.

2.5. Data Extraction Form

With data extraction, studies are selected to collect data that contribute to answering research questions related in this review. By designing cards to facilitate the extraction of data presented in Table 2. For easy synthesis of data, items in the Table will be grouped according to the research question. Table 2, shows that the extracted data is related to the experiments performed.

Table 2: Data extraction

Data extractor
Data checker
Field of study
Year of publication
Authors
Article title
Type of study (experiment)
RQ1 : What are the types of research topic trends chosen by researchers in the field of SEE?
- **Trends and topic research**
RQ2 : What types of datasets are most widely used in the field of SEE?
- **Datasets software effort**
RQ3 : What method is most often used for SEE?
- **Methods in terms of software effort estimation**
RQ4 : What types of validation and evaluation are used to measure the accuracy of the overall estimates of the model in the field of SEE?
- **Validation methods**
- **Metrics used to measure estimation accuracy**

2.6. Data Synthesis Methods

Data extracted, to be synthesized and tabulated in accordance with the research questions discussed, to collect evidence to answer them. Because this data includes both quantitative and qualitative data, and since the review discusses different types of research questions, various data synthesis approaches are used narrative synthesis. In this method, To improve the presentation of these findings, some visualization tools, including bar graphs, pie charts, and tables are also used to improve the presentation of data distribution and software effort estimation.

2.7. Threats to Validity

In this section will review searches in accordance with research questions and have used them to take relevant studies in five electronic databases. This search is done manually, by reading each title, abstract, and keywords in the journal and conference proceedings. It is done to avoid bias on journal selection searches on software effort estimation. According Wen et al (2012), to the validity of this review protocol is analyzed from the following three aspects: selection bias study, publication bias, and possible inaccuracies in data extraction [7].

3. RESULT AND DISCUSSION

The section present result and discussion in literature review. The first, present overview about selection study. Seconds, present report review findings according to the research questions. Thrid, present implications for research. Four, present limitations of this review. Finally, present conclusion and future research.

3.1. Selection Study

3.1.1. Significant Journal Publications

In this literature review, 74 main studies were used to analyze the SEE. Distribution is conducted from January 2000 to December 2017, this is to demonstrate how the research interest in software engineering on the topic of software effort estimation is changing over time. A brief overview of the distribution studies over the years is shown in Figure 3, indicating that research on SEE is still highly relevant today. Regarding the type of study selected, all the studies were experimental studies and no survey and review studies were used. Although most selected studies use at least one set of public data to validate machine learning and Non machine learning models, it does not mean that the validation results adequately reflect the real situation in the industry. In fact, the lack of case studies and industry surveys can imply that the application of models/techniques in SEE is still immature.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

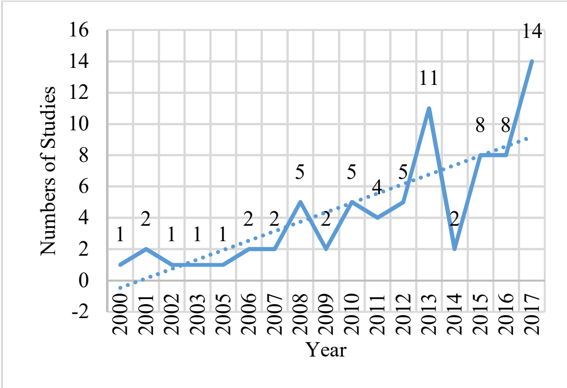

Figure 3: Distribution of Selected Studies

According to the main study selected in SEE the most important is the journal used is presented in figure 4, in this study did not use conference.

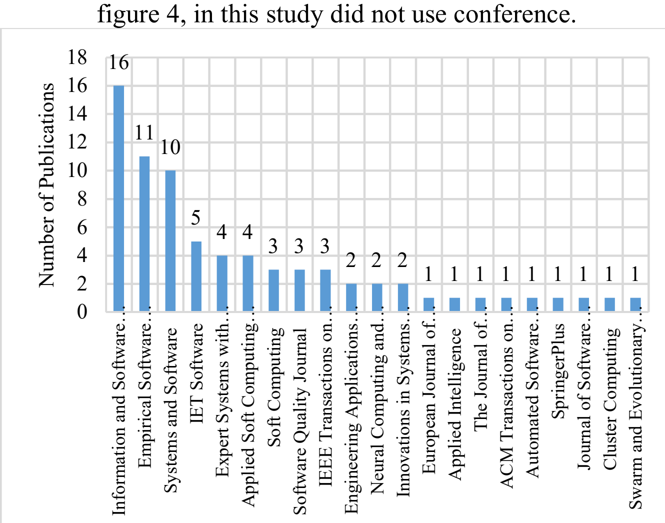

Figure 4: Journal Publications and Distribution of Selected Studies

In this study, will describe the Scimago Journal Rank (SJR) and Q (Q1-Q4) grades from the software estimation journals. The journal publication is ordered in accordance with its SJR score, presented in the table 3.

Table 3:. Scimago Journal Rank (SJR) of Selected Journals

Studies	SJR	Q-Category
European Journal of Operational Research	2.50	Q1 Information System and Management
Applied Soft Computing Journal	1.19	Q1 in Software
Expert Systems with Applications	1.43	Q1 in Computer Science
Soft Computing	1.30	Q1 in Software
Swarm and Evolutionary Computation	1.05	Q1 in Computer Science
Engineering Applications of Artificial Intelligence	1.04	Q1 in Artificial Intelligence
IEEE Transactions on Software Engineering	0.93	Q1 in Software
Information and Software Technology	0.78	Q1 in Software
Empirical Software Engineering	0.70	Q1 in Software
ACM Transactions on Software Engineering and Methodology	0.73	Q1 in Software
Applied Intelligence	0.66	Q2 in Artificial Intelligence
Systems and Software	0.64	Q2 in Software
Neural Computing and Applications	0.63	Q2 in Software
Cluster Computing	0.56	Q2 in Software
Automated Software Engineering	0.51	Q2 in Software
Software Quality Journal	0.45	Q2 in Software
The Journal of Supercomputing	0.44	Q2 in Software
SpringerPlus	0.43	Q1 in Multidisciplinary
Innovations in Systems and Software Engineering	0.37	Q3 in Software
IET Software	0.27	Q2 in Software
Journal of Software Engineering Research and Development	0.21	Q3 in Software

3.1.2. The Most Active and Influential Researcher

The most Active and Influential Researcher in the field of Software Effort Estimation From the main study selected, the researcher contributed very well and who very active in the field of Software Effort Estimation research. Here are the most active and influential researchers in the field of Software Effort Estimation, show in the figure 5, including: Mohammad Azzeh, Bardsiri Khatibi, Elham Khatibi, Ekrem Kocaguneli, Menzies Team, Ali Bou Nassif, Sun-Jen Huang, Nan-Hsing Chiu, Jawawi, D.N.A, Magne Jørgensen, Moataz A Ahmed, Satapathy Shashank Mouli, Rath Santanu Kumar, Hashim, S.Z.M, Ali Idri, Alain Abran and Mohamed Hosni.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

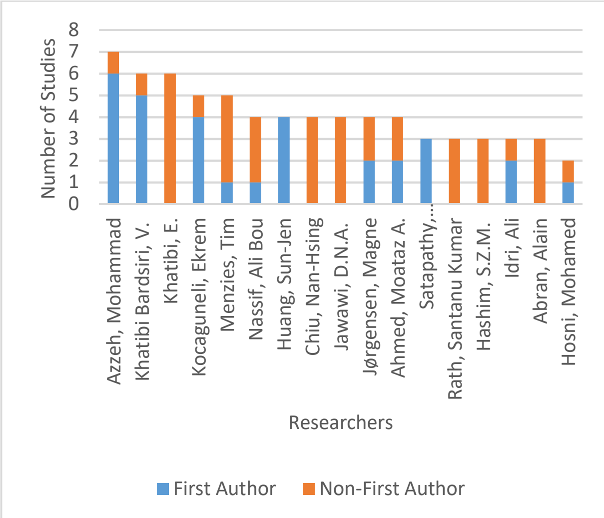

Figure 5: Influential Researchers and Number of Studies

3.2. Research Topic Trends (RQ1)

Although the use of methods in software development is increasing, the problem of effort estimation remains a challenge in software development efforts, largely because of the lack of many standard metrics that will be used for plan-based prediction [58]. Term estimation are applied when used to predict future award value, provided by the effort, in terms of monthly programmers, to conclude software development [59].

Discussed the issue of SEE projects, Research found in such areas [60] [61] [62][63] : 1) Creation and evaluation of estimation methods; 2) Calibration of estimation model; 3) Software system size measures; 4) Assessment of uncertainty; 5) Measurement and analysis of error estimation; 6) Organizational problems related to estimation; 7) Measure and analyze estimation errors; and 8) Data set properties.

The accuracy of estimation by analogue means that the estimation of software attempts by analogy is an appropriate estimation method. The estimation-based analogy also offers several advantages: easy to understand the approximate basis and this is useful where the domain is difficult to model [64]. Estimation by analogy, subjective choice of comparison criterion and process of difference identification and requires analogues project for comparison which is rarely achievable in software development [41]. The analysis of primary studies selected in this study, will focus on five topics in software effort estimation, among others:

The first type of **Classification**, presents a classification for embedded software development projects using ANN and SVM. After determining the classification, the effort estimation model was created for each class by using linear regression, ANN, and the SVR [65]. Context of effort-aware classification scenario, text mining based models perform similarly to software metrics based models in most cases [66] [67]. Using Model Tree has the advantage of dealing with categorical attributes, minimizing user interaction and improving the efficiency of the learning model through classification [68]. Selective classification of software projects based on fundamental attributes to localize the process of estimating development efforts in models using Analogy-Based Estimation (ABE) [69]. Hybrid model that consists of classification and prediction stages using a SVM and Radial Basis Neural Networks (RBNN) [70]. The Localized multi-estimator model avoids the blind classification methods and follows the classification of projects based on underlying attributes [2]. Using classification and data structures can also assist in optimizing the accuracy of Analogy Software effort Estimation. The option of a Fuzzy Feature Subset (FFSS) has a significant impact on accuracy [47].

The second type of **Clustering**, Although a clustering-based approach has been introduced, it still has potential problems to provide accurate and stable effort estimates [71] and SVR [72]. Clustering techniques, especially the clustering of K-Means [73] [74] [75] [76] [77], and Univariat, to know the unit test metrics that are less volatile, that is less influenced by the style adopted by the developer when writing unit test code [78]. In the case of clustering, each document is forced to join exactly one cluster. topic analysis and labeling have been combined to identify latent patterns and trends in the dataset. The two main topic modeling techniques are Latent Semantic Indexing (LSI) and Latent Dirichlet Allocation (LDA) [79]. This variant is clustering using Scott-Knott statistical test and is ranked by using four unbiased measurement errors [51]. Classified into an effort class, refer to the models generated in this study as duplex output models as they return two outputs [80]. The proposed fuzzy set generation process is based on the Fuzzy C-Means Clustering Technique (FCM) [11] [81] and a Real Coded Genetic Algorithm (RCGA) [82]. Used one-way ANOVA, t-tests, boxplots and Tukey's post-hoc test in order to examine if the clusters found by the clustering procedure have significant differences with respect to the size of the project [83]. Clusters using c-means clustering technique [2]. Fuzzy Subtractive Clustering and Artificial Neural Networks to estimate the development effort using class points [84]. The hybrid method is proposed to improve the accuracy of estimation of development

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

effort based on the combination of fuzzy clustering, ABE method and ANN [13]. The clustering algorithms used in this work are Density Based Spatial Clustering Of Application With Noise (DBSCAN) and unsupervised k-windows [85]. The k-means and Scheffe methods are adapted for constructing data clustering models [86]. Clustering techniques improves the estimation accuracy of analogy-based effort estimation techniques [87]. The fuzzy k-mode algorithm, a well-known clustering technique for large datasets containing categorical values [88].

The thrid type of **Estimation**, In this paper, results from using Linear Regression Model (LRM), compared with three Fuzzy Logic Models (FLM). There are two stages of the comparison model in the estimation model: (1) checking the adequacy of the model should be determined; and (2) the estimation model is validated using new data. The results show slightly better prediction accuracy between FLM and LRM to estimate development effort on a personal level when small programs are developed [89]. GRA (Gray Relational Analysis) is used to reduce the uncertainty in the distance of the distance between two software projects for both continuous and categorical features. That the use of GRA integration with fuzzy logic produces credible estimates when compared with Case Based Reasoning (CBR), Multiple Linear Regression (MLR) and ANN methods [43]. The combination of ANN into the fuzzy inference process for software effort estimation has the advantage of providing an objective set of fuzzy rules by utilizing the learning mechanisms of ANN methods [48]. Bagging ensembles of Regression Trees (RTs) show to perform well, being highly ranked in terms of performance across different data sets [20]. Need to consider outlier elimination and to conduct a detailed analysis of effort estimation results to improve the accuracy of software estimates within the software organization [90].

The fourth type of **Predicting**, Software development efforts can be predicted using various approaches and require large datasets from past projects while others require strong input from domain experts [91]. There are two current models that have been widely used to predict rework attempts for changing needs that are algorithmic and non-algorithmic models [92]. The uncertainty inherent in the software development process presents special challenges for predictive software attempts, systematically dealing with missing data values, outlier detection, selection of subset features and all of this in the context of noisy data [93]. To get predictions on better software projects, it is necessary to make more accurate predictions about their development efforts. Based on mathematical models, such as statistical regression or machine learning (ML) [94]. The accuracy of the prediction accuracy of the general regression neural network is statistically equal to or better than that obtained with the fuzzy logic model as well as by multiple linear regression [95] and CBR [96].

The five type of **Dataset Analysis**, the dataset obtained after the pre-processing and attribute selection of the original dataset. For a project's estimated effort, various machine learning models have been selected. There are various machine learning tools that will help for data analysis. Standard dataset for software effort estimation available mostly from data sources, such as the ISBSG (International Software Benchmarking Standars Group), Koten and Gray, COCOMO, NASA, Albrecht, Desharnais, Maxwell and Kemerer. Given that ISBSG is a large and heterogeneous data set, it is necessary to prepare data before applying any analysis to obtain minimal homogeneity in the sample to be studied [97]. Conduct a thorough statistical analysis of the five most popular datasets for estimating software effort to provide researchers with useful information and to help them choose an appropriate repository. The software engineering community must be aware of and responsible for the problem of software related data sets when evaluating the validity of research results [98]. The ISBSG has estimated it in the form of a normal effort and calculates a variable called ratio of normalization. The normalization ratio is derived from the division of normal effort by reported effort, which shows the difference between the reported effort and the estimated effort [2].

It can be concluded that most of the software researcher estimates choose classification as a research topic. Because the topic of research related to the classification is still a lot of opportunities in the industry, the reason is related to the cost and time, if there is a mistake in estimating software development will result in losses in a company. and subsequently relates to the use of public dataset which in testing software estimation. based on the total distribution of research topics on estimation of software efforts from 2000 to 2017. 6% obtained from research studies related to predicting technique topics, 28% of the studies focused on estimation topics, 57% of the studies focused on classification topics and 8% of the major studies related to the topic clustering. and the last is the research topic of dataset analysis of 1% coverage, show in the figure 6.

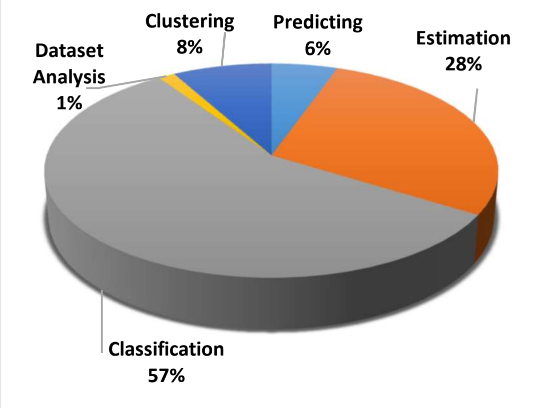

Figure 6: Distribution of Research Topics

3.3. Datasets Used (RQ2)

Raise awareness of how dataset properties affect results when evaluating estimation methods. Establishing effort estimation models from different historical datasets reveals various levels of accuracy in estimation [86]. However, in this case many studies have reported a comparison of the relative accuracy of data estimation efforts with dataset classification, but there are still many shortcomings in the classification of datasets. Thus, to determine the effect of estimation accuracy when the data classification method is used to determine the appropriate software group for the effort development estimation model, this is important in this study.

Historical data is very important and valuable for software development. The quality of the repository greatly affects the results and efficiency of the effort estimation model [99]. The dataset allows specialists to perform their analysis on a recurring and comparable basis in one field of study. However, it is impossible to compare the results of the research of a proposed model, because their datasets can not be assessed. so the importance of using public datasets, so as to compare the results of the model. In the past decade, many researchers have used various types of datasets for various purposes and have tried to find their own features, including: DPS (Data Processing Services), ISBSG (International standard benchmarking software group), Desharnais, Maxwell, and CF (Canadian financial) are the most popular among these data set for software effort estimation [98]. In the literature review on studies, the most dataset for software effort estimation is NASA, ISBSG, COCOMO, Desharnias and Albercht.

The datasets used for evaluation is important because its performance depends on dataset characteristics such as size, number of features, missing data and outliers [100]. They usage of public datasets for evaluate and compared these models in software development effort estimation [101] [102] [103]. Selecting an optimal feature subset that describes the software project is believed to provide the most accurate estimation [51][87][82]. That the prediction accuracy for each technique varies depending on the dataset used, with feature selection will produce the most accurate prediction across all datasets [91]. These datasets have been built and developed by various companies, some of which are cross-business and others are projects related to a single company [104]. The datasets is made publicly available in order to encourage repeatable, verifiable, refutable, and improvable predictive models of software engineering [105]. Different by Martín et al (2008), where the estimate of development effort at the personal level when small programs are developed [89].

In a review of this literature, 74 key studies that analyzed the performance of software effort estimation were included. Figure 7, shows the distribution of dataset types from 2000 to 2017. 86% of the study studies used public datasets and 14% of the study studies used a private dataset.

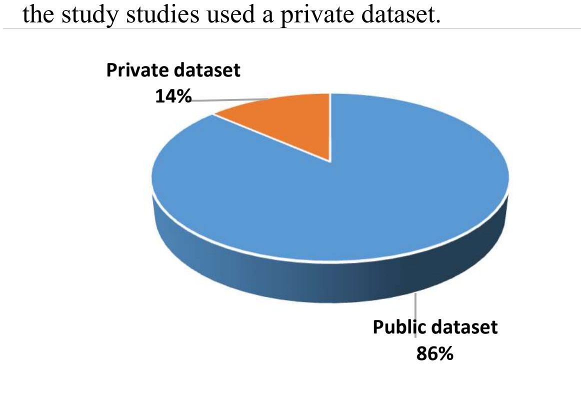

Figure 7: Total Distribution of Datasets

Shown in Figure 8, is a collection of published studies, mostly using more public datasets for software effort estimation studies since 2006 (see Appendix B).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

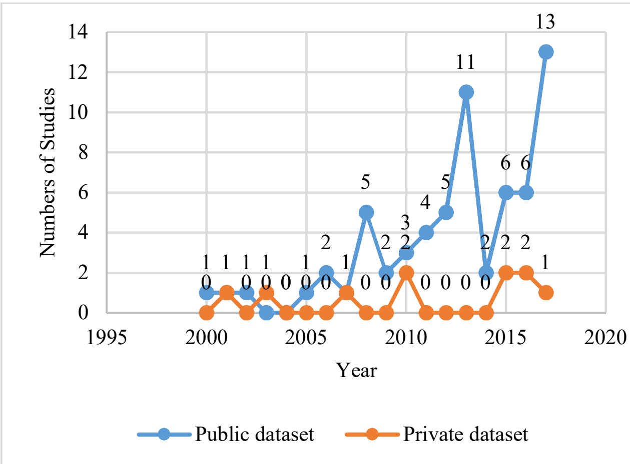

Figure 8: Distribution of Private and Public Datasets

In Appendix B, the most widely used dataset and those related to software effort estimation are Repository in PROMISE and ISBSG, which is one of the most popular datasets [98][106][107][108]. In using and selecting the right subset of data, it must fully understand the concepts and meanings of each dataset, because the problem of pre-processing and data preparation is an important task in the data mining domain [98]. The selection of inappropriate datasets will lead to unreal and biased results [98]. The level of accuracy in the algorithm is very dependent on the dataset used in the field of software effort estimation, because each dataset has different characteristics. So the importance of using datasets in this study.

Table 4 describes each feature of the dataset set, and summarizes the number of projects collected, the minimum and maximum values of the software effort in each data set.

Tabel 4: Data Set Summary

Dataset	Project	Features	Min Effort	Max Effort
ISBSG	148	10	24	60.270
COCOMO	63	17	5.9	11.400
NASA93	93	17	8.4	8.211
NASA	60	16	8.4	3.240
Desharnias	81	9	546	23.940
Albercht	24	7	0.50	105.20
Sdr	12	23	1	22
China	499	18	26	54.620
Kemerer	15	7	23	1.170
Miyaki	48	8	5.6	1.586
Maxwell	62	25	583	63.694
Finnish	38	5	460	26.670

Errors in the selection of inappropriate datasets can cause difficulties in developing the estimation model, so that it will get biased research results. in this section is used to analyze the characteristics of the most popular datasets used in the field of software effort estimation.

3.4. Most Used methods (RQ3)

3.4.1. Distribution of methods

From the selected study, we identified Sixteen (16) types of methods have been applied to software effort estimation (Appendix B). Nine (9) of the most widely applied, They are listed as follows: Neural Networks (NN); Case-Based Reasoning (CBR); Linear Regression (LiR); Fuzzy Logic (FL); Genetic Algorithms (GA); K-Nearest Neighbor (k-NN); Support Vector Regression (SVR); Logistic Regression (LR); and Decission Tree (DT).

Based on the results of a review of several studies, then obtained eight frequently used methods, presented in the figure 9.

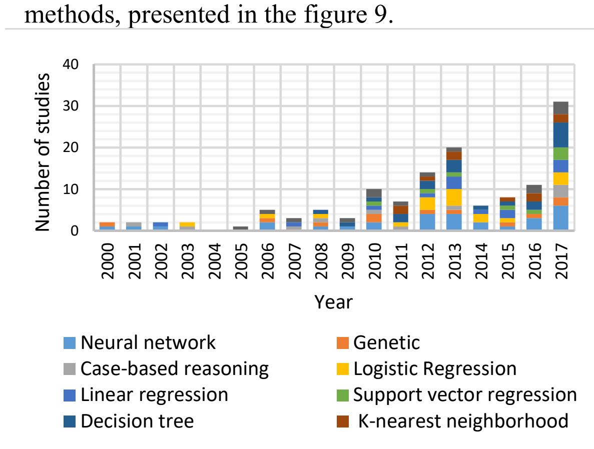

Figure 9: Distribution of the studies over publication year

Neural network (NN) and Decission tree (DT) are the two most commonly used algorithms. As illustrated in Figure 10.

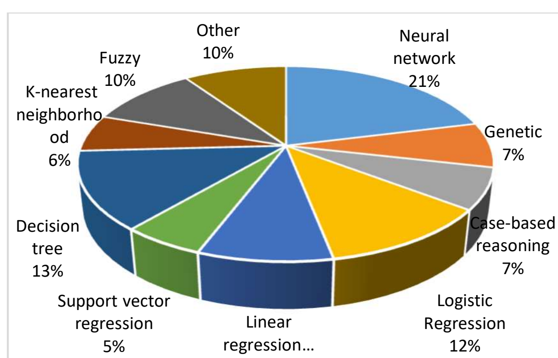

Figure 10: Distribution of Methods

Based on the previous figure, the comparison of the techniques used in software effort estimation is the most widely used NN and DT in recent decades. How to identify techniques in the software

Journal of Theoretical and Applied Information Technology
31st January 2019. Vol.97. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

effort estimation can be done by: stand alone or in combination, either a combination using two or more machine learning techniques (ML) or a combination of ML with Non Machine Learning (Non-ML).

Stand-alone algorithms using machine learning are NN [109] [110], CBR [3][111][112] and k-NN [82][113][114], while those using Non-ML are fuzzy [51][115]. Furthermore the combination method that is done using two or more ML is NN and Genetic [116] or using a combination of ML and Non-ML is NN and fuzzy [117][118].

We found out about the ML technique used for software effort estimation that has the highest and most relevant accuracy values in this literature review.

3.4.2. Machine Learning Method

An important process in software is to the right and accurate software efforts estimation. The current software estimates have switched to various machine learning methods (ML). The accurate estimate of software development efforts is closely related to the success or failure of software projects. The lack of accuracy and versatility in this field has attracted the attention of researchers over the past few years. Despite the improvements achieved in the estimating effort, there is no strong agreement on which individual models work best [2]. Some estimation models of software development efforts have been developed in recent decades. Determining which is the best estimation model is difficult to decide for the software management team [119].

Since 2008 interest in using Machine Learning for improved accuracy, the most widely used method is NN, followed by Model Tree, Classification and Regression Trees (CART), and GA [29]. Specific cross-validation approaches on different datasets to evaluate the accuracy of the learning model forecast and testing to analyze strengths and weaknesses in terms of accuracy, toughness and generalization [81]. A new comprehensive methodology for estimating software development efforts during the initial phase of development needs using the software's functional size as the main variable. Although, effort estimates in practice are mostly done by subjective evaluation, there are many works in this field trying to build parametric models for estimating effort [120]. Approaches for comparison of these models are often invalid and may make things worse. Identified several theoretical problems with a study comparing different estimation models in several common datasets to produce the best models [121]. This shows that developing a comparative study is an open issue, so it is still worth developing again.

GA-based approach significantly improves the classification accuracy and has fewer input features for SVM [122]. Integration of the GRA with GA method presents more precise estimates over the results using the CBR, CART, and ANN methods [123]. GA are applied to include feature selection and parameter optimization of machine learning methods, proving to be very efficient for the search for optimal or optimal solutions in a wide range of problems [16]. SVM and ANN models show better estimation capabilities compared to linear regression model [26].

NN result in performance improvements over conventional regression analysis in terms of average absolute error percentages [124]. k-NN parametric techniques in which the reaction reaction for a predetermined input value is obtained by finding out the average training case closest to a predetermined value, resulting in a minimum error and a higher prediction accuracy achieved by applying an effort estimation model [33]. Comparison of accuracy between Multiple Linear Regression (MLR), General Regression Neural Network (GRNN), and Fuzzy Logic model using Magnitude of Error Relative (MER) and Mean Square Error (MSE) to the estimate, statistically the accuracy results are the same [95].

The resulting model performance uses various neural networks compared and analyzed to improve the prediction accuracy of the software effort estimation process [73]. The ranking method ranks each feature in the dataset. The results are validated using different algorithms for classification. There are several classification algorithms available, where each algorithm has its own strengths and weaknesses. In all supervised learning problems, there is no learning algorithm that works best for individuals [38]. The existing machine learning algorithms provide good accuracy when classifying major class instances, but ignore/classify minority classification [125]. High levels of non-normality and variance and complex relationships between attributes and development efforts can cause serious problems for efficient project classification [31].

We found that ML techniques used in the field of software effort estimation are very consistent with the findings of several other relevant review works in a few years. For example, Huang et al (2008) conducting experiments on 5 learning machine methods, by integrating GRA with the GA method presents a more precise estimate of the results using CBR, CART, and ANN [123]. Hybrid estimators show a more accurate estimate than a single estimator for the dataset. Experimental results show

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

that both single and hybrids are used in the chosen combination proven that the approximate combination achieved by a single and hybrid estimator can reduce the bias in the final estimate [2].

3.4.3. Proposed Method Improvements

Some researchers have proposed the best technique in terms of increasing accuracy in software program estimation. The proposed technique has recently tried to improve the accuracy in the estimation of the resulting model by: the ensemble methods available generally improve the software effort estimation provided by the machine learning [126][127][25][52]; using bagging algorithm [9][15]; add feature selection [16][17][15][47]; using optimization parameters for classifier [16][10][128].

The value of the method parameters depends on the accuracy. In addition, the selection of input features may also have an important influence on the estimation accuracy [16]. Estimation by analogy is one of the machine learning techniques that predicts the software effort based on the premise that the more similar features the software project description [43]. Software effort estimation are optimization issues so that they can also be solved with Meta-heuristic algorithms. There are more than one algorithm available today to find the optimal solution for a particular problem [5].

Researchers proposed various ensemble methods such as boosting, bagging and random sampling techniques [129]. Sampling technique is a method of balancing data to classify unbalanced data [125]. Random Undersampling and oversampling are common types of sampling techniques [130]; and Synthetic Minority Over-sample Technique [131]. Some representative approaches combine oversampling and undersampling data preprocessing with classifier ensembles through boosting [132][133] or bagging [134][133][22]. The combined clustering-based undersampling approach yields optimal performance in small and large data sets [130]. Bagging techniques generally outperform boosting, and hence in noisy data environments, bagging is the preferred method for handling class imbalance [135]. Then in this bagging technique to handle the class imbalance.

Non-linear optimization problems can be solved effectively by Meta-heuristic Algorithms [5]. Implementation of this algorithm can be calculated in various ways to solve the optimization problem [136][5]. As for the use of meta-heuristics to explore the parameter setting with the aim to improve effort predictions [137]. The features adopted by the classifier are then selected as an optimal feature with the wrapper model, using a meta-heuristic approach to help search for the best feature parts [24]. Meta-heuristic methods tailored to solve this problem are: GA and three local search methods using annealing simulations, tabu search, and iterated local searches [138]. Ant Colony Optimization (ACO) and Bee Colony Optimization (BCO) are famous meta-heuristic search algorithms used in solving numerous combinatorial optimization problems [139]; satin bower bird optimization algorithm (SBO) [10]; Particle Swarm Optimization (PSO) [140]. PSO algorithm is chosen as an optimization algorithm because it can present acceptable performance in the field of estimating software development efforts [2].

Accurate software effort estimates are essential for efficient project planning software, because to the complex nature of the software. Estimation of development efforts has become a challenging issue that must be taken seriously. Although many models of effort estimation have been proposed over the last decade, the degree of accuracy is not satisfactory enough.

3.4.4. Feature Selection

Feature selection is how to identify and remove irrelevant and excessive features. Because in the selection of features is very important on large data to address a large number of input features. The feature selection is done by searching for subset feature space and evaluating each part. The search method is selected to perform searches and evaluators assign values to each feature section [141]. Feature selection to extract relevant data in feature space, so feature sets are more suitable for classification [139]. Feature selection is an important task in most classification problems, it still needs a new approach to determine the sub-feature options to improve the accuracy of classification [142]. Features selection in supervised learning has the primary goal of finding parts of features that produce higher classification accuracy [143][27]. The characteristic dataset affects the performance of feature selection techniques, this has an impact on classifier accuracy issues and the time complexity of various feature selection techniques.

Attribute noisy is caused by an error in the value of the attribute (the variable being measured incorrectly, the missing value) while the class interruption is caused by a sample that is labeled to be owned in more than one class [144]. In predictions or estimation problems, better performance can be achieved by removing some variables or features, that is, reducing the data dimension [15]. Estimates based on analogy, there have been a number of studies that investigate the impact of selecting feature subsets on prediction accuracy [47]. Lack of analogy-based systems such as noisy intolerance, intolerance of irrelevant features, sensitivity to choice of algorithms, similarity functions, etc [145]. With the feature selection most techniques provide higher predictive accuracy and this accuracy is more stable across different datasets [91]. Feature selection for the purpose of reducing dimension of dataset size by eliminating irrelevant and redundant features. datasets with relevant features that can lead to an increase in the accuracy of their estimates [39]. Feature selection must be created for the creation of a subset of candidate variables, because feature selection affects the prediction accuracy of each performance model [143].

Traditional method of feature selection has been widely used for some purposes, especially for better classification, but some specific feature data exist in the database that can change the class. So it needs to refine the feature data for several different classes compared to the traditional class. Additionally there are some sensitive feature values (sub-features) of individual features playing an important role leading to a new class or a unique class [142]. Feature selection is a difficult task in pattern recognition, because it requires searching through spaces that may be of high dimension. Complete search is a computing barrier especially when there are a large number of features that cause a reduction in dimensions [146].

Classification problems generally involve a number of features, as not all features are just as important for a particular task. there is even the possibility of excessive or even irrelevant. Will result in better performance by removing some features. In other circumstances, the dimensions of the input space can be reduced to save some computing effort, although this may slightly lower the classification accuracy [24]. Features selection based on feature prediction and redundancy by using cross validation for each method [141].

The feature selection algorithm is separated into three categories [147]: (1) method of this filter, because they filter attributes that are irrelevant before the induction process occurs; (2) the wrapper method, which produces a set of candidate features, run induction algorithm in the training data, and use the precision of the resulting description to evaluate the feature set; and (3) Embedded techniques that combine the feature selection step and classifier construction.

Here are six feature selection techniques, the purpose of this technique is to remove the irrelevant or redundant features of the feature vector provided. The filter method is used to evaluate each section. commonly used methods, statistics and entropy-based, with good performance across multiple domains: Information Gain (IG) attribute evaluation, Gain Ratio (GR) attribute evaluation, Symmetrical Uncertainty (SU) attribute evaluation, Relief-F (RF) attribute evaluation, One-R (OR) attribute evaluation, and Chi-Squared (CS) attribute evaluation [38].

3.4.5. Ensemble Machine Learning

Ensemble learning is one technique that combines at least two different solo variants of the same software effort estimation technique or a combination of one ensemble learning (such as Bagging, Negative Correlation or Random and one solo technique [51].

SEE is a strategic task that is important in software management. Several studies have used machine learning ensembles for this task. investigate the use of ensemble learning machines for SEE. Machine learning ensembles are a set of students who are trained to perform the same tasks and are combined with the aim of improving predictive performance. When combining students in an effort to get more accurate predictions, it is generally agreed that students must be different from each other. If not, the overall prediction will not be better than individual predictions. So, different ensemble learning approaches can be seen as different ways to produce diversity among basic learners [126].

This methodology has the following advantages compared to previous work using ensembles [126]:

1. Use of principled experiments, taking into account the choice of parameters and statistical tests.
2. Comparison using three different ensemble methods.
3. Use of a large number of data sets.
4. Experimental analysis that prioritizes the most frequent method behavior among the best to improve SEE.
5. Risk analysis for outlier evaluation.

This approach falls into two main categories: parametric models, and machine learning models. The importance of accurate effort estimates has led to extensive research efforts in this area. The current method can be classified into the following categories: (1) Parametric models: COCOMO, Software Lifecycle Management (SLIM) and Software Evaluation and Estimation of Resources - Software Estimating Model (SEER-SEM); (2) Expert judgment; (3) Learning oriented techniques:

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

machine learning methods and analogy based estimation; (4) Regression based methods: ordinary least square regression and robust regression; (5) Dynamics based models; and (6) Composite methods [10].

That ensembles are not statistically better than single learners, our study reports that (through the right strategy) ensembles can outperform single learners [129]. The main idea of ensemble is training every MultiLayer Perceptrons (MLP) with a special training set. Each training set is produced by a training project randomly selected from the original training device, which contains all previous projects. After each project is selected, it is replaced back to the original set. This method is called bootstrapping and it is considered the best way to form specific training sets for domains with very small datasets such as software estimation [148]. Bagging, like boosting, is a meta-learning technique that constructs an ensemble of models in order to improve classification performance [135].

3.5. Validation Methods to Accuracy (RQ4)

Accroding Idri et al (2015), accuracy in software effort estimation depends on several categories of parameters, including: (1) Dataset characteristics used (size, missing value, outliers, etc.); (2) Analogy process configuration (feature selection, uniformity measure, adaptation formula, etc.); and (3) The evaluation method used (out-of-out cross validation, disagreement, n-fold cross validation, evaluation criteria, etc.) [149].

Several methods are used to evaluate the approximate accuracy value of the software effort estimation approach. Accuracy of effort estimates can be measured by various metrics, and different metrics measure the accuracy of various aspects. these are some of the most popular accuracy assessments of these are Leave-One-Out Cross Validation (LOOCV), n-fold cross validation (n> 1) and holdout [150][7]. While selection criteria to define accuracy evaluation methods for software engineering estimation as follows; Mean Magnitude of Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), and predicted percentage (Pred (25)) [150][7]. which are calculated as follows [151][69]:

$$MRE = \frac{|estimated - actual|}{actual} \quad (1)$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE}{N} \quad (2)$$

$$PRED(X) = \frac{A}{N} \quad (3)$$

Cross-validation testing is a standard procedure used to evaluate many machine learning algorithms. Behind this test is to divide training data into a number of partitions, also known as folds. The classifier is evaluated by its classification accuracy on one partition after learning of the remaining. This procedure is then repeated until all partitions have been used for evaluation [152]. The cross validation methodology is used to compare the model by dividing the data into two segments: (1) to learn or train the model and (2) for testing to validate the model. In typical cross-validation, the training set and test set must be cross-over in successive rounds so that each data point has a chance to be validated [81]. It is shown in Table 4, that various historical project data sets are most often used and present relevant information to validate the Machine Learning (ML) model.

Accuracy metrics is an evaluation method in the ML model that must be considered, in addition to data sets and validation. besides that, accuracy metrics need to be used in testing to determine the effect of the reduction results on the work of the Machine Learning model. in this study the accuracy matrix used is MMRE (Mean Magnitude of Relative Error), Pred (25), and MdMRE (Median Magnitude of Relative Error) are the three most popular accuracy metrics (Appendix C) by adopting them to evaluate the accuracy of model estimates Machine Learning.

Table 5 in Appendix C, presents the results of the algorithm performance evaluation. in this case GA has a little performance evaluation, because the GA technique is more often to evaluate the weight that is most suitable for each software driver as feature selection and feature weighting in the combined ML model. Measurement metrics in this study, to measure the direction of the study. Because this study focuses on three known and widely used metrics, MMRE, MdMRE, and PRED (0.25) are used to measure estimator accuracy. this selection of metrics makes the results comparable to previous studies. In addition to evaluating performance metrics and measurement metric tables to show the results of accuracy in the estimation model, in an effort to reveal the truth or bias value of performance metrics.

The accuracy of the ML model is acceptable and better than the statistics, with an average MMRE relative error ranging from 35%-55%, PRED (25) 45%-75% and MdMRE of 30%-55%. But it also depends on the dataset that is applied to build the model and the preprocessing approach of the data taken, the ML algorithm can produce different results.

In the ML performance model measured in MMRE and Pred (25) (see Appendix C), NN has the most accurate performance with a median MMRE of about 35% and the Pred median (25) of around 70%. NN-based models have demonstrated the ability to estimate different predictions from previous experiences [153]. Followed by Fuzzy, LR, k-NN , CBR, and DT with median MMRE and median Pred (25) around 45%, while GA has the worst accuracy, because most GA is only used as feature selection and parameter optimization.

3.6. Implications for Research

The most important part in the process of estimating a soft device is an accurate estimate. because excessive or underestimated estimates can have consequences or result in losses in a company. because this is very much related to the cost and scheduling of the project. based on the results of the review there have been a number of estimation models proposed, but none of the models provide accurate estimation results on different datasets. many studies have developed an estimation model using ML and non-ML techniques, even doing hybrids with both models. The results in this literature review, by reviewing several techniques used in software effort estimation include Neural Networks (NN); Case-Based Reasoning (CBR); Linear Regression (LiR); Fuzzy Logic (FL); Genetic Algorithms (GA); K-Nearest Neighbor (k-NN); Support Vector Regression (SVR); Logistic Regression (LR); and Decision Tree (DT).

Therefore, researchers are encouraged to conduct research in this field by using ML techniques to produce even better accuracy. By looking for some ML techniques that are not presented in this literature review or doing hybrids with several algorithms. Because the field of software effort estimation using ML techniques can still be developed again. in addition, the problems that affect the accuracy of ML performance are very much dependent on historical software project data, because each dataset has different characteristics that affect the way to analyze the ML model. Without uniformity of use of datasets, it will produce various comparisons in each ML technique. The need for feature selection and parameter optimization in the dataset to improve accuracy. In this review study only limited to relevant studies and limited to experimental studies used to determine and compare each performance in ML techniques.

So that it is important to use historical software project data that can be used to make improvements, so that it can be developed again in the ML model to achieve significant accuracy and be based on a new estimation system model. After analyzing the results of the empirical study, that with different datasets and different machine learning algorithms shows different results with different algorithms.

3.7. Limitations of This Review

This section will review the performance of the ML model and compare the performance results of each technique in ML. because most of the reviews in this study, using accuracy accuracy to measure accuracy results, where measurement accuracy in ML techniques is very important. What is used is knowing the strengths and weaknesses of each ML technique by using several historical datasets.

The results of the analysis on RQ4 are very important to identify the software effort estimation model and ML technique that are precise and have very significant accuracy, so that it can be used as a reference for the development of research in this field. Researchers are encouraged to be able to develop the best ML model and technique. By referring to the results of this review, to find out which ML techniques have good performance and historical datasets that are most suitable for making estimates for accurate and unbiased results.

In this study, we conducted a review of the sharing of previous experimental studies involving several ML algorithms and several public datasets that were used to develop and build estimation models. As well as to find out the accuracy results of most experimental studies using validation methods (MMRE, PRED (25) and MdMRE). Besides that, data preparation is a very important process in building the ML model, which includes several stages such as: selection, cleaning, reduction, transformation and selection of features used to avoid bias in accuracy. In the data preparation process used to build the ML model, resulting in accurate accuracy in predictions.

Several studies that have been analyzed have found several strengths and weaknesses in estimating the ML model and historical dataset. Therefore, it is possible that some opinions are only used to represent the results of their studies. Here the importance of researchers is to be able to analyze and develop new models, by looking for several opportunities available from previous studies. As well as need to remember that the results of the accuracy are very dependent on the collection of historical datasets.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

In addition, the quality assessment process from the study can ensure that these strengths and weaknesses come from research, where quality results are acceptable.

Limitations and disadvantages of the empirical study, writing only applies to a number of studies selected in the SEE field in the year previously determined. Therefore there are several possibilities, that excellence, strength and weakness in the Machine learning technique presented based on the reviews in the selected study are only the opinions of the author and cannot be relied on fully. It is expected that the readers will be wiser in comparing the previous studies that are used as references in the SEE field. So that the author, in providing reliability in this emperical study, was supported by several studies that chose significantly. Besides, the quality of the study chosen has the strengths and weaknesses of each that can be accepted. Therefore it is important to sort out the results of studies from this empirical study.

4. CONCLUSION AND FUTURE WORKS

This paper presents an overview of related literature in the field of software effort estimation. The purpose of identifying and analyzing the methods used is in the literature review in research published between January 2000 and December 2017. The software effort estimation is a very important field of science, because The ability to accurately and consistently predict software development efforts is required in planning and conducting software development activities.

It can be concluded that some of the benefits of software effort estimation are as follows: 1) Establishment and evaluation of estimation methods in developing software; 2) improving software quality and knowing estimated effort; 3) identify effort estimation in the software; 4) Improved estimation techniques will facilitate time and budget; and 5) can predict, monitor, control, and assess software development. Note that this research question is a research question for literature review. They are different from research questions for the main research in this paper.

Research topic trends chosen by researchers in the field of software effort estimation, there are 9 topics: Estimation methods, Production functions, Calibration of models, Size measures, Organizational issues, Effort uncertainty assessments, Measures of estimation performance, and Data set properties.

We identified fiveteen (15) types of methods have been applied to software effort estimation (Appendix B). Nine (9) of the most widely applied, They are listed as follows: Neural Networks (NN); Case-Based Reasoning (CBR); Linear Regression (LiR); Fuzzy Logic (FL); Genetic Algorithms (GA); K-Nearest Neighbor (k-NN); Support Vector Regression (SVR); Logistic Regression (LR); and Decission Tree (DT).

What types of validation and evaluation are used to measure the accuracy of the overall estimates of the model in the field of software effort estimation are Leave-One-Out Cross-Validation (LOOCV), n-fold Cross-Validation, and Holdout. Selection criteria to determine the method of accuracy evaluation for software engineering estimation as follows; Average Relative Error (MMRE), Median Magnitude of Relative Error (MdMRE), and percentage prediction (Pred (25)).

This paper is to answer system questions and provide past and present works found in the literature. Many research opportunities are still available along this line and further investigations for SEE in different methods and classification of questions. Finally, the list of major studies is presented in Table 4. This list consists of 6 attributes (years, primary studies, publications, datasets, and methods) and 74 primary studies on SEE (Appendix B).

For future work, it is important to review the SEE field using complete and general machine learning techniques, by increasing the number of studies that must be done in machine learning techniques to compare performance.

REFERENCES:

[1] M. Jørgensen, K. H. Teigen, and K. Moløkken, "Better sure than safe? Over-confidence in judgement based software development effort prediction intervals," *J. Syst. Softw.*, vol. 70, no. 1–2, pp. 79–93, 2004.

[2] V. K. Bardsiri, D. N. A. Jawawi, A. K. Bardsiri, and E. Khatibi, "LMES: A localized multi-estimator model to estimate software development effort," *Eng. Appl. Artif. Intell.*, vol. 26, no. 10, pp. 2624–2640, 2013.

[3] D. Wu, J. Li, and Y. Liang, "Linear combination of multiple case-based reasoning with optimized weight for software effort estimation," *J. Supercomput.*, vol. 64, no. 3, pp. 898–918, 2013.

[4] L. L. Minku and X. Yao, "Which models of the past are relevant to the present? A

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

software effort estimation approach to exploiting useful past models," *Autom. Softw. Eng.*, vol. 24, no. 3, pp. 499–542, 2017.

[5] R. Kishore and D. . Gupta, "Software Effort Estimation using Satin Bowerbird Algorithm," vol. 5, no. 3, pp. 216–218, 2012.

[6] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens, "Data mining techniques for software effort estimation: A comparative study," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 375–397, 2012.

[7] J. Wen, S. Li, Z. Lin, Y. Hu, and C. Huang, "Systematic literature review of machine learning based software development effort estimation models," *Inf. Softw. Technol.*, vol. 54, no. 1, pp. 41–59, 2012.

[8] M. Hosni, A. Idri, A. Abran, and A. Bou, "On the value of parameter tuning in heterogeneous ensembles effort estimation," *Soft Comput.*, 2017.

[9] N. Saini and B. Khalid, "Effectiveness of Feature Selection and Machine Learning Techniques for Software Effort Estimation," *IOSR J. Comput. Eng.*, vol. 16, no. 1, pp. 34–38, 2014.

[10] S. H. Samareh Moosavi and V. Khatibi Bardsiri, "Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation," *Eng. Appl. Artif. Intell.*, vol. 60, no. May 2016, pp. 1–15, 2017.

[11] M. Azzeh, D. Neagu, and P. I. Cowling, "Analogy-based software effort estimation using Fuzzy numbers," *J. Syst. Softw.*, vol. 84, no. 2, pp. 270–284, 2011.

[12] S. Aljahdali and A. F. Sheta, "Software effort estimation by tuning COOCMO model parameters using differential evolution," *ACS/IEEE Int. Conf. Comput. Syst. Appl. - AICCSA 2010*, pp. 1–6, 2010.

[13] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "Increasing the accuracy of software development effort estimation using projects clustering," *IET Softw.*, vol. 6, no. 6, p. 461, 2012.

[14] T. Urbanek, Z. Prokopova, R. Silhavy, and V. Vesela, "Prediction accuracy measurements as a fitness function for software effort estimation," *Springerplus*, vol. 4, no. 1, pp. 1–17, 2015.

[15] P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, "A GA-based Feature Selection and Parameters Optimization for Support Vector Regression Applied to Software Effort Estimation Chromosome design," pp. 1788–1792, 2008.

[16] A. L. I. Oliveira, P. L. Braga, R. M. F. Lima, and M. L. Cornelio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," *Inf. Softw. Technol.*, vol. 52, no. 11, pp. 1155–1166, 2010.

[17] Q. Liu, J. Xiao, and H. Zhu, "Feature selection for software effort estimation with localized neighborhood mutual information," *Cluster Comput.*, vol. 3456789, no. 1, 2018.

[18] J. Huang, Y. F. Li, J. W. Keung, Y. T. Yu, and W. K. Chan, "An empirical analysis of three-stage data-preprocessing for analogy-based software effort estimation on the ISBSG data," *Proc. - 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2017*, pp. 442–449, 2017.

[19] P. Phannachitta, J. Keung, A. Monden, and K. Matsumoto, "A stability assessment of solution adaptation techniques for analogy-based software effort estimation," *Empir. Softw. Eng.*, vol. 22, no. 1, pp. 474–504, 2017.

[20] L. L. Minku and X. Yao, "Ensembles and locality: Insight on improving software effort estimation," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1512–1528, 2013.

[21] R. Malhotra, "Software Effort Prediction using Statistical and Machine Learning Methods," vol. 2, no. 1, pp. 145–152, 2011.

[22] J. Błaszczyński and J. Stefanowski, "Neighbourhood sampling in bagging for imbalanced data," *Neurocomputing*, vol. 150, no. PB, pp. 529–542, 2014.

[23] H. Velarde, C. Santiesteban, A. Garcia, and J. Casillas, "Software Development Effort Estimation based-on multiple classifier system and Lines of Code," *IEEE Lat. Am. Trans.*, vol. 14, no. 8, pp. 3907–3913, 2016.

[24] S. W. Lin, K. C. Ying, S. C. Chen, and Z. J. Lee, "Particle swarm optimization for parameter determination and feature selection of support vector machines," *Expert Syst. Appl.*, vol. 35, no. 4, pp. 1817–1824, 2008.

[25] K. Iwata, T. Nakashima, Y. Anan, and N. Ishii, "Effort Estimation for Embedded

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Software Development Projects by Combining Machine Learning with Classification," 2016.

[26] S. Aljahdali, A. F. Sheta, and narayan C. Debnath, "Estimating Software Effort and Function Point Using Regression , Support Vector Machine and Artificial Neural Networks Models," *IEEE Access*, 2015.

[27] İ. Babaoglu, O. Findik, and E. Ülker, "A comparison of feature selection models utilizing binary particle swarm optimization and genetic algorithm in determining coronary artery disease using support vector machine," *Expert Syst. Appl.*, vol. 37, no. 4, pp. 3177–3183, 2010.

[28] J.-C. Lin, C.-T. Chang, and S.-Y. Huang, "Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression," *2011 Int. Symp. Comput. Sci. Soc.*, pp. 349–352, 2011.

[29] M. Fernández-diego and F. González-ladrón-de-guevara, "Potential and limitations of the ISBSG dataset in enhancing software engineering research : A mapping review," vol. 56, pp. 527–544, 2014.

[30] A. Klair and R. Kaur, "Software Effort Estimation using SVM and kNN," *Int. Conf. Comput. Graph. Simul. Model.*, pp. 28–29, 2012.

[31] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A flexible method to estimate the software development effort based on the classification of projects and localization of comparisons," *Empir. Softw. Eng.*, vol. 19, no. 4, pp. 857–884, 2014.

[32] A. L. I. Oliveira, "Estimation of software project effort with support vector regression," *Neurocomputing*, vol. 69, no. 13–15, pp. 1749–1753, 2006.

[33] S. M. Satapathy, "Empirical Assessment of Machine Learning Models for Effort Estimation of Web-based Applications," *ISEC '17, ACM*, pp. 74–84, 2017.

[34] C. Catal, O. Alan, and K. Balkan, "Class noise detection based on software metrics and ROC curves," *Inf. Sci. (Ny).*, vol. 181, no. 21, pp. 4867–4877, 2011.

[35] V. Khatibi Bardsiri and E. Khatibi, "Insightful analogy-based software development effort estimation through selective classification and localization," *Innov. Syst. Softw. Eng.*, vol. 11, no. 1, pp. 25–38, 2015.

[36] V. Resmi, S. Vijayalakshmi, and R. S. Chandrabose, "An effective software project effort estimation system using optimal firefly algorithm," *Cluster Comput.*, 2017.

[37] E. Kocaguneli and T. Menzies, "Software effort models should be assessed via leave-one-out validation," *J. Syst. Softw.*, vol. 86, no. 7, pp. 1879–1890, 2013.

[38] J. Novaković, P. Strbac, and D. Bulatović, "Toward optimal feature selection using ranking methods and classification algorithms," *Yugosl. J. Oper. Res.*, vol. 21, no. 1, pp. 119–135, 2011.

[39] M. Hosni, A. Idri, and A. Abran, "Investigating Heterogeneous Ensembles with Filter Feature Selection for Software Effort Estimation," *ACM*, no. 2, 2017.

[40] N. Acir, Ö. Özdamar, and C. Güzeliş, "Automatic classification of auditory brainstem responses using SVM-based feature selection algorithm for threshold detection," *Eng. Appl. Artif. Intell.*, vol. 19, no. 2, pp. 209–218, 2006.

[41] P. Pospieszny, B. Czarnacka-Chrobot, and A. Kobyliński, "An effective approach for software project effort and duration estimation with machine learning algorithms," *J. Syst. Softw.*, 2017.

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

[43] M. Azzeh, D. Neagu, and P. I. Cowling, "Fuzzy grey relational analysis for software effort estimation," *Empir. Softw. Eng.*, vol. 15, no. 1, pp. 60–90, 2010.

[44] A. Bhardwaj, A. Tiwari, H. Bhardwaj, and A. Bhardwaj, "A genetically optimized neural network model for multi-class classification," *Expert Syst. Appl.*, vol. 60, pp. 211–221, 2016.

[45] K. L. Do Tuan, K. A. Yoon, Y. S. Seo, and D. H. Bae, "Filtering of inconsistent software project data for analogy-based effort estimation," *Proc. - Int. Comput. Softw. Appl. Conf.*, pp. 503–508, 2010.

[46] C. L. Huang and C. J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231–240, 2006.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[47] M. Azzeh, D. Neagu, and P. Cowling, "Improving Analogy Software Effort Estimation using Fuzzy Feature Subset Selection Algorithm," *PROMISE ACM*, pp. 71–78, 2008.

[48] S.-J. Huang and N.-H. Chiu, "Applying fuzzy neural network to estimate software development effort," *Appl. Intell.*, vol. 30, no. 2, pp. 73–83, 2009.

[49] Suharjito, S. Nanda, and B. Soewito, "Modeling software effort estimation using hybrid PSO-ANFIS," *2016 Int. Semin. Intell. Technol. Its Appl.*, pp. 219–224, 2016.

[50] U. R. Saxena and S. P. Singh, "Software effort estimation using Neuro-fuzzy approach," *2012 CSI Sixth Int. Conf. Softw. Eng.*, pp. 1–6, 2012.

[51] A. Idri, M. Hosni, and A. Abran, "Improved estimation of software development effort using Classical and Fuzzy Analogy ensembles," *Appl. Soft Comput. J.*, vol. 49, pp. 990–1019, 2016.

[52] J. J. Cuadrado-Gallego, P. Rodríguez-Soria, and B. Martín-Herrera, "Analogies and Differences between Machine Learning and Expert Based Software Project Effort Estimation," *2010 11th ACIS Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput.*, pp. 269–276, 2010.

[53] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering : An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, 2015.

[54] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," 2007.

[55] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences*. 2005.

[56] H. Zhang, T. Irish, S. Engineering, and M. A. Babar, "On Searching Relevant Studies in Software Engineering," pp. 1–10, 2005.

[57] B. A. Kitchenham, I. C. Society, E. Mendes, and G. H. Travassos, "Cross versus Within-Company Cost Estimation Studies : A Systematic Review," vol. 33, no. 5, pp. 316–329, 2007.

[58] S. Dragicevic, S. Celar, and M. Turic, "Bayesian network model for task effort estimation in agile software development," *J. Syst. Softw.*, vol. 127, pp. 109–119, 2017.

[59] R. D. A. Araújo, S. Soares, and A. L. I. Oliveira, "Hybrid morphological methodology for software development cost estimation," *Expert Syst. Appl.*, vol. 39, pp. 6129–6139, 2012.

[60] M. Sadiq, F. Mariyam, A. Alil, S. Khan, and P. Tripathi, "Prediction of software project effort using fuzzy logic," *2011 3rd Int. Conf. Electron. Comput. Technol.*, vol. 4, pp. 353–358, 2011.

[61] M. Shepperd and M. Cartwright, "Predicting with sparse data," *IEEE Trans. Softw. Eng.*, vol. 27, no. 11, pp. 987–998, 2001.

[62] G. C. Low and D. R. Jeffery, "Function Points in the Estimation and Evaluation of the Software Process," vol. 16, no. 893, pp. 64–71, 1990.

[63] M. Jorgensen and M. Shepperd, "A Systematic Review of Software Development Cost Estimation Studies," *IEEE Trans. Softw. Eng.*, vol. 33, no. 1, pp. 33–53, 2007.

[64] P. R. Hill, *Practical Software Project Estimation : A Toolkit for Estimating Software Development Effort & Duration*. Mc Graw Hill, 2011.

[65] K. Iwata, T. Nakashima, Y. Anan, and N. Ishii, "Effort prediction models using self-organizing maps for embedded software development projects," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, pp. 142–147, 2011.

[66] Y. Tang, F. Zhao, Y. Yang, H. Lu, Y. Zhou, and B. Xu, "Predicting Vulnerable Components via Text Mining or Software Metrics? An Effort-Aware Perspective," *Proc. - 2015 IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2015*, pp. 27–36, 2015.

[67] A. K. Pandey and N. K. Goyal, "Test effort optimization by prediction and ranking of fault-prone software modules," *2010 2nd Int. Conf. Reliab. Saf. Hazard, ICRESH-2010 Risk-Based Technol. Physics-of-Failure Methods*, pp. 136–142, 2010.

[68] M. Azzeh, "Model Tree based adaption strategy for software effort estimation by analogy," *Proc. - 11th IEEE Int. Conf. Comput. Inf. Technol. CIT 2011*, pp. 328–335, 2011.

[69] V. Khatibi Bardsiri and E. Khatibi, "Model to estimate the software development effort based on in-depth analysis of project attributes," *IET Softw.*, vol. 9, no. 4, pp. 109–118, 2015.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[70] M. Azzeh and A. B. Nassif, "A hybrid model for estimating software project effort from Use Case Points," *Appl. Soft Comput. J.*, vol. 49, pp. 981–989, 2016.

[71] Y. S. Seo, D. H. Bae, and R. Jeffery, "AREION: Software effort estimation based on multiple regressions with adaptive recursive data partitioning," *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1710–1725, 2013.

[72] E. Kocaguneli, A. Tosun, and A. Bener, "AI-based models for software effort estimation," *Proc. - 36th EUROMICRO Conf. Softw. Eng. Adv. Appl. SEAA 2010*, pp. 323–326, 2010.

[73] A. Panda, S. M. Satapathy, and S. K. Rath, "Empirical Validation of Neural Network Models for Agile Software Effort Estimation based on Story Points," *Procedia Comput. Sci.*, vol. 57, pp. 772–781, 2015.

[74] O. F. Sarac and N. Duru, "A novel method for software effort estimation: Estimating with boundaries," *Innov. Intell. Syst. Appl. (INISTA), 2013 IEEE Int. Symp.*, pp. 1–5, 2013.

[75] J. Lin, "Applying Particle Swarm Optimization to Estimate Software Effort by Multiple Factors Software Project Clustering," *Int. Comput. Symp.*, pp. 0–5, 2010.

[76] G. Nagpal, M. Uddin, and A. Kaur, "Analyzing Software Effort Estimation using k means Clustered Regression Approach," *ACM SIGSOFT Softw. Eng.*, vol. 38, no. 1, pp. 1–9, 2013.

[77] S. K. Sehra, J. Kaur, Y. S. Brar, and N. Kaur, "Analysis of Data Mining Techniques for Software Effort Estimation," *2014 11th Int. Conf. Inf. Technol. New Gener.*, pp. 633–638, 2014.

[78] F. Toure, M. Badri, and L. Lamontagne, "A metrics suite for JUnit test code: a multiple case study on open source software," *J. Softw. Eng. Res. Dev.*, vol. 2, no. 1, p. 14, 2014.

[79] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation," *Inf. Softw. Technol.*, vol. 91, pp. 1–21, 2017.

[80] S. Mensah, J. Keung, M. F. Bosu, and K. E. Bennin, "Duplex output software effort estimation model with self-guided interpretation," *Inf. Softw. Technol.*, vol. 94, pp. 1–13, 2018.

[81] A. Idri, A. Hassani, and A. Abran, "RBFN networks-based models for estimating software development effort: A cross-validation study," *Proc. - 2015 IEEE Symp. Ser. Comput. Intell. SSCI 2015*, no. Ml, pp. 925–932, 2015.

[82] A. Idri, I. Abnane, and A. Abran, "Missing data techniques in analogy-based software development effort estimation," *J. Syst. Softw.*, vol. 117, pp. 595–611, 2016.

[83] P. Chatzipetrou, E. Papatheocharous, L. Angelis, and A. S. Andreou, "A multivariate statistical framework for the analysis of software effort phase distribution," *Inf. Softw. Technol.*, vol. 59, pp. 149–169, 2015.

[84] S. Kanmani, "Class Point Based Effort Estimation of OO Systems using Fuzzy Subtractive Clustering and Artificial Neural Networks," pp. 141–142, 2008.

[85] T. R. Benala, S. Dehuri, and R. Mall, "ACM SIGSOFT Software Engineering Notes Computational Intelligence in Software Cost Estimation : An Emerging Paradigm Satchidananda Dehuri Rajib Mall ACM SIGSOFT Software Engineering Notes 1 . 2 Why Computational Intelligence in Software Cost Estimation ?," *ACM SIGSOFT Softw. Eng.*, vol. 37, no. 3, pp. 1–7, 2012.

[86] S. J. Huang, N. H. Chiu, and Y. J. Liu, "A comparative evaluation on the accuracies of software effort estimates from clustered data," *Inf. Softw. Technol.*, vol. 50, no. 9–10, pp. 879–888, 2008.

[87] F. A. Amazal, A. Idri, and A. Abran, "Improving Fuzzy Analogy Based Software Development Effort Estimation," *2014 21st Asia-Pacific Softw. Eng. Conf.*, pp. 247–254, 2014.

[88] F.-A. Amazal, A. Idri, and A. Abran, "An Analogy-Based Approach to Estimation of Software Development Effort Using Categorical Data," *2014 Jt. Conf. Int. Work. Softw. Meas. Int. Conf. Softw. Process Prod. Meas.*, pp. 252–262, 2014.

[89] C. López-Martín, C. Yáñez-Márquez, and A. Gutiérrez-Tornés, "Predictive accuracy comparison of fuzzy models for software development effort of small programs," *J. Syst. Softw.*, vol. 81, no. 6, pp. 949–960, 2008.

[90] Y. S. Seo and D. H. Bae, *On the value of outlier elimination on software effort estimation research*, vol. 18, no. 4. 2013.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[91] L. Radlinski and W. Hoffmann, "On predicting software development effort using machine learning techniques and local data," *Int. J. Softw. ...*, vol. 2, no. 2, 2010.

[92] S. Basri, N. Kama, F. Haneem, and S. A. Ismail, "Predicting Effort for Requirement Changes during Software Development," pp. 380–387, 2016.

[93] Q. Song and M. Shepperd, "Predicting software project effort: A grey relational analysis based method," *Expert Syst. Appl.*, vol. 38, no. 6, pp. 7302–7316, 2011.

[94] C. López-Martín, "Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects," *Appl. Soft Comput. J.*, vol. 27, pp. 434–449, 2014.

[95] C. Lopez-Martin, "Applying a general regression neural network for predicting development effort of short-scale programs," *Neural Comput. Appl.*, vol. 20, no. 3, pp. 389–401, 2011.

[96] S. G. Macdonell and A. R. Gray, "A Comparison of Modeling Techniques for Software Development Effort Prediction," pp. 869–872, 1997.

[97] M. Fernández-diego, "Discretization Methods for NBC in Effort Estimation : An Empirical Comparison based on ISBSG Projects," pp. 103–106, 2012.

[98] A. K. Bardsiri, S. M. Hashemi, and M. Razzazi, "Statistical Analysis of the Most Popular Software Service Effort Estimation Datasets," *J. Telecommun. Electron. Comput. Eng.*, vol. 7, no. 1, pp. 87–96, 2015.

[99] B. Turhan, "On the dataset shift problem in software engineering prediction models," *Empir. Softw. Eng.*, vol. 17, no. 1–2, pp. 62–74, 2012.

[100] S. El Koutbi, A. Idri, and A. Abran, "Systematic Mapping Study of Dealing with Error in Software Development Effort Estimation," *2016 42th Euromicro Conf. Softw. Eng. Adv. Appl.*, no. 2, pp. 140–147, 2016.

[101] S. Tariq, M. Usman, R. Wong, Y. Zhuang, and S. Fong, "On Learning Software Effort Estimation," *2015 3rd Int. Symp. Comput. Bus. Intell.*, pp. 79–84, 2015.

[102] F. Sarro, "Search-Based Approaches for Software Development Effort Estimation," pp. 38–43, 2011.

[103] A. Bakir, B. Turhan, and A. Bener, "A comparative study for estimating software development effort intervals," *Softw. Qual. J.*, vol. 19, no. 3, pp. 537–552, 2011.

[104] O. Hidmi and B. E. Sakar, "Software Development Effort Estimation Using Ensemble Machine Learning," *Int'l J. Comput. Commun. Instrum. Engg.*, no. March, 2017.

[105] K. Jeet and D. A. V Jalandhar, "A Model for Estimating Efforts required to make Changes in a Software Development Project," pp. 175–178, 2011.

[106] B. Vasilescu, A. Serebrenik, and T. Mens, "A historical dataset of software engineering conferences," *IEEE Int. Work. Conf. Min. Softw. Repos.*, pp. 373–376, 2013.

[107] tirimula rao Benala, R. Mall, P. Srikavya, and V. HariPriya, "Software Effort Estimation Using Data Mining Techniques," *Adv. Intell. Syst. Comput.*, vol. 248 VOLUME, pp. 85–86, 2014.

[108] D. Déry and A. Abran, "Investigation of the effort data consistency in the ISBSG repository," ... *15th Intern. Work. Softw. ...*, no. June, 2005.

[109] L. L. Minku and X. I. N. Yao, "Software Effort Estimation as a Multiobjective Learning Problem," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 4, 2013.

[110] A. B. Nassif, M. Azzeh, L. F. Capretz, and D. Ho, "Neural network models for software development effort estimation: a comparative study," *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2369–2381, 2016.

[111] J. Li, G. Ruhe, A. Al-Emran, and M. M. Richter, "A flexible method for software effort estimation by analogy," *Empir. Softw. Eng.*, vol. 12, no. 1, pp. 65–106, 2007.

[112] D. Wu, J. Li, and C. Bao, "Case-based reasoning with optimized weight derived by particle swarm optimization for software effort estimation," *Soft Comput.*, pp. 1–12, 2017.

[113] E. Kocaguneli, T. Menzies, J. Keung, D. Cok, and R. Madachy, "Active Learning and effort estimation: Finding the essential content of software effort estimation data," *IEEE Trans. Softw. Eng.*, vol. 39, no. 8, pp. 1040–1053, 2013.

[114] E. Kocaguneli, T. Menzies, and E. Mendes, "Transfer learning in effort estimation," *Empir. Softw. Eng.*, vol. 20, no. 3, pp. 813–843, 2015.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[115] Z. Muzaffar and M. A. Ahmed, "Software development effort prediction: A study on the factors impacting the accuracy of fuzzy logic systems," *Inf. Softw. Technol.*, vol. 52, no. 1, pp. 92–109, 2010.

[116] K. K. Shukla, "Neuro-genetic prediction of software development effort," *Inf. Softw. Technol.*, vol. 42, no. 10, pp. 701–713, 2000.

[117] X. Huang, D. Ho, J. Ren, and L. F. Capretz, "A soft computing framework for software effort estimation," *Soft Comput.*, vol. 10, no. 2, pp. 170–177, 2006.

[118] M. Azzeh, A. B. Nassif, and S. Banitaan, "Comparative analysis of soft computing techniques for predicting software effort based use case points," *IET Softw.*, vol. 12, no. 1, pp. 19–29, 2017.

[119] V. S. Dave and K. Dutta, "Comparison of Regression model , Feed-forward Neural Network and Radial Basis Neural Network for Software Development Effort Estimation," *ACM SIGSOFT Softw. Eng.*, vol. 36, no. 5, pp. 1–5, 2011.

[120] R. Abdukalykov, I. Hussain, M. Kassab, and O. Ormandjieva, "Quantifying the impact of different non-functional requirements and problem domains on software effort estimation," *Proc. - 2011 9th Int. Conf. Softw. Eng. Res. Manag. Appl. SERA 2011*, pp. 158–165, 2011.

[121] B. Kitchenham and E. Mendes, "Why Comparative Effort Prediction Studies may be Invalid," 2009.

[122] S. J. Huang and N. H. Chiu, "Optimization of analogy weights by genetic algorithm for software effort estimation," *Inf. Softw. Technol.*, vol. 48, no. 11, pp. 1034–1045, 2006.

[123] S.-J. Huang, N.-H. Chiu, and L.-W. Chen, "Integration of the grey relational analysis with genetic algorithm for software effort estimation," *Eur. J. Oper. Res.*, vol. 188, no. 3, pp. 898–909, 2008.

[124] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Inf. Softw. Technol.*, vol. 44, no. 15, pp. 911–922, 2002.

[125] Y. Arafat, S. Hoque, and D. Farid, "Cluster-based Under-sampling with Random Forest for Multi-Class Imbalanced Classification," pp. 1–6, 2017.

[126] L. L. Minku and X. Yao, "A Principled Evaluation of Ensembles of Learning Machines for Software Effort Estimation Categories and Subject Descriptors," *PROMISE ACM*, 2011.

[127] Monika and O. P. Sangwan, "Software Effort Estimation Using Machine Learning Techniques," *Ieee*, vol. 5, pp. 92–98, 2017.

[128] V. Khatibi Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "A PSO-based model to increase the accuracy of software development effort estimation," *Softw. Qual. J.*, vol. 21, no. 3, pp. 501–526, 2013.

[129] E. Kocaguneli, T. Menzies, and J. W. Keung, "On the value of ensemble effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1403–1416, 2012.

[130] W. C. Lin, C. F. Tsai, Y. H. Hu, and J. S. Jhang, "Clustering-based undersampling in class-imbalanced data," *Inf. Sci. (Ny).*, vol. 409–410, pp. 17–26, 2017.

[131] M. Hamill and K. Goseva-Popstojanova, "Analyzing and predicting effort associated with finding and fixing software faults," *Inf. Softw. Technol.*, vol. 87, pp. 1–18, 2017.

[132] R. E. Schapire, "The Strength of Weak Learnability (Extended Abstract)," *Mach. Learn.*, vol. 227, no. October, pp. 28–33, 1989.

[133] Y. Liu, E. Shriberg, A. Stolcke, and M. Harper, "Using machine learning to cope with imbalanced classes in natural speech: evidence from sentence boundary and disfluency detection.," *Interspeech*, no. 1, pp. 2–5, 2004.

[134] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[135] T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "Comparing boosting and bagging techniques with noisy and imbalanced data," *IEEE Trans. Syst. Man, Cybern. Part ASystems Humans*, vol. 41, no. 3, pp. 552–568, 2011.

[136] A. Kaveh and V. R. Mahdavi, "Colliding bodies optimization: A novel meta-heuristic method," *Comput. Struct.*, vol. 139, pp. 18–27, 2014.

[137] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes, "Using tabu search to configure support vector regression for effort estimation," *Empir. Softw. Eng.*, vol. 18, no. 3, pp. 506–546, 2013.

[138] J. Reca, J. Martínez, C. Gil, and R. Baños,

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

"Application of several meta-heuristic techniques to the optimization of real looped water distribution networks," *Water Resour. Manag.*, vol. 22, no. 10, pp. 1367–1379, 2008.

[139] P. Shunmugapriya and S. Kanmani, "A hybrid algorithm using ant and bee colony optimization for feature selection and classification (AC-ABC Hybrid)," *Swarm Evol. Comput.*, vol. 36, pp. 27–36, 2017.

[140] J. Mercieca and S. G. Fabri, "A Metaheuristic Particle Swarm Optimization Approach to Nonlinear Model Predictive Control," vol. 5, no. 3, pp. 357–369, 2012.

[141] D. Oreski, S. Oreski, and B. Klicek, "Effects of dataset characteristics on the performance of feature selection techniques," *Appl. Soft Comput. J.*, vol. 52, pp. 109–119, 2017.

[142] H. K. Bhuyan and N. K. Kamila, "Privacy preserving sub-feature selection in distributed data mining," *Appl. Soft Comput. J.*, vol. 36, pp. 552–569, 2015.

[143] M. Ramaswami and R. Bhaskaran, "A Study on Feature Selection Techniques in Educational Data Mining," *J. Comput.*, vol. 1, no. 1, pp. 7–11, 2009.

[144] Zena M. Hira and D. F. Gillies, "A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data," *Adv. Bioinformatics*, vol. 2015, no. 1, 2015.

[145] P. S. Bishnu and V. Bhattacherjee, "Software cost estimation based on modified K-Modes clustering Algorithm," *Nat. Comput.*, vol. 15, no. 3, pp. 415–422, 2016.

[146] Y. Chtioui, D. Bertrand, and D. Barba, "Feature selection by a genetic algorithm. Application to seed discrimination by artificial vision," *J. Sci. Food Agric.*, vol. 76, no. 1, pp. 77–86, 1998.

[147] P. (Institute for the S. of L. and E. Langley, "Selection of Relevant Features in Machine Learning," *Proc. AAAI Fall Symp. Relev.*, pp. 140–144, 1994.

[148] Y. Kultur, B. Turhan, and A. B. Bener, "ENNA : Software Effort Estimation Using Ensemble of Neural Networks with Associative Memory," pp. 330–338, 2008.

[149] A. Idri, F. A. Amazal, and A. Abran, "Analogy-based software development effort estimation: A systematic mapping and review," *Inf. Softw. Technol.*, vol. 58, pp. 206–230, 2015.

[150] A. Idri and A. Abran, "Analogy-based software development effort estimation : A systematic mapping and review," *Inf. Softw. Technol.*, 2014.

[151] I. (Norwegian S. of M. Myrtveit, E. (Buskerud U. C. Stensrud, and M. (Bournemouth U. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, 2005.

[152] J. Murillo-Morera, C. Castro-Herrera, J. Arroyo, and R. Fuentes-Fernández, "An automated defect prediction framework using genetic algorithms: A validation of empirical studies," *Iberamia*, vol. 19, no. 57, pp. 114–137, 2016.

[153] V. S. Dave and K. Dutta, "Neural network based models for software effort estimation: A review," *Artif. Intell. Rev.*, vol. 42, no. 2, pp. 295–307, 2014.

[154] M. Jorgensen and D. I. K. Sjoberg, "Impact of effort estimates on software project work," *Inf. Softw. Technol.*, vol. 43, no. 15, pp. 939–948, 2001.

[155] C. J. Burgess and M. Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation," *Inf. Softw. Technol.*, vol. 43, no. 14, pp. 863–873, 2001.

[156] S. G. MacDonell and M. J. Shepperd, "Combining techniques to optimize effort predictions in software project management," *J. Syst. Softw.*, vol. 66, no. 2, pp. 91–98, 2003.

[157] M. A. Ahmed, M. O. Saliu, and J. Alghamdi, "Adaptive fuzzy logic-based framework for software development effort prediction," *Inf. Softw. Technol.*, vol. 47, no. 1, pp. 31–48, 2005.

[158] S. Grimstad and M. Jørgensen, "Inconsistency of expert judgment-based estimates of software development effort," *J. Syst. Softw.*, vol. 80, no. 11, pp. 1770–1777, 2007.

[159] J. Li and G. Ruhe, *Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+*, vol. 13, no. 1. 2008.

[160] M. A. Ahmed and Z. Muzaffar, "Handling imprecision and uncertainty in software development effort prediction: A type-2 fuzzy logic based framework," *Inf. Softw. Technol.*, vol. 51, no. 3, pp. 640–654, 2008.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[161] M. O. Elish, “Improved estimation of software project effort using multiple additive regression trees,” *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10774–10778, 2009.

[162] M. Jorgensen, “Contrasting ideal and realistic conditions as a means to improve judgment-based software development effort estimation,” *Inf. Softw. Technol.*, vol. 53, no. 12, pp. 1382–1390, 2010.

[163] C.-J. Hsu and C.-Y. Huang, “Comparison of weighted grey relational analysis for software effort estimation,” *Softw. Qual. J.*, vol. 19, no. 1, pp. 165–200, 2011.

[164] C. Lopez-Martin, C. Isaza, and A. Chavoya, “Software development effort prediction of industrial projects applying a general regression neural network,” *Empir. Softw. Eng.*, vol. 17, no. 6, pp. 738–756, 2012.

[165] M. Azzeh, “A replicated assessment and comparison of adaptation techniques for analogy-based effort estimation,” *Empir. Softw. Eng.*, vol. 17, no. 1–2, pp. 90–127, 2012.

[166] E. Kocaguneli and T. Menzies, “The Journal of Systems and Software Software effort models should be assessed via leave-one-out validation,” *J. Syst. Softw.*, vol. 86, no. 7, pp. 1879–1890, 2013.

[167] J. Keung, E. Kocaguneli, and T. Menzies, “Finding conclusion stability for selecting the best effort predictor in software effort estimation,” *Autom. Softw. Eng.*, vol. 20, no. 4, pp. 543–567, 2013.

[168] M. Azzeh, A. B. Nassif, S. Banitaan, and F. Almasalha, “Pareto efficient multi-objective optimization for local tuning of analogy-based estimation,” *Neural Comput. Appl.*, vol. 27, no. 8, pp. 2241–2265, 2015.

[169] S. M. Satapathy, B. P. Acharya, and S. K. Rath, “Early stage software effort estimation using random forest technique based on use case points,” *IET Softw.*, vol. 10, no. 1, pp. 10–17, 2015.

[170] E. Løhre and M. Jørgensen, “Numerical anchors and their strong effects on software development effort estimates,” *J. Syst. Softw.*, vol. 116, pp. 49–56, 2015.

[171] W. Zhang, Y. Yang, and Q. Wang, “Using Bayesian regression and EM algorithm with missing handling for software effort prediction,” *Inf. Softw. Technol.*, vol. 58, pp. 58–70, 2015.

[172] M. Jørgensen, “Unit effects in software project effort estimation: Work-hours gives lower effort estimates than workdays,” *J. Syst. Softw.*, vol. 117, pp. 274–281, 2016.

[173] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, “Negative results for software effort estimation,” *Empir. Softw. Eng.*, vol. 22, no. 5, pp. 2658–2683, 2016.

[174] M. Bisi and N. K. Goyal, “Software development efforts prediction using artificial neural network,” *IET Softw.*, vol. 10, no. 3, pp. 63–71, 2016.

[175] D. Ph, H. Khademi, and M. Fallahnezhad, “Software effort estimation based on the optimal Bayesian belief network,” *Appl. Soft Comput. J.*, 2016.

[176] M. Hosni, A. Idri, A. Abran, and A. B. Nassif, “On the value of parameter tuning in heterogeneous ensembles effort estimation,” *Soft Comput.*, pp. 1–34, 2017.

[177] J. Murillo-Morera, C. Quesada-López, C. Castro-Herrera, and M. Jenkins, *A genetic algorithm based framework for software effort prediction*, vol. 5, no. 1. Journal of Software Engineering Research and Development, 2017.

[178] R. de A. Araújo, A. L. I. Oliveira, and S. Meira, “A class of hybrid multilayer perceptrons for software development effort estimation problems,” *Expert Syst. Appl.*, vol. 90, pp. 1–12, 2017.

[179] C. Lokan and E. Mendes, “Investigating the use of moving windows to improve software effort prediction: a replicated study,” *Inf. Softw. Technol.*, vol. 22, no. 2, pp. 716–767, 2017.

[180] T. R. Benala and R. Mall, “DABE: Differential evolution in analogy-based software development effort estimation,” *Swarm Evol. Comput.*, 2017.

[181] S. M. Satapathy and S. K. Rath, “Empirical assessment of machine learning models for agile software development effort estimation using story points,” *Innov. Syst. Softw. Eng.*, vol. 13, no. 2–3, pp. 191–200, 2017.

[182] F. Qi, X.-Y. Jing, X. Zhu, X. Xie, B. Xu, and S. Ying, “Software effort estimation based on open source projects: Case study of Github,” *Inf. Softw. Technol.*, 2017.

Journal of Theoretical and Applied Information Technology
31st January 2019. Vol.97. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Appendix A

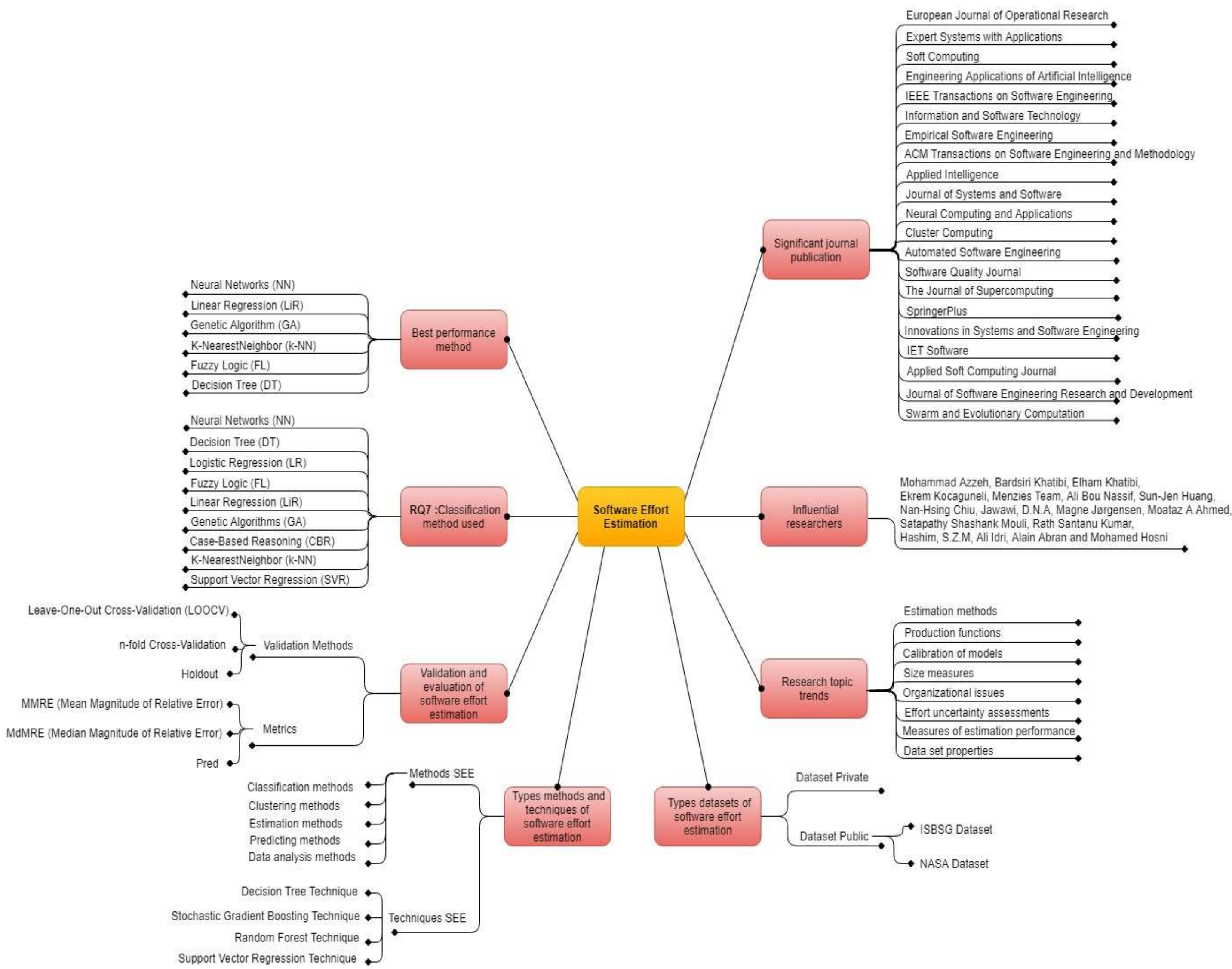

Figure 11: Mind Map of the SLR on Software Effort Estimation

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Appendix B

Table 4: The list Of primary studies

Year	Ref.	Publications	Dataset	Evaluation Methods	Method
2000	[116]	Information and Software Technology	COCOMO and Kemerer	-	Neural network (NN); Genetic
2001	[154]	Information and Software Technology	38 under-graduate computer science students	-	-
	[155]	Information and Software Technology	Desharnais and ASMA (Australian Software Metrics Association)	AMSE, MMRE, BMMRE, PRED (25)	Case-based reasoning (CBR); Genetic programming (GP); Neural networks (NN)
2002	[124]	Information and Software Technology	The IBM DP, Kemerer, and Hallmark data set	MAPE	Artificial neural network; Regression models
2003	[156]	Systems and Software	medical records information	MMRE	Expert judgment; Least squares regression (LSR); Case-based reasoning (CBR)
2005	[157]	Information and Software Technology	COCOMO	RMSRE, PRED (25)	Fuzzy logic
2006	[117]	Soft Computing	COCOMO81	PRED (25)	Neural networks (NN); Fuzzy logic
	[122]	Information and Software Technology	ISBSG and the IBM DP	MMRE, MdMRE, PRED (25)	Unweighted analogy (UA) ; Unequally weighted analogy (UWA); Linearly weighted analogy (LWA); Nonlinearly weighted analogy (NWA); Genetic algorithm; CART; ANN; Ordinary least square (OLS)
2007	[158]	Systems and Software	web-based database system	MMRE	Expert judgment
	[111]	Empirical Software Engineering	USP05-FT, USP05-RQ, ISBSG04, KEM87, Mends03, Leung02	MMRE, MdMRE, PRED (25)	Case-based reasoning (CBR); Collaborative Filtering; AQUA
2008	[89]	Systems and Software	COCOMO81	MMRE, MdMRE,	Fuzzy logic ;Linear regression
	[123]	European Journal of Operational Research	COCOMO and Albrecht	MMRE, PRED (25)	Grey relational analysis (GRA); Genetic; case- based reasoning (CBR); classification and regression trees (CART); artificial neural networks (ANN)
	[159]	Empirical Software Engineering	USP05-FT and USP05-RQ, ISBSG04, Mends03, Kemerer87, and Desharnais89	MMRE, MdMRE, PRED (25)	Rough set analysis; AQUA
	[86]	Information and Software Technology	ISBSG	MMRE, MdMRE, PRED (25)	Ordinary least square (OLS)
	[160]	Information and Software Technology	COCOMO	RMSRE, PRED (10), PRED (25)	Fuzzy logic
2009	[161]	Expert Systems with Applications	NASA	MMRE, PRED (25)	Multiple additive regression trees (MART); Classification and regression trees (CART)
	[48]	Applied Intelligence	COCOMO AND COCOMO II	MMRE, PRED (25)	Fuzzy logic; Artificial neural network (ANN); Fuzzy neural network
2010	[162]	Information and Software Technology	Historical dataset	-	Human judgment
	[43]	Empirical Software Engineering	ISBSG, COCOMO'81, Desharnais, The IBM DP, Kemerer	MRE, MMRE, MdMRE, MMER, PRED (25)	Grey relational analysis (GRA);Fuzzy set theory; Case-based reasoning (CBR); Artificial Neural network (ANN); Multi linear regression (MLR)
	[27]	Expert Systems with Applications	Exercise Stress Testing (EST) dataset	-	Genetic algorithm; Support vector machine (SVM); Exercise stress testing
	[16]	Information and Software Technology	Desharnais, NASA, COCOMO, Albrecht, Kemerer and Koten and Gray	MMRE, PRED (25)	Genetic algorithms; Support vector regression (SVR); Multi layer perceptron (MLP) ANN; M5P (Decision Tree)
	[115]	Information and Software Technology	COCOMO	RMSRE, PRED (10), PRED (25)	Fuzzy logic
2011	[11]	Systems and Software	COCOMO, Desharnias, kemerer, Albrecht	MMRE, MdMRE, PRED (25)	Fuzzy numbers; Case-based reasoning (CBR); Stepwise Regression (SR)
	[103]	Software Quality Journal	Promise (COCOMO_v1, COC'81, Desharnais_1_1, NASA93) and SoftLab (sdr05, sdr06, sdr07)	MRE, MMRE, MdMRE, PRED (25)	Linear discrimination; K-nearest neighborhood (k-NN); Decision tree (DT)
	[163]	Software Quality Journal	Desharnais, ISBSG, COCOMO, Kemerer	MMRE, PRED (25)	Grey relational analysis (GRA)
	[93]	Expert Systems with Applications	Desharnais, Albrecht, COCOMONASA, COCOMO'81, Kemerer	MMRE, PRED (25)	Grey relational analysis (GRA)

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Year	Ref.	Publications	Dataset	Evaluation Methods	Method
2012	[129]	IEEE Transactions on Software Engineering	Desharnais, Albrecht, finnish, NASA93, COCOMO'81, Kemerer, sdr, Maxwell, miyazaki94, telecom, china	MRE, MER, MMRE, PRED (25), MBRE, MIBRE	Regression trees (RT); Support vector machines (SVM); NN; K-nearest neighborhood (k-NN)
	[13]	IET Software	Desharnais, Maxwell	MMRE, PRED (25)	Artificial neural networks; Fuzzy; Analogy based-estimation (ABE); Classification and regression trees (CART); Stepwise regression (SWR); Multiple linear regression (MLR); ABE with MS function and inverse distance weighted mean solution function (ABEMA); ABE with ES function and inverse distance weighted mean solution function (ABEI); ABE with ES function and mean solution function (ABEM)
	[164]	Empirical Software Engineering	ISBSG	MER, MMER	Neural network (NN); General regression neural network
	[6]	IEEE Transactions on Software Engineering	the Experience, ESA, ISBSG, and Euroclear data	MRE, MMRE, MdMRE, PRED (25)	Ordinary least square (OLS)
	[165]	Empirical Software Engineering	Maxwell, Desharnias, COCOMO'81, Kemerer, Albrecht, Telecom, China	MMRE	Mean of closest effort (M); Weighted mean of closest effort (WM); Single size feature adaptation (SS); Multiple size feature adaptation (MS); AQUA; Regression Towards the mean (RTM); Genetic algorithm; Neural network (NN)
2013	[20]	Information and Software Technology	ISBSG, COCOMO81,NASA93, NASA, sdr, desharnais	MMRE, PRED (25)	Bagging; Regression tree (RT)
	[2]	Engineering Applications of Artificial Intelligence	ISBSG, Maxwell and COCOMO	MRE, MMRE, MdMRE, BMMRE, PRED (25)	Analogy based estimation (ABE); CART; Multi linear regression (MLR); Artificial neural networks (ANN); Genetic algorithm; Grey relational analysis (GRA); C-means; localized multi-estimator (LMES); stepwise regression (SWR); PSO
	[137]	Empirical Software Engineering	Albrecht, China, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, MaxwellA2, MaxwellA3, MaxwellS2, MaxwellT1, Miyazaki, Telecom	MRE, MMRE, PRED (25), MdAR, MAR	Support vector regression (SVR); Tabu search
	[3]	The Journal of Supercomputing	Miyazaki, Derhanais	MMRE, MdMRE, PRED (25)	Case-based reasoning (CBR)
	[166]	Systems and Software	Telecom, Kemerer, COCOMO'81, Desharnias, Albrecht, NASA93, Maxwell, sdr, Miyazaki, Finnish, China	-	Calculate bias and variance
	[109]	ACM Transactions on Software Engineering and Methodology	ISBSG	MMRE, PRED (25), LSD	Multi layer perceptron (MLP) NN; Pareto ensemble
	[71]	Information and Software Technology	ISBSG R9	MMRE, MdMRE, PRED (25), PRED (50), MMER, BMMRE	Least squares regression (LSR); Fuzzy
	[113]	IEEE Transactions on Software Engineering	Albrecht, China, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, Maxwell, Miyazaki, NASA93_1, NASA93_2, NASA93_3, COCOMO'81e, COCOMO'81o, COCOMO'81s	MRE, MAR, PRED(25), MBRE, MIBRE, MMER	K-nearest neighborhood (k-NN)
	[90]	Empirical Software Engineering	ISBSG R9, Bank and Stock data sets that are collected from financial companies, Desharnais	MMRE, MdMRE, PREDMRE (25) PREMMRE (50), BMMRE	Least trimmed squares; Cook's distance; K-means clustering; Box plot, and Mantel leverage metric; Least squares regression (LSR); Analogy Based Estimation (ABE)
	[167]	Automated Software Engineering	Albrecht, China, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, Maxwell, Miyazaki94, NASA93_1, NASA93_2, NASA93_3, COCOMO'81e, COCOMO'81o, COCOMO'81s, Finnish, telecom, china	RE, MRE, MER, MMRE, MdMRE, PRED (25), MBRE, MIBRE	Linear regression; Classification and regression trees (CART); Neural networks (NN)
	[128]	Software Quality Journal	IBM data processing services (DPS) organization, Canadian financial (CF) organization, ISBSG	RE, MRE, MMRE, PRED (25)	K-nearest neighborhood (k-NN); stepwise regression (SWR); multiple regression (MLR); CART; Analogy based Estimation (ABE); artificial neural network (ANN)
2014	[31]	Empirical Software Engineering	ISBSG, COCOMO'81	RE, MRE, MMRE, PRED (25)	Analogy Based Estimation(ABE) ; CART; ANN; SWR; MLR.
	[94]	Applied Soft Computing Journal	ISBSG R11	AR, MAR, MdAR	Radial Basis Function Neural Network; General regression neural network;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Year	Ref.	Publications	Dataset	Evaluation Methods	Method
					Feedforward multilayer perceptron (MLP); Statistical regression
2015	[168]	Neural Computing and Applications	Albrecht, China, Desharnais1, Desharnais2, Desharnais3, Finnish, Kemerer, Maxwell, Miyazaki, NASA, COCOMO, COCOMO'81e, COCOMO'81o, COCOMO'81s, Telecom, ISBSG	SA, BRE, IBRE, MBRE, MIBRE	Analogy Based Estimation (ABE): Genetic algorithm; AQUA; Regression toward the mean (RTM); linear size extrapolation (LSE).
	[14]	SpringerPlus	Poznan University of Technology dataset	MMRE, PRED (25), MSE	Analytical programming; Differential evolution generate regression functions
	[169]	IET Software	Albrecht, COCOMO, Desharnais, NASA	MSE, MMRE, PRE (100), PRED (75), PRED (50), PRED (25)	Random forest (RF)
	[170]	Systems and Software	423 software professionals from Romania, Ukraine, Argentina and Poland	-	Anchoring effects; Numerical preciseness
	[35]	Innovations in Systems and Software Engineering	ISBSG	RE, MRE, MMRE, PRED (25)	Analogy Based Estimation (ABE)
	[69]	IET Software	ISBSG	MRE, MMRE, PRED (25)	Analogy Based Estimation (ABE); Classified;); stepwise regression (SWR); multiple regression (MLR); CART; artificial neural network (ANN)
	[114]	Empirical Software Engineering	Tuku, NASA, COCOMO, NASA93, Desharnias, Finnish, Kemerer, Maxwell	AE, MRE, MER, MMRE, MdMRE, PRED (25), MBRE, MIBRE, SA	K-nearest neighborhood (k-NN)
	[171]	Information and Software Technology	ISBSG, CSBSG	MAR, BREM	Linear regression; Bayesian regression; Support Vector Regression (SVR)
2016	[172]	Systems and Software	online survey with 77 software professionals from Norway	-	Judgment bias
	[173]	Empirical Software Engineering	COCOMO	-	COCOMO ; Classification and regression trees (CART); Nearest neighbor
	[70]	Applied Soft Computing Journal	Historical dataset1, dataset2, dataset3	AE, MAE, MBRE, SA	Use Case Points (UCP); Radial basis neural networks; Support vector machine (SVM)
	[110]	Neural Computing and Applications	ISBSG	MR, MAR	Neural network (NN); Multilayer perceptron (MLP); General regression neural network (GRNN); Radial basis function neural network (RBFNN); Cascade correlation neural network
	[51]	Applied Soft Computing Journal	ISBSG, Albrecht, COCOMO81, China dataset, Desharnais, Kemerer, and Miyazaki	MAE, MIBRE, MBRE, LSD, SA, PRED (25)	Fuzzy logic
	[174]	IET Software	COCOMO, Derharnais, Albrecht	MMRE, PRED (25)	Genetic; Multilayer perceptron (MLP); Artificial neural network; SVR; Decision tree (M5P);
	[82]	Systems and Software	ISBSG repository (release 8), COCOMO81, Desharnais, Maxwell, Miyazaki, China and Albrecht	-	K-nearest neighborhood (k-NN)
	[175]	Applied Soft Computing Journal	COCOMO NASA	-	Bayesian belief network; Genetic; Fuzzy numbers
2017	[176]	Soft Computing	Albrecht, COCOMO'81, China, Desharnais, ISBSG, Kemerer, Miyazaki	AE, MRE, MMRE, PRED (25), MBRE, MIBRE, MAE, LSD, SA, Δ	K-nearest neighbor (k-NN); Support vector regression (SVR); Multilayer perceptron (MLP); Decision trees (DT)
	[177]	Journal of Software Engineering Research and Development	ISBSG R12	MdMRE, MMAR, SA, PRED (25)	Genetic; Gaussian Processes (GP); Least MedSq (LMS); LinearRegression (LR); MultilayerPerceptron (MP); RBFNetwork (RBFN; SMOreg (SMO); AdditiveRegression (AR); Bagging (BAG); ConjunctiveRule (CR); DecisionTable (DT); M5Rules (M5R); ZeroR (ZR); DecisionStump (DS); M5P (M5P); REPTree (RT)
	[41]	Systems and Software	ISBSG	ME, MAE, MSE, RMSE, MMRE, MMER, MBRE, PRED (25), PRED (30)	Support vector machines (SVM); Multi-Layer Perceptron Artificial Neural Network (MLP-ANN); Generalized linear models (GLM)
	[112]	Soft Computing	Maxwell, Desharnais	MMRE, PRED (25), MdMRE	Case-based reasoning (CBR)
	[178]	Expert Systems with Applications	Albrecht, COCOMO, Desharnais, Kemerer, KotenGray, NASA	MMRE, PRED (25)	Multilayer perceptrons (MLP); linear regression (LR); logistic regression; Morphological rank linear neural network, Radial basis function; Regression tree (RT); support vector regression (SVR).
	[179]	Information and Software Technology	ISBSG, Finnish	MAE	Linear Regression

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Year	Ref.	Publications	Dataset	Evaluation Methods	Method
	[10]	Engineering Applications of Artificial Intelligence	ISBSG R11, Kemerer	RE, MRE, MMRE, PRED (25)	OABE; Classification and regression trees (CART); SWR; Artificial neural network (ANN); Fuzzy inference system; Non linear adjustment to ABE (NABE); Multiple linear regression (MLR); PSO; SBO
	[36]	Cluster Computing	NASA 93, NASA 60, COCOMO81, Deshnaris	MMRE, MRE, PRED (25)	Artificial neural network (ANN); Fuzzy logic; Case-based reasoning (CBR)
	[180]	Swarm and Evolutionary Computation	Desharnais, NASA, COCOMO, China, Maxwell, Albrecht	MMRE, PRED (25), MdMRE, SA, Δ	Analogy Based Estimation (ABE); K-nearest neighborhood (k-NN); genetic
	[58]	Systems and Software	160 tasks from real agile project	MMRE, MRE, MAE, RMSE, RAE, RRSE	Bayesian Network
	[181]	Innovations in Systems and Software Engineering	Dataset of 21 software projects developed by six number of software houses	MAE, MMER, PRED (25)	Decision tree (DT); Random forest; Stochastic gradient boosting
	[182]	Information and Software Technology	Albrecht, China, Kitchenham, Kemerer, Maxwell, NASA93, COCOMO'81	MRE, PRED (25)	AdaBoost and Classification And Regression Tree (ABCART)
	[19]	Empirical Software Engineering	Albrecht, China, Desharnais, Finnish, Kemerer, Maxwell, Miyazaki94, NASA93-c1, NASA93-c2, NASA93-c5, COCOMO'81, COCOMO-sdr	MAR, MdAR, SD, LSD, RSD	Case-based reasoning (CBR); Analogy based effort estimation (ABE);AQUA; Multiple size adaptation (MSA); Linear size adaptation (LSA); Regression towards the mean (RTM); Unweighted mean of the k analogues (UAVG); Inverse-rank weighted mean of the k analogues (IRWM); Genetic; Neural network (NN)
	[118]	IET Software	234 projects from previous studies, 110 projects developed from information systems projects such as chains of hotels, multi-branch universities and multi-warehouse book stores, and 71 projects developed for different governmental and commercial sectors	AE, MAE, MBRE, MIBRE, SA, Δ	Use case points (UCP); Neural network; ANFIS; Support vector regression (SVR)

Journal of Theoretical and Applied Information Technology
31st January 2019. Vol.97. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Appendix C

Table 5: The list Of Accuracy Values

ID	Dataset	MMRE (%)	PRED (25) (%)	MdMRE (%)	ID	Dataset	MMRE (%)	PRED (25) (%)	MdMRE (%)
CBR									
[111]	Mends03	25^a	76.47^a	-	[11]	Desharnais	38.2	42.9	30.8
[112]	Derharnais	36^b	33^b	40^b	[11]	Albrecht	63.5	33.3	38.9
[112]	Maxwell	28^b	67	19^b	[11]	Kemerer	63.8	40	33.33
[156]	Medical records	54	-	-	[43]	ISBSG	53	41.1	36
[11]	ISBSG	52.32	42.71	30.23	[43]	Desharnais	38.2	42.9	30.8
[11]	COCOMO	47.3	35	33.8	[43]	COCOMO81	29	51.67	25
[123]	COCOMO	446	12	-	[43]	Kemerer	59.6	40	40.9
[123]	Albrecht	58	39	-	[43]	Albrecht	64	33.3	38.9
NN									
[155]	Desharnais	59.23^a	56^a	-	[109]	Desharnais	49.86	33.33	-
[48]	COCOMO81-1	38	33	41	[109]	NASA93	178.66	19.70	-
[48]	COCOMO81-2	44	43	30	[109]	ISBSG	203.17	17.44	-
[48]	COCOMO81-3	29	43	28	[2]	COCOMO	75	29	64
[48]	COCOMO81	37^a	40^a	28^a	[2]	ISBSG	122	17	108
[13]	Desharnais	51	33.641	-	[2]	maxwell	97	28	88
[13]	Maxwell	127.27	24.127	-	[167]	Kemerer	-	27	-
[2]	ISBSG	122	17	108	[167]	DesharnaisL3	-	40	-
[2]	Maxwell	97	28	88	[167]	NASA93_center_2	-	57	-
[2]	COCOMO	75	29	64	[167]	NASA93	-	39	-
[41]	ISBSG	21^b	64.65^b	-	[167]	COCOMO81s	-	18	-
[10]	Albrecht	92.5	25	-	[167]	Albrecht	-	42	-
[10]	Kemerer	57	40	-	[167]	Telecom1	-	39	-
[10]	ISBSG	100	24.1	-	[167]	COCOMO81	-	16	-
[10]	Albrecht	23.5^b	62.5^b	-	[167]	NASA93_center_5	-	33	-
[10]	Kemerer	26.8^b	60^b	-	[167]	DesharnaisL1	-	35	-
[10]	ISBSG	49^b	63.7^b	-	[167]	COCOMO81o	-	21	-
[16]	Desharnais	31.54	72.22	-	[167]	DerharnaisL2	-	40	-
[16]	NASA	19.50	94.44	-	[167]	COCOMO81e	-	7	-
[16]	COCOMO	21.94^a	78.74^a	-	[167]	Desharnais	-	32	-
[16]	Albrecht	68.63	61.67	-	[167]	Sdr	-	29	-
[16]	Kemerer	33.49	64	-	[167]	Miyazaki94	-	25	-
[16]	Koten & Gray	12.19	92.94	-	[167]	Maxwell	-	15	-
[117]	COCOMO81	-	71^b	-	[167]	Finnish	-	37	-
[165]	Maxwell	182.6^b	-	-	[167]	NASA93_center_1		33	
[165]	Desharnais	60.2^b	-	-	[167]	China	-	43	-
[165]	COCOMO	158.6^b	-	-	[36]	Desharnais	72	28.3	-
[165]	Kemerer	56.4^b	-	-	[36]	COCOMO81	143	47.6	-
[165]	Albrecht	80.6^b	-	-	[36]	COCOMONASA60	19	73	-
[165]	Telecom	60.3^b	-	-	[36]	COCOMONASA93	111	34	-
[174]	NASA	19.50	94.44	-	[165]	China	54.3^b	-	-
[174]	Desharnais	31.54	72.22	-	[43]	ISBSG	9.5	44.9	29.5
[174]	Albrecht	68.63	61.67	-	[43]	Desharnais	61.2	44	42.1
[122]	IBM DP	104	17	51	[43]	COCOMO81	55.5	50	42.2
[122]	ISBSG	170	12	94	[43]	Kemerer	47.9	50	37.6
[109]	COCOMO81	279.14	13	-	[43]	Albrecht	79.6	25	52.6
[109]	Sdr	192.54	14.44	-	[123]	COCOMO	143	11	-
[109]	NASA	108.05	42.67	-	[123]	Albrecht	86	21	-
[128]	DPS	90	22	-	[178]	Albrecht	14.84^a	95.83^a	-
[128]	CF	70	10	-	[178]	COCOMO	10.96^a	89.9^a	-
[128]	ISBSG	96	22	-	[178]	Desharnais	15.28^a	83.48^a	-
					[178]	Kemerer	45.81^a	40^a	-
					[178]	Kotengray	46.99^a	47.05^a	-
					[178]	NASA	15.38^a	77.77^a	-
LiR									
[89]	Gathered	26^b	67^b	13^b	[11]	Kemerer	161.73	6.7	74.88
[165]	Maxwell	48.2	-	-	[43]	COCOMO81	130.2	25	58.9
[165]	Desharnais	47.2	-	-	[43]	Kemerer	54.3	46.7	39.7
[165]	COCOMO	58.5	-	-	[43]	Albrecht	59.3	20.8	47.1
[165]	Kemerer	81.4	-	-	[2]	COCOMO	154	15	131
[165]	Albrecht	71.4	-	-	[2]	ISBSG	149	12	113
[165]	Telecom	-	-	-	[2]	maxwell	108	23	97
[165]	China	77.7	-	-	[13]	Desharnais	54	33.641	-
[167]	COCOMO	81	78	76	[13]	Maxwell	196.07	16.11	-
[43]	ISBSG	33.2	48.6	26.5	[10]	Albrecht	101	25	-
[43]	Desharnais	39.9	42	38.2	[10]	Kemerer	71	20	-
[11]	ISBSG	48.75	36.80	38.29	[10]	ISBSG	89.1	23.4	-
[11]	COCOMO	96.6	23.1	82.4	[128]	DPS	73	30	-
[11]	Desharnasi	34.6	45.5	28.6	[128]	CF	98	27	-
[11]	Albrecht	61.24	37.5	32.3	[128]	ISBSG	132	16	-

Journal of Theoretical and Applied Information Technology
31st January 2019. Vol.97. No 2

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

ID	Dataset	MMRE (%)	PRED (25) (%)	MdMRE (%)	ID	Dataset	MMRE (%)	PRED (25) (%)	MdMRE (%)
[178]	Albrecht	9.25^a	95.83^a	-					
[178]	COCOMO	11.33^a	94^a	-					
[178]	Desharnais	9.58^a	91.69^a	-					
[178]	Kemerer	18.75^a	73.33^a	-					
[178]	Kotengray	20.8^a	76.47^a	-					
[178]	NASA	18.07^a	77.77^a	-					
Fuzzy									
[89]	Gathered	23^b	67^b	18^b	[10]	Kemerer	26.8^b	60^b	-
[157]	COCOMO	-	70.59	-	[10]	ISBSG	49^b	63.7^b	-
[48]	COCOMO81-1	24^b	86^b	10^b	[11]	ISBSG	28.55	59.80	17.80
[48]	COCOMO81-2	22^b	71^b	15^b	[11]	COCOMO	33.37	62.33	20.36
[48]	COCOMO81-3	21^b	67^b	12^b	[11]	Desharnais	26.89	64.94	19.32
[48]	COCOMO81	22^a	75^a	12^a	[11]	Albrecht	50.08	50	30.75
[43]	ISBSG	33.3	55.2	22	[11]	Kemerer	55.65	53.33	24.24
[43]	Desharnais	30.6	64.7	17.5	[36]	Desharnais	4.10	79.63	-
[43]	COCOMO81	23.2	66.7	14.8	[36]	COCOMO81	15.6	81	-
[43]	Kemerer	36.2	52.9	33.2	[36]	COCOMONASA60	7.81	85.5	-
[43]	Albrecht	51.1	48.6	38	[36]	COCOMONASA93	5.62	88.25	-
[117]	COCOMO81	-	71^b	-	[160]	COCOMO	-	45.7^a	-
[10]	Albrecht	23.5^b	62.5^b	-	[115]	COCOMO	-	97.35^a	-
GA									
[165]	Maxwell	159.7^b	-	-	[180]	Albrecht	1.8^b	25^b	1.8^b
[165]	Desharnais	56.7^b	-	-	[180]	China	10^b	16.7^b	10^b
[165]	COCOMO	76.3^b	-	-	[180]	COCOMO81	9.7^b	73.5^b	9.8^b
[165]	Kemerer	33.7^b	-	-	[180]	NASA93	0.9^b	11.8^b	0.9^b
[165]	Albrecht	55.8^b	-	-	[2]	COCOMO	62^b	41^b	50^b
[165]	Telecom	53.1^b	-	-	[2]	ISBSG	69^b	28^b	55^b
[165]	China	53.2^b	-	-	[2]	maxwell	81^b	31^b	76^b
SVM									
[41]	ISBSG	13	76.91	-					
GRA									
[43]	ISBSG	33.3	55.2	22	[163]	Desharnais	36	57.1	-
[43]	Desharnais	30.6	64.7	17.5	[163]	ISBSG	269.3	19.2	-
[43]	COCOMO81	23.2	66.7	14.8	[123]	COCOMO	69	38	-
[43]	Kemerer	36.2	52.9	33.2	[123]	Albrecht	31	48	-
[43]	Albrecht	51.1	48.6	38	[93]	Albrecht	66.2	42.1	26.7
[2]	COCOMO	41^b	53^b	36^b	[93]	COCOMONASA	29.5	58.3	18.1
[2]	ISBSG	41^b	49^b	33^b	[93]	COCOMO81	59.5	30.2	55.6
[2]	maxwell	59^b	50^b	50^b	[93]	Desharnais	49.75	45.5	29.8
[163]	Kemerer	65.3	20	-	[93]	Kemerer	47.8	53.3	23.2
[163]	COCOMO	86.5	14.2	-					
k-NN									
[103]	COCOMO81	189^a	33^a	183^a	[180]	Albrecht	2^b	37.5^b	1.9^b
[103]	COCOMONASA_V1	69^a	42^a	45^a	[180]	China	6.8^b	57.3^b	3.9^b
[103]	Desharnais_1_1	13^a	84.14^a	12^a	[180]	COCOMO81	5^b	81.6^b	4.2^b
[103]	NASA93	69^a	55.5^a	52^a	[180]	NASA93	7.4^b	15.7^b	1.8^b
[103]	Sdr05	45^a	45.5^a	28^a	[128]	DPS	26	62	-
[103]	Sdr06	30^a	50.5^a	31^a	[128]	CF	38	69	-
[103]	Sdr07	14^a	81.33^a	13^a	[128]	ISBSG	64	51	-

''+'' means combining data sets, ''-'' means not applicable
^{a} mean of accuracy values.
^{b} accuracy value under optimal model configuration