IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 4 October 2023, accepted 29 November 2023, date of publication 5 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3339994

RESEARCH ARTICLE

Early Software Defects Density Prediction: Training the International Software Benchmarking Cross Projects Data Using Supervised Learning

TOUSEEF TAHIR[1], CIGDEM GENCEL[2], GHULAM RASOOL[1], TARIQ UMER[1], (Senior Member, IEEE), JAWAD RASHEED[3,4,5], (Member, IEEE), SOOK FERN YEO[6,7], AND TANER CEVIK[8]

[1]Department of Computer Science, COMSATS University Islamabad, Lahore Campus, Lahore 54000, Pakistan
[2]Department of Management Information Systems, Ankara Medipol University, 06050 Ankara, Turkey
[3]Department of Computer Engineering, Istanbul Sabahattin Zaim University, 34303 İstanbul, Turkey
[4]Department of Software Engineering, Istanbul Nisantasi University, 34398 İstanbul, Turkey
[5]Deep Learning and Medical Image Analysis Laboratory, Boğaziçi University, 34342 İstanbul, Turkey
[6]Faculty of Business, Multimedia University, Malacca 75450, Malaysia
[7]Department of Business Administration, Daffodil International University, Dhaka 1207, Bangladesh
[8]Department of Computer Engineering, Istanbul Arel University, 34537 İstanbul, Turkey

Corresponding authors: Jawad Rasheed (jawad.rasheed@boun.edu.tr) and Sook Fern Yeo (yeo.sook.fern@mmu.edu.my)

ABSTRACT Recent reviews of the literature indicate the need for empirical studies on cross-project defect prediction (CPDP) that would allow aggregation of the evidence and improve predictive performance. Most empirical studies predict defects at granularity levels of method, class, file, and module/package during the coding phase, and thereby avoid external failure costs. The main goal of this study is to perform an empirical study on early defect prediction at the beginning of a project at the product level of granularity for using it as input in planning quality activities of the project. Hence, both internal and external failure costs could be avoided as much as possible through proper planning of quality. We first made a systematic mapping study (SMS) on secondary studies (literature reviews) on defect prediction to identify the most used datasets, the project attributes and metrics utilized as estimators, and the supervised learning methods employed for training the data. Then, we made an empirical study on defect density prediction using cross-project data. We collected 760 project data from the International Software Benchmarking (ISBSG) dataset version 11, which reported both defects and functional size attributes. We trained the prediction models using: i) the complete set of project attributes, ii) the individual attributes, and iii) multiple subsets of attributes. We employed classification and regression approaches of machine learning. The machine learning models are trained using original values of the dataset, and z-score and logged transformations of original values to explore the effects of data normalization on prediction. Most machine learning models trained on the z-score transformation of the dataset performed best for classifying defects. The Multilayer-Perceptron (Neural Network) model trained on the z-score transformation of complete dataset predicted defects with the highest F1-score of 0.89 using binary classification. The logged transformation and feature selection methods improved the results for multivariable regression. The multivariable regression predicted defects with the highest Root Mean Squared Error (RMSE) and R^2 (r-squared) values of 0.4 and 0.9, respectively, with a subset of 11 features using logged transformation. The results of classification and regression approaches indicate that defects can be predicted with reasonable accuracy at the software product level using cross-project data.

INDEX TERMS Cross projects dataset, defect prediction, feature selection, fault prediction, ISBSG dataset, machine learning.

The associate editor coordinating the review of this manuscript and approving it for publication was Ahmed Farouk.

I. INTRODUCTION

The costs associated with project quality consist of the cost of quality (money spent during the project to avoid failures)

and the cost of poor quality (money spent during and after the project for failures) [1]. Software companies require predicting time and resources for quality activities such as reviews, inspections, and testing at the beginning of a project as well as throughout the development life cycle as the data on defects are being collected in the software requirements specification, design, coding, and testing phases [2]. Most of the literature on defects estimation focused on predicting defects during the testing phase to allocate the required time and resources for removing them before delivering to the customers [3], [4] and thereby avoiding external failure costs (i.e., the cost of failures found by the customer). The identification and fixation of post-deployment defects typically require more effort as compared to the software development and testing phases. Therefore, the defect prediction term is usually considered as referring to predicting software defects in an intermediate or final software product, or its releases by the project team during testing activities [5]. Indeed, a recent SLR and meta-analysis on defect prediction [4] identified that most of the defect prediction studies predict defects at the method, class, file, and module/package granularity levels. They utilize the attributes of source code artifacts which are produced during the software coding phase [6], [7].

On the other hand, internal failure costs (i.e., the cost of rework for the failures found by the team during testing) are also very high [8]. Therefore, early software defect prediction for the final product is crucial for planning quality assurance and control activities to avoid both internal and external costs of failure [9].

The main goal of the study is to perform an empirical study on defects prediction of software at the beginning of a project for using it as an input in planning quality activities of the project. Hence, both internal and external failure costs could be avoided as much as possible.

To this end, we first made a Systematic Mapping Study (SMS) of literature reviews on defect prediction for identifying i) the machine learning methods applied, ii) the best estimator attributes, and iii) the project datasets utilized. According to the findings of the SMS, we designed our empirical study. In our SMS, we did not find any study that predicted defects for a software product at the beginning of a project using a publicly available dataset. This finding is in line with what Hosseini et al. [7] mentioned in their literature review and meta-analysis. Hence, most previous work is focused on defect prediction during the testing phase, which is too late to avoid costs of poor quality for a project.

Software defect prediction typically requires fitting a mathematical model on a training dataset that learns an estimator (e.g., software project and/or product attributes), and later using the model on unseen or new data [2]. In this study, we used binary classification and regression-based machine learning models for model fitting.

Most of the primary studies we identified in the SMS used machine learning approaches of binary classification and regression. The binary classification approaches predict the code as either defective or non-defective. The regression approaches predict the number of defects on a continuous scale. Hosseini et al. [7] found that 29 out of 30 primary studies on defect prediction predicted binary classes of defects and only one study performed regression. The regression provides more information because the prediction of several defects helps to plan testing and development resources that will be required to find and fix the defects.

Software companies have been reported to have difficulty in collecting and organizing useful defect-related data [4], [10]. Therefore, they tend to collect cross-project data (CPD) from open-source projects or publicly available datasets [3], [11]. Radjenovicetal et al. [11] found that 58% of the defect prediction studies used private datasets, 21% used partially public datasets, and 22% used public datasets. The private datasets including defect data, or source code are not publicly available. The partially public datasets share the source code of the project and defect data without the metrics' values (i.e., project attributes). The public datasets share the metrics values and the defects data for all the modules [11]. The most common estimators among the publicly available datasets include object-oriented and static code metrics. The object-oriented metrics contain information such as the number of classes, weighted method per class, and depth of the inheritance tree. The static code metrics contain information such as code churn, lines of code, number of modules/files, cyclomatic complexity, and function calls.

When the training dataset includes data from the same project, the process is referred to as within-project defects prediction (WPDP), and when data from other projects is used, it is referred to as cross-project defects prediction (CPDP) [10]. A recent systematic literature review (SLR) on CPDP [4] found that most of the primary studies on CPDP were published since 2010 and the interest has increased since 2015. For example, Malhotra [12] trained models using the NASA MDP dataset and predicted defects for several telecommunication projects in Turkcell. Turhan et al. [5] trained models using the Promise dataset [13] and predicted defects in projects of the SOFTLAB. Both studies predicted software defects at the module level using static code metrics such as cyclomatic complexity, number of lines of code, decision count, and the total number of operands and operators.

In this study, we used a publicly available software projects dataset, namely, the International Software Benchmarking Standards Group (ISBSG) dataset version 11. According to our SMS results, this dataset was not previously utilized for CPDP studies. We used a chunk of projects and their attributes for training the prediction model and the model predicted the software defects density (DD) of the products in the other chunk of projects.

The CPDP presents the challenge of effectively using a dataset of multiple projects in such a way that the effect of the heterogeneous data on prediction is reduced [4]. The data can be heterogeneous in terms of sources (e.g., databases and spreadsheets), formats (e.g., CSV, JSON, and XML), types (e.g., text, and numerical) and it might suffer from different data qualities from different sources. The data coming from

different sources needs to be integrated and its features are needed to be pre-processed to make them consistent in terms of their formats and data types.

In this study, we collected both numerical and categorical data from the ISBSG dataset. The ISBSG provides data in the MS Excel format. The data formats are consistent in the dataset and the data types were already defined by the group. The quality of the data was also assessed and provided. Hosseini et.al [7] pointed out a problem of lack of normal distribution in all the CPDP datasets, which we also observed in the ISBSG dataset, and thus normalized the original data. In CPDP, the challenge of dimensionality reduction is significant. This would help train generalized machine learning models and avoid overfitting. In this study, we catered to this problem. We also followed some of the relevant guidelines of Hosseini et al. [7] and Hall et al. [14] to our study (e.g., data quality; performance measures, and statistical tests) and we provided our reflections in the light of both studies.

Hence contributions of the study are:

- A systematic mapping study (SMS) on secondary studies (literature reviews) on CPDP.
- First empirical study on software product defects prediction for planning purposes using a publicly available dataset that contains cross project data.
- Collection of both numerical and categorical data from the ISBSG dataset. Transformed original data into z-scores and log values to cater lack of normal distribution.
- For the dimensionality reduction challenge, the study trained the models on i) the complete set of attributes, ii) the individual attributes, and iii) multiple subsets of attributes identified by feature selection methods.
- Comparisons of the study results to the findings of the secondary studies.

This article is organized as follows; Section II presents problem definition and data preparation phases of the empirical study and Section III presents experimentation of machine learning models and analysis of the results. Section IV presents the conclusions of the study and reflections on the findings of the study with respect to previous studies. Appendix A presents the details of our SMS on the literature reviews on defects prediction, Appendix B provides links for the source codes of experiments and details of the obtained results, and Appendix C presents names and descriptions of attributes/measures used in the ISBSG dataset.

II. EMPIRICAL STUDY

The empirical study included three phases: problem definition, data preparation, and experimentation as shown in Figure 1. Below, in the following subsections, we discuss these phases.

A. PROBLEM DEFINITION

In our SMS on secondary studies on defect prediction (Appendix A), we identified the most used datasets as NASA MDP, ECLIPSE, PROMISE, Mozilla, Apache, Jureczko, and Softlab. According to our findings, the ISBSG dataset has not been previously used for defect prediction. The ISBSG dataset contains data from 5,052 projects. We used a subset of this dataset that reported data on defects. The subset contains 760 projects with 23 attributes (Table 12). As for the defects, the ISBSG defines the Defects Density (DD) attribute for software products. The defect density measure is in general calculated as shown in (1).

$$Defect\ density = \frac{number\ of\ defects}{software\ size} \tag{1}$$

The primary studies used in the SMS count number of defects, however measuring DD for CPDP offers more advantages. The DD provides a normalized count of defects concerning size and it enables fair comparison of projects with varying sizes. The ISBSG dataset measures the size in terms of functionality provided by the software (i.e., functional size units) instead of counting source lines of code. The functional size units are a more useful measure of size when software projects are implemented in different programming languages.

The ISBSG defines DD as ''the Number of defects reported per 1,000 Functional Size Units of delivered software in the first month of use of the software''. The availability of values/labels for the DD attribute [target class] makes its prediction suitable for supervised learning [15], [16]. The DD attributes (dependent variable or target class) are available for 760 projects in the ISBSG dataset.

Two validity threats concerning the constructs of the empirical study might have been the fact that the ISBSG dataset has a mixture of different functional size measurement methods, and the defect counting methods differ from organization to organization. The ISBSG clearly defines what is a defect and states that it has worked with the data-providing companies to report defects consistently and using well-defined defect categories.

As for the units of functional sizes reported, we analyzed the dataset. Among the functional size measurement methods in the dataset, the Common Software Measurement International Consortium (COSMIC) method was developed to measure both business applications and real-time software. The rest of the methods have a similar scale for reporting software business applications' functional size. The COSMIC scale for reporting the sizes of business applications is also similar to those of previous methods. Cuadrado-Gallego et al. [17] reported an approximated conversion factor of 1:1, within a range between 0.9 and 1.1, but moving from a larger number of data points analyzed than in past studies. As most of the projects measured by the COSMIC method are business applications (57/69 projects), we decided to keep all 760 projects in our dataset.

The DD is predicted by using regression and classification approaches of machine learning. The regression uses a variable (x) or set of variables (x_1, x_2, ..., x_n) to learn an output variable y as a mapping function such that f(x) = y [15], [16]. The function approximating a numerical value of y based on a

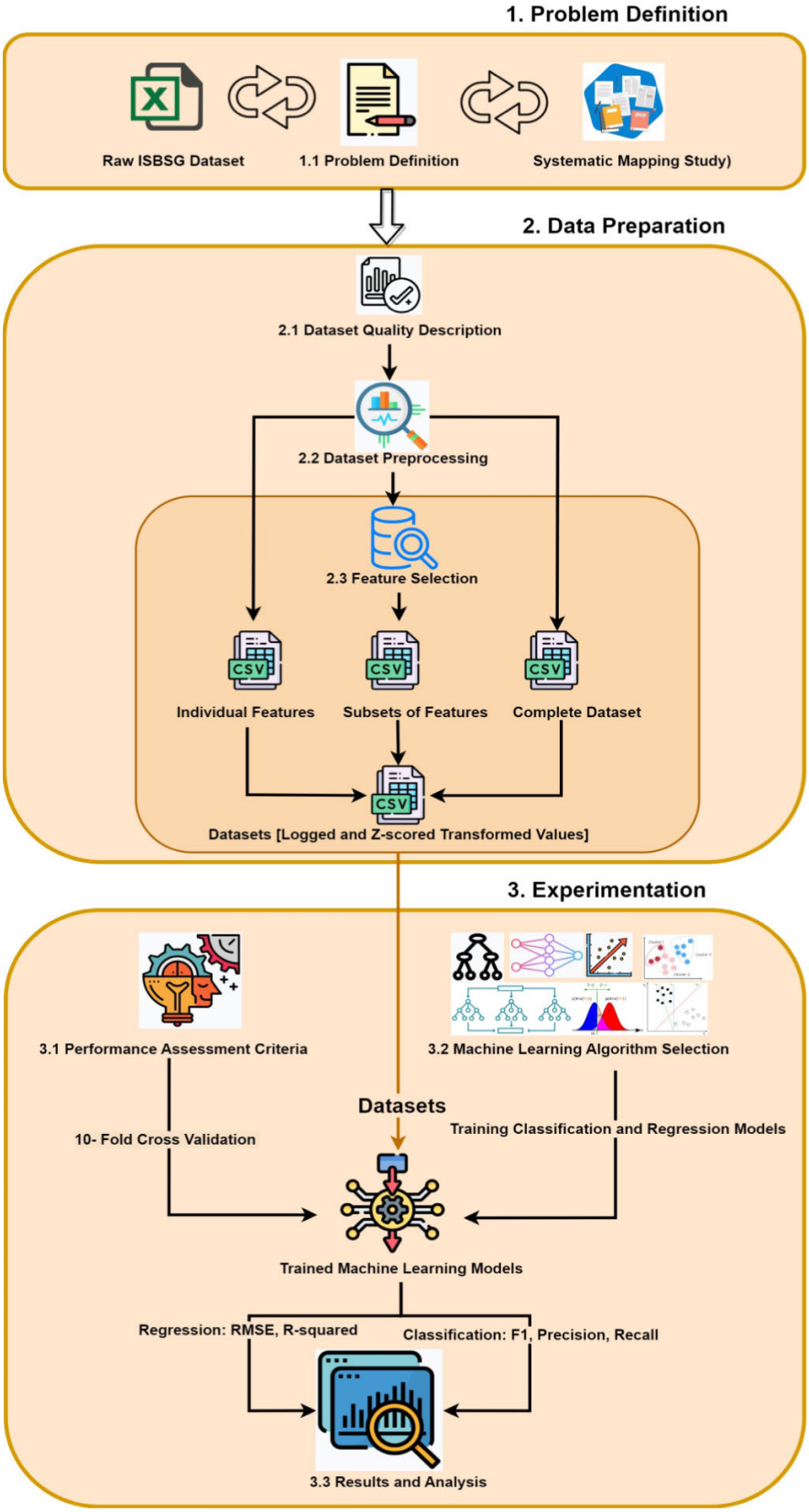

FIGURE 1. Cross-project defect density prediction study.

single variable is linear regression; otherwise, it is a multivariable (multiple) regression. The x_i values are either numerical or ordinal. Examples of regression include estimating the number of defects, function points, and effort (man-hours) in the upcoming project. Similarly, the classification predicts the value of y in terms of a variable (x) or a set of variables ($x_1, x_2, \ldots, x_n$). The x_i values are either numerical, ordinal or feature vectors. The value of y is either predicted as a binary class (e.g., non-faulty, faulty) or multi-class (e.g., non-faulty, very faulty, extremely faulty) [15], [16].

B. DATA PREPARATION

The data preparation consists of three sub-tasks i.e., dataset quality description, dataset description, and dataset construction.

1) DATASET QUALITY DESCRIPTION

The ISBSG dataset mostly includes numerical, categorical, and Boolean types of values. The coding schemes are consistent in the dataset and a meta-data file is available along the dataset. The dataset used in this study has no typographical errors or coding inconsistencies. The ISBSG dataset provides quality ratings for the project's data. The ISBSG organization claims that their quality ratings are trusted by software development organizations because they provide structured guidelines and questionnaires for data collection, and they rigorously check and recheck the quality of data before adding it to their repository.

Table 1 presents a description of quality ratings and the number of projects with each quality rating in the ISBSG dataset. The largest part of the data (92%) is classified as 'A' and 'B'. According to [7] and [14], NASA, PROMISE, and ECLIPSE are the top 3 most used datasets for WPDP and according to [4], NASA, Jureczko, and Softlab are the most used datasets for CPDP. We have observed that data quality information is missing in these datasets and in general among publicly available datasets. The quality rating is very important as if we are unaware of the quality of data then it makes the prediction less credible.

The software development organizations planning to use the ISBSG dataset along with their dataset for CPDP can develop their data quality guidelines in line with ISBSG. It includes defining data quality criteria such as given in Table 1. In addition, they should consistently define software attributes such as those given in Appendix C. It should be followed by identifying the sources of data collection (e.g., source code, software requirement specification (SRS) document, test cases, etc.) from software development processes, software products, and software development resources. The data quality information can be improved through the data profiling process i.e., defining meta-data such as statistics about the data and dependencies among the features. In addition, it involves defining use cases of data, profiling non-relational data, and exploring tools for automated quality assurance. A comprehensive data profiling mechanism can be defined by following the guidelines of Abedjan et al. [18]. The quality of data is maintained through educating data collection teams, organizing training and workshops, conducting data audits, and treating data quality improvement as a continuous process that is backed by project managers [19].

2) DATASET PRE-PROCESSING

This section presents descriptive statistics of the dataset and its relevant pre-processing. Tables 2 and 3 represent types of attributes/metrics and descriptive statistics of the dataset. The definition of each attribute is presented in Appendix C. Fenton and PFleeger defined three types of metrics (i.e., process, product, and resource) which are discussed in most of the software measurement studies [20], [21].

A software development organization with a defined measurement process has all of these three types of metrics data collected from past projects. However, it is observed in SMS studies (Appendix-A) that most of the defect prediction datasets used mostly the product type metrics (e.g., size, object-oriented, and complexity) followed by process metrics (e.g., testing effort, deployment time, etc.). The resource metrics are occasionally found in publicly available datasets. This study explores machine learning approaches using all three types of metrics. We have used 23 metrics/features including process (6), product (11), and resource (6) metrics in this study.

Table 2 presents descriptive statistics of attributes/metrics measured in numerical scale. The count attribute represents the number of values available without imputation. The number of missing values is due to the reasons that every software development organization does not collect all types of data due to heterogeneous domains of projects, available budget, and allocated time and objectives of the measurement process. The analysis of standard deviation (std), min (minimum value), max (maximum value), and quartiles (25%, 50%, and 75%) indicate skewness of the data, which is dealt with logarithmic transformation, given in (2), of attribute/metrics values.

$$X' = \log_{10}(X) \tag{2}$$

The heterogeneity of values in the dependent variable (DD) and independent variables (23 features in Table 1) is dealt with using the z-score, given in (3), where x is the value of attribute/metrics, μ is the mean and σ is the standard deviation. Table 3 presents descriptive statistics of attributes/metrics containing categorical/discrete values.

$$Z - score = \frac{x - \mu}{\sigma} \tag{3}$$

The codes of discrete values in available data are found consistent. The missing values are replaced with the help of the KNN-imputer algorithm (proposed in [22]) by using the sci-kit-learn API. It uses Euclidian distance between data points to determine nearest neighbor values for missing data. The use of logarithmic transformation and data normalization using a z-score helps to mitigate large disparities among the attribute/metric values. The data normalized using z-score

TABLE 1. ISBSG dataset quality description.

Quality Rating	Description	No. of Projects
A	"The data submitted was assessed as being sound with nothing being identified that might affect its integrity".	329 (43%)
B	"The submission appears fundamentally sound, but there are some factors which could affect the integrity of the submitted data".	377 (49%)
C	"Due to significant data not being provided, it was not possible to assess the integrity of the submitted data".	31 (5%)
D	"Due to one factor or a combination of factors, little credibility should be given to the submitted data".	23 (3%)

TABLE 2. The attribute measured in numerical scale in the ISBSG dataset.

Attribute Name	Type of Attribute/Metrics	Count	Mean	Std	Min	25%	50%	75%	max
Adjusted_Function_Points	Product	759	461.2	856.9	3	99.5	234	493	16148
Value_Adjustment_Factor	Product	479	1.0	0.1	0.3	1	1	1.09	1.3
Normalised_Work_Effort_Level 1	Resource	708	5316.4	10713.9	24	749.2	2000.5	5246.7	126133
Normalised_Work_Effort	Resource	760	5958.0	12308.8	24	866.2	2288	5725.5	150040
Normalised_Level_1_PDR	Resource	708	15.9	27.2	0.2	4.9	9	17.525	330.6
Normalised_PDR	Resource	760	17.0	27.3	0.2	5.5	10	19.125	330.6
Speed_of_Delivery	Resource	503	10.4	16.7	0.1	2.6	5.3	11.2	200
Project_Elapsed_Time	Process	730	8.9	9.5	0.2	4	7	12	131
Effort_Plan	Process	214	608.9	1587.6	0	65.5	194	509.5	17668
Effort_Specify	Process	227	961.5	2291.2	0	79.5	258	710	21578
Effort_Design	Process	195	621.5	1246.8	0	65	198	586	8766
Effort_Build	Process	265	3261.7	6769.2	4	455	997	2847	63852
Effort_Test	Process	287	1171.9	3138.3	0	124	349	868.5	37335
Effort_Implement	Process	195	372.0	810.1	0	24	95	362.5	5995
Effort_Unphased	Process	676	2853.3	7003.4	572	0	628	2880.5	106480
Average_Team_Size	Resource	195	5.2	5.6	1	2	3.3	6.15	42
Defect Density (DD)	Product	760	36.7	187.19	0	0	7.1	24.7	4236

TABLE 3. The attribute measured in the categorical scale in the ISBSG dataset.

Attribute Name	Type of Attribute/Metrics	Count	Categorical values	Frequency
Development_Type	Process	760	Enhancement	589
			New Development	418
			Re-development	28
			Other	7
Client_Server	Product	443	Yes	305
			No	232
			Not Applicable	4
Development_Platform	Product	691	MF	340
			Multi	285
			PC	266
			MR	71
Language_Type	Product	710	3GL	602
			4GL	321
			ApG	62
			2GL	1
CASE_Tool_Used	Process	389	Yes	228
			No	213
			Don't Know	61
How_Methodology_Acquired	Process	468	Combined Developed/Purchased	192
			Developed In-house	172
			Developed In-house	75
			Traditional	57
			Purchased	18
			Developed In-house	18
Defect Density (DD)			Yes	484
			No	276

ranges between −3 and +3. Log10 transformed the values of metrics ranging between 0.1 and 5.1 with an average of 2.1. The available dataset is transformed from an MS Excel sheet to CSV files.

The dataset is used for training regression and classification models. The regression models predict the DD i.e., number of defects per unit of function point. The regression approaches approximate a mapping function (f)

TABLE 4. Dataset construction for supervised learning (classification).

Experiments (EXP)	Exp-1
Construction of Datasets	All 23 attributes combined: 1 dataset Each individual attribute: 23 datasets Subsets selected using feature selection techniques: 5 datasets
Prediction Tasks	Predicting defect density (DD) using linear and multivariable regression

from a set of continuous/categorical input variables of training data (x) to a continuous output variable y. The DD attribute originally contains numerical/continuous values. The classification models are trained to predict binary classes i.e., defective, non-defective. The conversion of continuous variables into discrete ones (nominal intervals) is critical in machine learning as it increases learning efficiency, prediction accuracy, and comprehension of prediction results in a machine-learning context [15].

It is also critical because the majority of machine learning algorithms perform better while predicting target class in terms of discrete values for supervised learning [15]. A typical discretization method starts with sorting the numerical values in ascending/descending order. The discretization bins are created for ranges of values (i.e., intervals) by assigning a nominal label (i.e., Yes for defective, No for non-defective) to each interval. The binary classification for defect prediction is most widely used for most of the supervised learning tasks and especially for CPDP [7], [12], [15], [23].

3) FEATURES SELECTION

The cross-project datasets contain data from heterogeneous projects therefore it has chances of containing irrelevant and redundant features which might add to the curse of dimensionality (i.e. lack of performance in prediction due to irrelevant features) [7], [15], [24], [25], [26]. The features/metrics subsets selection techniques are used for dimensionality reduction [7], [15],. These techniques are specifically identified and applied for regression and classification approaches in the following sections.

Feature Selection for Classification: There is no fixed set of metrics in the software measurement domain that is used to predict a specific attribute [20], [21], [27]. Therefore, this study predicts DD classes by using the dataset in three different ways as illustrated in Table 4. Each attribute is used to train the machine learning model to evaluate its ability to predict DD. The complete dataset is also used to predict DD. In addition, five feature selection techniques are also used to select subsets of data. The description of feature selection techniques and attributes/features selected with each technique are presented in the following section.

It can be performed in three ways i.e., wrapper, filters, and embedded methods [28]. The wrapper method creates subsets of a dataset using a prediction model. These subsets are randomly created, and they range from an individual to many features. The exhaustive search is time-consuming for a large number of features. In our case, due to 23 features, it was appropriate to use to method. The wrapper method iteratively creates a learning scheme by using cross-validation to select the best subset of attributes. Each subset is split between train and test sets. Each subset trains a prediction model, and the best subset is selected based on less number of incorrect predictions (error rate) on the test set. The filter methods use a proxy measure instead of an error rate. Filter methods are more computationally intensive than wrapper methods. The proxy measures include feature ranking based on different methods such as chi-square and information gain. The chi-squared statistic of an attribute concerning the target class is evaluated to select an attribute. It is calculated in the (4):

$$X_C^2 = + \sum \frac{(O_i - E_i)^2}{E_i} \log_{10}(X) \tag{4}$$

where: C is the degree of freedom, O is the observed value(s), and E is the expected value(s). The information gain of an attribute is measured concerning the target class to select an attribute. It is calculated in (5). Embedded methods are a catch-all group of techniques that involve feature selection as a part of building a prediction model such as using a recursive feature elimination algorithm with principal component analysis (PCA) and gain ratio to iteratively build a predictive model by removing features with low weights [23].

$$\begin{aligned} &InfoGain\,(Class, Attribute) \\ &\quad = H\,(Class) - H(Class|Attribtue) \end{aligned} \tag{5}$$

The gain ratio of an attribute is measured with respect to the target class to select an attribute. It is calculated as follows:

$$GainR\,(Class, Attribute) = \frac{InfoGain}{H(Attribute)} \tag{6}$$

The PCA is achieved through performing data transformation and eigenvectors of the attribute are mapped on principal component space to eliminate eigenvectors with noise and later transform back to the original space. The PCA is used in combination with the Ranker search. The given dataset is used to identify the kth principal component of a data vector x(i) in the coordinates of tk(i) = x(i). w(k), (where w(k) is the k^{th} eigenvector of xTx) after the data transformation. The subsets of features are presented in Table 14 of Appendix C.

Feature Selection for Regression: The regression approaches approximate a mapping function (f) from a set of continuous/categorical input variables of training data (x) to a continuous output variable y. The dataset is a combination of attributes that contain discrete values or real numbers. The non-numeric discrete values are converted into numeric values before training regression algorithms. This study uses the dataset for regression in three different ways as illustrated in Table 5. Each attribute is used to train the machine learning model to evaluate its ability in predicting DD.

The complete dataset is also used to predict DD. In addition, four subsets of features are created with the help of feature selection techniques i.e., multicollinearity, variance inflation factor (VIF), P-value, and Bayesian information

TABLE 5. Dataset construction for supervised learning (regression).

Experiments (EXP)	Exp-2
Construction of Datasets	All 23 attributes combined: 1 dataset Each individual attribute: 23 datasets Subsets of attributes: 6 datasets
Prediction Tasks	Predicting defect density (DD) using linear and multivariable regression

criterion (BIC). Statistics-API[1] and Scikit-learn[2] are used to perform the aforementioned methods. A subset of features is also created by adding values of effort attributes/measures in the software development life cycle i.e., planning, specifying, designing, building, testing, and implementation efforts as 'total effort' attribute/measure. In addition, this shortened dataset is further reduced using multicollinearity analysis. In total, this makes 6 subsets of features.

Multicollinearity in the dataset is detected when independent variables are linearly correlating with each other. The drawback of predicting with a dataset having multicollinearity is the inability to know which features are contributing toward predicting a dependent variable (DD in this study). The heat map in Figure 2 is used to identify multicollinearity in the dataset using Spearman correlation and features having a higher correlation than 0.7 (moderately strong relationship) are dropped to create a subset [29]. Estimating the p-values of each feature helps to identify the statistically significant features ($p < 0.05$) for predicting the dependent variable. It is calculated by evaluating the results of ordinary least square (OLS) regression using two-tailed T-tests using stats models API. The features having a p-value less than 0.05 are selected to create a subset [30].

The VIF is used to select a subset of features (independent variables) by measuring the impact of their multicollinearity on the preciseness of predicting dependent variables in regression models [31]. Multicollinearity is observed when two or more variables in a regression model have a high correlation. The features having a VIF score less than 10 are selected to create a subset [31]. The VIF score of 10 means that the variance of the coefficient of prediction (such as R^2, RMSE, etc.) will be 10 times more than in a case where there will not have been a correlation. The high VIF score causes inflated standard errors which are uncertain in identifying the unique contribution of each predicting variable in the regression model.

This deteriorates the interpretability of the regression model because it reduces the reliability and stability of the coefficients of independent variables in a regression model. The VIF score of 10 is most commonly used as a threshold of feature selection because VIF less than 5 shows a low correlation between predicting attributes. A VIF score between 5 and 9 is considered as moderate and a VIF score greater than 9 or more is considered as high correlation [31].

The BIC is a statistical method that is used to select among competing regression models. A regression model having the smallest value of BIC is considered the most suitable for prediction [32]. The formula to calculate the BIC score i.e., $BIC = -2 \times LL + \log(N) \times k$, where LL is the log-likelihood of the model which means how well the model fits the training data, N is the number of training examples and k is the number of independent variables in the model. The BIC method aims to select a model that best fits the training data (higher LL) and has a smaller number of independent variables (K). It reduces the complexity and improves the interpretation of the model. The lowest BIC score indicates the best tradeoff between the model fitting the data (LL) and its complexity in terms of several variables (K). The BIC score calculated for models based on original, shortened, multicollinearity, P-value, and VIF-based selection of features is 4048, 4009, 4023, 4006, and 4017 respectively. This hints that if complexity and performance both are considered then a subset of three features that are selected based on P-value might be best predicting DD. However, if the complexity is ignored then subsets of eleven features selected based on multicollinearity performed slightly better than the p-value-based subset of features. The subsets of features are presented in Table 14 of Appendix C.

III. EXPERIMENTATION

This phase discusses performance assessment criteria and results and analysis of prediction.

A. PERFORMANCE ASSESSMENT CRITERIA

This task is performed in two steps. One is to decide how the measurement dataset is used to train and test the machine learning models, second is to decide how the performance of machine learning models will be assessed.

1) TRAINING-TESTING METHOD

The selection of an appropriate training-testing method depends on several factors such as dataset size, reliability of testing/evaluation results, and required computation resources. The dataset used in this study is cross-project data of 760 heterogeneous projects. The size of the dataset is smaller in CPDP studies (SMS, Appendix A) therefore it does not require special training resources. The heterogeneity of the projects requires avoiding methodological mistakes of learning parameters of prediction function from a chunk of data and applying it to the other chunk. This can cause higher variability in the performance of training models based on how data is split into chunks, and it also has a limitation of using a chunk of data for testing which could have been used for training as well. This reduces the generalization of a training model and intends to perform accurately on training data but cannot accurately predict based on yet unseen data which is called overfitting. This problem can be avoided through cross-validation. To implement cross-validation, the

[1] Statemodels 0.14.0: https://www.statsmodels.org/stable/index.html [Last accessed: June 01, 2023].

[2] Scikit-learn: machine learning in Python – sickit-learn 1.2.2.: https://scikit-learn.org/stable/ [Last accessed: June 01, 2023].

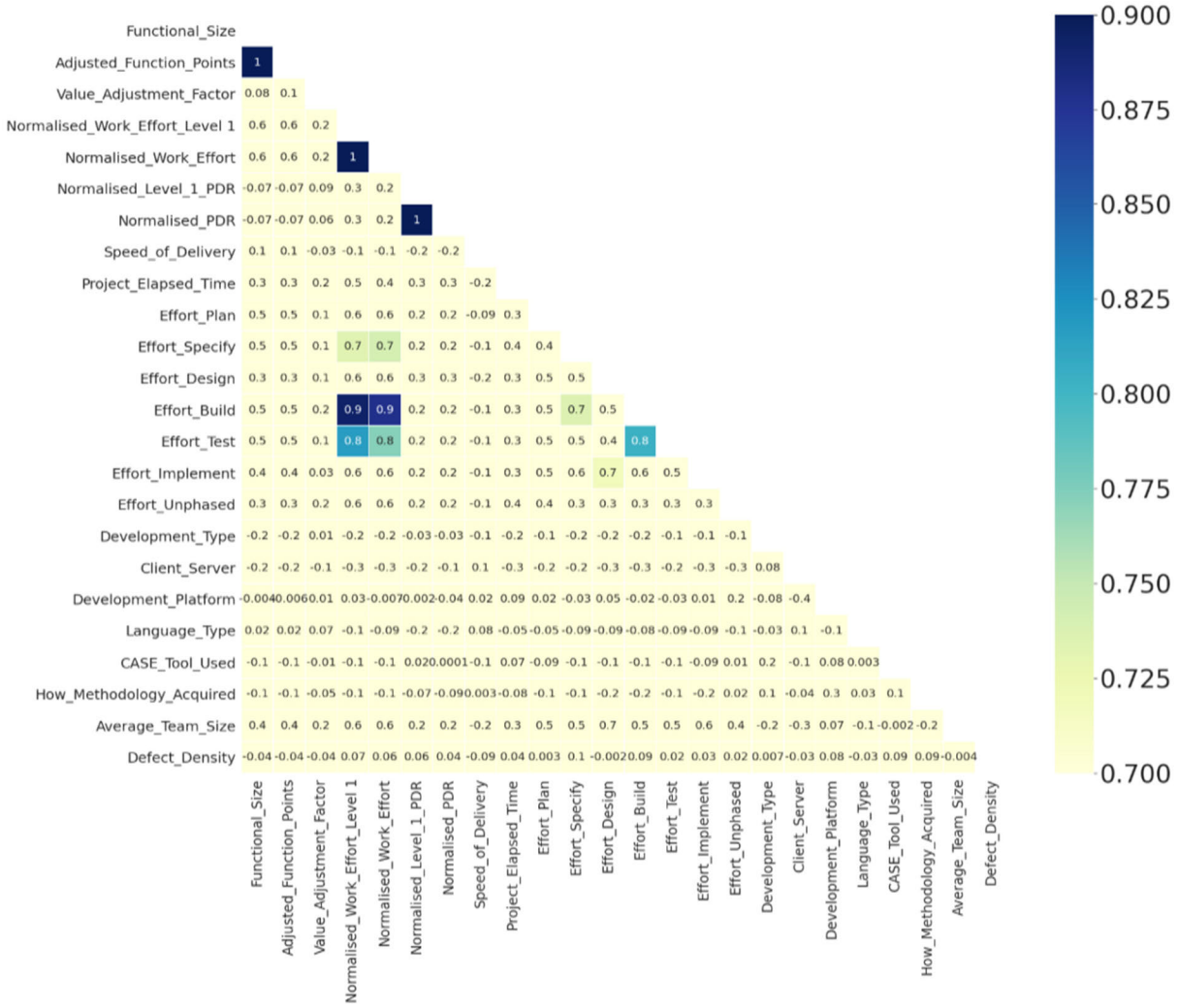

FIGURE 2. The heatmap of multicollinearity among all features.

predictive models are trained in three steps by dividing the dataset into three subsets i.e., training data, validation data, and test data. The training data is used to construct the predictive model and validation data is used to fine-tune the predictive model. The test set is then used to estimate the performance of the predictive model [15]. N-fold cross-validation is the approach to better estimate the performance of ML algorithms. Using this approach, data is split into N equal disjoint sets. N-1 sets (or folds) are used for training of ML model and the nth set is used for testing. The data sets for training and testing are different in each of the N iterations [15]. We used 10-fold cross-validation for our classification and regression experiments. This approach utilizes the whole dataset for training and testing, and it reduces overfitting.

2) PERFORMANCE EVALUATION

Classification Measures: According to studies in the SMS, precision, recall, and F1-Score are the most commonly used measures for performance evaluation [6], [7], [12], [14], [33], [34], [35], [36]. The possible outcomes of a prediction based on a machine learning algorithm are evaluated using a number of True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) measures. The precision, recall and are calculated with the help of TP and FP, and F1-Score is calculated with the help of Precision and Recall as given in (7) – (9).

$$Precision = \frac{TP}{TP + FP} \tag{7}$$

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1\ score = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{9}$$

According to Hosseini et al [7], the F1-score and area under the curve (AUC) are the most used evaluation measures among CPDP studies. The ISBSG dataset like other cross-project industrial datasets contains imbalance class distributions.

The choice of evaluation measures for a training model is critical for understanding its performance. Precision measures the correctly classified defects among all the predictions that classify the product as defective as shown in equation (7) above. The high precision means that actual defects are accurately identified through the training model and defect-finding resources will be correctly deployed. The recall measures correctly classified defects concerning all the actual defects (true positive cases) and wrongly classified defects (false positive cases) in supervised learning as shown in equation (8). It helps to maintain the quality of software products because high recall means that there will be fewer chances of missing a real defect. The F1-score is the harmonic mean (H.M) of precision and recall as shown in equation (9). An imbalanced dataset suffers from a disparity between the number of positive and negative classes. The imbalanced data causes training models to be biased towards a certain class. The F1-score is suitable for evaluating the performance of the training models on datasets with imbalanced class distributions [8]. The F1-meaasure depicts a lower score if the training model fails to detect a defect or incorrectly classifies the defect.

Regression Measures: According to SMS, most studies use Mean Squared Error (MSE) [36], Root Mean Squared Error (RMSE) [6], [7], [14], [35], [36], and r-squared (R^2) [7] to evaluate regression results. MSE is based on the average squared difference between the predicted values and the original values in the dataset. It measures how close a fitted (regression) line is to actual data points. The smaller value of MSE shows a better fit of the actual data set. RMSE is the square root of MSE. The r-squared (R^2) is also called the coefficient of determination and it measures how well the predicted values approximate the actual data points. Ideally, a R^2 value of 1 and an RMSE value of 0 represent that predicted values are equal to actual data points.

MSE and RMSE estimated over n samples are defined in equations (10) and (11), respectively.

$$MSE\left(y,\hat{y}\right)=\frac{1}{n_{samples}}\sum_{i=0}^{n_{sample}-1}\left(y,\hat{y}\right)^2 \tag{10}$$

$$RMSE\left(y,\hat{y}\right)=\sqrt{MSE\left(y,\hat{y}\right)} \tag{11}$$

where: $\hat{y}$ is the predicted value of the i^{th} sample, and y is the corresponding true value. R^2 estimated over n samples is defined in (12).

$$R^2=1-\frac{SS_{Residual}}{SS_{total}} \tag{12}$$

$SS_{Residual}$ is the sum of squares of residuals defined in (13).

$$SS_{Residual}=\sum_{i=0}^{n}\left(y_i-\left(\hat{\alpha}+\hat{\beta}x_i\right)\right)^2 \tag{13}$$

where $\hat{\alpha}$ is the estimated value of constant term α and $\hat{\beta}$ is the estimated value of coefficient β. x_i is the i^{th} value of an independent variable. SS_{total} in (14) describes how well the regression model fits the training data. Its higher value denotes that the regression model does not fit the training data.

$$SS_{total}\left(y,\hat{y}\right)=\sum_{i=1}^{n}\left(y_i-\hat{y}\right) \tag{14}$$

Each of the evaluation measures provides an important insight into the performance of the training model. The MSE and RMSE are measured in the same units of the target variable which supports direct interpretation of results. Both measure the standard deviation among the differences between actual and predicted values i.e., residuals. The MSE takes a square of the residuals, which makes the interpretation of the results difficult. Taking the square root of the MSE (i.e., RMSE) helps to normalize the results. RMSE is preferred over MSE when comparing the performance of multiple training models. The large residuals will cause the RMSE value away from its ideal value of 0. The R^2 provides the goodness of fit of a training model. It indicates how well the variances in the target variable are explained by the model. A higher value of R^2 (ideally close to 1) indicates that the training model has learned patterns and relationships among independent and dependent variables during the training phase. It indicates how well the model might perform on the unseen data.

In summary, the RMSE provides insight into the accuracy of the predictions and R^2 explains how well the training model adapts to the variance in the target variable. Therefore, a combination of both is used to select the best training models.

B. MACHINE LEARNING ALGORITHM SELECTION

The supervised machine learning algorithms are selected based on the most commonly used and successful machine learning approaches on CPDP datasets that are identified in the SMS in Appendix A.

1) PARAMETER SETTING

The machine learning models are implemented using a renowned machine learning API i.e., scikit-learn. The classical machine learning algorithms such as linear regression, multivariable regression, logistic regression, decision tree, support vector machine, and Naive Bayes are trained on default settings. The K-NN model is trained after finding the optimum value of a number of clusters. The hyper-parameter tuning is performed for ensemble decision tree models i.e., random forest and extra-tree classifiers. These settings include the number of decision trees (100 to 1200), criterion ('gini', 'entropy'), the maximum number of tree levels (5, 23), minimum number of samples required to split a node (2, 5, 10, 15, and 100), minimum number of samples required for leaf node (1, 2, 5, and 10). The multilayer perceptron (ANN) model is also trained with multiple combinations of the parameters and retained with the best performance. These settings include activation functions ('identity', 'logistic', 'tanh', 'relu'), solver ('lbfgs', 'sgd', 'adam'), learning rate (0.1 to 0.5), number of neurons in the hidden layer (up to 100), number of iterations (up to 200), kernel regularizer (L2 penalty, alpha=0.0001).

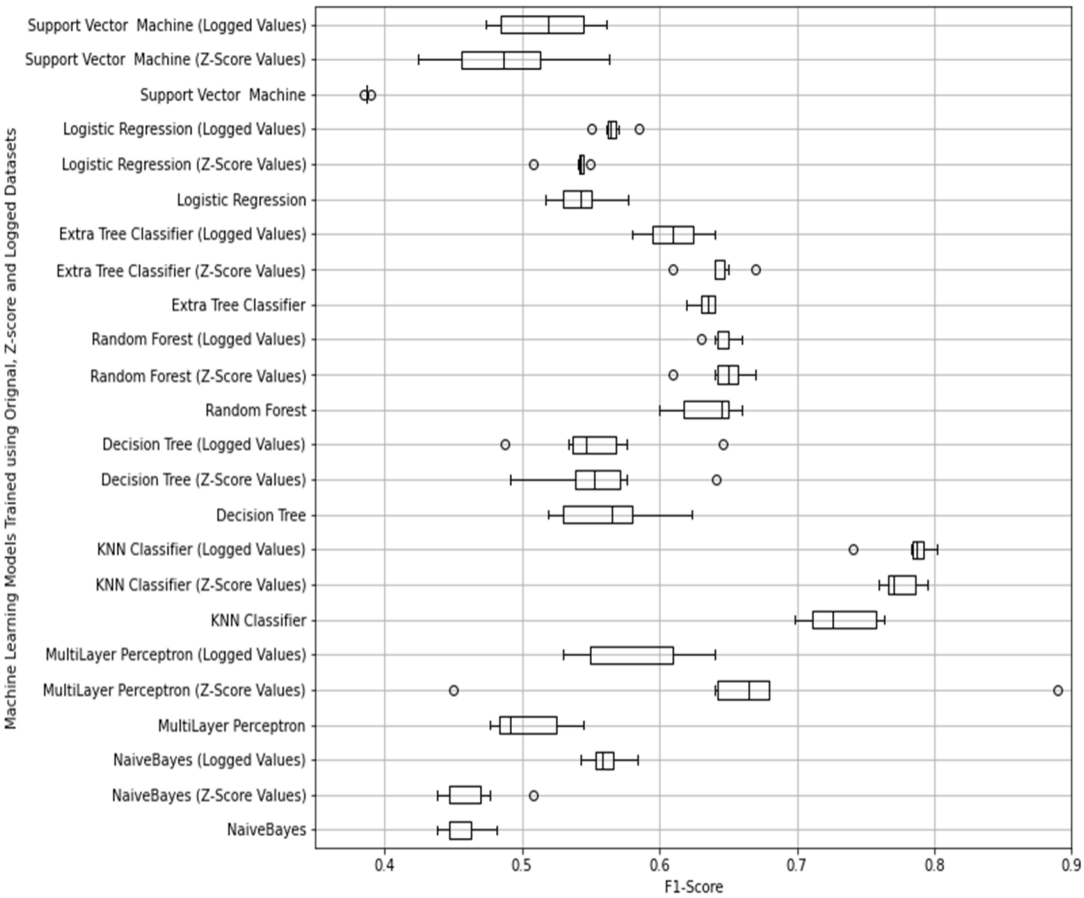

FIGURE 3. Performance of machine learning models using datasets and subsets.

C. RESULTS

The following sections present results and analysis of classification and regression tasks.

The complexity of a training model and its predictive performance are considered in deciding the most suitable machine learning model for CPDP. The simpler models due to the smaller number of features contain more biases because they do not process all the information of the complete dataset. In addition, their adaptation to lower variance makes them more vigorous to noisy data. However, simpler models are prone to underfitting the training data. On the other hand, complex models might become overfitting i.e., fitting too well on the training data and poor performance on unseen data. These models tend to adapt to higher variance and missing underlying patterns among features in the training data. To tackle the issue of underfitting and overfitting, we have used N-fold cross-validation to train and test all chunks of the dataset. In addition, we have used data transformations to normalize the effect of data heterogeneity. Furthermore, more than one evaluation measure is used to mitigate the bias in evaluating the performance of training models. After taking the aforementioned steps, the variations of simple to complex models are also experimented with using a feature selection process to select the best set of features to train the machine learning model and predict DD.

1) CLASSIFICATION

This study has used the dataset in multiple ways to predict DD classes using 8 machine learning algorithms for identifying the best set of metrics. The variations of the dataset include:

Original dataset:

- Original dataset containing all 23 metrics/features combined. {1 dataset}.
- The original dataset is transformed into Logged values of its features. {1 dataset}
- The original dataset is transformed into Z-score values of its features. {1 dataset}

Subsets:

- Six feature selection methods are used to create subsets of metrics/features from the original dataset. {6 datasets}
- Each subset is transformed into Logged values of its features. {6 dataset}
- Each subset is transformed into Z-score values of its features. {6 dataset}

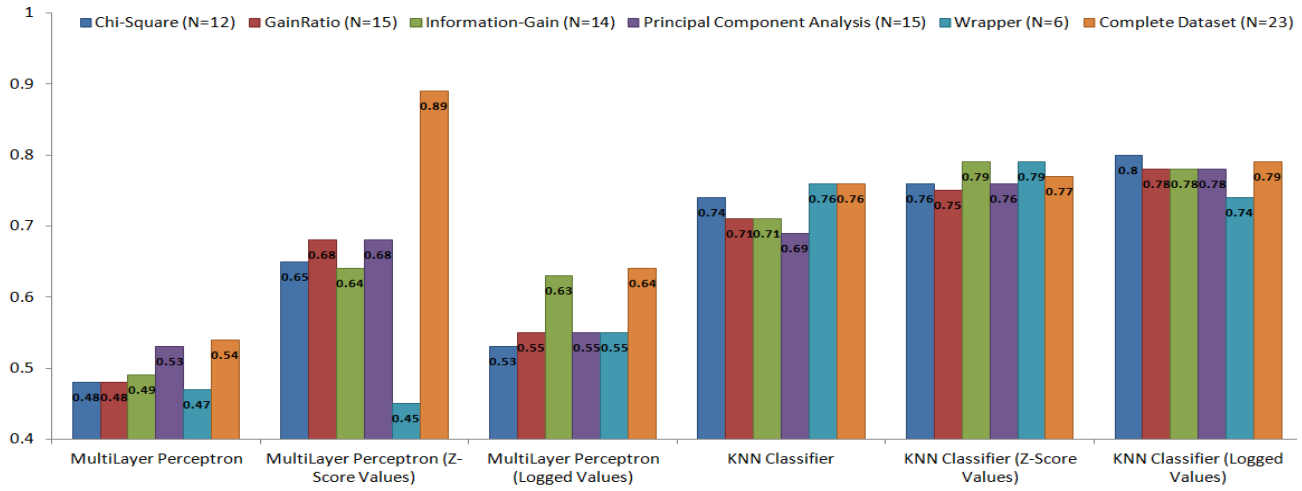

FIGURE 4. Highest performances of multilayer perceptron and KNN classifier models using datasets and subsets with various feature selection techniques.

Individual features:

- Each metric/feature is separately used to build machine learning models. {23 metrics used as 23 datasets}.
- Values of each feature are transformed into Logged values. {23 dataset}
- Values of each feature are transformed into Z-score values. {23 dataset}

Due to space constraints, we have made all results available in Google Drive (Appendix B). The box plot in Figure 3 depicts the overall performance of machine learning algorithms on the subsets and complete datasets. The datasets are also transformed into logged values and z-score values to normalize them. Multilayer Perceptron performed best with an F1-score of 0.87 followed by the KNN classifier with an F1-score of 0.80. Random Forest and Extra Trees classifiers predicted the highest F1 score of 0.67.

Figure 4 depicts the performance of the top two models, and it is evident that the Z-score transformation of the dataset helped to train the model with the highest F1 score. Logged transformation of 12 features extracted using chi-square feature selection helped to train the model with the second highest F1-score of 0.80. This shows that data transformations and feature selection are useful on the dataset because all the higher F1-scores are achieved using transformations as compared to using original values as shown in Figure 3 and Figure 4. The box plot in Figure 5 presents the overall performance of machine learning models on individual features.

The KNN classifier model outperformed other models with best highest F1-score of 0.77. All other machine learning models mostly predicted DD around an F1-score of 0.50, which is considered equal to random guessing [37]. KNN classifier model has performed consistently over individual features, complete dataset, and its subsets.

Figure 6 presents the best results of DD prediction by training the KNN classifier model using individual features. The features with an F1-score of 0.7 or above are presented. The 'functional size' feature predicted DD with an F1-score of 0.77 followed by 'normalized work effort level 1', 'normalized level 1 PDR (ufp)', and 'normalized work effort' with F1-scores of 0.75, 0.75, and 0.74, respectively.

Receiver Operating Characteristics (ROC): In this study, the ISBSG dataset has 64 percent of projects classified as non-defected which means that the dataset is skewed [38], [39]. The Receiver Operating Characteristic (ROC) graph is used to measure the algorithm's performance being insensitive to class skewness and unequal classification error costs. The performance metrics (e.g. precision, recall, F1-score) are based on confusion metrics and they are sensitive to class skewness [15], [38], [39]. These performance metrics depend on the distribution of instances in a dataset, for example, faulty (e.g., positive) and non-faulty (e.g., negative) instances for binary classification of defects.

The Multilayer Perceptron and KNN classifier models predicted DD with their highest F1-score of 0.89 and 0.80 respectively. Therefore, we used ROC curves to visualize their performance concerning the true positive rate and false positive rate in Figure 7.

The Multilayer Perceptron algorithm has True-Negative Rate = 45, False-Positive Rate = 10, False-Negative Rate = 7, True-Positive Rate = 90, and area under the curve (AUC) = 0.96. The KNN classifier model has a True-Negative Rate = 37, False-Positive Rate = 18, False-Negative Rate = 10, True-Positive Rate = 87, and area under the curve (AUC) = 0.88. Both algorithms performed very little sensitivity to skewness; however, Multilayer Perceptron is a better predictor of DD based on higher AUC and F1-score.

2) REGRESSION

Linear regression and multivariable regression are used to predict DD using individual features and combined features respectively.

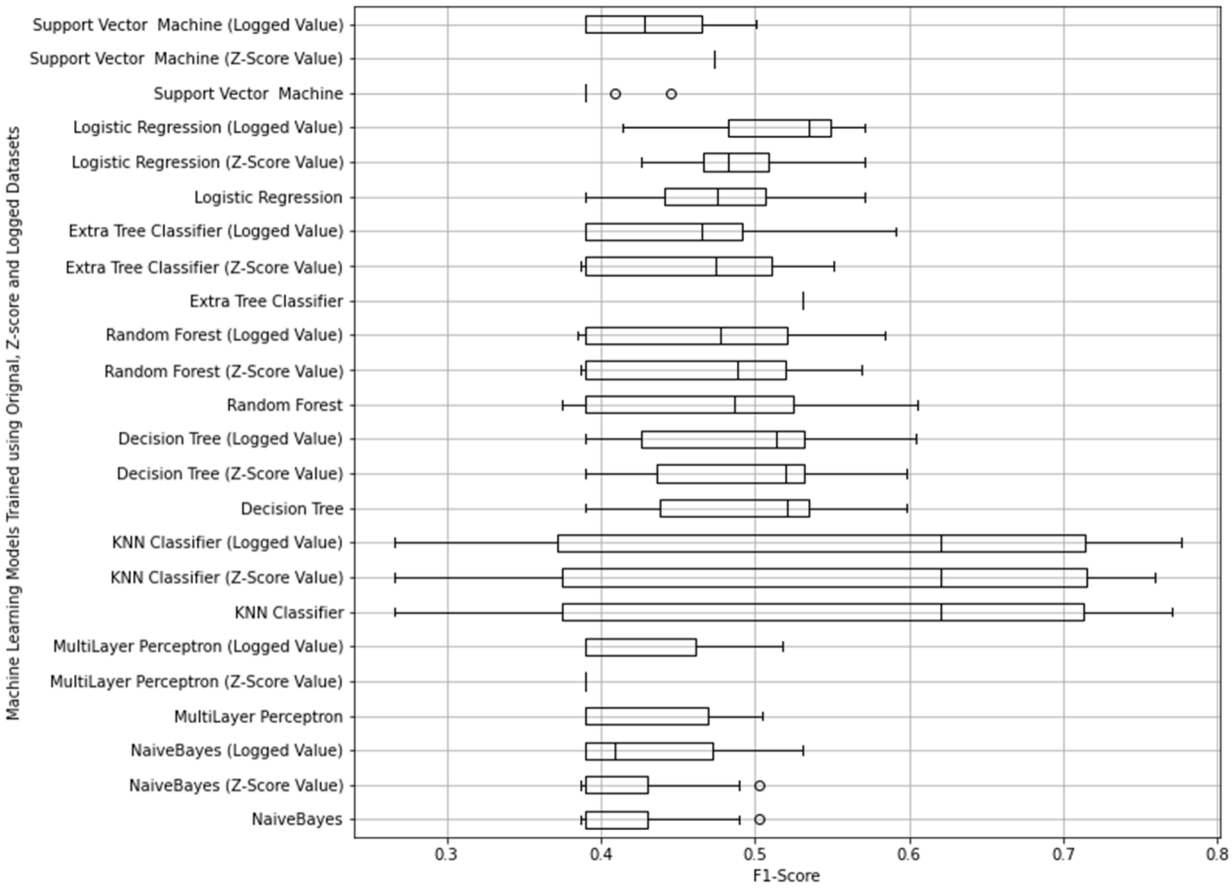

FIGURE 5. Performance of various machine learning models using individual features.

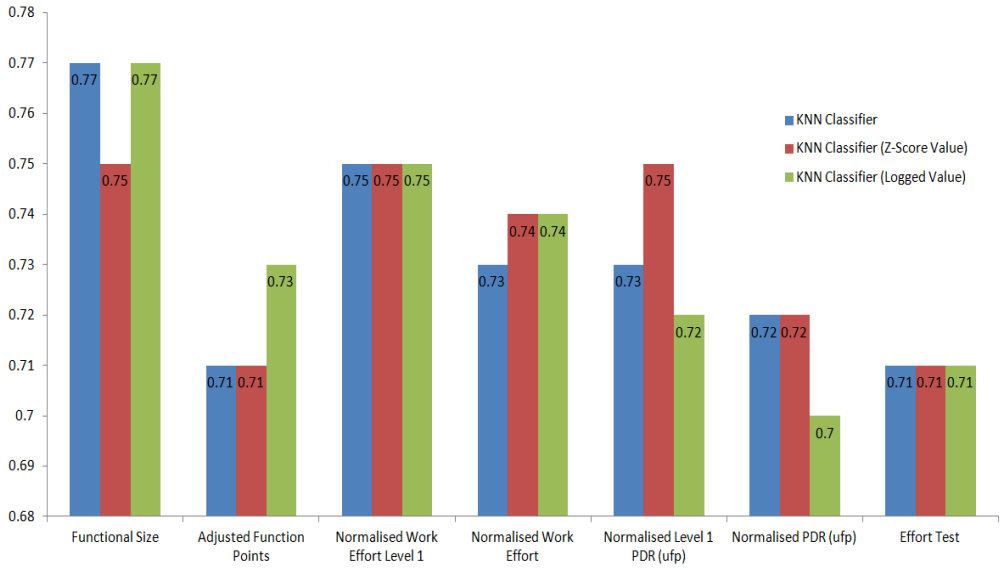

FIGURE 6. The F1-score of various KNN-based models using individual features.

Linear Regression: The individual features are evaluated for their ability to predict DD using their original values, logged values, and z-scores. The R^2 and RMSE are used to evaluate the performance of regression. The ideal values of R^2 and RMSE are 1 and 0, respectively.

Figure 8 represents the performance of linear regression using original, logged, and z-score values of individual features and DD features. The r-squared results of only those features are presented which are closer to 1 (i.e., between the range of 0.4 and 0.9). Logged feature values predicted DD

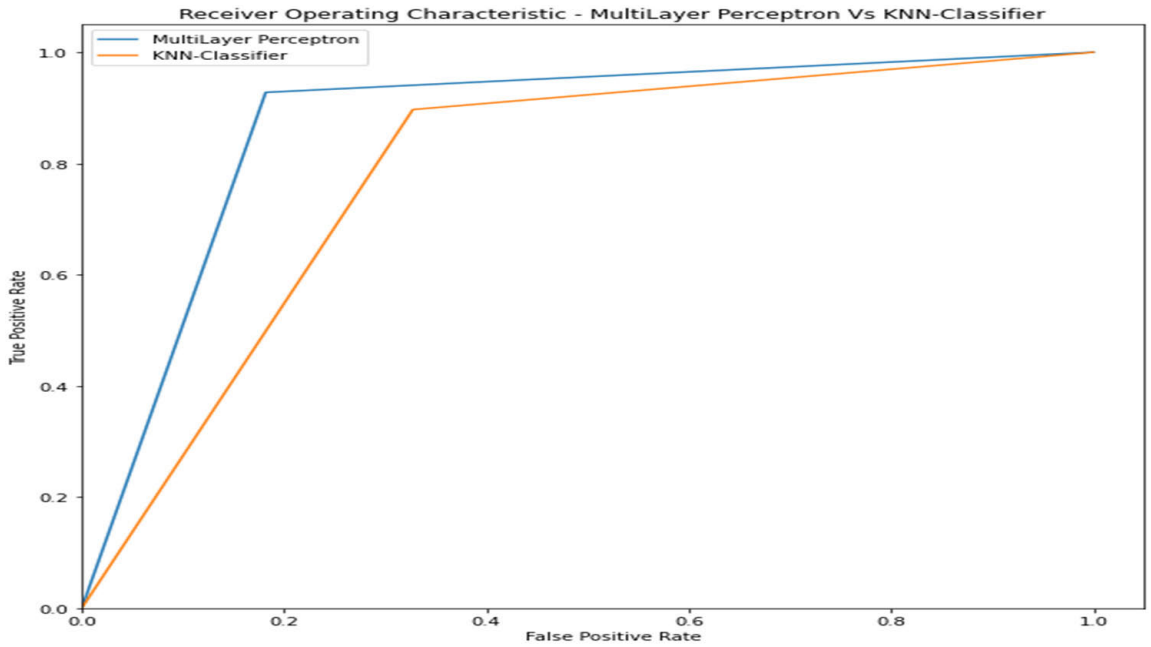

FIGURE 7. The comparison between ROC curves of multilayer perceptron and KNN classifier.

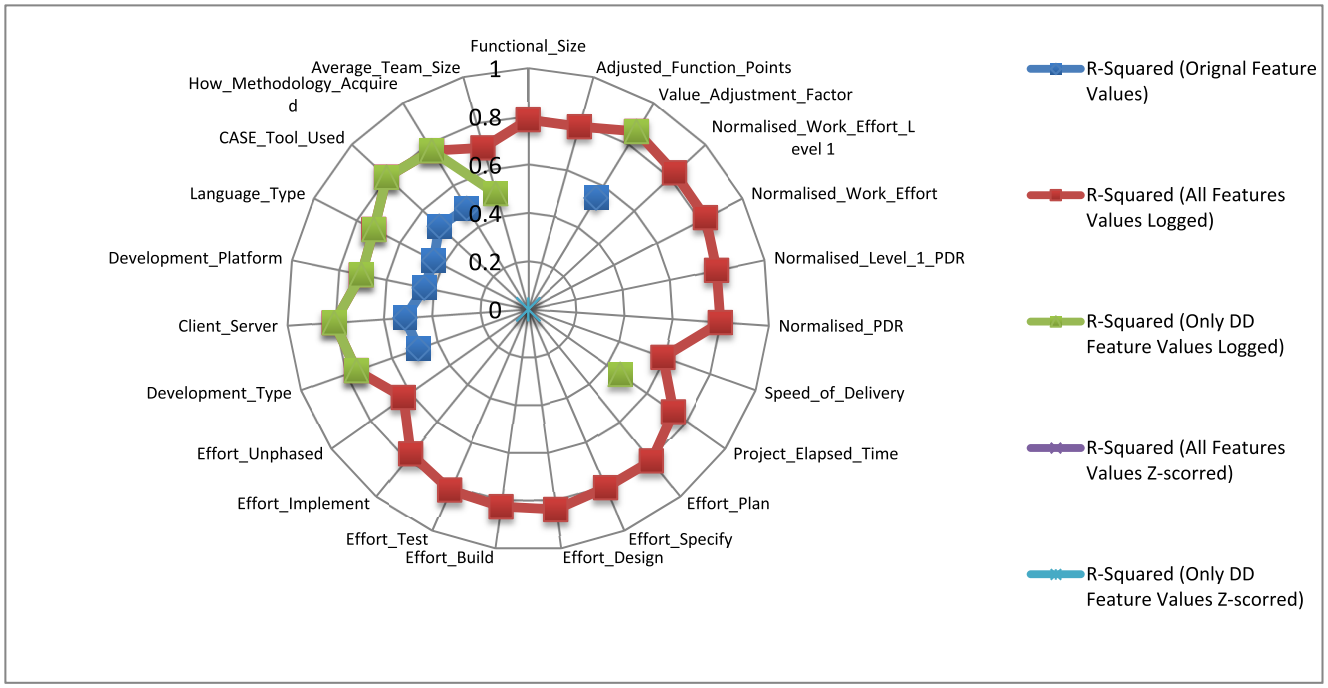

FIGURE 8. The R-squared (R^2) values of defect density prediction using individual features.

better than using z-score values of features. Original features values of 'Developement_Type', 'Client_Server', 'Developement_Platform', 'Language type', 'CASE tools used', and 'How_methodology_Acquired' predicted logged values of DD equally well as with all logged values. The logged values of productivity and effort features (e.g., normalized_PDR, effort_specify) predicted DD with r-squared values of more than 0.8 which is close to the ideal value of 1.

Figure 9 represents the performance of linear regression in terms of RMSE by using original, logged, and z-score values of individual features and DD features. The RMSE results of only those features presented that are closer to zero (i.e., between the range of 0 and 0.6). It is evident that the logged feature values predicted DD better than using z-score values of features. Logged values of 'Developement_Type', 'Client_Server',

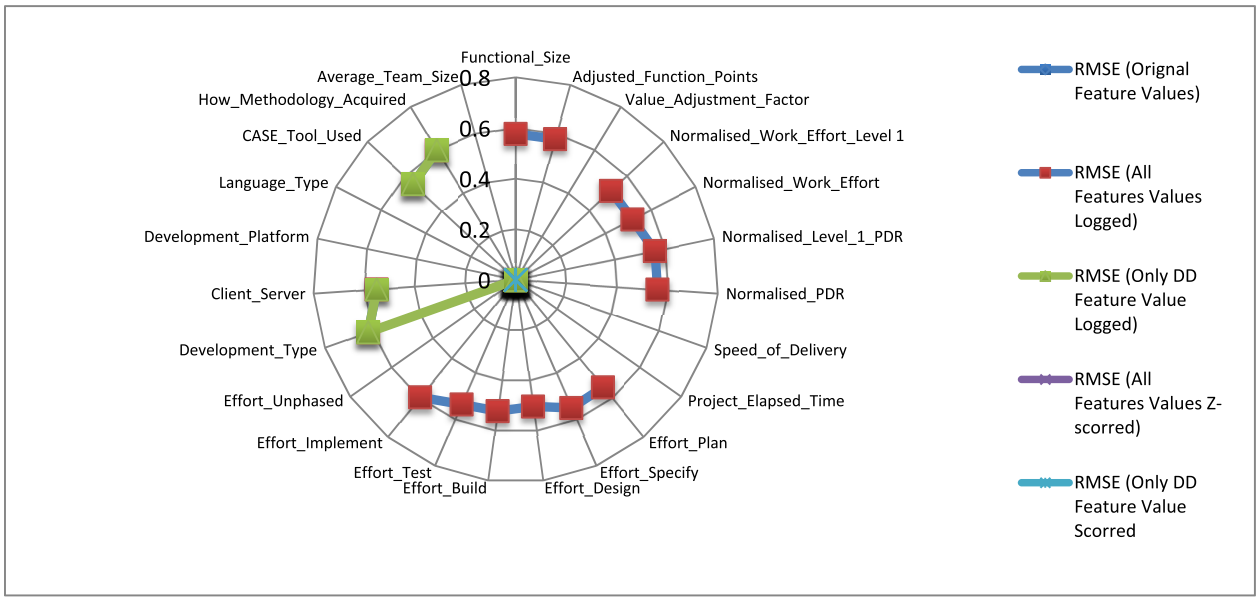

FIGURE 9. The RMSE values defect density prediction using individual features.

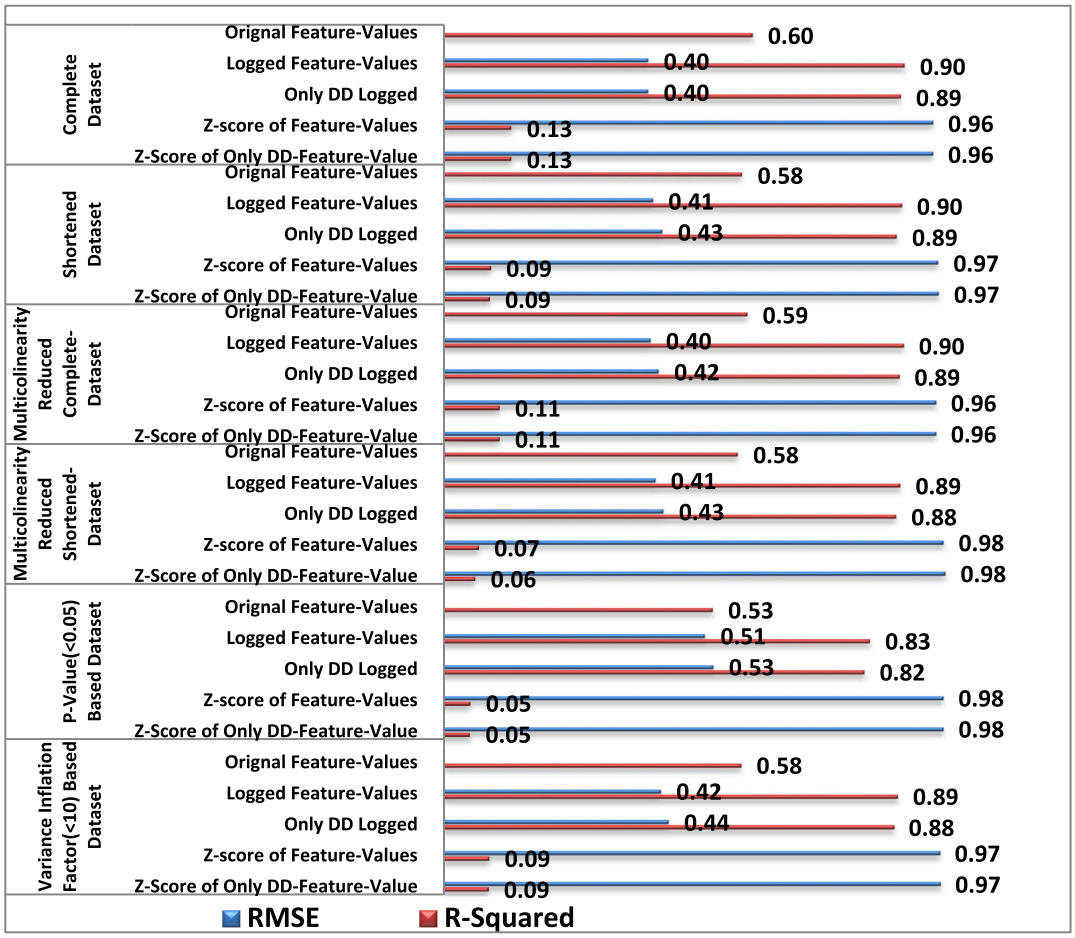

FIGURE 10. Multivariable regression using multiple subsets of a complete dataset.

'CASE tools used', and 'How_methodology_Acquired' predicted logged values of DD equally well as with only all logged values of DD. The logged values of productivity and effort features (e.g., normalized_PDR, effort_specify) predicted DD with RMSE values between 0.5 and 0.56.

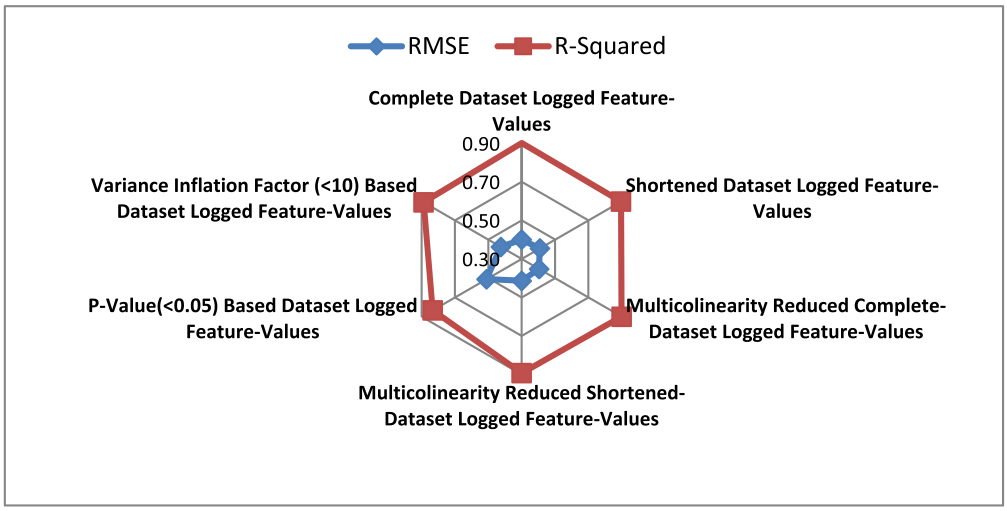

FIGURE 11. Higher results using multiple subsets of the complete dataset.

Overall, the features related to effort and productivity of software development predicted DD best in terms of r-squared and RMSE measures.

Multivariable Regression: This section presents results and analysis of DD prediction using complete datasets and subsets. Figure 10 represents RMSE and r-squared scores of multivariable regressions on the subsets as discussed above. Each subset and dataset are transformed in two ways i.e., either complete data is transformed or only DD (target) feature is transformed. The best prediction performance is achieved with the dataset that is reduced based on a statistical filter of multicollinearity and transformed using logged values as shown in Figure 10. Overall, logged transformations on the datasets enabled the best scores of RMSE in the range (0.4, 0.98) and r-squared values in the range (0.9, 15).

Figure 11 represents higher results of RMSE (i.e., closer to 0) and r-squared (i.e., closer to 1). The best results (with RMSE $=$ 0.4 and r-squared $=$ 0.9) are achieved based on logged feature values of the complete dataset and a multicollinearity-based reduced dataset of 11 features. The logged subsets outperformed the subsets of original values and z-score values. It shows the benefits of feature reduction and data transformations. The prediction of bugs in the planning phase can help project managers make mitigation strategies; however, fewer amounts of data are available for prediction at the start of a project. In that case, requiring three attributes that are related to available project time, the architecture of the project, and the number of resources available can make prediction work convenient.

The VIF, BIC, and multicollinearity-based feature selection methods use the analysis of correlation coefficients among independent variables in different ways. The simplest way of finding multicollinearity among the independent variables is to find a pair-wise correlation among independent variables and their correlation with the target variable as well. The variables having a high correlation with other variables and a lower correlation with the target variable are dropped. The VIF score provides a more structured process of finding multicollinearity as compared to the manual approach because it processes the r-square values of predicting the target attribute. The BIC score is used to predict the best subset of data that can predict the target variable and its criteria is to process the number of parameters and coefficients of prediction. The p-value is calculated by analyzing the coefficients of the regression model through the Wald test. This test is performed with the help of dividing linear regression confidence by its standard errors and later normalizing the results through z-transformation. The P-value for each independent variable informs its level of significance for predicting a dependent variable. The P-value is the probability that the absolute value of a feature is more extreme than the one calculated through the Wald test.

In terms of predicting DD, the subsets of features selected with P-value performed better than the ones selected based on correlation. The fundamental difference between these approaches is multicollinearity analyzes pair-wise correlation coefficients among independent variables as well as with target variables to drop/keep a variable in a subset. The P-value uses analysis of variance between each independent variable and dependent variable. The analysis of training models using RMSE and R2 supports a comprehensive analysis because the residuals and the structure of training models are evaluated through both measures.

IV. CONCLUSION

This study presented an empirical study on early defect prediction to plan quality activities at the beginning of a project to minimize costs of poor quality (that is both internal and external failure costs). The results indicate that software companies may utilize regression and classification-based training models to predict post-deployment defects of a software product at a reasonable accuracy by using the dataset.

The main contribution of this study is twofold: i) a systematic mapping study (SMS) on secondary studies (literature

TABLE 6. Dataset construction for supervised learning (classification).

Findings of the Secondary Studies on Defect Prediction	Reflections on Our Finding
Hosseini et. al [7] reported Bayesian Learners (BL), Logistic Regression (LR), Random Forest (RF), Decision Tree (DT), and Support Vector Machine (SVM) are the most used algorithms. DT, NN, and SVM performed the best on CPDP datasets in terms of F1-measure while LR and RF did not perform as per primary studies. NN, SVM, and DT-based models achieved the highest median F1-score of 0.5 on CPDP datasets.	This study experimented with the most common and most successful machine learning algorithms reported in the literature reviews (Table A-IV, Appendix A). We found that Multilayer Perceptron performed the best with an F1-score of 0.87 using the z-transformation of the complete dataset (Figure 4). The multivariable regression model performed best with the highest scores of RMSE = 0.4 and r-squared = 0.9 by using a subset of 11 features. The subset of features is selected using a statistical filter of multicollinearity and values of the subset are transformed to logged values.
Hosseini et al [7] identified 30 CPDP primary studies and 29 studies only performed binary classification (defective, non-defective). They highlighted a research gap that regression is least performed in CPDP studies.	We performed binary classification and regression analysis as well.
Hosseini et al [7] reported that most studies utilize software measures without feature selection/extraction. They recommended further exploiting and filling this gap.	This study used five different feature selection methods applied to extract subsets of features for classification i.e., Chi-square, Gain ratio, Information gain, Principal component analysis, and Wrapper. The statistical filters of multicollinearity, P-values, variance information factor (VIF), and Bayesian Information Criterion (BIC) are applied to extract subsets of features for regression. In addition, we explored the potential of classifying and estimating DD based on individual features as well. The feature selection methods for the classification approach performed close to the highest prediction results as compared to using the complete dataset. The subset selected based on multicollinearity performed with the highest prediction results for regression.
Hosseini et al [7] recommend hyper-parameter tuning for existing and proposed approaches in CPDP because it is generally proven that it improves the prediction results of machine-learning approaches.	We used ensemble methods for the Decision Tree by using Extra Trees, Random Forests, and Randomized Search CV. The Neural Network is experimented with hyper-parameter tuning. Due to hyperparameter tuning NN performed best among machine learning approaches with an F1-score of 0.89.
NASA (54 percent), Jureczko (65 percent), and Softlab (25 percent) datasets are the most widely used in CPDP studies. Hall et al. [14] recommended that data quality should be seriously considered before conducting a defects prediction study. The majority of datasets are missing information on the quality of data (e.g., NASA dataset). This makes predictions based on these datasets less credible. Therefore, there is a need for datasets with quality information.	The quality information of data is discussed in the 'dataset quality description' section and Table I.
It is recommended to use precision, recall, F1-score, and AUC for CPDP datasets due to data heterogeneity and imbalanced class issues [7].	The complete results based on these measures are reported online (Appendix B). The f1-measure that is based on precision and recall is presented and discussed in this study. The AUC is used and it assures that the results are not affected due to the heterogeneity of data and class imbalance issues.
The CPDP studies use defects count and features/attributes that are collected at the method, class, file, and package module level within a coding phase. Therefore, these studies predict defects at the code level.	The ISBSG measures/attributes are collected during the completion of software projects and defects are counted after one month of deployment of each product. This study predicts post-deployment defects at the product level.
The Z-score normalization and log transformation are widely used in CPDP studies to minimize the effect of skewness and heterogeneity of data.	This study applied both transformations on the datasets before training regression and classification models. The z-score transformations improved results for classification models and log transformations improved results for regression models. Overall, the training models performed better on the transformations than the original values.
Hosseini et al [7] reported prediction models built upon a combination of different types of process and product metrics to perform the best. Most of the studies used a combination of traditional metrics (size and complexity metrics), process metrics (code delta, code churn), and Object-Oriented metrics.	We found in our SMS that resource-related measures (e.g., team size, and productivity) are rare in the CPDP datasets. In this study, we used a total of 23 attributes/measures which are distributed among processes (6), products (11), and resources (6) types of metrics.

reviews) on CPDP (cross-project defect prediction) and ii) an empirical study on early defects prediction using a publicly available benchmarking dataset. In Table 6, we provide our reflections on the findings of this study by comparing them to the findings of recent SLR and meta-analysis on CPDP [7] and SLR of Hall et al. [14].

In the future, we are planning to further extend the study by exploring deep learning and transformer-based training models to predict defects. The software development organization can use the findings of this study and predict their defects by using the source code of training models in this study. The source code is given in Appendix B. In addition, the ISBSG cross-project dataset is further extendible through the addition of their in-house project data.

APPENDIX A
SMS ON SECONDARY STUDIES

Here, we present a summary of our SMS on secondary studies (i.e., literature reviews (LRs), systematic literature reviews (SLRs), and SMSs) on defect prediction. These studies reviewed in total of 509 defect prediction studies published between 1990 and 2019. The research questions

TABLE 7. Research questions of the SMS.

	Research Question	Motivation
RQ1	What are the most used machine learning approaches in defect prediction studies?	To find and apply the most commonly used machine learning approaches on the ISBSG dataset.
RQ2	What datasets are used for cross-project defect prediction studies, and which relevant attributes/measures are collected in these datasets?	To identify types of attributes/measures used among the datasets.

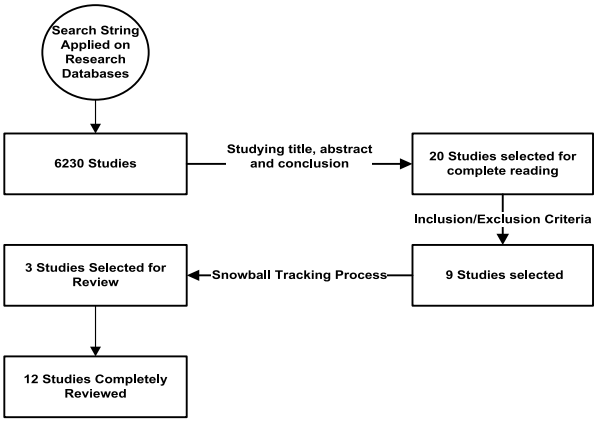

FIGURE 12. The selection process of secondary studies.

TABLE 8. Selection of studies.

Research Databases	No. of Studies Extracted [Search Result]	No. of Selected Studies
Scopus	45	1
Web of Sciences	4	1
Springer	2619	1
IEEE Xplore	325	3
ACM	122	0
Science Direct	3115	6

(RQ) of this study are shown in Table 7. The key findings of SMS are presented and compared with the results and findings of this study in Section III.

A. SEARCH PROCESS

1) SEARCH STRING 1

(Literature review OR systematic review OR systematic literature review OR SLR OR Systematic mapping OR systematic mapping study) AND (Defect OR Fault OR Error OR Failure) AND (Prediction OR estimation).

2) SEARCH STRING 2

'ISBSG AND (Defect OR Fault OR Error OR Failure) AND (Prediction OR estimation)' and within the SLRs with an objective to identify the use of the ISBSG dataset for defects prediction. We did not find a study that used the ISBSG dataset for defect predictions at a project level.

B. STUDY INCLUSION CRITERIA

We have selected LRs, SLRs, and SMSs on defect predictions which are published in peer-reviewed journals. A study is selected if it at least has an analysis of types of machine learning algorithms, datasets, or metrics used in the defect prediction studies.

TABLE 9. The list of secondary studies included.

Reference	No. of Studies	Coverage Time
[11]	106	1991 – 2011
[12]	64	1991 – 2013
[7]	30	2007 – 2019
[6]	50	2006 – 2015
[14]	208	2000 – 2010
[34]	156	1995 – 2018
[35]	52	2000 – 2016
[36]	29	1993 – 2016
[33]	90	1990 – 2009
[40]	118	2010 – 2018
[41]	74	1990 – 2007
[42]	71	2000 – 2013

TABLE 10. Machine learning methods used in the list of secondary studies included.

Ref	Machine Learning Methods (Top 5)							TS
	ANN	BL	DT	EL	LR	RF	SVM	
[12]	✓ (26%)	✓ (47%)	✓ (47%)	✓ (18%)			✓ (27%)	64
[7]		✓ (43%)	✓ (16%)		✓ (32%)	✓ (16%)	✓ (16%)	30
[6]		✓ (48%)	✓ (22%)		✓ (50%)	✓ (32%)	✓ (26%)	50
[14]		✓	✓		✓	✓	✓	36
[34]	✓	✓	✓		✓		✓	156
[36]	✓	✓	✓		✓		✓	12
[33]	✓	✓	✓		✓			90
[42]	✓ (12%)	✓ (19%)	✓ (15%)		✓ (7%)	✓ (8%)		71

ANN: Artificial Neural Network, BL: Bayesian Learners, DT: Decision Tree, EL: Ensemble Learner, LR: Logistic Regression, RF: Random Forest, SVM: Support Vector Machine, and TS: Total Studies.

C. SEARCHING ACTIVITIES

Figure 12 represents an overall process that is used for the selection of secondary studies. First, we selected 20 studies based on title, abstract, and conclusion for the complete review and evaluated them according to inclusion criteria. The complete review resulted in a selection of 9 studies. The snowball tracking process identified 3 more studies.

1) SEARCH RESULTS

Table 8 represents the research databases, search results, and the number of SLR, SMS, and LR studies selected after applying the search string. Table 9 represents the title, coverage time, and number of studies included.

D. RESULTS OF THE SMS

Table 10 presents the most used machine learning methods. Few studies explicitly mentioned the percentage of machine learning methods used. In the other studies, we have identified the most used algorithms based on the titles of primary studies and overall discussion in the SLRs, LRs, and SMSs. The methods mentioned against the Study IDs in Table 10 are according to their frequency of use in the research studies or frequency of appearance in the SLRs, LRs,

TABLE 11. Most used defect prediction datasets and metrics for defect prediction.

Ref	Datasets Repositories	Metrics Type	Prediction	Projects Count	No. of Metrics or Attributes	Programming Language
[6], [7]	MDP-NASA	Product: Object Oriented (OO), Source Code Metrics (SCM) Process: Code Delta, Code Churn	Binary Classification	11 Projects	39	C++, Java
	SOFLAB-Turkish company	Product: OO, Static Code Metrics	Binary Classification	3 Projects	30	C
	Jureczko and Madeyski	Product: OO	Regression	92 Products	20	Java
	Relink (JUR)	Product: OO	Binary Classification	3 Projects	20	
	AEEM	Product: OO	Binary Classification	5 Projects	61	Java
	MOCKUS	Product: Static Code Metrics Process: Metrics related to actions performed on Github	Binary Classification	1385 Projects	21	Multiple
	NETGENE	Product: Static Code Metrics, OO, Code dependency metrics	Binary Classification	4 Projects	465	
	AUDI Electronics	Product: Static Code Metrics	Binary Classification	3 Projects	17	C
[11], [12], [14], [33], [35], [36], [41]	ECLIPSE	Product: Static Code Metrics, OO	Eclipse	3 Releases	71	Java
	PROMISE	Product: Static Code Metrics, OO	Binary Classification	10 Projects	22	C, C++
[34]	Mozilla	Process: Temporal activities from bug reporting to fixing. Resource: Committer experience	Binary Classification	Mozilla Releases (2006 – 2012)	12	Ruby
	Apache	Process: Temporal activities from bug reporting to fixing.	Binary Classification	Apache Releases (2004 – 2008)		C

TABLE 12. Description of dataset attributes/metrics.

Feature Label	Description
Functional_Size	The unadjusted function point count (before any adjustment by a Value Adjustment Factor if used)
Adjusted_Function_Points	For IFPUG, NESMA, FiSMA, and MARK II counts this is the adjusted size (the functional size is adjusted by a Value Adjustment Factor).
Value_Adjustment_Factor	The adjustment to the function points, applied by the project submitter, takes into account various technical and quality characteristics
Normalised_Work_Effort_Level 1	The development team's full life-cycle effort.
Normalised_Work_Effort	Full life-cycle effort for all teams reported.
Normalised_Level_1_PDR	Project productivity delivery rate in hours per functional size unit calculated from Normalized Level 1 Work Effort for the development team only divided by Functional Size (Unadjusted Function Points)
Normalised_PDR	Project productivity delivery rate in hours per functional size unit calculated from Normalized Work Effort divided by the Functional Size (Unadjusted Function Point count).
Speed_of_Delivery	Functional Size Units per person per elapsed month calculated as Functional Size / Project Elapsed Time * Max Team Size
Project_Elapsed_Time	Total elapsed time for the project in calendar months.
Effort_Plan	Breakdown of Summary Work Effort reported for Plan phase within the range: Plan, Specify, Design, Build, Test, and Implement.
Effort_Specify	Breakdown of Summary Work Effort reported for Specify phase within the range: Plan, Specify, Design, Build, Test, and Implement.
Effort_Design	Breakdown of Summary Work Effort reported for the Design phase within the range: Plan, Specify, Design, Build, Test, and Implement.
Effort_Build	Breakdown of Summary Work Effort reported for the Build phase within the range: Plan, Specify, Design, Build, Test, and Implement.
Effort_Test	Breakdown of Summary Work Effort reported for the Test phase within the range: Plan, Specify, Design, Build, Test, and Implement.
Effort_Implement	Breakdown of the Summary Work Effort reported for the Implementation phase within the range: Plan, Specify, Design, Build, Test and Implement.
Effort_Unphased	Where no phase breakdown is provided in the submission, this field contains the same value as the Summary Work Effort.
Development_Type	This field describes whether the development was a new development, enhancement, or redevelopment.
Client_Server	Indicator of whether the application or product requires more than one computer to operate different components or parts of it.
Development_Platform	Defines the primary development platform, (as determined by the operating system used). Each project is classified as PC, Mid-Range, Main Frame, or Multi-platform.
Language_Type	Defines the language type used for the project.
CASE_Tool_Used	Whether the project used any CASE tool.
How_Methodology_Acquired	Describes whether the development methodology was purchased or developed in-house, or a combination of these.
Average_Team_Size	The average number of people that worked on the project, (calculated where available from the team sizes per phase).
Defect Density (DD)	Defects per 1000 FP calculated as Total Defects Delivered * 1000 / Functional Size.

TABLE 13. Feature selection methods applied on the dataset for classification models.

Subsets Selection Method	Feature Selection	Selected Features / Attributes	Number of Features (N)
Filter	Chi-Square	Functional_Size, Adjusted_Function_Points, Value_Adjustment_Factor, Normalised_Work_Effort_Level 1, Normalised_Work_Effort, Effort_Plan, Effort_Build, Effort_Test, Effort_Implement, Development_Type, Client_Server, CASE_Tool_Used	12
Embedded	Gain Ratio	Functional_Size, Adjusted_Function_Points, Normalised_Work_Effort_Level 1, Project_Elapsed_Time, Effort_Plan, Effort_Specify, Effort_Design, Effort_Build, Effort_Test, Effort_Implement, Development_Type, Client_Server, CASE_Tool_Used, How_Methodology_Acquired,Average_Team_Size	15
Filter	Information Gain	Adjusted_Function_Points, Normalized_Work Effort_Level 1, Project_Elapsed_Time, Normalized_Work_Effort, Development_Platform, Value_Adjustment_Factor, Development_Type, Effort_Build, Client_Server, Case_Tools, Language_Type, How_Methodology_Acquired, Effort_Design	14
Embedded	Principal Component Analysis	Functional_Size Adjusted_Function_Points, Value_Adjustment_Factor, Normalised_Work_Effort_Level 1, Normalised_Work_Effort, Normalised_Level_1_PDR, Normalised_PDR, Speed_of_Delivery, Project_Elapsed_Time, Effort_Plan, Effort_Specify, Effort_Design, Effort_Build, Effort_Test, Effort_Implement	15
Wrapper	Wrapper	Project_Elapsed _Time, Effort_Specification, Effort_Testing, Effort_Implementation, Development_Type, Client_Server, Development_Platform	6

TABLE 14. Feature selection methods applied on the dataset for regression models.

Subsets Selection Method	Selected Features / Attributes	Number of Features (N)
Multicollinearity	Functional_Size, Adjusted_Function_Points, Value_Adjustment_Factor, Normalised_Level_1_PDR, Speed_of_Delivery, Project_Elapsed_Time, Effort_Plan, Effort_Specify, Effort_Design, Effort_Build, Effort_Unphased	11
Variance Inflation Factor	Speed_of_delivery, Project_Elapsed_Time, Effort_Plan,Effort_Specify, Effort_Design, Effort_Implement, Effort_Unphased, Development_Type, Client_Server, Development_Platform, Language_Type,CASE_Tool_Used, How_Methodology_Acquired, Average_Team_Size	14
P-Value	Speed_of_delivery, Client_server, average_team_size	3
Adding effort-related attributes as "total effort"	Functional_Size, Adjusted_Function_Points, Value_Adjustment_Factor, Normalised_Work_Effort_Level 1, Normalised_Work_Effort, Normalised_Level_1_PDR, Normalised_PDR, Speed_of_Delivery, Project_Elapsed_Time, Total_Effort, Development_Type, Client_Server, Development_Platform, Language_Type, CASE_Tool_Used, How_Methodology_Acquired, Average_Team_Size	17

and SMSs. According to the findings, Bayesian Learners, Regression, Random Forests, Decision Trees and Support Vector Machines, Rule-based methods, and Artificial Neural Networks are the most used methods.

Table 11 presents the most used CPDP datasets that are publicly available and attributes/measures collected about process, product, and resource entities. NASA MDP, ECLIPSE, PROMISE, Mozilla, Apache, Jureczko, and Softlab are found to be the most used. It is observed that resource-related measures are the least used ones.

APPENDIX B
CODE AND RESULTS

The code and detailed results are available at https://drive.google.com/drive/folders/174ER7vzZxw2onDArqxl7yw2nFyEPkxAW?usp=sharing

APPENDIX C
DATASET ATTRIBUTES AND DETAILS

Table 12 lists the attributes or metrics' description of the dataset. Tables 13 and 14 present subsets of features selected for classification and regression models.

DECLARATION OF COMPETING INTEREST

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

Data sharing does not apply to this article, as no new datasets were generated or analyzed during the current study.

REFERENCES

[1] *A Guide to the Project Management Body of Knowledge PMBOK Guide Seventh Edition and The Standard for Project Management*, 7th ed. PA, USA: Project Manag. Inst., 2021.

[2] R. Pietrantuono, P. Potena, A. Pecchia, D. Rodriguez, S. Russo, and L. Fernández-Sanz, "Multiobjective testing resource allocation under uncertainty," *IEEE Trans. Evol. Comput.*, vol. 22, no. 3, pp. 347–362, Jun. 2018, doi: 10.1109/TEVC.2017.2691060.

[3] T. DeMarco, *Controlling Software Projects: Management, Measurement, and Estimates*. Upper Saddle River, NJ, USA: Prentice-Hall, 1986.

[4] B. W. Boehm and P. N. Papaccio, "Understanding and controlling software costs," *IEEE Trans. Softw. Eng.*, vol. 14, no. 10, pp. 1462–1477, Oct. 1988, doi: 10.1109/32.6191.

[5] B. Turhan, T. Menzies, A. B. Bener, and J. Di Stefano, "On the relative value of cross-company and within-company data for defect prediction," *Empirical Softw. Eng.*, vol. 14, no. 5, pp. 540–578, Oct. 2009, doi: 10.1007/s10664-008-9103-7.

[6] S. Herbold, "A systematic mapping study on cross-project defect prediction," 2017, *arXiv:1705.06429*.

[7] S. Hosseini, B. Turhan, and D. Gunarathna, "A systematic literature review and meta-analysis on cross project defect prediction," *IEEE Trans. Softw. Eng.*, vol. 45, no. 2, pp. 111–147, Feb. 2019, doi: 10.1109/TSE.2017.2770124.

[8] C. Jones, *Estimating Software Costs: Bringing Realism to Estimating*, 2nd ed. New York, NY, USA: McGraw-Hill, 2007. [Online]. Available: http://www.amazon.com/Estimating-Software-Costs-Bringing-Realism/dp/0071483004

[9] R. Özakinci and A. K. Tarhan, "A decision analysis approach for selecting software defect prediction method in the early phases," *Softw. Quality J.*, vol. 31, no. 1, pp. 121–177, Mar. 2023, doi: 10.1007/s11219-022-09595-0.

[10] B. Turhan, G. Kocak, and A. Bener, "Data mining source code for locating software bugs: A case study in telecommunication industry," *Ex. Syst. Appl.*, vol. 36, no. 6, pp. 9986–9990, Aug. 2009, doi: 10.1016/j.eswa.2008.12.028.

[11] D. Radjenović, M. Hericko, R. Torkar, and A. Živkovic, "Software fault prediction metrics: A systematic literature review," *Inf. Softw. Technol.*, vol. 55, no. 8, pp. 1397–1418, Aug. 2013, doi: 10.1016/j.infsof.2013.02.009.

[12] R. Malhotra, "A systematic review of machine learning techniques for software fault prediction," *Appl. Soft Comput.*, vol. 27, pp. 504–518, Feb. 2015, doi: 10.1016/j.asoc.2014.11.023.

[13] J. S. Shirabad and T. J. Menzies. (2005). *The PROMISE Repository of Software Engineering Databases*. [Online]. Available: http://promise.site.uottawa.ca/SERepository

[14] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Trans. Softw. Eng.*, vol. 38, no. 6, pp. 1276–1304, Nov. 2012, doi: 10.1109/TSE.2011.103.

[15] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques*, 3rd ed. Amsterdam, The Netherlands: Elsevier, 2011, doi: 10.1016/C2009-0-19715-5.

[16] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, 1997.

[17] J. J. Cuadrado-Gallego, L. Buglione, M. J. Domínguez-Alda, M. F. D. Sevilla, J. Antonio Gutierrez de Mesa, and O. Demirors, "An experimental study on the conversion between IFPUG and COSMIC functional size measurement units," *Inf. Softw. Technol.*, vol. 52, no. 3, pp. 347–357, Mar. 2010, doi: 10.1016/j.infsof.2009.12.001.

[18] Z. Abedjan, L. Golab, F. Naumann, and T. Papenbrock, *Data Profiling*. Cham, Switzerland: Springer, 2019, doi: 10.1007/978-3-031-01865-7.

[19] W. Fan and F. Geerts, *Foundations of Data Quality Management*. Cham, Switzerland: Springer, 2012, doi: 10.1007/978-3-031-01892-3.

[20] T. Tahir, G. Rasool, and M. Noman, "A systematic mapping study on software measurement programs in SMEs," *E-Inform. Softw. Eng. J.*, vol. 12, no. 1, pp. 133–165, 2018, doi: 10.5277/e-Inf180106.

[21] T. Tahir, G. Rasool, and C. Gencel, "A systematic literature review on software measurement programs," *Inf. Softw. Technol.*, vol. 73, pp. 101–121, May 2016, doi: 10.1016/j.infsof.2016.01.014.

[22] O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman, "Missing value estimation methods for DNA microarrays," *Bioinformatics*, vol. 17, no. 6, pp. 520–525, Jun. 2001, doi: 10.1093/bioinformatics/17.6.520.

[23] D. Zhang and J. J. P. Tsai, "Machine learning and software engineering," *Softw. Quality J.*, vol. 11, no. 2, pp. 87–119, 2003, doi: 10.1023/A:1023760326768.

[24] Y. Zhang, D. Lo, X. Xia, and J. Sun, "An empirical study of classifier combination for cross-project defect prediction," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 264–269, doi: 10.1109/COMPSAC.2015.58.

[25] P. He, B. Li, X. Liu, J. Chen, and Y. Ma, "An empirical study on software defect prediction with a simplified metric set," *Inf. Softw. Technol.*, vol. 59, pp. 170–190, Mar. 2015, doi: 10.1016/j.infsof.2014.11.006.

[26] J. Nam and S. Kim, "Heterogeneous defect prediction," in *Proc. 10th Joint Meeting Found. Softw. Eng.*, Aug. 2015, pp. 508–519, doi: 10.1145/2786805.2786814.

[27] A. Meneely, B. Smith, and L. Williams, "Validating software metrics," *ACM Trans. Softw. Eng. Methodol.*, vol. 21, no. 4, pp. 1–28, Nov. 2012, doi: 10.1145/2377656.2377661.

[28] V. Bolón-Canedo, I. Porto-Díaz, N. Sánchez-Maroño, and A. Alonso-Betanzos, "A framework for cost-based feature selection," *Pattern Recognit.*, vol. 47, no. 7, pp. 2481–2489, Jul. 2014, doi: 10.1016/j.patcog.2014.01.008.

[29] B. Ratner, "The correlation coefficient: Its values range between +1/−1, or do they?" *J. Targeting, Meas. Anal. Marketing*, vol. 17, no. 2, pp. 139–142, Jun. 2009, doi: 10.1057/jt.2009.5.

[30] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, vol. 44, no. 2, 6th ed. Hoboken, NJ, USA: Wiley, 2021.

[31] D. A. Belsley, E. Kuh, and R. E. Welsch, *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. Hoboken, NJ, USA: Wiley, 1980, doi: 10.1002/0471725153.

[32] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: MIT Press, 2012.

[33] C. Catal, "Software fault prediction: A literature review and current trends," *Exp. Syst. Appl.*, vol. 38, no. 4, pp. 4626–4636, Apr. 2011, doi: 10.1016/j.eswa.2010.10.024.

[34] L. Son, N. Pritam, M. Khari, R. Kumar, P. Phuong, and P. Thong, "Empirical study of software defect prediction: A systematic mapping," *Symmetry*, vol. 11, no. 2, p. 212, Feb. 2019, doi: 10.3390/sym11020212.

[35] R. Özakinci and A. Tarhan, "Early software defect prediction: A systematic map and review," *J. Syst. Softw.*, vol. 144, pp. 216–239, Oct. 2018, doi: 10.1016/j.jss.2018.06.025.

[36] S. S. Rathore and S. Kumar, "A study on software fault prediction techniques," *Artif. Intell. Rev.*, vol. 51, no. 2, pp. 255–327, Feb. 2019, doi: 10.1007/s10462-017-9563-5.

[37] E. Alpaydin, *Introduction to Machine Learning*. Cambridge, MA, USA: MIT Press, 2020.

[38] Y. Ma and B. Cukic, "Adequate and precise evaluation of quality models in software engineering studies," in *Proc. 3rd Int. Workshop Predictor Models Software Eng.*, May 2007, p. 1, doi: 10.1109/PROMISE.2007.1.

[39] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006, doi: 10.1016/j.patrec.2005.10.010.

[40] E. A. Felix and S. P. Lee, "Systematic literature review of preprocessing techniques for imbalanced data," *IET Softw.*, vol. 13, no. 6, pp. 479–496, Dec. 2019, doi: 10.1049/iet-sen.2018.5193.

[41] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Exp. Syst. Appl.*, vol. 36, no. 4, pp. 7346–7354, May 2009, doi: 10.1016/j.eswa.2008.10.027.

[42] R. S. Wahono, "A systematic literature review of software defect prediction: Research trends, datasets, methods and frameworks," *J. Softw. Eng.*, vol. 1, no. 1, pp. 1–16, 2018.

TOUSEEF TAHIR received the master's degree in software engineering from BTH, Sweden, and the Ph.D. degree in computer science from COMSATS University Islamabad, Pakistan. He has been a Software-Engineering Researcher and an Educator for over ten years. Currently, he is an Assistant Professor with COMSATS University Islamabad. His research interests include data-driven software engineering involving machine learning, deep learning, natural language processing, and empirical case studies to improve software project management. He has experience teaching introductory and specialized courses spanning across complete software development life cycles.

CIGDEM GENCEL received the Ph.D. degree from Middle East Technical University, Turkey. She is currently an Associate Professor in computer engineering. After receiving the Ph.D. degree, she was with the Blekinge Institute of Technology, Sweden, and then the Free University of Bolzano, Italy. She is also with Ankara Medipol University, Turkey. Her research has provided novel solutions to real industrial challenges. In particular, she has been involved in empirical research studies in academy-industry collaboration contexts. Her research interests include software size and project effort estimating, software measurement process, empirical software engineering, software project management, and software quality.

GHULAM RASOOL received the M.Sc. degree in computer science from BZU, Multan, Pakistan, in 1998, the M.S.C.S. degree from the University of Lahore, Pakistan, in 2008, and the Ph.D. degree in reverse engineering from the Technical University of Ilmenauu, Germany, in 2011. He is currently an Associate Professor with COMSATS University Islamabad, Lahore Campus. He has more than 22 years of teaching and research experience at national and international levels. His research interests include reverse engineering, design patterns, antipattern recovery, program comprehension, and source code analysis.

TARIQ UMER (Senior Member, IEEE) received the master's degree in computer science from Bahauddin Zakariya University, Multan, Pakistan, in 1997, and the Ph.D. degree in communication systems from the School of Computing & Communication, Lancaster University, U.K., in 2012.

He served in the IT education sector in Pakistan for more than 13 years. Since January 2007, he has been an Assistant Professor with the CS Department, COMSATS University Islamabad, Lahore Campus, Lahore. His research interests include vehicular ad-hoc networks, the Internet of Things, wireless sensor networks, and telecommunication network design. He served in the TPC for various international conferences. He is currently an Editor of *Future Generation Computer Systems* (Elsevier) and an Associate Editor of IEEE Access. He is serving as a Reviewer for IEEE Communications Letters, IEEE Access, *Computers and Electrical Engineering* (Elsevier), *Journal of Network and Computer Applications* (Elsevier), *Ad Hoc Sensor Wireless Networks*, *Wireless Networks* (Springer) journal, and the *Journal of Communications and Networks*. He is an active member of the Pakistan Computer Society and Internet Society Pakistan.

JAWAD RASHEED (Member, IEEE) received the B.S. degree in telecommunication engineering from the National University of Computer and Emerging Sciences, Pakistan, the M.S. degree in electrical and electronics engineering in Turkey, and the Ph.D. degree in computer engineering in Turkey. He is currently an Associate Professor with Istanbul Sabahattin Zaim University, Turkey. He is also a Senior Researcher with the Deep Learning and Medical Image Analysis Laboratory, Boğaziçi University, Turkey, and a Research Fellow with Istanbul Nisantasi University, Turkey. He is the author/coauthor of more than 50 articles published in well-reputed journals and highly-ranked conferences. His research interests include artificial intelligence and image processing, pattern recognition, the IoT, and data analytics.

He was a Gold Medalist and was awarded the Academic Excellence Award for securing straight A's in O' Level exams held by Cambridge University. Later, he also received a prestigious Doctorate and Research Scholarship for the Ph.D. studies (for three years). He serves as a guest/lead-guest/topic editor for special issues at the *Symmetry*, *Mathematics*, *Healthcare*, *Applied Sciences*, *Electronics*, and *Journal of Sensor and Actuator Networks*. Recently, he served as a book editor for *Lecture Notes on Data Engineering and Communications Technologies* (Springer) (Forthcoming Networks and Sustainability in the IoT Era). In addition, he is the General Chair of IEEE ICAIoT and IEEE/Springer FoNeS-AIoT and chairs the Technical Program Committee of Springer FoNeS-IoT 2021.

SOOK FERN YEO received the Bachelor of Business Administration degree (Hons.) in management, the Master of Business Administration degree in multimedia marketing from Multimedia University, Malaysia, and the Doctor of Philosophy (Ph.D.) degree in service innovation from Universiti Sains Malaysia (USM), Malaysia. She is currently the Deputy Dean of Research & Industrial Collaborations with Multimedia University. Moreover, she is also a Lecturer with the Faculty of Business, where she teaches fundamental of marketing, understanding management, service marketing, brand management, and strategic marketing at the undergraduate level. Her current research interests include social media marketing, consumer behavior, service marketing, branding, and international business. Before joining MMU, she was a Program Coordinator with the School of Secretarial and Administrative Studies (2000–2009) and later a Program Coordinator with the School of Business (2009–2010), Stamford College Group. She has over 20 years of experience as an Academician.

TANER CEVIK received the B.Sc. degree in computer engineering from Istanbul Technical University, İstanbul, in 2001, and the Ph.D. degree from Istanbul University, in 2012. He joined the Department of Computer Engineering, Istanbul Arel University, in 2023, and continues to work as a Professor. His research interests include image processing, machine learning, and wireless communications.