NOVA IMS Information Management School

MGI

Mestrado em Gestão de Informação
Master Program in Information Management

Statistical Analysis for Revealing Defects in Software Projects.

Alia Nabil Mahmoud Faried Elsayed

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

STATISTICAL ANALYSIS FOR REVEALING DEFECTS IN SOFTWARE PROJECTS

by

Alia Nabil Mahmoud Faried Elsayed

Dissertation presented as partial requirement for obtaining the Master's degree in Information Management/ Master's degree in Statistics and Information Management , with a specialization in Systems Management and Information Technologies.

Advisor :Prof. Vítor Manuel Pereira Duarte dos Santos

ACKNOWLEDGEMENTS

First, I would like to thank Prof. Vitor Santos of the Information Management School at Nova University.

He was always available to help whenever I faced any trouble or had a questions about my research.

He leaded me in the right the way whenever he thought I needed it.

Finally, I must express my very deep acknowledge to my parents, my husband, my sisters, my brothers and my work friends for providing me with constant support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

This achievement would not have been possible without them.

Thank you.

PUBLICATIONS RESULTING FROM THIS DISSERTATION

Mahmoud, A.; Santos, V. (2021) Statistical Analysis for Revealing Defects in Software Projects: Systematic Literature Review. The International Journal of Advanced Computer Science and Applications – vol.12, pp.

ABSTRACT

Defect detection in software is the procedure to identify parts of software that may comprise defects. Software companies always seek to improve the performance of software projects in terms of quality and efficiency. They also seek to deliver the soft-ware projects without any defects to the communities and just in time. The early revelation of defects in software projects is also tried to avoid failure of those projects, save costs, team effort, and time. Therefore, these companies need to build an intelligent model capable of detecting software defects accurately and efficiently.

This study seeks to achieve two main objectives. The first goal is to build a statistical model to identify the critical defect factors that influence software projects. The second objective is to build a statistical model to reveal defects early in software pro-jects as reasonable accurately. A bibliometric map (VOSviewer) was used to find the relationships between the common terms in those domains. The results of this study are divided into three parts:

In the first part The term "software engineering" is connected to "cluster," "regression," and "neural network." Moreover, the terms "random forest" and "feature selection" are connected to "neural network," "recall," and "software engineering," "cluster," "regression," and "fault prediction model" and "software defect prediction" and "defect density."

In the second part We have checked and analyzed 29 manuscripts in detail, summarized their major contributions, and identified a few research gaps.

In the third part Finally, software companies try to find the critical factors that affect the detection of software defects and find any of the intelligent or statistical methods that help to build a model capable of detecting those defects with high accuracy.

Two statistical models (Multiple linear regression (MLR) and logistic regression (LR)) were used to find the critical factors and through them to detect software defects accurately. MLR is executed by using two methods which are critical defect factors (CDF) and premier list of software defect factors (PLSDF). The accuracy of MLR-CDF and MLR-PLSDF is 82.3 and 79.9 respectively. The standard error of MLR-CDF and MLR-PLSDF is 26% and 28% respectively. In addition, LR is executed by using two methods which are CDF and PLSDF. The accuracy of LR-CDF and LR-PLSDF is 86.4 and 83.8 respectively. The standard error of LR-CDF and LR-PLSDF is 22% and 25% respectively. Therefore, LR-CDF outperforms on all the proposed models and state-of-the-art methods in terms of accuracy and standard error.

KEYWORDS

Defects; Software projects; Statistical model; Linear regression; Logistic regression.

INDEX

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MLR: MULTIPLE LINEAR REGRESSION;

LR: LOGISTIC REGRESSION;

CDF: CRITICAL DEFECT FACTORS;

PLSDF: PREMIER LIST OF SOFTWARE DEFECT FACTORS;

PRISMA : PREFERRED REPORTING ITEMS FOR SYSTEMATIC REVIEWS;

OLS : ORDINARY LEAST SQUARES;

SE: STANDARD ERROR;

ARS: ADJUSTED-R-SQUARED;

PV: P-VALUE;

SD: SOFTWARE DEFECTS;

OOP: OBJECT-ORIENTED PROGRAMMING;

RQ: RESEARCH QUESTION;

MCC: MATHEWS CORRELATION COEFFICIENT;

RF: RANDOM FOREST;

SVM: SUPPORT VECTOR MACHINE;

NN: NEURAL NETWORK;

RS: R-SQUARED;

1. INTRODUCTION

Software companies aim to improve the quality of software projects in terms of their accuracy and efficiency. Software companies consume from 50% to 75% of the total budget of software projects in finding and fixing defects in those projects (Koroglu,2016). In the CHAOS report, many software projects vary in size (small, medium, and large projects) and, therefore, cost. These projects use many software development methods such as waterfall and agile. Several software projects failed due to the development and testing phase, as shown in table 1. A standard software development cycle has six phases, namely, planning, analysis, design, implementation, testing, and maintenance. In the development phase, developers modify source code that may lead to many defects in a software project. In modifications, developers should be careful not to produce any new defects in these projects. The testing phase is crucial to soft-ware projects. It is responsible for delivering the final project or product efficiently to customers without any defects and in time. Many factors, such as McCabe and Halstead, help developers find and fix defects in those projects, as shown in table 2. Nevertheless, there is difficulty in using these factors in medium and large-scale pro-jects. Thus, developers need a statistical or intelligent model capable of predicting defects in software projects accurately and efficiently.

Many reasons lead to the failure of software development projects. These are the lack of experience of the project team, lack of knowledge of the code language, insufficient experience in the field, etc. Software defects in the development phase are among the most critical problems facing software companies because the many defects lead to those projects' failure. The avoidance of software defects is to gain clients' trust by providing a quality product. According to the CHAOS report, many software projects still fail because of the many reasons that have been mentioned earlier (Abdelaziz Mohamed et al., 2017). However, the direct reason for these projects' failure is the emergence of many software defects, as shown in Table 1 (Abdelaziz Mohamed et al., 2017).

Therefore, this study looks to realize two main objectives. The primary objective is to construct a statistical model to distinguish the critical defect factors that impact software projects. The second objective is to construct a statistical model to reveal defects early in software projects with sensitivity and accuracy.

We made a compressive study about the relevant related work using PRISMA methodology. The PRISMA explanation gives the minimum set of items for detailing a precise audit. It comprises the four-phase flow diagram, which permits us to utilize the Clarification and Elaboration document to go through cases and clarifications and find the meaning and method of reasoning for each item on the checklist. For a clear under-standing of PRISMA, perusing the Clarification and Elaboration document is unequivocally recommended. The PRISMA Stream Graph delineates the stream of data through the diverse stages of a Precise Audit. It maps out the number of records recognized, included, and prohibited and the reasons for avoidances.

The contribution of our study has 4 dimensions:

1. Create a bibliometric map to determine statistical or intelligent techniques that have been adopted for revealing defects in software projects.

2. Create a bibliometric map to determine performance metrics that have been adopted in the literature in the detection of software defects.

3. Build a statistical model to determine critical factors that influence on reveal defects in software projects.

4. Build a statistical model for revealing defects in software projects with reasonable accuracy.

Table 1. CHAOS Report by Agile Versus Waterfall (Abdelaziz Mohamed et al., 2017)

Size	Method	Successful	Challenged	Failed
All Size Projects	Agile (Scrum)	39%	52%	9%
	Waterfall	11%	60%	20%
Large Size Projects	Agile (Scrum)	18%	29%	53%
	Waterfall	3%	55%	42%
Medium Size Projects	Agile (Scrum)	27%	62%	11%
	Waterfall	7%	68%	25%
Small Size Projects	Agile (Scrum)	58%	38%	4%
	Waterfall	44%	45%	11%

The study is organized as follows. Section 2 presents the materials and methods, PRISMA, search questions, and search strategy. Section 3 presents the results with an analysis, and discussion, visualizing analysis and analysis per topic. Section 4 presents the methodology. Finally, in section 5, we discuss the conclusion.

2. LITERATURE REVIEW

2.1. Background

2.1.1. Software Defects:

Software Defect is a defect, error or bug in the Software which has an adverse effect on the occurrence, operation, implementation, or performance of the Software.

Many researchers such as Sirshar, M. (2019) and others (Sharma, D., & Chandra, P. (2019); Sukanya, V. S., & Saraswathy, S. (2017)) have suggested many factors to detect software defects . However, to date, there is no formal study to determine the critical factors to help software companies detect software defects with a reasonable degree of accuracy. Most researchers such as Rathore, S. S., & Kumar, S. (2015) and others also used scientific methods and models to detect software defects, but these models were weak in accuracy and results. Thus, software companies need a formal study to determine the critical factors to build a statistical model capable of detecting soft-ware defects with high results and accuracy.

Table 2. Software Metrics of McCabe and Halstead to reveal software defects (Yousef, 2015*)*

Factor ID	Factor	Description
1	Loc	McCabe's line count of code
2	v(g)	McCabe ''cyclomatic complexity''
3	eV(g)	McCabe ''essential complexity''
4	Iv(g)	McCabe ''design complexity''
5	N	Halstead total operators + operands
6	V	Halstead ''volume''
7	L	Halstead ''program length''
8	D	Halstead ''difficulty''
9	I	Halstead ''intelligence''
10	E	Halstead ''effort'': effort to write program
11	B	Halstead ''Number of Delivered Bugs''
12	T	Halstead's time estimator: time to write program
13	LOCode	Halstead's line count
14	LOComment	Halstead's count of lines of comments
15	LOBlank	Halstead's count of blank line
16	LOCodeAndComment	Halstead's count of lines which contain both code and comments
17	uniq_Op	Unique operators
18	uniq_Opnd	Unique operands
19	total_Op	Total operators
20	total_Opnd	Total operands
21	branchCount	Of the flow graph
22	defects	Module has/has not one or more reported defects

2.1.2. Regression Analysis:

This section is composed of two parts which are multiple linear regression and logistic regression, as follows:

2.1.2.1. Multiple Linear Regression:

Multiple regression analysis consists of one dependent variable and many independent variables, but it is persistent such as the reveal of a software defect, number of hours, and etc [(T. Pushpavathi, V. Suma, and V. Ramaswamy,2014),(A. Mohamed, N. Darwish and H. Hefny,2017)]. In linear regression, the dependent variable (status of software defect (True and False)) has an indefinite number of potential values. The degree of independent variables is unpretentious. It utilizes the ordinary least squares (OLS) approach to decrease errors and achieve the best potential fit. It utilizes the generic linear equation, as follows (T. Hovorushchenko and A. Krasiy,2015):

$$Y = B0 + \sum (BiXi) + \epsilon \quad (1)$$

Where

Y: dependent variable

Xi: independent variables.

B0 : intercept (the value of y when x = 0).

Bi: the slope of the line.

ϵ: terminology of the distinction that isn't explained by the model and it's called "error".

2.1.2.2. Logistic Regression:

Logistic regression is a compilation algorithm used to portend a definite variable (True or False) based on a set of separate variables or to portend the probability of an entity pertinence to one class or another class. Logistic regression algorithm uses one or more predictor variables that may be continuous or definite to portend the entity classes. This method helps to identify important factors (Xi) affecting the target variable (Y) and the quality of the relationship between each of these factors and the dependent variable [5,18].

Logistic regression offers decile such as standard error (SE), Adjusted-R-squared (ARS), and P-value (PV). SE measures the precision that represents a sample division of the population by using norm variation. ARS is used to show the impose of the logistic regression model. It calculates the rate of the impact of separate variables on the dependent variable. PV is a statistical rate that shows how each separate variable affects the dependent variable and is a number between (zero, one), a large P-value (> 0.05) indicates weak evidence against the null supposition; as an outcome, the model rejects the supposition (N. Darwish, A. Mohamed, and A. Abdelghany,2016),(T. Chow and D. Cao,2008). The main equation of logistic regression (T. Chow and D. Cao,2008). as follows:

$$g(E(y)) = \alpha + \beta x1 + \gamma x2 \quad (2)$$

Where,

g() is the link function,

E(y) is the expectation of target variable

$\alpha + \beta x1 + \gamma x2$ is the linear predictor (α,β,γ to be predicted).

2.2. MATERIALS AND METHODS

The systematic literature survey presents an evaluation of the scientific community's contributions to the topic of revealing software defects by using a rigorous and auditable methodology based on the PRISMA approach.

The PRISMA method is composed of five phases, as follows:

1. Identification of relevant manuscripts of the domain or domains.
2. Screening of titles, abstracts, papers without experiments, and position papers.
3. Eligibility analysis.
4. Full-text screening exclusion.
5. Final papers to be analyzed in detail.

We also adopted a bibliometric map; the bibliometric map is used to find the relationships between common software defects domain terms (Moral-Muñoz et al., 2020). To this end, we followed three phases, evaluating the following quantities:

1. Words frequency.
2. Most common words.
3. Frequency of these common words in the final manuscripts of the study.

By following PRISMA (Moher,2009), this section is structured in the following way: (1) our research questions, (2) followed paper search strategy, (3) bibliometric map, (4) inclusion and exclusion criteria, and (5) final paper selection.

2.2.1. Research Questions

Our study aims to provide a state-of-the-art review of current research efforts in revealing software projects. We start by introducing the reader to specific topics con-cerning research objectives and employed methods. Particularly, the survey addresses the following research questions, aiming to identify the adoption techniques that have been applied in the overall domain of revealing software defects:

RQ1: What kinds of metrics have been adopted in software defects (SD)?

RQ2: Which statistical or intelligent techniques have been adopted for SD?

RQ3: What performance metrics have been adopted in the literature in the prediction of SD?

2.2.2. Search Strategy

A literature survey generally recommends searching several available journal and conference paper repositories to determine if similar work has already been per-formed, aiding in locating potentially relevant studies. The papers counted were searched in two electronic repositories, Scopus and Web of Science. This study's covered topics were multidisciplinary, including, Software, Computer Science, Engineering, Mathematics, Environmental Science, Telecommunications, and Multidisciplinary Sciences. However, both repositories were used. The analysis showed that most of the publications from Web of Science were in Scopus as well. A repeated search process was performed to identify publications that have in their titles, abstracts, or keywords the following expressions: "software-defects" (or software defects, or defect or projects defects), and "machine learning" in Figure 1.

"(software-defects OR defect OR projects) AND (OR “data mining OR forecasting OR “machine learning” OR “neural network" OR “clustering" OR “artificial intelligence” OR “prediction” OR “predictive” OR “statistical” OR analysis”)

Fig. 1 search query for scientific manuscripts to extract the best studies in software defects

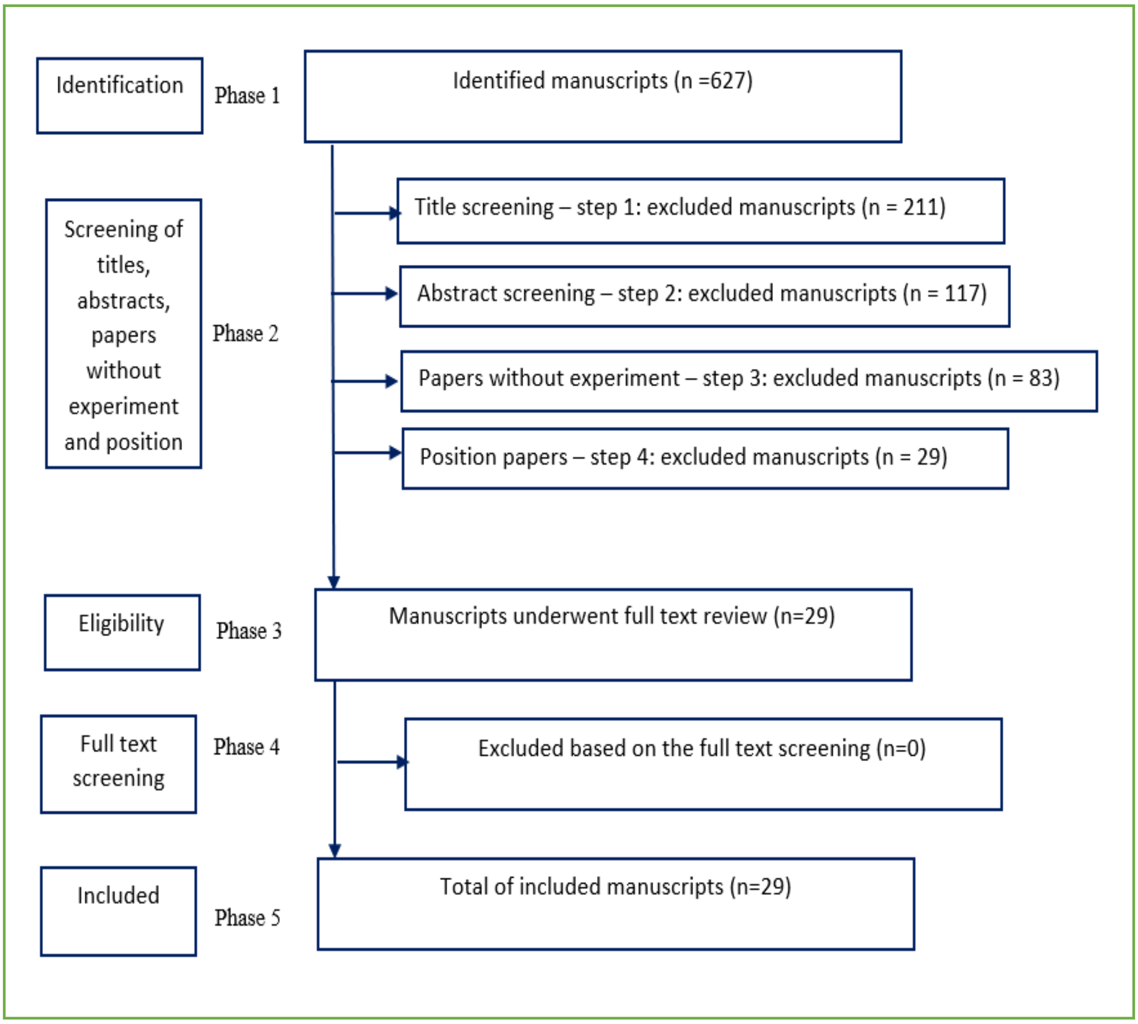

Fig. 2. scientific steps for analyzing the proposed manuscripts "PRISMA flow chart"

In phase 1, we applied the search string to all electronic repositories looking for papers published between 2015 to 2019, which resulted in 627 publications.

In Phase 2, followed a 5-step approach. In step 1, we excluded manuscripts based on titles (e.g., software defects, regression, and machine learning), which narrowed the set to 211 publications. In step 2, we excluded manuscripts based on abstracts screening, which resulted in 117 publications. In the following step 3, we excluded manuscripts reporting research without experiments, resulting in 83 publications.

Subsequently, in step 4 of phase 2, we excluded position manuscripts which gave us the final figure of 29 publications.

In phase 3, manuscripts underwent a full-text reading and review, which lead to no exclusions (the result of phase 4).

As a result of our paper selection approach, the final list included 29 manuscripts (phase 5), analyzed in detail in this paper. These were further divided into the following four categories, as shown in tables 3 and 4.

1. Regression analysis studies to reveal Software Defects.
2. Studies of Software Defects Prediction.

Table 3. Regression analysis studies to reveal Software Defects

No	Ref	Application	Dimensions	Method of Solution and Performance Metrics
1	**S.N. Umar**	Software testing defect prediction model-a practical approach	Total number of test cases executed, test team size, allocated development effort, test case execution effort, and the total number of components delivered	Multiple linear regression. R square and standard error
2	**(Dhiauddin & Ibrahim, 2012)**	A Prediction Model for System Testing Defects using Regression Analysis	Software complexity, test process, errors, the severity of the defect, and validity of defect	Multiple linear regression. Adjusted R square
3	**E. A. FELIX and et al**	Integrated Approach to Software Defect Prediction	Defect acceleration, namely, the defect density, defect velocity, and defect introduction time	Statistical analysis. Adjusted R square and correlation coefficient
4	**D. VERMA and et al**	Prediction of defect density for open source software using repository metrics	software size, number of developers, commits, and the total number of defects	Multiple linear regression. R square
5	**D. Sharma and et al**	Identification of latent variables using factor analysis and multiple linear regression for software fault prediction	Coupling between object classes, depth of inheritance tree, lack of cohesion of methods, and weighted methods per class	Multiple linear regression. R square and Adjusted R square
6	**O. Sari and et al**	Use of Logistic Regression Analysis for Bug Prediction	Weighted method count, depth of inheritance tree, lack of cohesion in methods, number of attributes, and number of methods	Logistic regression. Standard error

7	**G. MAUSA and et al**	Software Metrics as Identifiers of Defect Occurrence Severity	Software size, number of code lines, and the total number of defects.	Correlation coefficients and logistic regression. Error rate
8	**Peng H. and et al**	presented a model for predicting defects in software projects	Software size, number of code lines, and the total number of defects.	Logistic regression. Standard error
9	**M. Dhillon and et al**	An empirical model for fault prediction on the basis of regression analysis	Weighted method count, depth of inheritance tree, lack of cohesion in methods, number of attributes, and number of methods	Logistic regression. Precision, recall, and f1 measure
10	**X. Chen and et al**	Multi-Objective Effort-Aware Just-in-Time Software Defect Prediction	diffusion [Number of modified subsystems], size [line of codes], history [The number of unique changes to the modified files], and finally, experience [Developer experience].	Logistic regression. Accuracy

Table 4. Studies of Software Defects Prediction

No	Ref	Application	Dimensions	Method of Solution and Performance Metrics
1	**A. H. Yousef**	Extracting software static defect models using data mining	McCabe and Halstead metrics	Data mining techniques. Accuracy, Precision, Recall, and F1 score
2	**Karuna P and et al**	Statistical analysis of metrics for software quality improvement	Violation of programming standards, error in data representation, error in design logic, and assorted error type	Statistical analysis. Mean and standard deviation
3	**Sukanya. V and et al**	An enhanced evolutionary model for software defect prediction	McCabe and Halstead metrics	Enhanced genetic algorithm, genetic algorithm, and

				neural network. Precision
4	**Y. Koroglu and et al**	Defect prediction on a legacy industrial software: a case study on software with few defects	Product and process metrics	Data mining techniques. AUC
5	**L. KUMAR and et al**	An effective fault prediction model developed using an extreme learning machine with various kernel methods	Complexity, coupling, cohesion, and inheritance in the code	Extreme learning machine with various kernel methods (e.g., Linear kernel, Polynomial kernel, and Sigmoid kernel). Accuracy
6	**F. Zhang and et al**	Towards building a universal defect prediction model	The weighted method programming language, issue tracking, total lines of code, total number of files, the total number of commits, and the total number of developers	K-mean clustering. AUC
7	**A. Marandi and et al**	An approach of statistical methods for improving software quality	Post-delivery rework effort, actual effort, cost of the appraisal, cost of prevention, and cost of failure	Statistical analysis. Standard error
8	**G. RajBahadur and et al**	The impact of using regression models to build defect classifiers	Object-oriented metrics	Linear regression, logistic regression, random forest, support vector machine, and neural network. AUC
9	**S. Rathore and et al**	Predicting the number of faults in a software system using genetic programming	Total number of modules, number of lines of code, and number of faulty modules	Genetic programming. Recall and error rate
10	**M. Sirshar and et al**	Comparative Analysis of Software Defect Prediction	Product and process metrics	Neural Network, Naive Bayes, Deep

		Techniques		Forest technique. Error rate
11	**M. Rawat and et al**	Software defect prediction models for quality improvement: a literature study	Object-oriented code, product, and process metrics	Regression models. Accuracy
12	**S. Feng and et al**	Complexity-based Oversampling Technique to alleviate the class imbalance problem in software defect prediction	Line of code, number of children, and weighted method per class	Complexity-based Oversampling. Error rate
13	**S. Patil and et al**	Predicting software defect type using concept-based classification	Interface, syntax, and standard [build-config-install]	Concept-based Classification. F1 score
14	**J. Jiarpakdee and et al**	The impact of automated feature selection techniques on the interpretation of defect models	inconsistent and correlated	Automated Spearman correlation. Error rate
15	**A. Bangash and et al**	On the time-based conclusion stability of cross-project defect prediction models	Time, types of the projects, software development process	Mathews Correlation Coefficient. F-score
16	**S. Morasca and et al**	On the assessment of software defect prediction models via ROC curves	Lines of code and complexity	Receiver Operating Characteristic. Error rate

2.3. Results, Analysis, and Discussion

This section introduces two main parts, which are bibliometric analysis and analyzing previous works in detail. The first part shows the relationships between common terms in intelligence, statistical techniques, and performance metrics used in the previous study. The second part seeks to find the scientific gap between proposed manuscripts in this study to build a novel model to overcome the issues for revealing defects in software projects.

2.3.1. Visualizing Analysis

We used VOS viewer ("VOS viewer," n.d.), a Visualizing bibliometric network, to find common terminology in two areas: software defects and statistical techniques, across the 29 manuscripts under analysis. This tool supported our study, with visual information enabling us to explore the relations between the domains of software defects and statistical techniques. Moreover, it helped us find the most common dimensions, clustering, and variety techniques able to answer our research questions.

Figure 3 represents the visualization of a network map that displays the relations between the most popular terminology, how it is linked. The larger node represents the popular terminology in manuscripts, and the size of it represents the number of times these words appeared in manuscripts. VOS viewer splits the terminology into clusters according to the relevance concerning each other.

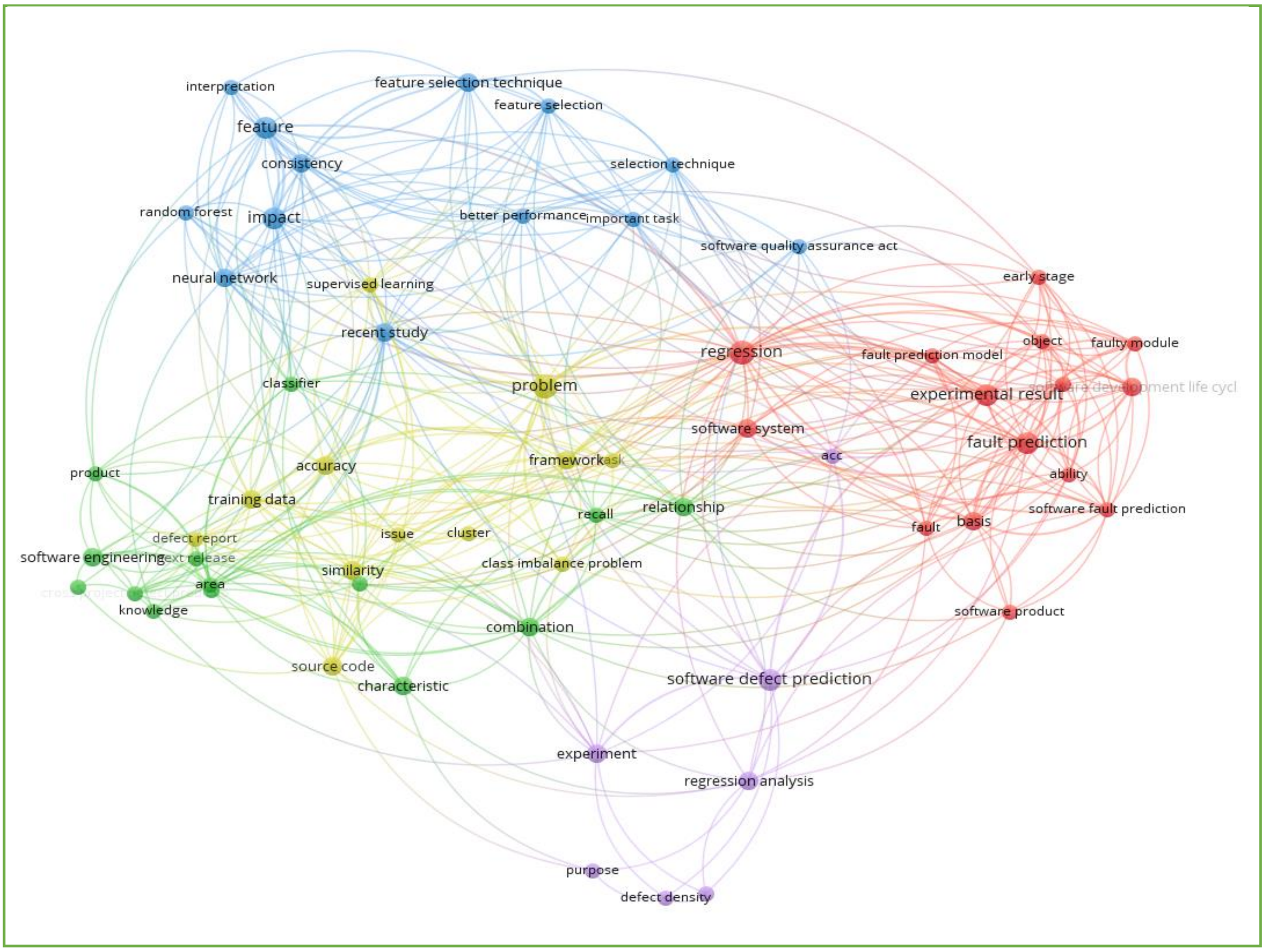

Fig. 3. The relationships between the common terms using the bibliometric map.

We performed the analysis on the title and abstract using a binary counting method of 759 examined keywords with a minimum threshold of 2 occurrences, resulting in 57 terminologies, as shown in the figure. The largest nodes representing the important nodes of each cluster in the network map are determined as" Regression" (red), "cluster" (yellow), "software engineering" (green), "neural network" (blue), and finally "software defect prediction" (purple)

Looking closer at the network map in figure1, we can see that the 5 clusters are connected between them; for instance, the "regression" term is connected to "fault prediction model" in the same red cluster, it connected to "cluster" and "accuracy" in the yellow cluster, it is also connected to "software engineering" and "recall" in the green cluster. Finally, it is also connected to "neural network" and "feature selection" in the blue cluster; it is also connected to "software defect prediction "and "defect density." Besides, the term "software engineering" in the green cluster is connected to "cluster" in the yellow cluster, "regression" in the red cluster, and "neural network" in the blue cluster. Moreover, the terms "random forest" and "feature selection" are connected to "neural network" in the blue cluster, "recall" and "software engineering" in the green cluster, "cluster" in yellow cluster, "regression" and "fault prediction model" in the red cluster and "software defect prediction" and "defect density" in the purple cluster.

Finally, by analyzing the network map in figure1, we can identify the important terms in each cluster, as follows:

- In the red cluster: "regression" and "software prediction model."
- In the yellow cluster: "cluster" and "accuracy."
- In the green cluster: "recall" and "software engineering."
- In the blue cluster: "random forest", "feature selection" and "neural network"
- In the purple cluster: "software defect prediction" and "defect density."

2.3.2. Analysis Per Topic:

RQ1 drove to look for metrics, data sources, and critical factors able to reveal software defects. Our review of papers S1 to S26 allowed us to extract such critical factors. Dimensions such as software status [No. of defects], OOP [Depth of Inheritance Tree and No. of Methods], McCabe Metrics [Line Count of Code], and Halstead Metrics [Effort to Write Program and Time to Write Program] seem to be highly considered when studying the revealing of software defects in software companies. Table 5 shows the variety of metrics used in predicting defects in software projects. The studies of S1, S4, and S16 relied on team dimension (team size and the number of developers) to predict software defects in software projects. The studies of S2, S3, S4, S7, S8, S12, S15, and S26 relied on software status dimensions (software complexity, number of defects, and software size) to detect defects in those projects. Moreover, the studies of S5, S6, S9, S15, S16, S18, and S21 relied on the OOP dimension (coupling between object classes, depth of inheritance tree, number of methods) also to reveal defects in those projects. Also, the studies of S7, S8, S10, S11, S13, S16, S19, S22, and S26 relied on McCabe metrics (line count of code, cyclomatic complexity, essential complexity, and design complexity) to find the optimal intelligent techniques to predict defects in software projects. Finally, the studies of S1, S3, S11, S13, S16, S17, S25 relied on Halstead Metrics (total operators + operands, effort to write the program, number of delivered bugs, count of lines of comments, and time to write a program) to forecast defects in various software projects. We observed that four factors are the most used in predicting defects in software projects. These are the number of defects, depth of inheritance tree, number of methods, and line count of code.

Table 5. Major factors in software defect projects

		Dimensions																	
		Team		**Software status**			**OOP**			**McCabe Metrics**				**Halstead Metrics**					**Other Factors**
		Size	No. Developers	software complexity	No. of Defects	Software Size	Coupling between Object classes	Depth of Inheritance Tree	No. of Methods	Line Count of Code	Cyclomatic Complexity	Essential Complexity	Design Complexity	Total Operators + Operands	The effort to Write Program	Number of Delivered Bugs	Count of Lines of Comments	Time to Write Program	
S.N. Umar [9] S1	Software testing defect prediction model-a practical	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓	-	-	✓
M.D. Suffian and et al S2 [10]	A Prediction Model for System Testing Defects using Regression Analysis	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
E. A. FELIX and et al. S3 [11]	Integrated Approach to Software Defect Prediction	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	✓	✓
D. VERMA and et al. S4 [12]	Prediction of defect density for open source software using repository metrics	-	✓	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-	-	-
D. Sharma and et al. S5 [13]	Identification of latent variables using factor analysis and multiple linear regression for software fault prediction	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	✓
O. Sari and et al. S6 [14]	Use of Logistic Regression Analysis for Bug Prediction	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-
G. MAUSA and et al. S7 [15]	Software Metrics as Identifiers of Defect Occurrence Severity	-	-	-	✓	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-
Peng H. and et al. s8 [16]	presented a model for predicting defects in software projects	-	-	-	✓	✓	-	-	-	✓	-	-	-	-	-	-	-	-	-

M. Dhillon and et al s9 [17]	An empirical model for fault prediction on the basis of regression analysis	-	-	-	-	-	-	✓	✓	-	-	-	-	-	-	-	-	-	✓
X. Chen and et al. s10 [17]	An empirical model for fault prediction on the basis of regression analysis	-	-	-	-	-	-	-	-	✓	-	-	-	-	-	-	-	-	✓
A. H. Yousef s11 [7]	Extracting software static defect models using data mining	-	-	-	-	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Karuna P and et al. s12 [18]	Statistical analysis of metrics for software quality improvement	-	-	-	✓	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
Sukanya.V and et al s13 [8]	An enhanced evolutionary model for software defect prediction	-	-	-	-	-	-	-	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	-
Y. Koroglu and et al s14 [19]	Defect prediction on a legacy industrial software: a case study on software with few defects	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	✓
L. KUMAR and et al. s15 [20]	An effective fault prediction model developed using an extreme learning machine with various kernel methods	-	-	✓	-	-	✓	✓	-	-	-	-	-	-	-	-	-	-	-
F. Zhang and et al. s16 [21]	Towards building a universal defect prediction model	-	✓	-	-	-	-	-	✓	✓	-	-	-	-	-	-	✓	-	-
A. Marandi and et al s17 [22]	An approach of statistical methods for improving software quality	-	-	-	-	-	-	-	-	-	-	-	-	-	✓	-	-	-	✓
G. RajBahadur and et al s18	The impact of using regression models to build defect classifiers	-	-	-	-	-	✓	✓	✓	-	-	-	-	-	-	-	-	-	-

S. Rathore and et al. s19 [24]	Predicting the number of faults in a software system using genetic programming	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-
M. Sirshar and et al. s20 [25]	Comparative Analysis of Software Defect Prediction Techniques	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
M. Rawat and et al. s21 [26]	Software defect prediction models for quality improvement: a literature study	-	-	-	-	-	√	√	√	-	-	-	-	-	-	-	-	-	√
S. Feng and et al. s22	Software defect prediction models for quality improvement: a literature study	-	-	-	-	-	-	-	-	√	-	-	-	-	-	-	-	-	√
S. Patil and et al. s23	Software defect prediction models for quality improvement: a literature study	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
J. Jiarpakdee and et al s24	Software defect prediction models for quality improvement: a literature study	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√
A. Bangash and et al. s25	Software defect prediction models for quality improvement: a literature study	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	√	√
S. Morasca and et al s26	Software defect prediction models for quality improvement: a literature study	-	-	√	-	-	-	-	-	√	-	-	-	-	-	-	-	-	-

While addressing RQ2, we examined the techniques applicable in predicting defects in software projects. With this goal, we analyzed manuscripts S1 to S26 and noticed that techniques such as multiple linear regression, logistic regression, and machine learning are the most adopted, as shown in table 6. Moreover, multiple linear regression was adopted by 23% of the analyzed manuscripts, whereas statistical analysis and data mining were the choices in 27% of manuscripts. Logistic regression accounted for 27% of the revised manuscripts. Also, machine learning techniques accounted for 19% of the revised manuscripts. Finally, the remaining 4% corresponded to the other intelligent techniques. We noticed four points.

Firstly, the studies (S1, S2, S4, S5, and S21) relied on multiple linear regression where S1 presented a model to predict defects in software projects to enhance the quality of software testing. This study seeks to find a suitable model to predict software defects to save effort, costs, and software companies' time. The results of this study show that R square and standard errors are 0.91 and 5.90%, respectively. S2 presented a model for predicting defects in software projects to improve the testing process in those projects. Besides, the adjusted R square in multiple linear regression is 90%. S4 presented a framework to predict defect density in open-source software projects. The results of this study show that the R square in multiple linear regression is 0.86. S5 presented a model to predict faults in software projects. Furthermore, the results of this study show that R square and adjusted R square are 83% and 80%, respectively. S21 presented a review study to detect defects in a software project. It also seeks to find an optimal model to detect defects efficiently to save costs and time. Also, this study confirmed that regression models have achieved high results in terms of accuracy in detecting defects of software projects.

Secondly, the studies (S6, S7, S8, S9, and S10) relied on logistic regression, where S6 presented an approach to improve the quality of software projects by detecting bugs in software projects efficiently. Also, the standard error in the proposed statistical technique is 0.24. S7 presented a study to detect defects in software projects in the early-stage to save effort, money, and time. This study also depends on statistical techniques such as correlation coefficients and logistic regression. The results show that the accuracy in logistic regression is 91.2%, and the correlation coefficient is 0.95. S8 presented a model for predicting defects in software projects. The result of this study shows that the standard error in logistic regression is 0.19. S9 presented an empirical model to predict fault in software projects. This study also depends on the binary logistic regression technique to predict defects in software projects. The results also show that the precision, recall, and f1 measures are 0.65, 0.9, and 0.79. S10 presented a study to predict software defects by using logistic regression just in time. The results of this study show that the proposed technique is better than the state-of-the-art methods in terms of accuracy. The accuracy of the proposed technique is 0.73.

Thirdly, the studies (S3, S11, S12, S14, S17, S24, S25) relied on statistical analysis and data mining techniques where S3 presented an approach to forecasting defects in software projects. It also depends on statistical regression such as multiple linear regression to predict defects in those projects. Besides, the adjusted R square in statistical regression is 98.6%, and the correlation coefficient is 0.98. S11 presented a model to extract software static defects by using data mining techniques. The results of this study show that the accuracy in Association Rules, Decision Tree, Naive Bayes, and Neural Network is 77.2%, 76.6%, 73.2%, and 73.2%, respectively. Thus, Association Rules is better than Decision Tree, Naive Bayes, and Neural Network in terms of accuracy. S12 presented a study to improve the quality of software projects using statistical analysis. The results of this study were evaluated in terms of projection of errors (total errors) and cumulative projection of severity errors (e.g., series, moderate and minor). It also shows that total errors in 2016 are more than in 2015 by 1.5%.

Moreover, most severity errors are minor types. S14 presented a study to predict defects in legacy industrial software using data mining techniques. The results of this study show that the area under the curve (AUC) in Random Forest, Logistic Regression, Decision Tree, Naive Bayes, and a combination of Random Forest + Logistic regression is 0.73, 0.72, 0.66, 0.67, and 0.75. Thus, a combination of Random Forest + Logistic regression is better than Random Forest, Logistic Regression, Decision Tree, Naive Bayes. S17 presented an approach to improve software quality and cost minimization using statistical analysis. The results of this study were evaluated in terms of standard error. The standard error in the statistical model is 0.13. S24 presented a study to evaluate the impact of automated feature selection techniques on the interpretation of defect models. This study investigated 12 automated feature selection techniques in terms of consistency, correlation, performance, computational cost. By analyzing 14 publicly-available defect datasets, the results showed that the most important inconsistent metrics are highly correlated with the automated Spearman correlation of 0.85–1. S25 presented a study to predict defects in software models. This study applied the Mathews Correlation Coefficient-MCC to avoid defects in software models. MCC in F-score is less than 0.01. Therefore, the proposed technique is better than the state-of-the-art methods in terms of MCC.

Fourthly, the studies (S13, S15, S16, S18, S20) relied on machine learning techniques where S13 presented a model to predict software defects by using an enhanced genetic algorithm. The results of this study were evaluated in terms of precision. It also confirmed that precision in enhanced genetic algorithm, genetic algorithm, and neural network is 0.93, 0.81, and 0.80, respectively. Thus, the enhanced genetic algorithm is better than the genetic algorithm and neural network. S15 presented a model to predict effective faults in software projects using extreme learning machines with various kernel methods (e.g., Linear kernel, Polynomial kernel, and Sigmoid kernel). The results of this study were evaluated in terms of accuracy metrics. The accuracy in the linear kernel, Polynomial kernel, and Sigmoid kernel is 0.88, 0.93, and 0.91. Thus, an extreme learning machine using the Polynomial kernel is better than linear kernel and Sigmoid kernel. S16 presented a model to predict universal defects in software projects using clustering techniques. The results of this study were evaluated in terms of AUC. The AUC in K-mean clustering is 0.76. S18 presented a model to detect defects in a software project. This study depends on object-oriented metrics. It also relies on many intelligent techniques such as linear regression (LR), logistic regression (LG), random forest (RF), support vector machine (SVM), and neural network (NN). The results of this study were evaluated in terms of AUC. The AUC in LR, LG, RF, SVM and NN is 0.86, 0.94, 0.91, 0.90 and 0.90. Thus, LG is better than LR, RF, SVM, and NN. S20 presented a review analysis to predict defects in a software project. This study depends on many metrics, such as product and process metrics. It also introduced a comparative analysis between Neural Network, Naive Bayes, Deep Forest technique. This study relies on previous works in the analysis of these techniques. Besides, this study confirmed that Deep Forest is better than Neural Network, Naive Bayes in terms of error rate.

Fifthly, the studies (S19, S22, S23, and S26) relied on other intelligent and statistical techniques where S19 presented an approach to predict many faults in a software system by using a genetic algorithm. The results of this study were evaluated in terms of error rate and recall. The error rate and recall in the genetic algorithm are 0.11, 0.91, respectively. S22 presented a new technique in software defect prediction by Complexity-based Oversampling. This paper relied on three main factors: a line of code, number of children, and weighted method per class. By analyzing the results, the proposed technique is better than the other oversampling techniques under the statistical Wilcoxon rank-sum test and Cliff's effect size. S23 presented a framework to predict software defect type using concept-based classification. This paper's main objective is to minimize the labeled training data's dependence for automation of the software defect type classification task. The results show that the proposed framework outperforms the state-of-the-art semi-supervised [LeDEx] in terms of the F1 score. F1 score in the proposed framework and LeDEx is 63.16% and 62.30%, respectively. S26 presented a study to assess the software prediction model by using Receiver Operating Characteristic. The results showed that the proposed technique is better than all other state-of-the-art methods in terms of recall and accuracy by 0.4 and 0.8, respectively.

Table 6. Intelligent and statistical techniques in software defect project

NO	Multiple Linear Regression	Logistic Regression	Statistical Analysis	Data Mining	Machine Learning	Other
S1	√	-	-	-	-	-
S2	√	-	-	-	-	-
S3	-	-	√	-	-	-
S4	√	-	-	-	-	-
S5	√	-	-	-	-	-
S6	-	√	-	-	-	-
S7	-	√	-	-	-	
S8	-	√	-	-	-	-
S9	-	√	-	-	-	-
S10	-	√	-	-	-	-
S11	-	-	-	√	-	-
S12	-	-	√	-	-	-
S13	-	-	-	-	√	
S14	-	-	-	√	-	-
S15	-	-	-	-	√	
S16	-	-	-	-	√	-
S17	-	-	√	-	-	-
S18	√	√	-	-	√	-
S19	-	-	-	-	-	
S20	-	-	-	-	√	-
S21	√	√	-	-	-	-
S22	-	-	-	-	-	
S23	-	-	-	-	-	
S24	-	-	√	-	-	-
S25	-	-	√	-	-	-
S26	-	-	-	-	-	

The literature study also analyzed the performance evaluation metrics in the scope of our RQ3. Results are shown in table 7 and table 8. 21% of the selected manuscripts (S10,11,15, 9, 13, and 21) adopted accuracy and precision. 21% of them (S9, 11, 19, 23, and 25) selected only recall and F1 score. The error rate was used by 30% of the analyzed manuscripts (S1, 6, 7, 8, 17, 19, 20, 22, 24, and 26). 15% of the manuscripts adopted the R Square measure (S1, 2, 3, 4, and 5). We also realized that 13% (S12 S14, S16, and S18) did not use any defined evaluation metric.

Table. 7. Sample of performance metrics rate in previous work

	Performance Metrics	**Rate**
1	Accuracy and precision	21%
2	Recall and F1 Score	21%
3	Error Rate	30%
4	R Square Measure	15%
5	Other	13%

Table 8. Majority of performance metrics used in software defect projects

NO	**Accuracy**	**Precision**	**Recall**	**F1 score**	**Error Rate**	**R-Square**	**Other**
S1	-	-	-	-	√	√	-
S2	-	-	-	-	-	√	-
S3	-	-	-	-	-	√	-
S4	-	-	-	-	-	√	-
S5	-	-	-	-	-	√	-
S6	-	-	-	-	√	-	-
S7	-	-	-	-	√	-	-
S8	-	-	-	-	√	-	-
S9	-	√	√	√	-	-	-
S10	√	-	-	-	-	-	-
S11	√	√	√	√	-	-	-
S12	-	-	-	-	-	-	√

S13	-	✓	-	-	-	-	-
S14	-	-	-	-	-	-	✓
S15	✓	-	-	-	-	-	
S16	-	-	-	-	-	-	✓
S17	-	-	-	-	✓	-	
S18	-	-	-	-	-	-	✓
S19	-	-	✓	-	✓	-	-
S20	-	-	-	-	✓	-	-
S21	✓	-	-	-	-	-	-
S22	-	-	-	-	✓	-	-
S23	-	-	-	✓	-	-	-
S24	-	-	-	-	✓	-	-
S25	-	-	-	✓	-	-	-
S26	-	-	-	-	✓	-	-

Our research helped us to determine several research gaps. We only identified a few manuscripts (S11 and S13) tackling specific metrics impacting defects in software projects. For example, some studies (S5, S6, S9, S18, and S21) are concentrated on the OOP metric in general, with no mention of the line count of code and the number of developers. There are only simple manuscripts (S14, S20, S23, and S24) regarding finding defects in all types of software projects (small, medium, and large projects). However, stakeholders in software companies seem to find this topic pertinent and are willing not only to enhance software efficiency in those projects but interested to predict early defects in software projects to save costs and money. The results of this survey also showed a significant gap in the field of "intelligent and statistical models," particularly relating to the automatic prediction of defects in software projects. Some of the most promising algorithms are not yet being utilized. Only a few studies (S18 and S21) tackle the application of "hybrid statistical and intelligent techniques, for instance, logistic regression with multiple linear regression and regression analysis with deep learning," which is a promising technique for forecasting defects in software projects. Moreover, there is a lack of official studies to identify critical factors that influence defects in software projects.

3. METHODOLOGY

Proposal of a new proposed model based on a statistical model able to predict defects in software projects. This section presents an approach for a statistical model able to predict defects in software projects. The proposed model has been used in several scientific data science researches like is the case of (Yousef, A. H. ,2015). As shown in Figure 4, the detailed the proposed model will cover the following phases:

1. State-of-the-art analysis: Review the literature to extract important metrics, data sources, mathematical and computational approaches used for predicting defects of software projects.

2. Data collection: data is collected from the NASA data sets online. We have two reasons to select the NASA Data set. The first reason is it is too hard to collect huge data from software companies to reveal the defects in software projects. The second reason for selecting Nasa is based on its vast and high-quality data. It explains the static measures and other variables that are used to detect static defects in software projects. It also shows a binary variable indicating whether the module is defective or not.

3. Data Analysis and Pre-Processing: Analyze the data in detail and, if necessary, transform it to expose its information content better. Different mathematical techniques may be used, namely, outlier removal, discretization, reduction of the number of variables, and/or dimensionality (adopting regression models).

4. Feature selection: determine critical metrics and detect defects that will be adopted in the proposed IST study by using logistic regression and multiple linear regression. Create a mapping between logistic regression and multiple linear regression to determine the final list of critical metrics capable of predicting defects in software projects.

5. Build a model: present a statistical model capable of predicting defects in software projects using multiple linear regression and logistic regression.

6. Training and verification model: train the model with data set and verify its ability to predict defects in software projects.

7. Also, we will present a comparison between logistic regression and multiple linear regression by using the final list of critical metrics to determine which one is better than the other in terms of accuracy, precision, recall, F1 measure, and error rate.

Following this holistic approach, we built a methodology composed of five phases, as shown in figure 4.

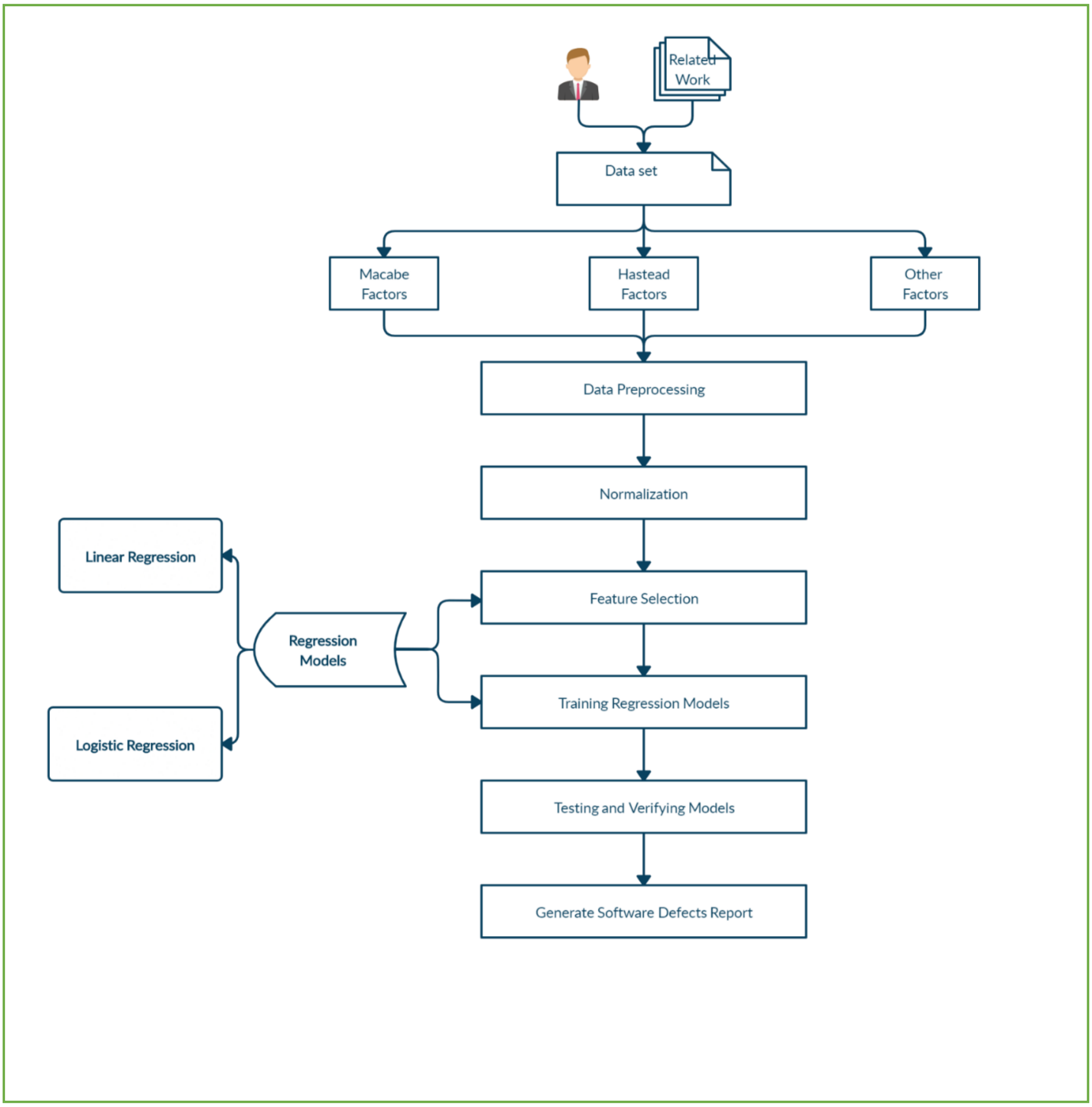

Fig 4. A Proposed Statistical Model for Software Defects Prediction

4. RESULTS AND DISCUSSION

4.1. MULTIPLE LINEAR REGRESSION:

In our trial to detect which features can impact positively the defeat of software projects. This trial is used multiple Linear regression analysis, where the relationship between multiple independent variables (factors in software projects) and the dependent variable (grade of impacting the factors in software projects) is specific. Based on the Eq. (3), the multiple Linear regression analysis can be specified as follows:

$$\mathbf{Y = \beta 0 + \beta 1x1 + \beta 2x2 + \ldots. + \beta 11x11 + \varepsilon \ (3)}$$

Where:

Y: is degree of effect the defeat factors in software projects

$\beta 0$: is the y-intercept

βi: is the regression coefficient

Xi: critical failure factors

ε: the random error term

The applied model has been executed by a set of steps. To begin with, the proposed data is split into dependent and independent variables (refer to Table 2). Furthermore, it consists of seventy percent training and thirty percent testing data. The ordinary least squares (OLS) is used to verify the proposed model that assumes a robust linear relationship between the dependent and the independent variables. After that, L2 Regularization is used to upgrade the quality of the proposed model.

This part introduces a flow chart for determining the critical factors that impact software projects by MLR and LR analysis. Fig. 5 shows the flow chart for the proposed model. MLR and LR analysis introduced regression statistics like standard error (SE), R-squared (RS), adjusted R-squared (ARS), and P-value (PV). SE shows a first handle on how fully the provided equation is suitable for the sample data. It is critical to the units of the stander of the dependent variable. RS shows the explanatory impose of the regression model. ARS is an updated version of RS that has been modified for the set of predictors in the model. It raises only if the relative terms promote the model more than would be expected. It is minimized when a predictor promotes the model by less than expected. It is always lower than the RS.ARS of MLR and LR is 0.78. PV helps to set the importance of the statistical results. It is a number between 0 and 1 and is explained in the following method: a small PV (typically ≤ 0.05) indicates a powerful proof versus the null supposition. The null supposition is rejected from the statistical results.

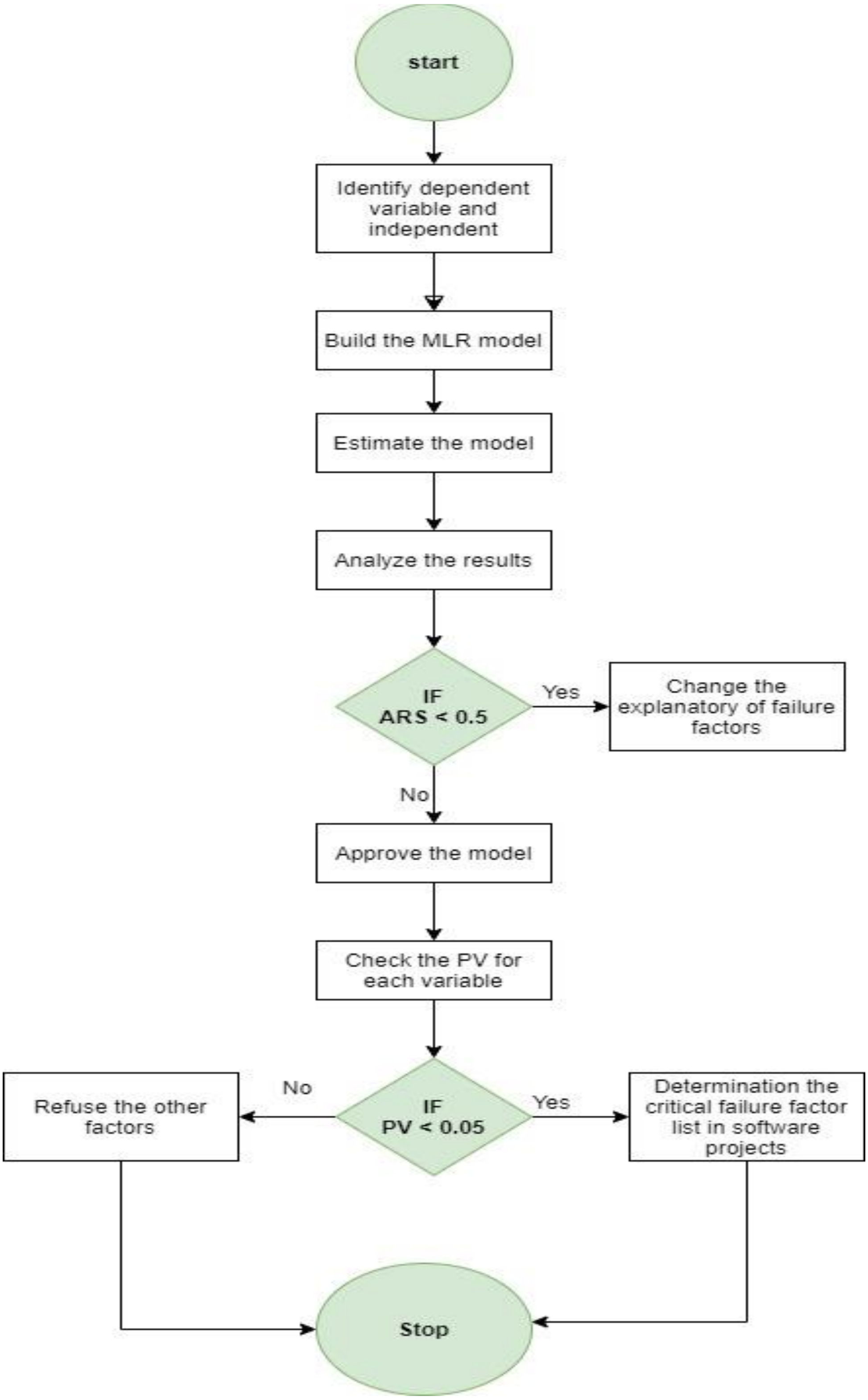

Fig.5 Flow chart of the proposed algorithm for MLR and LR model

Table 9. Summary of critical factors that affect software projects in MLR

No	Factor ID		P-Value
1	loc	✓	**0**
2	v(g)	✓	**0**
3	ev(g)	**X**	0.4341
4	iv	**X**	0.0537
5	n	✓	**0**
6	v	**X**	0.4804
7	l	**X**	0.2107
8	d	✓	**0.0002**
9	i	✓	**0.0075**
10	e	**X**	0.9454
11	b	**X**	0.7833
12	t	**X**	0.9454
13	IOCode	✓	**0**
14	IOComment	✓	**0.0461**
15	IOBlank	**X**	0.0809
16	locCodeAndComment	**X**	0.0667
17	Column1op	✓	**0**
18	Column2opnd	**X**	0.2169
19	Column1totalopnd	✓	**0.0001**
20	Column1totalop	✓	**0.0003**
21	Column1branch	✓	**0**

The model based on critical defect factors (CDF) based on the model-based premier list of software defect factors (PLSDF) to the accuracy and standard error ratio, as shown below in Figs. 6 and 7.

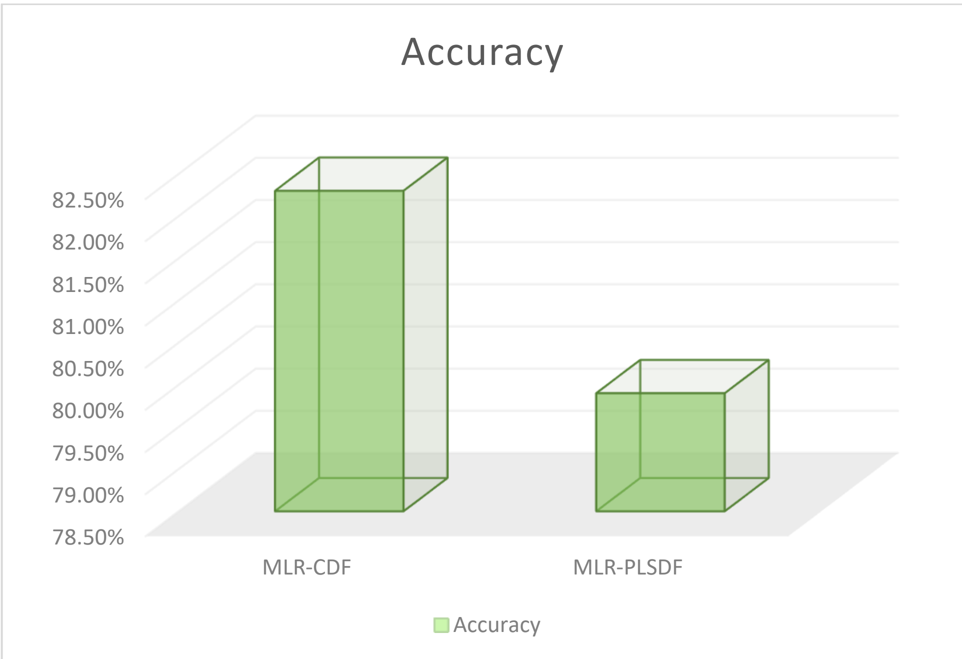

Fig 6. The comparison between model of MLR-CDF and model of MLR-PLSDF to accuracy

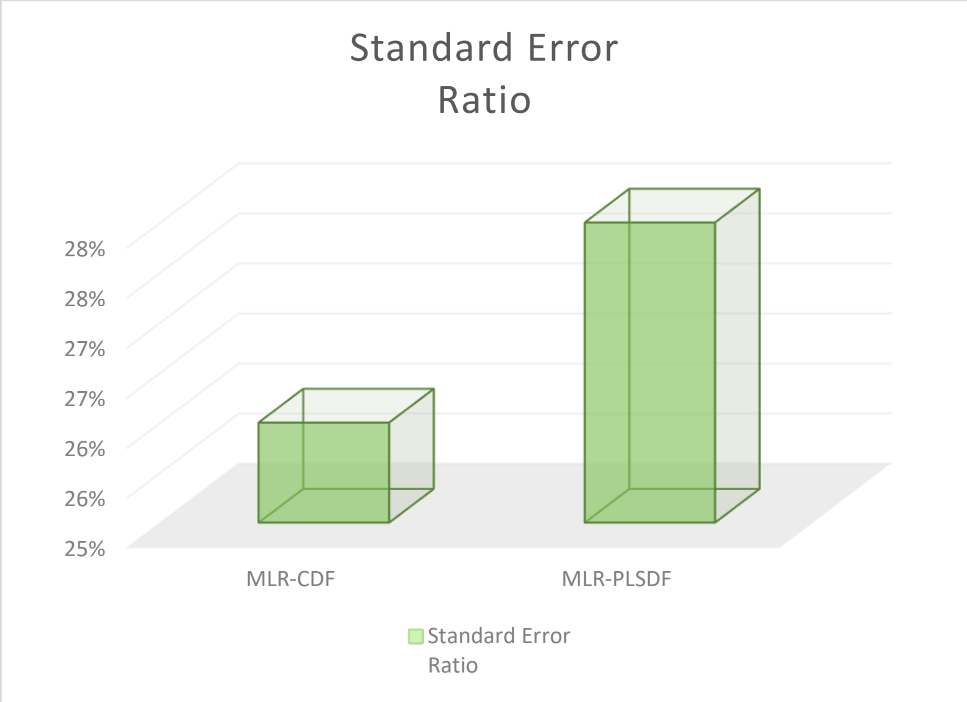

Fig 7. The comparison between model of MLR-CDF and model of MLR-PLSDF to stander error ratio

4.2. Logistic Regression:

In our trial to show the experimental results of our proposed approach. The approach is executed using a different technique, which is logistic regression. A group of pre-processing steps does the proposed model. First, the dataset attributes are split into defect factors in software projects as separate variables and the degree of effecting defect factors in software projects which will be as the dependent variable. Second, the dataset also is split into 80% training data and also 20% testing data. Third, the dependent variable was changed from categorical values (False, True) to binary values (0:1). Fourth, the independent variables were run between 0 and 1. assume that Xmin and Xmax are the minima and maximum values of an attribute X, as shown in Eq. (5).

$$X_{new} = \frac{X - X_{min}}{X_{max} X_{min}} \qquad \text{Eq (5)}$$

We used logistic regression to define critical defect factors that impact software projects. Two methods do it. The first method relies on the critical defect factors (LR-CDF) of software projects. The second model relies on the premier list of software defect factors (LR-PLSDF). In the PLSDF method, the relationship between independent variables (premier list of software defect factors) and the dependent variable (degree of effecting defect factors in software projects) is fixed, as shown in Eq. (6). The logistic regression of PLSDF can be identified as follows:

$$P = \frac{1}{1+e^{-(\beta 0 + \beta 1x1 + \beta 2x2 + \ldots + \beta 13x13)}} \qquad \text{Eq (6)}$$

Where:

- P = degree of effecting defect factors in software projects
- β0 = P-intercept
- βi = regression coefficient
- Xi = LR-CDF

Table 10 present the statistical results of the PLSDF method. It includes two significant results (Adjusted R squared and P-value). ARS is - 0.82. The PV shows the significant separate features that affect detect defects in software projects. If P-value>0.05, the degree is not significant statistically. For example, the P-value for (v(g)=0.5970) is greater than 0.05; thus, this feature should refuse. therefore, those features with a value level (P-Value<0.05) would be known as elects for being important features affecting defect factors selection.

Table 10. Summary of critical factors that affect software projects in LR

No	Factor ID		P-Value
1	loc	✓	0.0000
2	v(g)	X	0.5970
3	ev(g)	✓	0.0267
4	iv	✓	0.0447
5	n	X	0.1740
6	v	X	0.6973
7	l	✓	0.0003
8	d	✓	0.0072
9	i	✓	0.0084
10	e	X	0.9994
11	b	X	0.7338
12	t	X	0.9995
13	IOCode	✓	0.0001
14	IOComment	✓	0.0047
15	IOBlank	✓	0.0111
16	locCodeAndComment	X	0.0747
17	Column1op	✓	0.0032
18	Column2opnd	✓	0.0000
19	Column1totalopnd	✓	0.0016
20	Column1totalop	✓	0.0437
21	Column1branch	X	0.1599

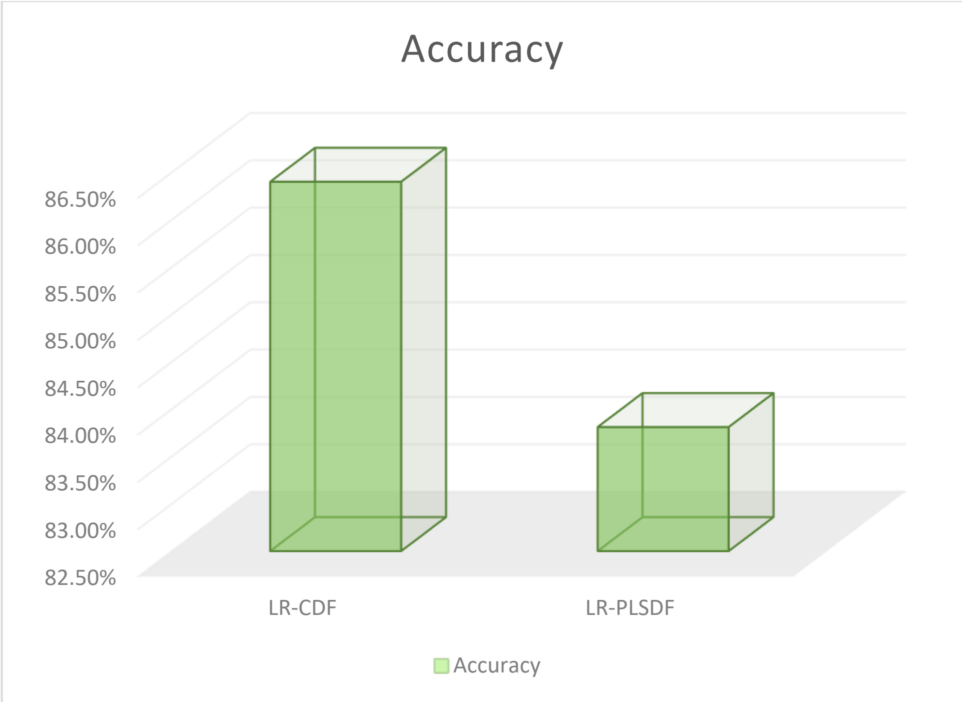

Fig 8. The comparison between model of LR-CDF and model of LR-PLSDF to accuracy

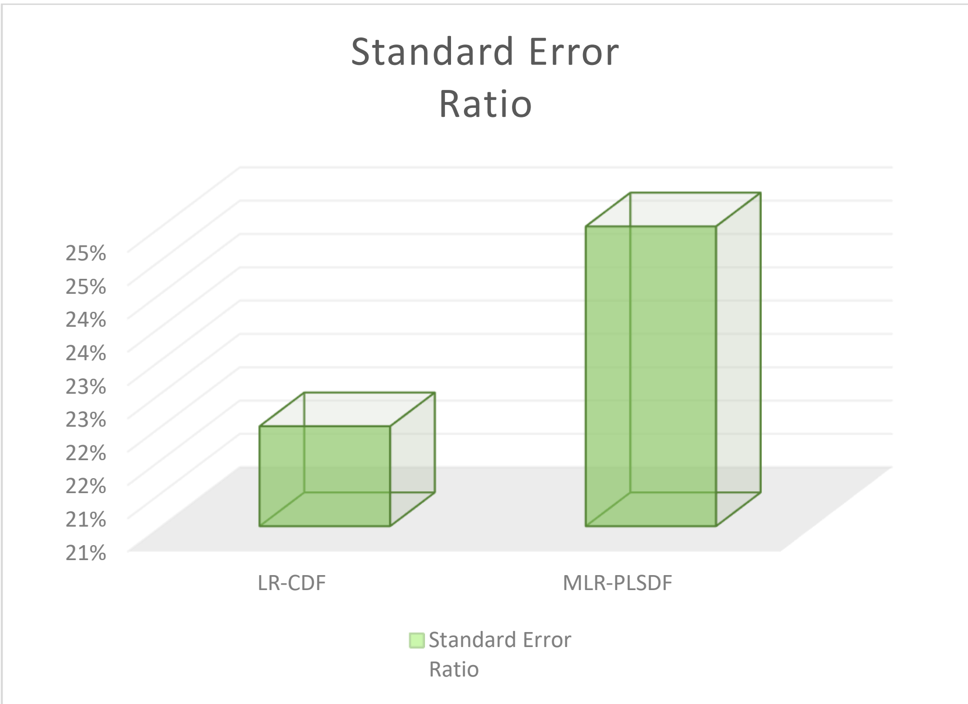

Fig 9. The comparison between model of LR-CDF and model of LR-PLSDF to stander error ratio

In this part we do comparison between the accuracy and stander error ratio in multilinear regression and logistic regression to extract the best result, as show in figure 10,11, which shows that the best result extracts from (LR-CDF).

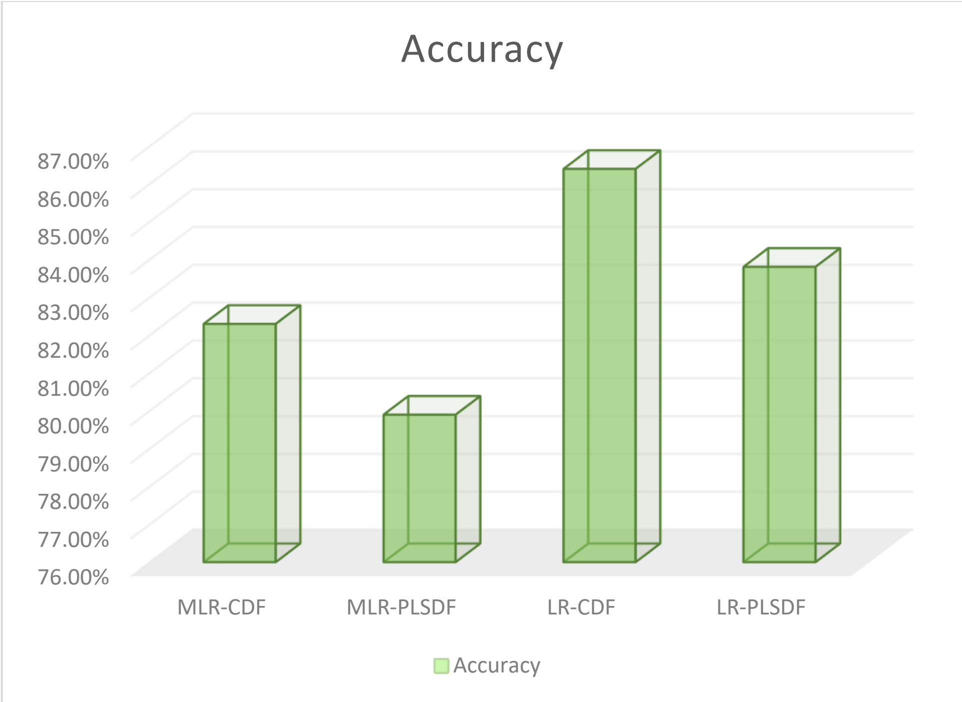

Fig.10 The accuracy comparison of all proposed statistical model

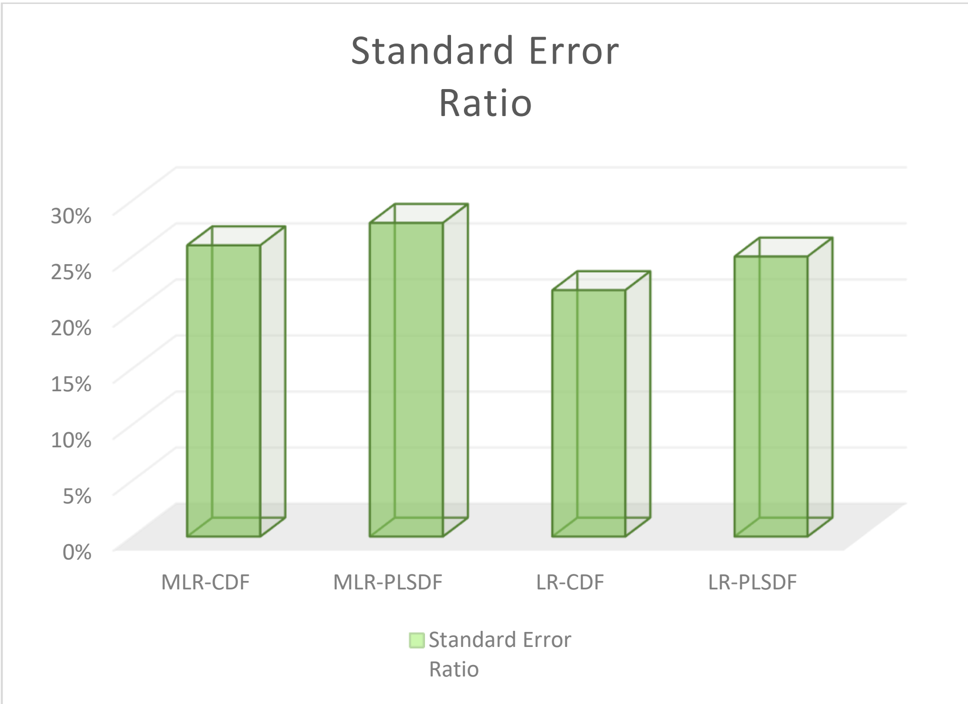

Fig.11 The standard error ratio comparison of all proposed statistical model

The LR-CDF model outperforms the state-of-the-art methods in previous works of accuracy, as shown in figure 12.

Fig12. The comparison between the proposed model and state of the art methods

The LR-CDF model outperforms the intelligent techniques (Association rule, Decision tree, Naive Bayes, and neural network) in Ahmed H. Yousef's study in terms of accuracy by 9.1%, 10.3%, 13.1%, and 13.1% respectively.

5. CONCLUSIONS

5.1. MAIN CONTRIBUTION

This paper presented a systematic review on the topic of revealing defects in software projects, concentrating on finding replies to our research questions, a diplomatic map was used to find the most used terminology in the statistical technique's software projects domains. By following a Prisma approach in our systematic review, we started by determining 627 papers and ended with VP analyses of 26 papers. The research questions covered three major points. Firstly, we identified the factors of our metrics that influence revealing defects in software projects. Secondly, we concentrated our research on identifying the production techniques used in the context. After, we determined the evaluation criteria used by those techniques. Thus, there is still a chance for enhancement regarding our topic to use statistical and intelligent techniques to reveal defects in software projects.

Finally, a new methodology based on a statistical model able to predict defects in software projects was proposed.

This study succeeded in identifying the critical factors that affect the detection of defects in the programs. Statistical analysis is executed by four methods, which are MLR-CDF, MLR-PLSDF, LR-CDF, and LR-PLSDF. LR-CDF outperforms on all the proposed methods in order to accuracy and standard error. In addition, LR-CDF outperforms on state-of-the-art methods (Association rule, Decision tree, Naive Bayes, and neural network) related to the accuracy by 9.1%, 10.3%, 13.1%, and 13.1% respectively.

5.2. LIMITATIONS TO THE CURRENT WORK

The study has some limitations. it was restricted by the search keywords selected and the time of the manuscripts (last six years). In addition, it utilized a fixed number of electronic sources. Furthermore, this study only handled English scientific papers, and we cannot warranty to have picked all the worthy substance for our review.

As mentioned, the study has not enveloped all scientific papers in 2021, which may include novel intelligent techniques. The emergence of novel intelligent techniques may assist in enhancing the accuracy of revealing defects in different software projects.

5.3. FUTURE WORK

It is recommended as future work to utilize other techniques in terms of improving the model accuracy and identifying critical factors for revealing defects in software projects.

This study proposes processing the revealed defects of software projects by integrating optimization techniques and deep learning techniques such as long short-term memory, convolutional neural networks, and deep forest, which are some of the recent trends found in research aiming to improve the accuracy of the proposed model and state-of-the-art method in previous works.

Bibliography

S. N. U. (2013). Software Testing Defect Prediction Model - a Practical Approach. International Journal of Research in Engineering and Technology, 02(05), 741–745. https://doi.org/10.15623/ijret.2013.0205001

Bangash, A. A., Sahar, H., Hindle, A., & Ali, K. (2020). On the time-based conclusion stability of cross-project defect prediction models. Empirical Software Engineering. https://doi.org/10.1007/s10664-020-09878-9

Chen, X., Zhao, Y., Wang, Q., & Yuan, Z. (2018). MULTI: Multi-objective effort-aware just-in-time software defect prediction. Information and Software Technology, 93, 1–13. https://doi.org/10.1016/j.infsof.2017.08.004

Dhiauddin, M., Suffian, M., & Ibrahim, S. (2012). A Prediction Model for System Testing Defects using Regression Analysis. International Journal of Soft Computing And Software Engineering, 2(7), 2251–7545. https://doi.org/10.7321/jscse.v2.n7.6

Dhillon, M. K., Singh, P. B., & Singh, P. J. (2016). Empirical Model for Fault Prediction On the Basis of Regression Analysis. International Journal of Science and Research (IJSR), 5(6), 163–168. https://doi.org/10.21275/v5i6.nov164139

Felix, E. A., & Lee, S. P. (2017). Integrated Approach to Software Defect Prediction. IEEE Access, 5, 21524–21547. https://doi.org/10.1109/ACCESS.2017.2759180

Feng, S., Keung, J., Yu, X., Xiao, Y., Bennin, K. E., Kabir, M. A., & Zhang, M. (2020). COSTE: Complexity-based OverSampling TEchnique to alleviate the class imbalance problem in software defect prediction. Information and Software Technology, 129(September 2020), 106432. https://doi.org/10.1016/j.infsof.2020.106432

He, P., He, Y., Yu, L., & Li, B. (2018). An Improved Method for Cross-Project Defect Prediction by Simplifying Training Data. Mathematical Problems in Engineering, 2018. https://doi.org/10.1155/2018/2650415

Jiarpakdee, J., Tantithamthavorn, C., & Treude, C. (2020). The impact of automated feature selection techniques on the interpretation of defect models. Empirical Software Engineering, 25(5), 3590–3638. https://doi.org/10.1007/s10664-020-09848-1

T. Pushpavathi, V. Suma, and V. Ramaswamy, "Defect Prediction in Software Projects-Using Genetic Algorithm based Fuzzy C-Means Clustering and Random Forest Classifier", International Journal of Scientific & Engineering Research, Vol. 5, pp. 888-898, 2014.

A. Mohamed, N. Darwish and H. Hefny, "Towards a Machine Learning Model for Predicting Failure of Agile Software Projects", International Journal of Computer Applications, Vol. 168, No. 6, pp. 20-26, 2017

T. Hovorushchenko and A. Krasiy, "Realization of the Neural Network Model of Prediction of the Software Project Characteristics for Evaluating the Success of its Implementation", International Conference on Intelligent Data Acquisition And Advanced Computing Systems, pp. 348-353, 2015.

N. Darwish, A. Mohamed, and A. Abdelghany, "A Hybrid Machine Learning Model for Selecting Suitable Requirements Elicitation Techniques", International Journal of Computer Science and Information Security, Vol. 14, No. 6, pp. 1-12, 2016.

T. Chow and D. Cao, "A survey study of critical success factors in agile software projects", Journal of System and Software, Vol.81, pp. 961-971, 2008.

Kondo, M., Bezemer, C. P., Kamei, Y., Hassan, A. E., & Mizuno, O. (2019). The impact of feature reduction techniques on defect prediction models. In Empirical Software Engineering (Vol. 24, Issue 4). Empirical Software Engineering. https://doi.org/10.1007/s10664-018-9679-5

Koroglu, Y., Sen, A., Kutluay, D., Bayraktar, A., Tosun, Y., Cinar, M., & Kaya, H. (2016). Defect prediction on a legacy industrial software: A case study on software with few defects. Proceedings - International Conference on Software Engineering, 17-May-201, 14–20. https://doi.org/10.1145/2896839.2896843

Kumar, L., Tirkey, A., & Rath, S. K. (2018). An effective fault prediction model developed using an extreme learning machine with various kernel methods. Frontiers of Information Technology and Electronic Engineering, 19(7), 864–888. https://doi.org/10.1631/FITEE.1601501

Marandi, A. K., & Khan, D. A. (2017). An approach of statistical methods for improve software quality and cost minimization. International Journal of Applied Engineering Research, 12(6), 1054–1061.

Mauša, G., Grbac, T. G., Brezočnik, L., Podgorelec, V., & Heričko, M. (2019). Software metrics as identifiers of defect occurrence severity. CEUR Workshop Proceedings, 2508(September), 22–25.

Morasca, S., & Lavazza, L. (2020). On the assessment of software defect prediction models via ROC curves. Empirical Software Engineering, 25(5), 3977–4019. https://doi.org/10.1007/s10664-020-09861-4

Patil, S., & Ravindran, B. (2020). Predicting software defect type using concept-based classification. Empirical Software Engineering, 25(2), 1341–1378. https://doi.org/10.1007/s10664-019-09779-6

Rajbahadur, G. K., Wang, S., Kamei, Y., & Hassan, A. E. (2017). The impact of using regression models to build defect classifiers. IEEE International Working Conference on Mining Software Repositories, 135–145. https://doi.org/10.1109/MSR.2017.4

Rathore, S. S., & Kumar, S. (2015). Predicting number of faults in software system using genetic programming. Procedia Computer Science, 62(Scse), 303–311. https://doi.org/10.1016/j.procs.2015.08.454

Rawat, M. S., & Dubey, S. K. (2012). Software Defect Prediction Models for Quality Improvement: A Literature Study. International Journal of Computer Science Issues, 9(5), 288–296.

Sari, O., & Kalipsiz, O. (2015). Bug prediction for an ATM monitoring software use of logistic regression analysis for bug prediction. ICEIS 2015 - 17th International Conference on Enterprise Information Systems, Proceedings, 2, 382–387. https://doi.org/10.5220/0005382803820387

Sharma, D., & Chandra, P. (2019). Identification of latent variables using, factor analysis and multiple linear regression for software fault prediction. International Journal of Systems Assurance Engineering and Management, 10(6), 1453–1473. https://doi.org/10.1007/s13198-019-00896-5

Sirshar, M. (2019). Comparative Analysis of Software Defect Prediction Techniques. December, 456頁、453頁、603頁.

Sukanya, V. S., & Saraswathy, S. (2017). An Enhanced Evolutionary Model for Software Defect Prediction. 7(10), 15323–15328.

Verma, D., & Kumar, S. (2017). Prediction of defect density for open source software using repository metrics. Journal of Web Engineering, 16(3–4), 294–311.

Yousef, A. H. (2015). Extracting software static defect models using data mining. Ain Shams Engineering Journal, 6(1), 133–144. https://doi.org/10.1016/j.asej.2014.09.007

Zhang, F., Mockus, A., Keivanloo, I., & Zou, Y. (2016). Towards building a universal defect prediction model with rank transformed predictors. Empirical Software Engineering, 21(5), 2107–2145. https://doi.org/10.1007/s10664-015-9396-2

Abdelaziz Mohamed, A., Ramadan Darwish, N., & Ahmed Hefny, H. (2017). Towards a Machine Learning Model for Predicting Failure of Agile Software Projects. International Journal of Computer Applications, 168(6), 975–8887.

Moral-Muñoz, J. A., Herrera-Viedma, E., Santisteban-Espejo, A., & Cobo, M. J. (2020). Software tools for conducting bibliometric analysis in science: An up-to-date review. Profesional de La Informacion, 29(1), 1–20. https://doi.org/10.3145/epi.2020.ene.03

Moher, D., Liberati, A., Tetzlaff, J., Altman, D. G., & Group, T. P. (2009). Preferred Reporting Items for Systematic Reviews and Meta-Analyses : The PRISMA Statement. 6(7). https://doi.org/10.1371/journal.pmed.1000097

Jiarpakdee, J., Tantithamthavorn, C., & Hassan, A. E. (2021). The Impact of Correlated Metrics on the Interpretation of Defect Models. IEEE Transactions on Software Engineering, 47(2), 320–331. https://doi.org/10.1109/TSE.2019.2891758

Li, N., Shepperd, M., & Guo, Y. (2020). A systematic review of unsupervised learning techniques for software defect prediction. Information and Software Technology, 122(February), 106287. https://doi.org/10.1016/j.infsof.2020.106287

Maddeh, M., Ayouni, S., Alyahya, S., & Hajjej, F. (2021). Decision tree-based Design Defects Detection. IEEE Access, 9, 71606–71614. https://doi.org/10.1109/ACCESS.2021.3078724