Received 28 July 2022, accepted 12 August 2022, date of publication 18 August 2022, date of current version 1 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3200149

RESEARCH ARTICLE

Adaptive Discretization Using Golden Section to Aid Outlier Detection for Software Development Effort Estimation

SWARNIMA SINGH GAUTAM AND VRIJENDRA SINGH, (Senior Member, IEEE)
Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj 211015, India

Corresponding author: Vrijendra Singh (vrij@iiita.ac.in)

This work was supported by Ministry of Education, Government of India through Indian Institute of Information Technology Allahabad.

ABSTRACT The software engineering researchers have worked on different dimensions to facilitate better software effort estimates, including those focusing on dataset quality improvement. In this research, we specially investigated the effectiveness of outlier removal to improve estimation performance of 5 machine learning (ML) methods (Support Vector Regression, Random Forest, Ridge Regression, K-Nearest Neighbor, and Gradient Boosting Machines) for software development effort estimation (SDEE). We propose a novel discretization method based on Golden Section (dubbed as Golden Section based Adaptive Discretization, GSAD) to identify optimal number of outliers for SDEE dataset. The results signify the importance of optimal number of outliers' removal to improve estimations. Moreover, the results obtained after applying GSAD technique have been compared with IQR and Cooks' distance based outlier identification methods over 4 datasets: ISBSG Release 2021, UCP, NASA93 and China. The empirical results confirm that the performance of ML based SDEE methods is generally improving by employing GSAD and the proposed GSAD method has the ability to compete with the other prevalent outlier identification methods.

INDEX TERMS Software development effort estimation, machine learning, discretization, outlier identification, golden section method.

I. INTRODUCTION

The need to have utmost accuracy in estimated effort has become reign supreme for software development industry to better support the decision making process. Both overestimation and underestimation of required effort are highly undesirable and occurrence of these can cause failure of a software project and resource wastage [1], [2]. To achieve accurate effort estimates for the proposed software project, it is required to build SDEE model using historical projects' datasets. The quality of dataset affects estimation accuracy and reliability of empirical research, thus it is suggested that improving dataset quality can be an important factor in improving accuracy of SDEE models and it can facilitate more clarity to project managers to take better decisions [1], [3], [4], [5], [6], [7], [8].

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

Outlier handling is one of the important pre-processing task not only just to improve the quality of data but also for the reliability of model generated using that data, since SDEE datasets greatly suffer due to outliers [8], [9], [10], [11], [12]. The presence of outliers may be the indication of: error in reporting the measurements; since at times a project stakeholder (more specifically development team member) works on more than one project at the same time so it becomes difficult to keep track of the correct measurements, lack in project continuity and/ or other unstable environmental factors during project development. Therefore, the identification of outliers is essential before building an estimation model in order to ensure the reliability of the model.

The outlier identification is relatively less researched in the field of SDEE. No research has been performed to consider the extent of outlierness of data points for the identification of optimal number of outliers and the extent to which they influence model's performance. In this research work, we aim to introduce a novel method called Golden Section based Adaptive Discretization (GSAD) to identify outlier data points. GSAD discretizes the dataset using Golden Section to partition each feature domain and subsequently finds the data points which are very distant from the other data points. It further cumulates the distant data points of each feature to come up with true outlier data points. The idea behind using the discretization method for outlier identification is that discretization leads to finite set of non-overlapping partitions. Since discretization not only aims to improve the efficiency in a particular field to wherever it is applied as a pre-processing step but the very basis of discretization is to represent the information in a more compact form. Thus, a discretization method can be capable of providing a set of data points which do not comply with the compact representation of a particular dataset.

In the conventional major discretization methods that follow equal-width or equal-frequency concept, all occurrences of a repetitive value may belong to different interval. This conception may not be useful in facilitating best results for every case, specifically where the continuous attributes are not uniformly distributed. Whereas, the proposed GSAD method places all repetitive entries of a value in the same interval which is beneficial for identifying distant values in a dataset.

SDEE has been one of the most widely researched domain of study in the field of software engineering. Several studies have carried out vast review of the promising work done by researchers in the past [13], [14], [15]. As per these review studies, some researchers assume expert based SDEE methods to be better than the model based SDEE, some favour model based estimation while some found no difference in them [16]. As per a comparative study done in [17], nine studies found analogy-based SDEE model outperforming regression based SDEE, while four studies found regression based SDEE model to be the best performing. The enormous amount of work has been done in the domain of SDEE to assist continually growing software development industry. But, it is also prominent that studies often affirm the findings by relying on limited experimental settings, which also emanates to a very crucial "conclusion instability" problem [18], [19].

The reliability of comparative analysis of models' performance broadly depends on following factors:

1) Dataset characteristics
2) Design of experiments; more specifically validation method and data splits [18], [20], [21]
3) Specific aspect of accuracy; more specifically different evaluation criteria [22], [23], [24]

The proposed outlier identification method has been evaluated on five machine learning methods using four widely acknowledged benchmark SDEE datasets: NASA93, China, UCP and ISBSG release 2021. We have analysed and evaluated model performances with systematically designed experimental settings so as to alleviate the conclusion instability concern. The novel contributions of this paper are two folds:

1) A novel method named GSAD is introduced to identify outliers in SDEE dataset based on Golden Section method.
2) Additionally, in the literature the concept of outlierness has never been studied for the purpose of outlier identification in the field of software development effort estimation. In order to detect optimal number of outliers, we have conceptualized the identification of outlierness of each data point and the effect of removing those data points on the basis of level of outlierness.

With the consideration of all the aforementioned challenges and the objectives of this study, this paper more specifically investigates following research questions:

RQ1. **How to use golden section based discretization method to detect optimal number of outliers in SDEE data, which is one of the factors associated with the data quality in SDEE?**

RQ2. **Can our golden section based outlier identification method help in improving the performance of ML based SDEE method that does not consider the presence of outliers?**

RQ3. **How the performance of ML based SDEE method varies using our proposed outlier identification method in comparison to other prevalent outlier identification methods?**

The prime focus of this paper is to provide detailed insight on how to use golden section method to deal with the previously mentioned challenges and to further validate golden section-based approach with reference to outlier identification to improve effort estimation. To the best of our knowledge, Golden Section method has never been explored to discretize dataset for the purpose of outlier identification in the field of SDEE.

The organization of this paper is as follows: section II discusses about the most relevant selected studies that form the basis of this research work. Thereafter, section III discusses the background techniques. Section IV reflects upon the experimental design including dataset description, the proposed outlier identification method, validation and accuracy measures used in this study. This is followed by section V which elaborates results and discussion. Further, section VI discusses about the threats to validity. Finally, section VII outlines general conclusion and gives direction for future research.

II. RELATED WORK

This section provides the review of studies done in the the field of SDEE with a focus on outlier identification. Keung *et al.* [25] have applied Mantel's correlation to assess the dataset quality and identify outliers. The major attention in their proposed method is that it can specifically and only be used for analogy based estimation.

Chan *et al.* [26] have proposed least median squares based statistical methodology to identify and eliminate outliers.

Their proposed method has two major concerns: they have used only MMRE as evaluation criteria, which is considered highly biased [27], [28], [29], [30], also their proposed approach is dependent of dataset distribution. While considering the concerns of approach proposed in [26], Seo *et al.* [3] have proposed two approaches: statistics based least trimmed square and data mining based K-means to identify outliers. Their evaluation criteria were also only MRE based measures which is considered sensitive to high MRE values [27], [31].

In another study, Seo *et al.* [1] have explored least trimmed squares, Boxplot, K-means clustering, Mantel leverage, and Cook's distance to study the influence of outliers. They observed that initially the experimental results found no noticeable improvements with outlier removal but statistical analysis depicted significant improvement with the removal of outliers on some datasets. Further, they suggested that outlier removal has the potential to improve the likelihood of better estimates.

Kocaguneli *et al.* [5] have proposed QUICK method to filter out the outliers present in the dataset. Their approach relied on Euclidean distance to prune out unpopular rows as outliers. They further advocated that essential information can be represented with reduced content in SDEE datasets. In a recent study, Silhavy *et al.* [32] have explored the process of outlier identification using relative percentage error, median absolute deviation (MAD), and inter-quartile range (IQR). They further identified that MAD based approach has best performance when being used with stepwise model. In a study [33], Nassif *et al.* have used IQR to identify outliers before building fuzzy regression models. This study also recommended to remove outliers for the purpose of improving prediction performance of estimation models.

All the previous researches identify outliers using either a single attribute such as 'effort' or consider all the points that are identified as outliers using each individual attribute of dataset. We propose GSAD to determine the outlierness of each data point based on groups of outlier positive attributes. As a data point may be a potential outlier for one or more attributes. If we consider all such data points as true outliers that are potential outlier to each individual attribute, it may classify a large portion of dataset as final outliers. Whereas, it is not sufficient to use any single specific attribute such as 'effort' to identify final set of outliers because there may be some other important attributes such as 'size' or some productivity influencing attributes [13] that may also contain outlier values. Unlike the previous outlier identification methods, through GSAD we need not to manually decide which attribute(s) should be utilized for final outlier set identification, rather the idea is to use GSAD in order to automatically find those attribute groups which are actually containing outlier values.

To the best of our knowledge, there is only one previous research work [34] in the field of SDEE, which has considered extent of outlierness and the influence of it on the performance. Please note that the focus of our research paper is to improve the performance of estimation models by identification and removal of outliers that are already present in dataset. Whereas, the study [34] is not focused towards outlier identification rather they have experimentally added outliers in dataset to study the extent of adding different degrees of outliers in existing dataset. Therefore, the work mentioned in [34] can not be used for comparative analysis purposes in our work.

III. BACKGROUND TECHNIQUES

A. DEFINITION

In order to eliminate the ambiguities in comprehension of various concepts in the remainder of this article, we mention following key terms:

- **Cutpoint:** A value which is found in a continuous range of values to partition the range into two intervals. e.g. a cutpoint C_p divides the continuous range [a, b] into intervals [a, C_p] and [C_p, b] [35].
- **Frequency Count:** The total number of values which actually belong to a particular interval [36].
- **Frequency Threshold:** In partitioning algorithms a frequency threshold is required to assess the intervals in order to re-partition the crowded interval, which is the necessary measure for the convergence of the algorithm [37].

B. OUTLIER IDENTIFICATION METHODS

In this work, we have proposed a new outlier identification method using Golden Section method. In order to enhance the integrity of findings, we have also evaluated the competency of our method with other prevalent outlier identification methods: Box-plot or inter-quartile based and Cooks' Distance measure [38] based methods.

1) GOLDEN SECTION METHOD

The Golden Section (GS) is surprisingly one of the most appearing pattern in the universe [39]. Its presence can be noticed in nature as well as man-made systems such as arrangement of flower petals, solar system, famous ancient architectures, paintings etc. It is not just perceived as a premise to measure beauty but also facilitates the optimal solution to space distribution. Owing to its wide applicability, it has been used in several fields of research including Flight Management [40], power point tracking [41], computer science [42], signal processing [43], image processing [44]. The term (GS) has its inception in the classical problem for the division of line segment in an specific manner [41], [43]. As specified in Fig.1, problem states that the line segment defined by search space [lb, ub] of length L is divided into two

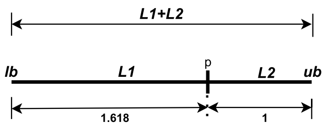


FIGURE 1. Representation of GS proportion.

sub-segments (major and minor) of lengths $L1$ and $L2$ respectively with a point p in line such that the ratio between $L1$ and $L2$ is equal to the ratio between L and $L1$, i.e.

$$\frac{L1}{L2} = \frac{L}{L1} = \frac{L1 + L2}{L1} = \varphi \tag{1}$$

where φ is the *golden ratio* representing quotient of $L1$ (i.e. major sub-segment) to $L2$ (i.e. minor sub-segment). The resulting quadratic equation for Eq.1 in terms of φ is:

$$\varphi^2 - \varphi - 1 = 0 \tag{2}$$

The solution to problem of dividing the line segment is achieved by preserving the *golden ratio* which is a positive root of Eq.2 as:

$$\varphi = \frac{1 + \sqrt{5}}{2} \approx 1.618 \tag{3}$$

The ratio of $L2$ (i.e. minor sub-segment) to $L1$ (i.e. major sub-segment) is called the GS (ϕ) which is defined as:

$$\phi = \frac{1}{\varphi} = 0.618 \tag{4}$$

Using GS optimization technique for line search optimization, GS is used to obtain two cutpoints C_1 and C_2 from a line segment which is defined by interval $[min, max]$ as:

$$C_1 = min + \phi(max - min) \tag{5}$$
$$C_2 = max - \phi(max - min) \tag{6}$$

In this research paper, initially the search space is spanning across the entire set of values (i.e. from minimum to maximum value) of an attribute and this search space is later on getting dynamically reduced. The sub-spaces are being identified by GS method along with the application of our explained procedure. The complete process of the proposed GS based outlier identification and removal method has been explained in Section IV-C.

2) INTER-QUARTILE RANGE (IQR)

IQR based outlier identification method has been one of the most adapted statistical outlier identification method in the field of SDEE [32]. The process of outlier identification relies on the measurement of the inter-quartile range between the lower quartile (Q1) and upper quartile (Q3). It is measured as $IQR = Q3 - Q1$. The data points which are lower than lower boundary $Q1 - 1.5 * IQR$ or greater than upper boundary $Q3 + 1.5 * IQR$ are considered as outliers. For comparison purposes, in the present study we have followed the same approach as that of [1], that if a data point contains its values outside the lower and upper boundaries for one or more variables then that data point will be identified as outlier data point.

3) COOKS' DISTANCE

Cooks' distance measures the change in residual values of all data points from full regression model to refitted regression model after omitting i^{th} data point [38]. To identify outliers using Cooks' distance, we have followed the same approach based on previous studies [1], [45], [46].

C. EFFORT ESTIMATION METHODS

Machine learning (ML) based prediction methods have been widely researched in the field of SDEE to improve the predictions [14], [47]. In this study, we have compared the performance of our GSAD based outlier identification and removal method with other prevalent outlier identification and removal methods using widely researched five different ML based methods that are also being explored by data science competition community [48].

1) SUPPORT VECTOR REGRESSION (SVR)

SVR has been widely studied by researchers in the area of SDEE [24], [49], [50]. It is machine learning technique which works by mapping non-linear separable patterns in data into higher feature space with an aim of minimizing the loss function along with maximizing support vector bounds.

2) RANDOM FOREST REGRESSION (RF)

RF is a state-of-art ensemble learning approach adapted for both classification and regression purposes [51]. In ensemble learning, the final results are proposed by combining the individual results of multiple similar or distinct type of estimation models. For regression purposes, it functions by constructing multiple decision trees using the training data and further provides the final solution in terms of the mean prediction of all individual decision trees.

3) RIDGE REGRESSION (RIDGE)

Ridge is an alternative regression approach which attempts to provide improvement over ordinary least square (OLS) regression. OLS suffers due to the presence of highly correlated attributes. The objective in regression problem is to capture the variation occurring in response variable(s) with the proportional variations in explanatory variables, but due to high collinearity among explanatory variables, these variations do not reflect clear true patterns while explaining these variations. The Ridge works to alleviate this concern by adding a penalty factor (λ) to all the diagonal elements of matrix X^TX before finding inverse of the matrix. The mathematical representation of Ridge parameter estimation is of following form:

$$\hat{\beta}_{ridge} = (X^TX + \lambda I)^{-1} X^TY \tag{7}$$

4) K-NEAREST NEIGHBOR REGRESSION (KNN)

KNN has been chosen in this study since it is one of the simplest estimation method which is perceived as similar to human based expert-judgement [52], [53]. KNN is a case based reasoning (CBR) or analogy based estimation (ABE) approach which assumes that a new project (which is characterized by a set of n features) will most likely have similar effort as to its similar past projects from the case base [2], [52]. A distance function is used to identify the similarity between projects on the basis of values of n features. Thereafter, the average effort value of all k most similar projects is considered to be as the effort for target project.

5) *GRADIENT BOOSTING MACHINE (GBM)*

GBM [54] is an ensemble based approach which relies on gradient descent [55] approach to optimize error reduction to build model with minimal error. GBM is a popular variant of boosted trees which is preferred because of its strength of facilitating better predictions even without much data preprocessing.

In this study, all the models using SVR, RF, Ridge, KNN, and GBM have been implemented in Python using $Scikit-learn$ library [56].

IV. EXPERIMENTAL DESIGN

A. DATASET

For the empirical evaluation of SDEE models proposed in this study, we have used 4 well-known benchmark datasets, namely ISBSG release 2021 [57], NASA93 [58], China [59], UCP [60]. These datasets have been most widely used by researchers in SDEE domain [5], [24], [61], [62], [63]. It is noteworthy that each of these datasets represent quite diversity in terms of number of instances, type and number of features, technical specifications, sources of data collection, different application domain etc. These datasets are representing different model-based counting approaches: China and ISBSG 2021 (selected subset) datasets are FP based, NASA93 is LOC based, while another dataset belongs to UCP methodology.

ISBSG release 2021 contains 10,531 cross-company projects including data from different countries, organization types, and development types. We have selected the dataset instances following the guidelines from ISBSG [64] and the procedure mentioned in [63]. This has resulted in 1179 new development type IFPUG version 4+ projects of quality A and B (data and function point quality). The feature selection was performed using the same protocol as suggested by Dejaeger *et al.* [65]. We have discarded the dataset instances which were having missing values, together with this some features have also been discarded due to irrelevancy/ unavailability of such attributes at the time of initial estimation. These features include project sequencing related features, features which are supposed to not be available at the time of initial estimation (e.g. development duration, time), features which are highly correlated and features which are having only one value (i.e. constant). The descriptive statistics of these datasets have been listed in table 1.

B. DATASET PRE-PROCESSING

Most of the SDEE datasets are skewed in nature, so studies have suggested the use of natural logarithm in order to make the distribution of data closer to normal distribution [28], [66]. Boehm has also suggested to use natural logarithm of attribute values for regression purpose since prima facie effort varies exponentially with an increase in software size [67]. We have observed that several SDEE datasets are having multiple '0' entries corresponding to different attributes. In such scenarios, consideration of normal log transformation is not a solution because it can not transform '0' values. Even if we leave a '0' value as it is and consider taking log(x) of only non-zero values then logarithmic of '1' will result in a '0', so the attribute having value either a '0' or '1' will ultimately have same effect. In this study, we have transformed each attribute of dataset using log1p transformation function (8) before the generation of regression models in order to reduce the skewness of dataset.

$$log1p(x) = log(1 + |x|) \tag{8}$$

Here, x belongs to the feature vector, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})$ and $i \in (1, 2, \ldots, m)$ for a dataset of size nxm. log1p transformation successfully alleviates the aforementioned concerns that are associated with logarithmic transformation by adding one to the value before taking its logarithm.

C. OUTLIER IDENTIFICATION AND ELIMINATION

This section answers RQ1, which is aimed to determine the use of GS based discretization method to identify outlier points in the dataset. GSAD is an adaptive discretization method that has been proposed in this research work to identify outliers with an aim to assess the impact of outliers in SDEE. The method forms the intervals automatically on the basis of dataset values and hence we do not need to fix the interval length or the number of intervals beforehand. GSAD works as follows (see Fig. 2):

1) Discretization of each feature vector separately.
2) Identify true outlier set.
3) Find most significant data points and discard outliers from dataset.

The following sections provide detailed description of the proposed method and its steps.

1) GSAD BASED OUTLIER IDENTIFICATION (OI@GSAD)

As stated earlier, the objective of this research is to identify the optimal set of outliers in a SDEE dataset by performing discretization to improve the quality of dataset. We propose GS based discretization method to filter out the outliers and find the most influential projects for the context of SDEE dataset. Consider a SDEE dataset D of size $n \times m$ which consists of a set of $m - 1$ independent features (that are accountable for the required effort amount for a project) and one dependent feature that is the effort value. It can be visualized in the form of a matrix as:

$$D = \begin{pmatrix} A_{1,1} & A_{1,2} & \cdots & A_{1,m} \\ A_{2,1} & A_{2,2} & \cdots & A_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \cdots & A_{n,m} \end{pmatrix}$$

An outlier identification method identifies a project as outlier if the feature values of that project are very distant to the feature values of other projects in the dataset [68]. It is required to identify and discard the set of outlier projects and make the use of only significant projects for the purpose of effort estimation of a new project. This is to specify that we

TABLE 1. Descriptive statistics of benchmark datasets used in this study.

Dataset	Feature	Min	Max	Mean	Std. Dev.
China	Input	0.00	9404.00	167.10	486.34
	Output	0.00	2455.00	113.60	221.27
	Enquiry	0.00	952.00	61.60	105.42
	File	0.00	2955.00	91.23	210.27
	Interface	0.00	1572.00	24.23	85.04
	Resource	1.00	4.00	1.46	0.82
	Effort	26.00	54620.00	3921.05	6480.86
UCP	Simple_Actors	0	4	0.71	0.90
	Average_Actors	0	3	0.94	0.88
	Complex_Actors	0	6	2.62	1.50
	Simple_UC	0	20	2.7	2.89
	Average_UC	3	30	15.84	5.37
	Complex_UC	5	27	14.28	4.45
	T1	0	5	0.41	1.14
	T3	0	5	1.35	2.02
	T4	0	5	2.3	2.37
	T5	0	5	2.92	2.37
	T6	0	5	3.14	2.43
	T7	0	5	3.71	2.06
	T9	0	5	4	1.95
	T10	0	5	4.47	1.43
	T11	3	5	4.9	0.38
	ENV2	0	2	0.04	0.26
	ENV3	0	5	0.18	0.72
	ENV4	0	4	1.21	1.37
	ENV5	0	5	2.64	1.88
	ENV6	1	5	4.15	1.01
	ENV7	0	5	1.57	1.55
	ENV8	0	5	3.84	1.25
	Real_Effort	5775	7970	6558.72	664.23
NASA93	rely	0.88	1.40	1.11	0.13
	data	0.94	1.16	1.00	0.07
	cplx	0.85	1.65	1.18	0.15
	time	1.00	1.66	1.13	0.20
	stor	1.00	1.56	1.13	0.19
	virt	0.87	1.15	0.92	0.09
	turn	0.87	1.15	0.96	0.09
	acap	0.71	1.00	0.89	0.09
	aexp	0.82	1.13	0.93	0.06
	pcap	0.70	1.00	0.91	0.10
	vexp	0.90	1.21	1.00	0.08
	lexp	0.95	1.14	0.97	0.05
	modp	0.82	1.24	0.98	0.09
	tool	0.83	1.24	1.00	0.09
	sced	1.00	1.08	1.04	0.04
	loc	0.90	980.00	94.02	133.60
	actual	8.40	8211.00	624.41	1135.93
ISBSG 2021	Input count	0.0	2019.0	147.32	219.99
	Output count	0.0	1337.0	123.99	171.12
	Enquiry count	0.0	952.0	90.74	136.59
	File count	0.0	1252.0	129.61	166.42
	Interface count	0.0	977.0	49.10	90.59
	Developer	70.0	6610.0	2182.82	2097.72
	Functional Size	6.0	16148.0	620.67	947.46
	Value Adjustment Factor	0.65	1.29	1.01	0.08
	Normalised Work Effort Level 1	40.0	230514.0	6679.57	13336.10

will be using partition, region, and interval interchangeably in further sections.

OI@GSAD consists of three major steps: the identification of potential outliers, generation of outlier commonality matrix and finally distinguishing false and true outliers. The entire process of outlier identification is as follows:

2) DATASET PARTITIONING AND IDENTIFICATION OF POTENTIAL OUTLIERS

For a dataset D which consists of m features, we perform the potential outlier identification process to each of the feature separately. To begin with the process, we firstly partition the entire region of feature values into three sub-regions as per the GS method with the help of two cutpoints found using Eq. 5

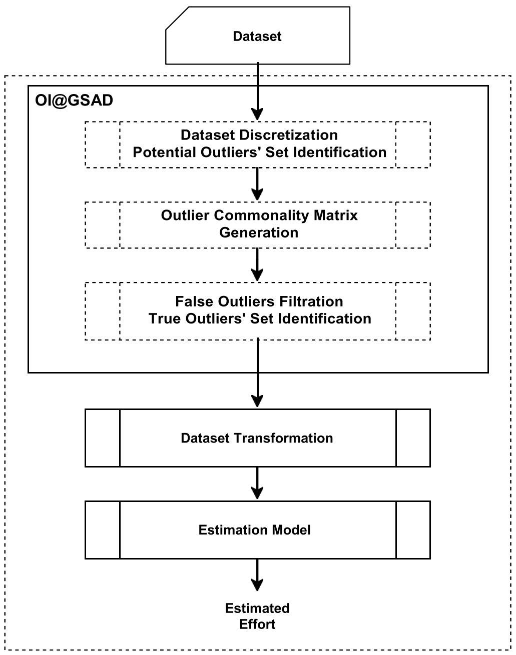

Dataset
OI@GSAD
Dataset Discretization
Potential Outliers' Set Identification
Outlier Commonality Matrix
Generation
False Outliers Filtration
True Outliers' Set Identification
Dataset Transformation
Estimation Model
Estimated
Effort

FIGURE 2. **GSAD based effort estimation.**

and Eq. 6. The algorithmic representation of the method is given in Algorithm 1.

GSAD based partitioning is an iterative partitioning approach where at each iteration t, the feature vector A_j of size n (where $X_i \in R^n$) is partitioned and assessed for re-partitioning on the basis of its frequency count value $f_{A_j}(t)$, where A_j is the j^{th} feature vector of the dataset. We consider the partitioning problem $p(X) s.t. X \in r_0 \subset R^n$ where r_0 is a sub-partition defined by boundaries $lFreq \leq f_{r_0}(t) \leq uFreq$, where $lFreq$, $uFreq$ are the lower and upper frequency thresholds for the dataset D. The sub-partition with its $f_{r_0}(t)$ value greater than the upper frequency threshold is subjected to be re-partitioned in the next splitting iteration. Here, each such partition which is selected for re-partitioning is divided into three sub-regions formed with the help of two cutpoints. The maximum number of new partitions/ intervals at a splitting iteration t is equal to $t + 2$.

The partitions that do not cross the upper frequency threshold need not to be re-partitioned and thus made fixed as the final sub-partitions (i.e. partition.fixed = TRUE). Also the partitions have been made in such a way that if a value has multiple occurrences then all such occurrences will belong to the same partition. In this particular scenario, the compulsion of having the interval frequency to be less than the upper frequency threshold is relaxed. The same has been shown through statement $Int_i.fixed = TRUE$ of Algorithm 1, where interval Int_i has been made fixed to avoid its further partitioning.

Partitioning Process Convergence Measure: The frequency thresholds $lFreq$ and $uFreq$ are the factors which decide when to partition the current interval further and when to stop partitioning. These frequency thresholds have been identified empirically. If n is the number of samples in dataset, then the thresholds will be defined as:

$lFreq = 10\%$ of n - 40% of (10% of n) and

$uFreq = 10\%$ of $n + 40\%$ of (10% of n),

For example, if dataset D has 100 samples, then at any iteration t, the boundaries for an interval r_0 will be considered as $(10 - 4) \leq f_{r_0}(t) \leq (10 + 4)$, i.e. $6 \leq f_{r_0}(t) \leq 14$.

The partitioning process automatically stops further partitioning after total q splitting iterations when the convergence criterion meets. Thereafter, the process of diagnosing potential outliers begins. For this purpose, we consider partitions obtained after q splitting iterations and identify those partitions which defy the potential outlier frequency threshold OF_{Th}. We have set the OF_{Th} to 5% of n which signifies that those partitions for which the $f_{r_0}(t)$ is less than 5% of total values in the attribute will contain the potential outliers. Thereafter all the elements of such partitions are added to the set of potential outliers (PO_j), for j^{th} attribute. We repeat the entire process to identify the set of potential outliers for all the remaining attributes in dataset. The statistics of obtained intervals and resulting potential outliers have been displayed in Table 2.

In this research work, we have empirically decided OF_{Th} to be equal to 5% of n for the studied datasets. This threshold is depending on the size of dataset. Note that, finding an optimal value for a particular threshold to identify outliers has always been a difficult issue in the domain of outlier identification. The Tukey's method (i.e. IQR) has fixed 1.5 * IQR while defining lower and upper boundaries to ensure optimal number of outliers. Tukey has taken value '1.5' because 2 was too big and 1 was too small [69]. Similarly, in our method, we have observed that taking a OF_{Th} value below 5% identifies very few number of potential outliers by individual attributes, which ultimately results in no or too few true outliers of high degree of outlierness. In a nutshell, the lower the value of OF_{Th}, the lower is the number of outliers. Therefore, we have recommended OF_{Th} value to be taken as at least 5% of n for the selected set of datasets.

3) GENERATION OF OUTLIER COMMONALITY MATRIX $O_{cn}[i, j]$

Initially we have as many potential outlier sets as the number of features in the dataset. Then out of these temporary sets of potential outliers, we obtain a final set of outliers by taking intersection over all these temporary outlier sets.

$O_{cn}[i, j]$ matrix shows the popularity of a data point as potential outlier among all the features of the considered dataset. The dimension of this matrix is $n \times m$, where n is the number of instances and m is the number of attributes in the dataset. As an example, we have shown the matrix entries for data points o_1, o_2, o_{95} and o_{499} of China dataset in Figure 3. The matrix entry $O_{cn}[i, j] = 1$ signifies that i^{th} data point has been chosen as potential outlier from j^{th} attribute and for the contrary $O_{cn}[i, j]$ will be equal to a 0 entry. So, the total number of 1s in a row of this matrix corresponds to

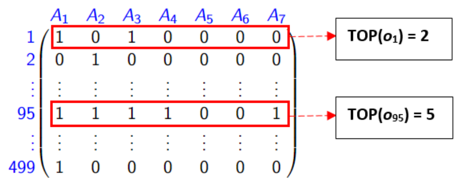
Algorithm 1: Partitioning and Identification of Potential Outliers

```
Input:
        $A_j$–sorted $j^{th}$ feature vector of dataset $D$ where $j \in \{1, 2 \ldots, m\}$, with $min$ as smallest and $max$ as largest value of
        $j^{th}$ feature
        $lfreq$–lower frequency threshold for dataset $D$
        $ufreq$–upper frequency threshold for dataset $D$
        $OF_{Th}$–outlier frequency threshold for dataset $D$
Output: $PO_j$–The potential outliers in $j^{th}$ attribute for dataset $D$, where $j \in \{1, 2 \ldots, m\}$
Initialize $I_{set} \leftarrow NULL, PO_j \leftarrow NULL$
Function CreateInterval($min, max$):
    $Int_i.start \leftarrow min$
    $Int_i.end \leftarrow max$
    $Add\ Int_i\ to\ I_{set}$
    return $I_{set}$
for $all\ intervals\ Int_i\ in\ I_{set}$ do
    $C_1 \leftarrow Int_i.start + \phi(Int_i.end - Int_i.start)$
    $C_2 \leftarrow Int_i.end - \phi(Int_i.end - Int_i.start)$
    // Form new sub-intervals
    $Int_1.start \leftarrow Int_i.start,\ Int_1.end \leftarrow C_1$
    $Int_2.start \leftarrow C_1,\ Int_2.end \leftarrow C_2$
    $Int_3.start \leftarrow C_2,\ Int_3.end \leftarrow Int_i.end$
    $Int_1.fixed \leftarrow FALSE$
    $Int_2.fixed \leftarrow FALSE$
    $Int_3.fixed \leftarrow FALSE$
    $Replace\ Int_i\ with\ Int_1,\ Int_2,\ Int_3$ in $I_{set}$
end
for $all\ Intervals\ Int_i\ in\ I_{set}$ do
    if $Int_i.freq > ufreq_{A_j}\ \&\ Int_i.freq > lfreq_{A_j}\ \&\ Int_i.fixed = FALSE$ then
        if $Int_i\ contains\ all\ same\ values$ then
            $Int_i.fixed = TRUE$
        end
        Repeat $Step\ 7\ to\ 18$
    end
    else
        $Int_i.fixed = TRUE$
    end
end
for $all\ Intervals\ Int_i\ in\ I_{set}$ do
    if $Int_i.freq < OF_{Th}$ then
        Add all elements of $Int_i$ to $PO_j$
    end
end
```

the total potential outlier positive features (i.e. $TOP(o_i)$) in dataset with reference to a data point o_i. The more the value of $TOP(o_i)$, the point o_i will be said to have more degree of outlierness.

4) FORMATION OF FINAL TRUE OUTLIERS' SET AND FILTRATION OF FALSE OUTLIERS

For each of the data point $o_i \in PO_j$, we calculate its $TOP(o_i)$, where $j \in \{1, 2, \ldots, m\}$. As an example, we have highlighted the TOP values of data points o_1 and o_{95} in Figure 3 as 2 and 5 respectively. The value of TOP is 1 for o_2 and o_{499}.



$$\begin{array}{c|ccccccc} & A_1 & A_2 & A_3 & A_4 & A_5 & A_6 & A_7 \\ \hline 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 95 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 499 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

FIGURE 3. Outlier Commonality Matrix for China Dataset.

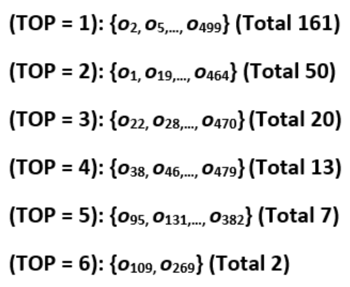
Similarly, the TOP values for other data points have been obtained.

TABLE 2. Statistics of obtained intervals and potential outliers for each attribute of datasets.

Dataset	Feature	Intervals	Potential Outliers
China	Input	18	119
	Output	15	50
	Enquiry	13	29
	File	16	72
	Interface	12	30
	Resource	3	0
	Effort	15	120
UCP	Simple_Actors	4	2
	Average_Actors	4	3
	Complex_Actors	7	6
	Simple_UC	9	4
	Average_UC	13	9
	Complex_UC	14	8
	T1	4	6
	T3	5	4
	T4	5	6
	T5	4	3
	T6	2	0
	T7	3	0
	T9	4	3
	T10	3	0
	T11	3	5
	ENV2	3	2
	ENV3	4	6
	ENV4	4	1
	ENV5	6	0
	ENV6	4	0
	ENV7	6	3
	ENV8	5	3
	Real_Effort	12	3
NASA93	rely	4	2
	data	4	0
	cplx	5	3
	time	4	0
	stor	4	0
	virt	3	0
	turn	4	2
	acap	3	0
	aexp	4	1
	pcap	3	0
	vexp	4	4
	lexp	3	0
	modp	5	1
	tool	5	3
	sced	3	0
	loc	11	2
	actual	15	10
ISBSG 2021	Input count	17	164
	Output count	15	194
	Enquiry count	15	205
	File count	18	276
	Interface count	15	125
	Developer	12	138
	Functional Size	19	233
	Value Adjustment Factor	10	124
	Normalised Work Effort Level 1	19	273

Further, it can be seen in Figure 4, that for China dataset, a total of 161 data points have their TOP = 1, 50 data points have TOP = 2, 20 data points have TOP = 3 and so on.

The rationale behind final outlier set identification is to achieve better performance (minimum error) along with the removal of minimum number of data points as outliers.



(TOP = 1): $\{o_2, o_5, \ldots, o_{499}\}$ (Total 161)

(TOP = 2): $\{o_1, o_{19}, \ldots, o_{464}\}$ (Total 50)

(TOP = 3): $\{o_{22}, o_{28}, \ldots, o_{470}\}$ (Total 20)

(TOP = 4): $\{o_{38}, o_{46}, \ldots, o_{479}\}$ (Total 13)

(TOP = 5): $\{o_{95}, o_{131}, \ldots, o_{382}\}$ (Total 7)

(TOP = 6): $\{o_{109}, o_{269}\}$ (Total 2)

FIGURE 4. Statistics of TOP values of data points for China Dataset.

To analyse that, GSAD categorizes all the data points as per their $TOP(o_i)$ values into different sets. The set $TOP \geq m$ includes all those data points as outliers that have been found as PO for at least m attributes. For instance, with the help of Figure 4, it can be easily observed that set $TOP \geq 1$ will include all those data points which have been found as PO with $TOP = 1$ up to $TOP = 6$, i.e. $161+50+20+13+7+2 = 253$. This value can be reconfirmed from the first row of Table 3.

The analysis to obtain optimal number of outliers starts with $TOP \geq 1$ and stops at $TOP \geq n$, where n is the total number of attributes in a dataset. For the simplicity, we have shown the analysis results for three categories (up to $TOP \geq 3$) in Table 3. It can be seen from the table, $TOP \geq 2$ is yielding minimum error % for two datasets (UCP and ISBSG 2021). For China and NASA93, $TOP \geq 1$ is resulting in minimum error but it is also noteworthy that $TOP \geq 1$ is resulting in a significantly large portion of data to get removed as outliers. Which is again encouraging us to use $TOP \geq 2$ for final outlier set identification. The similar patterns of results appeared even beyond $TOP \geq 3$. Thus, for the present study, we have considered all those data points (POs) as the final true outliers for which $TOP \geq 2$. So the rest of the data points to which only single feature has considered as potential outlier will be filtered out to the false outlier set O_{False}. Therefore, the false outlier set $O_{False} = PO - TOP \geq 2$ needs to be filtered out. Thus, after following this process, we have filtered 161 (i.e. 253-92) data points from China, 33 (i.e. 52-19) data points from UCP, 14 (i.e. 19-5) data points from NASA93 and 347 (i.e. 881-534) data points from ISBSG 2021 dataset as false outliers. Even though we have used Mean Square Error (MSE) of SVR based estimation model to direct the search for final outlier set, the entire set of estimation outcomes scored well across an exhaustive range of ML based SDEE models along with multiple performance measures.

We have also compared the performance of GSAD with other two previous methods of outlier identification. Table 4 shows the statistics of outliers after running GSAD (with $TOP \geq 2$) and the other two methods for each of the dataset. The datasets are of varying sizes, but it is clearly evident that GSAD is identifying varying and optimal number of outliers for different datasets.

TABLE 3. The number of outliers and error outcomes using GSAD with different number of potential outlier positive attributes.

Dataset	Total Instances	Total Outlier Positive Attributes	Total Outliers	Model Error (MSE%)
China	499	$TOP \geq 1$	253	60.00
		$TOP \geq 2$	92	80.50
		$TOP \geq 3$	42	87.00
UCP	70	$TOP \geq 1$	52	0.80
		$TOP \geq 2$	19	0.60
		$TOP \geq 3$	5	0.70
NASA93	93	$TOP \geq 1$	19	48.90
		$TOP \geq 2$	5	53.00
		$TOP \geq 3$	2	63.30
ISBSG 2021	1179	$TOP \geq 1$	881	0.09
		$TOP \geq 2$	534	0.06
		$TOP \geq 3$	224	0.70

TABLE 4. The number of outliers identified using GSAD, IQR [1] and Cooks [1], [45], [46] methods in all datasets.

Dataset	Instances (no missing value)	Outliers using GSAD	Outliers using IQR	Outliers using Cooks
China	499	92	156	23
UCP	70	19	35	2
NASA93	93	5	64	4
ISBSG 2021	1179	534	862	19

D. EFFORT ESTIMATION MODEL GENERATION

All the models have been generated using 5 learning algorithms: *SVR*, *RF*, *Ridge*, *KNN*, and *GBM*. We have considered them as baseline to assess whether the GSAD based outlier identification and removal can improve the performance. For the completeness, we have compared the results with prevalent outlier identification and removal approaches (IQR and Cooks). In order to assess the estimation performances, each type of machine learning method has been applied to generate four different versions of estimation models:

1) with all data points, i.e. without using any outlier identification and removal method
2) with only non-outlier data points identified after using proposed method GSAD
3) with only non-outlier data points identified after using existing outlier identification method IQR [1]
4) with only non-outlier data points identified after using existing outlier identification method Cooks based on studies [1], [45], [46]

Table 5 lists all the parameter values that have been investigated for each SDEE method in this study. The best hyper-parameters for each learning algorithm have been selected using grid-search technique with 10 $\times$ 3 repeated cross-fold validation.

E. EFFORT ESTIMATION MODEL EVALUATION

In this study, we have validated the performance of all SDEE models using Repeated Cross-Validation (RCV) and the most recommended Leave-One-Out Cross Validation (LOOCV) [12]. RCV can be beneficial over Cross-Fold validation in order to reduce the high-variance between multiple predictions by repeating the process of validation [18]. T. Menzies *et al.* [18] have suggested to use more than one validation methods, for example LOOCV and RCV to study the variability of results while reporting the final conclusions. To study the variability in results, we have undertaken the sensitivity analysis [70]. Through this, we intend to determine the sensitivity of the findings of this study by repeating all the experiments with different experimental settings. To achieve this, we have repeated all the experiments by switching between LOOCV and 10×3 RCV that involves 10 repeats of 3-fold cross-validation.

TABLE 5. Hyper-parameter values of investigated SDEE methods.

Estimation Method	Hyper-parameters and values

SVR	Kernel = "linear, sigmoid"
	C (regularization parameter) = {0.01, 0.1}
	ϵ (slack variable) = {0.1, 0.3, 1}

RF	max depth = {5, 10}
Ridge	α = {0.00, 0.05, ..., 1}
KNN	k (number of neighbors) = {1, 2, 3, 4, 5} metric = "euclidean"

GBM	number of estimators = {2, 4, ..., 100}
	max depth = {3, 5, ..., 9}
	learning rate = {0.05, 0.01, 0.1}

F. PERFORMANCE EVALUATION

A variety of performance measures have been used by researchers to showcase the performances of SDEE models. These performance measures do not measure and/ or represent the same facet of performance. The reliability in measurement of performance largely depends on performance evaluation measure [20], [71]. Some of the measures have been criticized for their biasness, but none of the measure has been unanimously accepted to compare all type of SDEE models [13]. Therefore, for proper empirical analysis, we have used Mean Absolute Error (MAE) (9), Mean Square Error (MSE) (11), Median Absolute Error (MdAE) (10), Standardized Accuracy (SA) (12).

The use of MAE has been recommended by several studies [24], [29], [72] in the past for measuring average absolute difference between actual and predicted effort. MSE has also been recommended for the SDEE field in study [73]. MSE measures the average squared loss from actual to the predicted effort. MAE focuses on central tendency, therefore, it is considered as unbiased for both under and overestimation. On the other hand, MSE penalizes large error values more as the square will be comparatively much larger if the error (difference between actual and predicted effort) is large. In other words, with the use of MSE, the training model will give much attention to improving the predictions of those software projects for which the error is high. MdAE measures the median value from the absolute errors of all projects in observation. It has been recommended as more robust to large outliers' scenario [74]. SA gives an idea about the performance of a model in comparison to random guessing. We have also used effect size (Δ) measure to verify whether there is any improvement in comparison to random guessing or if the predictions are generated by chance.

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y_i}| \tag{9}$$

$$MdAE = Median\{|y_i - \widehat{y_i}|\} \tag{10}$$

$$MSE = \frac{1}{n}\sum_{i=1}^{n} (y_i - \widehat{y_i})^2 \tag{11}$$

$$SA_{M_i} = 1 - \frac{MAE_{M_i}}{\overline{MAE}_{M_0}} \tag{12}$$

$$\Delta = MAE_{M_i} - \frac{\overline{MAE}_{M_0}}{SP_0} \tag{13}$$

where $y_i/\widehat{y_i}$ represents the actual/estimated effort value, $1 \leq i \leq n$ and n is the number of projects in test set. MAE_{M_i} is the MAE of SDEE model for which the performance is being measured and $\overline{MAE}_{M_0}$ is the MAE of large number of randomly guessed effort values (generally 1000) from the dataset. SP_0 is the standard deviation of randomly guessed effort values for entire test sample.

This is to note that smaller values of MAE, MdAE, MSE error measures signify better performance while SA is the accuracy measure for which the larger the value, the better the performance. The values of Δ are interpreted with the help of scales proposed by Cohen [75], according to this scale the values can be categorized into small (around 0.2), medium (around 0.5) and large (around 0.8). A Δ value falling in these categories signify real effect in model's performance in comparison to random guessing.

To the best of our knowledge in the field of SDEE, it is improper to consider that any specific performance measure can always be preferred over others, rather each one of them may be useful in measuring different aspects. In this manuscript, we are not comparing the performances of error and accuracy measures to choose the best measure among all. We have used a stack of unbiased performance measures to evaluate the performances of all SDEE models over different aspects of error and accuracy.

G. STATISTICAL TEST

In order to evaluate the performance of proposed methods in this research work, Mann-Whitney U tests have been employed. Mann-Whitney U test is non-parametric test which has been adopted since the error distributions of the studied techniques are not normally distributed as identified using Shapiro-Wilk test [76]. Through these tests, we aim to identify whether the error distributions (absolute error, square error or MAE, MSE) of two techniques T_i and T_j are significantly different or not. More specifically, we have tested **null hypothesis** (H_0) which states that techniques T_i and T_j are statistically equivalent. The **alternative hypothesis** (H_A) states that techniques T_i and T_j are significantly different. Here, T_i represents technique employing our GSAD while T_j signifies one of the techniques employing baseline (no outlier removal), IQR, or Cooks. All statistical tests have been performed at a significance level (α) of 0.05.

To summarize the results of statistical comparisons and further assess the competitiveness of our GSAD based models, we have used win-tie-loss statistics (Algorithm 2) as used in previous studies [5], [24], [77], [78].

Algorithm 2: Pseudo Code for Win-Tie-Loss Calculation Between Techniques T_i, T_j w.r.t. Error Distributions Err_i and Err_j

```
Initialize $win_i \leftarrow 0$, $tie_i \leftarrow 0$, $loss_i \leftarrow 0$, $win_j \leftarrow 0$, $tie_j \leftarrow 0$, $loss_j \leftarrow 0$;
if Function MannWhitneyU($Err_i$, $Err_j$) says they are same
then
    $tie_i = tie_i + 1$;
    $tie_j = tie_j + 1$;
end
else
    if $Err_i < Err_j$ then
        $win_i = win_i + 1$;
        $loss_j = loss_j + 1$;
    end
end
else
    $win_j = win_j + 1$;
    $loss_i = loss_i + 1$;
end
```

V. EXPERIMENTAL RESULTS AND DISCUSSION

This section is aimed to elaborate the significance of the proposed GSAD method through the performance comparison of models with and without the consideration of outliers as well as with other schemes of outlier identification. To simplify the readability, the SDEE model that do not incorporate any outlier removal scheme (that is as per previous study [79]) is referred as SDEE_BASE, the model that uses only non-outlier data points after applying GSAD is referred as SDEE_GSAD, whereas SDEE_IQR and SDEE_Cooks signify the models that are using non-outlier data points obtained after applying IQR and Cooks schemes respectively. This is to also note that an SDEE model is one of the ML method specified in Section IV-D.

A. RQ2. CAN OUR GOLDEN SECTION BASED OUTLIER IDENTIFICATION METHOD HELP IN IMPROVING THE PERFORMANCE OF ML BASED SDEE METHOD?

To answer this question, we have compared the performance of 5 estimation methods (namely SVR, RF, Ridge, KNN, and GBM) with and without the application of GSAD based outlier removal method by using a stack of unbiased performance measures (namely, SA, Δ, MAE, MSE, and MdAE), and statistical test of significance for both 10 $\times$ 3 RCV and LOOCV validation methods. The values of performance measures for all datasets are listed in Tables 6 and 7.

In order to empirically assess whether our GSAD based models are actually predicting and not guessing, the SA, Δ values have been shown in Table 6. It can be observed that all the SDEE_GSAD models are achieving mostly positive values for SA (with very few exceptions for ISBSG 2021) that are ranging from 0.12 (12%) to 0.67 (67%) for RCV while from 0.53 (53%) to 0.99 (99%) for LOOCV validation. The Δ values obtained for all the SDEE_GSAD models are showing medium to large effect size improvement over random guessing. Therefore, it can be concluded that SDEE_GSAD models are not yielding their predictions by chance. It can be further confirmed from the table that SDEE_GSAD models are obtaining better values than their competitive SDEE_BASE models for most cases.

Further, we have assessed the performance of SDEE_GSAD and SDEE_BASE using MAE, MSE and MdAE. The results can be seen in Table 7 for all the datasets using RCV and LOOCV validation methods. For a total of 20 cases of comparison with SDEE_BASE, the SDEE_GSAD models are achieving lower MAE values for 17 cases using RCV. For MSE measure, SDEE_GSAD models are yielding lower values for 17 cases and equal values for 3 cases, while for MdAE measure, SDEE_GSAD models are achieving lower values in 15 cases and equal values in 1 case in comparison to SDEE_BASE models. The results for MAE, MSE measures using 10 $\times$ 3 RCV can be further visualized in Fig. 5. This is to note that due to space limitations, we have not shown the plots for other performance measures. For LOOCV method, SDEE_GSAD models are obtaining lower values of MAE and MSE measures for 17 and 18 cases, equal values for 2, 2 cases respectively. While for MdAE, SDEE_GSAD models are achieving lower values for 17 cases.

The p-values obtained after statistical comparison of SDEE_GSAD and SDEE_BASE are shown in Table 9. The MAE and MSE distributions obtained by SDEE_GSAD

TABLE 6. Performance comparison of all models in terms of Standardized Accuracy (SA) and Effect Size (Δ) for each dataset using 10 $\times$ 3 RCV and LOOCV. The models with superscript † represent the models that are as per previous paper [79].

Dataset	Estimation Method	RCV		LOOCV	
		SA	$\|\Delta\|$	SA	$\|\Delta\|$
China	SVR_BASE†	0.17	3.92	0.89	5.33
	SVR_GSAD	0.22	5.80	0.89	5.88
	SVR_IQR	0.12	2.68	0.88	5.78
	SVR_Cooks	0.19	4.77	0.89	5.62
	RF_BASE†	0.15	3.46	0.89	5.33
	RF_GSAD	0.21	5.58	0.89	5.87
	RF_IQR	0.10	2.27	0.88	5.77
	RF_Cooks	0.18	4.58	0.89	5.62
	Ridge_BASE†	0.17	3.92	0.89	5.33
	Ridge_GSAD	0.23	6.04	0.89	5.88
	Ridge_IQR	0.11	2.75	-6.5	42.78
	Ridge_Cooks	0.19	4.61	-6.5	41.12
	KNN_BASE†	0.10	2.28	0.88	5.29
	KNN_GSAD	0.16	4.24	0.89	5.84
	KNN_IQR	0.05	1.35	0.87	5.74
	KNN_Cooks	0.14	3.57	0.88	5.58
	GBM_BASE†	0.14	3.27	0.89	5.33
	GBM_GSAD	0.21	5.47	0.89	5.86
	GBM_IQR	0.08	1.99	0.88	5.76
	GBM_Cooks	0.17	4.17	0.89	5.61
UCP	SVR_BASE†	0.21	5.93	0.99	87.62
	SVR_GSAD	0.12	3.31	0.99	94.15
	SVR_IQR	0.19	5.83	0.99	81.49
	SVR_Cooks	0.19	5.16	0.99	87.37
	RF_BASE†	0.41	10.87	0.99	87.8
	RF_GSAD	0.34	8.67	0.99	94.36
	RF_IQR	0.22	7.58	0.99	81.63
	RF_Cooks	0.44	11.54	0.99	87.56
	Ridge_BASE†	0.33	8.86	0.99	87.71
	Ridge_GSAD	0.35	8.87	0.99	94.34
	Ridge_IQR	0.46	13.76	-7.78	639.84
	Ridge_Cooks	0.34	9.71	-7.78	685.18
	KNN_BASE†	0.22	5.55	0.99	87.63
	KNN_GSAD	0.25	7.04	0.99	94.23
	KNN_IQR	0.18	6.01	0.99	81.58
	KNN_Cooks	0.22	6.00	0.99	87.37
	GBM_BASE†	0.43	12.06	0.99	87.74
	GBM_GSAD	0.33	8.96	0.99	94.37
	GBM_IQR	0.34	10.65	0.99	81.77
	GBM_Cooks	0.41	11.87	0.99	87.48
NASA93	SVR_BASE†	0.55	14.27	0.90	3.30
	SVR_GSAD	0.56	15.00	0.90	3.30
	SVR_IQR	0.77	22.98	0.91	2.58
	SVR_Cooks	0.56	14.09	0.90	3.41
	RF_BASE†	0.58	15.14	0.91	3.34
	RF_GSAD	0.59	16.09	0.91	3.35
	RF_IQR	0.74	25.70	0.90	2.57
	RF_Cooks	0.57	14.32	0.91	3.43
	Ridge_BASE†	0.63	16.9	0.92	3.37
	Ridge_GSAD	0.67	17.74	0.93	3.4
	Ridge_IQR	0.83	28.23	-5.00	14.19
	Ridge_Cooks	0.67	18.13	-4.68	17.61
	KNN_BASE†	0.52	13.71	0.89	3.28
	KNN_GSAD	0.55	14.41	0.90	3.29
	KNN_IQR	0.73	24.06	0.89	2.54
	KNN_Cooks	0.53	13.93	0.89	3.36
	GBM_BASE†	0.56	14.79	0.90	3.33
	GBM_GSAD	0.59	16.24	0.91	3.36
	GBM_IQR	0.75	22.52	0.90	2.55
	GBM_Cooks	0.56	13.85	0.90	3.40
ISBSG 2021	SVR_BASE†	-2.13	36.20	-0.65	0.60
	SVR_GSAD	-3.12	42.72	-1.57	1.79
	SVR_IQR	-0.21	2.41	0.01	0.02
	SVR_Cooks	-2.83	43.78	-1.26	1.41
	RF_BASE†	0.23	4.10	0.55	0.51
	RF_GSAD	0.26	3.64	0.55	0.63
	RF_IQR	0.48	5.52	0.62	0.97
	RF_Cooks	0.27	4.36	0.57	0.64
	Ridge_BASE†	0.24	4.23	0.55	0.51
	Ridge_GSAD	0.24	3.30	0.53	0.60
	Ridge_IQR	0.43	4.84	0.99	1.55
	Ridge_Cooks	0.28	4.47	0.97	1.08
	KNN_BASE†	0.24	4.33	0.55	0.51
	KNN_GSAD	0.27	3.81	0.56	0.64
	KNN_IQR	0.48	5.45	0.61	0.96
	KNN_Cooks	0.28	4.39	0.57	0.64
	GBM_BASE†	0.23	4.06	0.55	0.51
	GBM_GSAD	0.26	3.67	0.54	0.61
	GBM_IQR	0.49	5.51	0.63	0.99
	GBM_Cooks	0.29	4.58	0.58	0.65

models are found to be significantly different than SDEE_BASE in 15 cases for RCV validation method. While for LOOCV, the distributions are found to be significantly different for 5 cases with both MAE and MSE. The win-tie-loss statistics of Mann-Whitney U tests for statistically comparing the distributions of MAE and MSE measures have been summarized in Table 12. We can observe that SDEE_GSAD achieves best win-tie-loss outcomes (14+14+4+4=36 wins and 5+5+15+15=40 ties, out of total 80 cases) against SDEE_BASE across both error measures over all datasets for both RCV and LOOCV validation schemes. This proves that SDEE_GSAD models have not been outperformed by SDEE_BASE models for more than 5% (4 out of 80) cases.

Therefore, it can be concluded from the results that when we remove the identified outliers using GSAD approach, the performance of ML based SDEE models is generally improving.

B. RQ3. HOW THE PERFORMANCE OF ML BASED SDEE METHOD VARIES USING OUR PROPOSED OUTLIER IDENTIFICATION METHOD IN COMPARISON TO OTHER PREVALENT OUTLIER IDENTIFICATION METHODS?

We have empirically assessed the competitiveness of our GSAD based method to other prevalent outlier identification methods (IQR, Cooks) in improving the performance of ML based SDEE methods. The comparative results of all measures can be seen in Tables 6 and 7. From Table 6, it is clearly evident that GSAD based models are mostly achieving positive values for SA (with only 2 exceptions with one dataset), while IQR and Cooks based methods are yielding negative SA values with all 4 datasets. Thus, IQR and Cooks based models are performing poorer than random guessing in some cases with all datasets.

Table 7 shows the summary of error measures' results in terms of MAE, MSE and MdAE for all datasets with RCV and LOOCV methods. We have further assessed the statistical significance of results using Mann-Whitney U tests between the competing models. The p-values obtained after statistical comparison of SDEE_GSAD with SDEE_IQR and SDEE_Cooks models over MAE and MSE distributions are shown in Table 10 and Table 11.

For completeness, win-tie-loss statistics between the competing methods have been summarized in Table 12. We can observe that SDEE_GSAD models achieve 28 (12+8+7+1) wins and 46 (8+10+12+16) tie outcomes against SDEE_Cooks models. Which means that SDEE_GSAD models have not been outperformed for more than 7.5% (5 out of 80) cases by GSAD_Cooks models. In comparison to SDEE_IQR models, the proposed SDEE_GSAD models achieve 19 (9+7+3) wins and 24 (1+3+9+11) ties. The proposed SDEE_GSAD models have been outperformed by SDEE_IQR models for 46.25% (37 out of 80) cases. Here, it is also noteworthy to remember that IQR based outlier identification method has discarded the highest percentage of data as outliers (refer Table 4), leaving only a small portion of data as non-outlier set for model training and testing. With this, there is very high possibility that IQR might classify a new project as outlier and also the model

TABLE 7. Performance comparison of all models in terms of MAE, MSE and MdAE for each dataset using 10 × 3 RCV and LOOCV. The proposed models have been highlighted in bold fonts. The models with superscript † represent the models that are as per previous paper [79].

Dataset	Estimation Method	RCV			LOOCV		
		MAE	MSE	MdAE	MAE	MSE	MdAE
China	SVR_BASE†	0.778	0.930	0.674	0.776	0.919	0.678
	SVR_GSAD	0.726	0.805	0.653	0.723	0.798	0.645
	SVR_IQR	0.761	0.854	0.700	0.757	0.850	0.707
	SVR_Cooks	0.761	0.887	0.670	0.759	0.877	0.674
	RF_BASE†	0.787	0.946	0.688	0.779	0.926	0.687
	RF_GSAD	0.741	0.846	0.651	0.731	0.832	0.642
	RF_IQR	0.778	0.897	0.688	0.776	0.884	0.682
	RF_Cooks	0.767	0.903	0.660	0.755	0.878	0.655
	Ridge_BASE†	0.773	0.919	0.667	0.773	0.918	0.664
	Ridge_GSAD	0.725	0.807	0.644	0.724	0.805	0.651
	Ridge_IQR	0.761	0.859	0.681	49.588	0.856	0.686
	Ridge_Cooks	0.759	0.881	0.663	54.831	0.879	0.655
	KNN_BASE†	0.839	1.089	0.717	0.830	1.068	0.692
	KNN_GSAD	0.788	0.955	0.677	0.767	0.915	0.659
	KNN_IQR	0.814	1.029	0.692	0.802	1.005	0.683
	KNN_Cooks	0.806	1.011	0.696	0.806	1.010	0.685
	GBM_BASE†	0.792	0.963	0.697	0.780	0.941	0.665
	GBM_GSAD	0.752	0.861	0.658	0.745	0.840	0.638
	GBM_IQR	0.788	0.903	0.702	0.778	0.873	0.718
	GBM_Cooks	0.778	0.916	0.689	0.768	0.899	0.671
UCP	SVR_BASE†	0.069	0.006	0.066	0.062	0.005	0.057
	SVR_GSAD	0.070	0.006	0.070	0.062	0.005	0.064
	SVR_IQR	0.079	0.008	0.082	0.074	0.007	0.068
	SVR_Cooks	0.070	0.007	0.066	0.063	0.005	0.057
	RF_BASE†	0.049	0.004	0.037	0.044	0.004	0.030
	RF_GSAD	0.052	0.004	0.045	0.044	0.003	0.037
	RF_IQR	0.071	0.008	0.061	0.059	0.005	0.052
	RF_Cooks	0.052	0.005	0.037	0.044	0.004	0.030
	Ridge_BASE†	0.057	0.005	0.048	0.053	0.004	0.046
	Ridge_GSAD	0.052	0.003	0.046	0.047	0.003	0.043
	Ridge_IQR	0.054	0.004	0.046	77.506	0.003	0.033
	Ridge_Cooks	0.053	0.004	0.048	77.168	0.003	0.045
	KNN_BASE†	0.065	0.007	0.059	0.062	0.006	0.052
	KNN_GSAD	0.063	0.006	0.056	0.058	0.005	0.051
	KNN_IQR	0.074	0.008	0.069	0.065	0.006	0.048
	KNN_Cooks	0.068	0.007	0.059	0.062	0.006	0.054
	GBM_BASE†	0.051	0.005	0.039	0.050	0.004	0.040
	GBM_GSAD	0.053	0.004	0.043	0.045	0.003	0.037
	GBM_IQR	0.061	0.007	0.047	0.044	0.004	0.031
	GBM_Cooks	0.053	0.005	0.040	0.051	0.004	0.044
NASA93	SVR_BASE†	0.550	0.616	0.374	0.541	0.609	0.380
	SVR_GSAD	0.518	0.536	0.361	0.503	0.514	0.371
	SVR_IQR	0.302	0.130	0.286	0.271	0.106	0.263
	SVR_Cooks	0.519	0.528	0.375	0.497	0.503	0.369
	RF_BASE†	0.504	0.491	0.344	0.477	0.454	0.297
	RF_GSAD	0.484	0.436	0.354	0.440	0.340	0.344
	RF_IQR	0.341	0.191	0.294	0.283	0.130	0.207
	RF_Cooks	0.499	0.452	0.361	0.461	0.398	0.308
	Ridge_BASE†	0.443	0.389	0.317	0.429	0.371	0.320
	Ridge_GSAD	0.382	0.298	0.275	0.367	0.276	0.248
	Ridge_IQR	0.214	0.073	0.185	18.438	0.068	0.168
	Ridge_Cooks	0.376	0.305	0.243	30.364	0.291	0.221
	KNN_BASE†	0.589	0.628	0.460	0.567	0.589	0.422
	KNN_GSAD	0.535	0.498	0.426	0.524	0.443	0.412
	KNN_IQR	0.352	0.189	0.323	0.310	0.152	0.273
	KNN_Cooks	0.555	0.524	0.454	0.556	0.512	0.446
	GBM_BASE†	0.529	0.558	0.374	0.500	0.523	0.327
	GBM_GSAD	0.476	0.414	0.348	0.425	0.337	0.304
	GBM_IQR	0.318	0.177	0.257	0.298	0.155	0.252
	GBM_Cooks	0.514	0.495	0.381	0.504	0.451	0.409
ISBSG 2021	SVR_BASE†	0.076	0.006	0.082	0.070	0.006	0.074
	SVR_GSAD	0.063	0.006	0.082	0.077	0.006	0.084
	SVR_IQR	0.018	0.000	0.020	0.018	0.000	0.021
	SVR_Cooks	0.079	0.006	0.086	0.080	0.007	0.088
	RF_BASE†	0.018	0.001	0.009	0.018	0.001	0.009
	RF_GSAD	0.013	0.000	0.007	0.013	0.000	0.007
	RF_IQR	0.007	0.000	0.005	0.007	0.000	0.004
	RF_Cooks	0.015	0.000	0.008	0.014	0.000	0.008
	Ridge_BASE†	0.018	0.001	0.010	0.018	0.001	0.010
	Ridge_GSAD	0.014	0.000	0.009	0.014	0.000	0.009
	Ridge_IQR	0.008	0.000	0.006	0.000	0.000	0.006
	Ridge_Cooks	0.015	0.000	0.009	0.001	0.001	0.009
	KNN_BASE†	0.019	0.001	0.008	0.018	0.002	0.008
	KNN_GSAD	0.013	0.000	0.007	0.013	0.001	0.006
	KNN_IQR	0.007	0.000	0.005	0.007	0.000	0.004
	KNN_Cooks	0.015	0.000	0.008	0.015	0.001	0.007
	GBM_BASE†	0.019	0.001	0.010	0.019	0.001	0.010
	GBM_GSAD	0.013	0.000	0.007	0.014	0.000	0.008
	GBM_IQR	0.007	0.000	0.005	0.007	0.000	0.005
	GBM_Cooks	0.014	0.000	0.008	0.015	0.000	0.009

TABLE 8. Performance comparison of proposed SDEE_GSAD models and ATLM [80] in terms of MAE, MSE and MdAE for each dataset using 10 × 3 RCV and LOOCV. The models with superscript * represents that our SDEE_GSAD model performs better than ATLM model w.r.t given performance measure.

Dataset	Estimation Method	RCV			LOOCV		
		MAE	MSE	MdAE	MAE	MSE	MdAE
China	ATLM	0.784	0.923	0.710	0.784	0.927	0.691
	SVR_GSAD	0.726*	0.805*	0.653*	0.723*	0.798*	0.645*
	RF_GSAD	0.741*	0.846*	0.651*	0.731*	0.832*	0.642*
	Ridge_GSAD	0.725*	0.807*	0.644*	0.724*	0.805*	0.651*
	KNN_GSAD	0.788	0.955	0.677*	0.767*	0.915*	0.659*
	GBM_GSAD	0.752*	0.861*	0.658*	0.745*	0.840*	0.638*
UCP	ATLM	0.068	0.007	0.044	0.060	0.007	0.041
	SVR_GSAD	0.070	0.006*	0.070	0.062	0.005*	0.064
	RF_GSAD	0.052*	0.004*	0.045	0.044*	0.003*	0.037*
	Ridge_GSAD	0.052*	0.003*	0.046	0.047*	0.003*	0.043
	KNN_GSAD	0.063*	0.006*	0.056	0.058*	0.005*	0.051
	GBM_GSAD	0.053*	0.004*	0.043*	0.045*	0.003*	0.037*
NASA93	ATLM	0.435	0.448	0.269	0.448	0.491	0.231
	SVR_GSAD	0.518	0.536	0.361	0.503	0.514	0.371
	RF_GSAD	0.484	0.436*	0.354	0.440*	0.340*	0.344
	Ridge_GSAD	0.382*	0.298*	0.275	0.367*	0.276*	0.248
	KNN_GSAD	0.535	0.498	0.426	0.524	0.443*	0.412
	GBM_GSAD	0.476	0.414*	0.348	0.425	0.337*	0.304
ISBSG 2021	ATLM	0.077	0.096	0.066	0.076	0.094	0.062
	SVR_GSAD	0.063*	0.006*	0.082	0.077	0.006*	0.084
	RF_GSAD	0.013*	0.000*	0.007*	0.013*	0.000*	0.007*
	Ridge_GSAD	0.014*	0.000*	0.009*	0.014*	0.000*	0.009*
	KNN_GSAD	0.013*	0.000*	0.007*	0.013*	0.001*	0.006*
	GBM_GSAD	0.013*	0.000*	0.007*	0.014*	0.000*	0.008*

TABLE 9. The p-values from the Mann-Whitney U test between each pair of SDEE_GSAD vs SDEE_BASE models across 4 datasets over MAE and MSE for 10 × 5 RCV and LOOCV validation. The significant values have been highlighted in bold.

Dataset	Method	RCV		LOOCV	
		MAE	MSE	MAE	MSE
China	SVR_GSAD vs SVR_BASE	**0.000**	**0.000**	0.111	0.111
	RF_GSAD vs RF_BASE	**0.000**	**0.001**	0.100	0.100
	Ridge_GSAD vs Ridge_BASE	**0.000**	**0.000**	0.142	0.142
	KNN_GSAD vs KNN_BASE	**0.000**	**0.000**	0.085	0.085
	GBM_GSAD vs GBM_BASE	**0.000**	**0.000**	0.302	0.302
UCP	SVR_GSAD vs SVR_BASE	0.237	0.450	0.301	0.301
	RF_GSAD vs RF_BASE	0.295	0.109	0.267	0.267
	Ridge_GSAD vs Ridge_BASE	**0.001**	**0.001**	0.232	0.232
	KNN_GSAD vs KNN_BASE	0.237	**0.020**	0.369	0.369
	GBM_GSAD vs GBM_BASE	**0.018**	0.156	0.344	0.344
NASA93	SVR_GSAD vs SVR_BASE	0.129	0.088	0.380	0.380
	RF_GSAD vs RF_BASE	0.174	0.098	0.450	0.450
	Ridge_GSAD vs Ridge_BASE	**0.000**	**0.002**	0.167	0.167
	KNN_GSAD vs KNN_BASE	**0.010**	**0.004**	0.480	0.480
	GBM_GSAD vs GBM_BASE	**0.003**	**0.000**	0.288	0.288
ISBSG 2021	SVR_GSAD vs SVR_BASE	**0.018**	**0.014**	**0.000**	**0.000**
	RF_GSAD vs RF_BASE	**0.000**	**0.000**	**0.000**	**0.000**
	Ridge_GSAD vs Ridge_BASE	**0.000**	**0.000**	**0.000**	**0.000**
	KNN_GSAD vs KNN_BASE	**0.000**	**0.000**	**0.001**	**0.001**
	GBM_GSAD vs GBM_BASE	**0.000**	**0.000**	**0.000**	**0.000**

trained using such low volume of data might not reflect the ground truth.

Therefore, it is apparent that GSAD based outlier identification can compete with and even outperform the other outlier identification approaches for ML based SDEE methods.

The performance comparison with previous works is a difficult task for researchers in the field of SDEE due to lack of complete implementation details, discrepancies in usage of validation schemes and performance measures. However, we have compared the performance of our proposed models with ML models obtained as per previous study [79] and IQR [1] and Cooks distance [1], [45], [46] outlier identification and removal techniques based models. In addition to this, we have compared the results with well-known state-of-art baseline SDEE method Automatically Transformed Linear Model (ATLM) [80]. The ATLM model is regression based publicly available model. ATLM chooses between *log*, *sqrt*, and none type of transformation as per the skewness of dataset attributes. Most of the real SDEE datasets are of extremely heterogeneous nature with 0 being the minimum value and a very large maximum value. With 0 values, it is not possible to choose between *log*, *sqrt*, and none type of options for dataset transformation. Therefore, we needed to use $log1p$ transformation for implementation of ATLM models as well. Our proposed models' performances in comparison ATLM can be seen in Table 8. The results have been examined and shown in Table using three performance measures over two validation schemes for fair comparison. We observed

TABLE 10. **The p-values from the Mann-Whitney U test between each pair of SDEE_GSAD vs SDEE_IQR models across 4 datasets over MAE and MSE for 10 × 5 RCV and LOOCV validation. The significant values have been highlighted in bold.**

Dataset	Method	RCV		LOOCV	
		MAE	MSE	MAE	MSE
China	SVR_GSAD vs SVR_IQR	**0.000**	**0.016**	0.166	0.166
	RF_GSAD vs RF_IQR	**0.004**	0.067	0.078	0.078
	Ridge_GSAD vs Ridge_IQR	**0.001**	**0.016**	**0.000**	0.128
	KNN_GSAD vs KNN_IQR	**0.005**	**0.000**	0.260	0.260
	GBM_GSAD vs GBM_IQR	**0.001**	0.083	0.123	0.123
UCP	SVR_GSAD vs SVR_IQR	**0.003**	**0.001**	0.192	0.192
	RF_GSAD vs RF_IQR	**0.000**	**0.000**	0.101	0.101
	Ridge_GSAD vs Ridge_IQR	0.427	0.285	**0.000**	0.479
	KNN_GSAD vs KNN_IQR	**0.000**	**0.000**	0.349	0.349
	GBM_GSAD vs GBM_IQR	**0.024**	**0.011**	0.293	0.293
NASA93	SVR_GSAD vs SVR_IQR	**0.000**	**0.000**	**0.042**	**0.042**
	RF_GSAD vs RF_IQR	**0.000**	**0.000**	**0.024**	**0.024**
	Ridge_GSAD vs Ridge_IQR	**0.000**	**0.000**	**0.000**	**0.005**
	KNN_GSAD vs KNN_IQR	**0.000**	**0.000**	**0.004**	**0.004**
	GBM_GSAD vs GBM_IQR	**0.000**	**0.000**	0.094	0.094
ISBSG 2021	SVR_GSAD vs SVR_IQR	**0.000**	**0.000**	**0.000**	**0.000**
	RF_GSAD vs RF_IQR	**0.000**	**0.000**	**0.000**	**0.000**
	Ridge_GSAD vs Ridge_IQR	**0.000**	**0.000**	**0.000**	**0.000**
	KNN_GSAD vs KNN_IQR	**0.000**	**0.000**	**0.001**	**0.001**
	GBM_GSAD vs GBM_IQR	**0.000**	**0.000**	**0.000**	**0.000**

TABLE 11. **The p-values from the Mann-Whitney U test between each pair of SDEE_GSAD vs SDEE_Cooks models across 4 datasets over MAE and MSE for 10 × 5 RCV and LOOCV validation. The significant values have been highlighted in bold.**

Dataset	Method	RCV		LOOCV	
		MAE	MSE	MAE	MSE
China	SVR_GSAD vs SVR_Cooks	**0.000**	**0.000**	0.203	0.203
	RF_GSAD vs RF_Cooks	**0.013**	**0.012**	0.273	0.273
	Ridge_GSAD vs Ridge_Cooks	**0.001**	**0.002**	**0.000**	0.216
	KNN_GSAD vs KNN_Cooks	0.065	**0.010**	0.194	0.194
	GBM_GSAD vs GBM_Cooks	**0.018**	**0.028**	0.322	0.322
UCP	SVR_GSAD vs SVR_Cooks	0.387	0.280	0.323	0.323
	RF_GSAD vs RF_Cooks	0.056	0.255	0.266	0.266
	Ridge_GSAD vs Ridge_Cooks	**0.002**	**0.002**	**0.000**	0.375
	KNN_GSAD vs KNN_Cooks	**0.009**	**0.001**	0.386	0.386
	GBM_GSAD vs GBM_Cooks	0.103	0.295	0.248	0.248
NASA93	SVR_GSAD vs SVR_Cooks	0.387	0.285	0.453	0.453
	RF_GSAD vs RF_Cooks	0.142	0.163	0.434	0.434
	Ridge_GSAD vs Ridge_Cooks	0.468	0.305	**0.000**	0.267
	KNN_GSAD vs KNN_Cooks	0.159	0.353	0.360	0.360
	GBM_GSAD vs GBM_Cooks	**0.029**	**0.037**	0.115	0.115
ISBSG 2021	SVR_GSAD vs SVR_Cooks	**0.013**	**0.000**	**0.000**	**0.000**
	RF_GSAD vs RF_Cooks	**0.000**	0.392	**0.000**	**0.000**
	Ridge_GSAD vs Ridge_Cooks	**0.000**	0.114	**0.000**	0.362
	KNN_GSAD vs KNN_Cooks	**0.000**	**0.018**	**0.009**	**0.009**
	GBM_GSAD vs GBM_Cooks	**0.000**	0.285	**0.014**	**0.014**

TABLE 12. **Win-tie-loss results of Mann-Whitney U test obtained by comparing MAE and MSE distributions using 10 × 3 RCV and LOOCV validation.**

SDEE_GSAD vs.	RCV						LOOCV					
	MAE			MSE			MAE			MSE		
	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss	Win	Tie	Loss
SDEE_BASE	14	5	1	14	5	1	4	15	1	4	15	1
SDEE_IQR	9	1	10	7	3	10	3	9	8	0	11	9
SDEE_Cooks	12	8	0	8	10	2	7	12	1	1	16	3
Total	35	14	11	29	18	13	14	36	10	5	42	13

some instabilities in the performance of ATLM. It resulted with extremely high error values for some validation folds of the studied dataset especially ISBSG, so after removing the results of those outlying folds, ATLM models' results came down to those which have been reported in Table 8. We can see that our proposed models have much better performances than ATLM especially for China, UCP and ISBSG datasets. It can further be concluded that GSAD based our proposed ML models have the potential to compete with previously proposed ATLM method.

Computational Code Availability: The computational code have been made freely available online at: Code Archive_GSAD.

VI. THREATS TO VALIDITY

There are multiple factors which may introduce biasness to the validity of an empirical study. This section discusses these threats in respect to internal, construct, conclusion and external validity pertaining to our study and the measures that have been taken to address these threats.

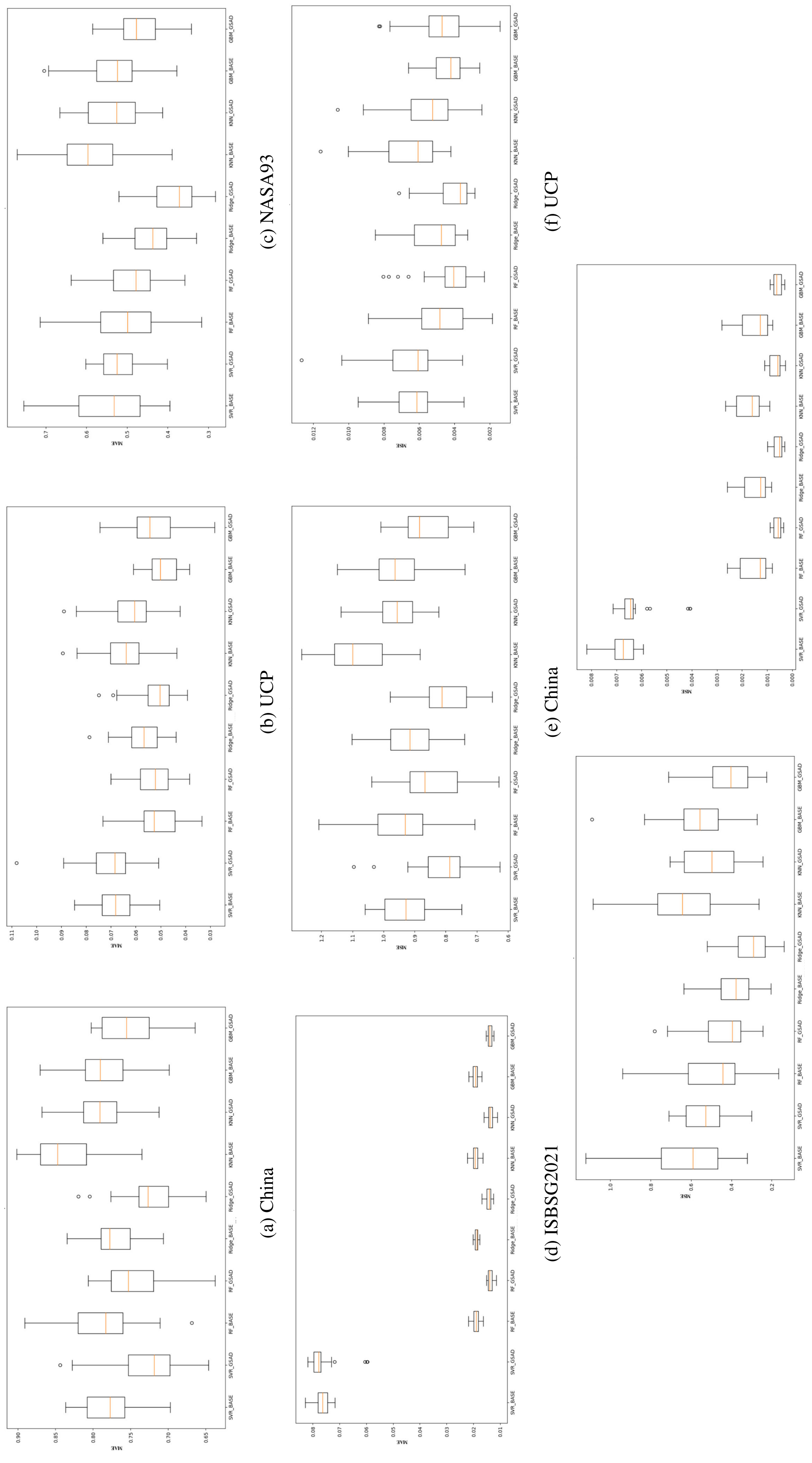


FIGURE 5. The boxplots of each pair of SDEE_BASE and SDEE_GSAD models across 4 datasets over MAE (a-d), MSE (e-h) for 10 × 3 RCV validation.

A. INTERNAL VALIDITY

The major threat to internal validity is concerned with selection bias of data studied. We have used 4 datasets that follow different model-based methodologies. China and ISBSG 2021 datasets are FP based, NASA93 is LOC based, while another dataset belongs to UCP methodology. These datasets also vary in terms of dimensions, application domain, size, complexity. Thus, for this study, internal threat is not a cause of major concern.

B. CONSTRUCT VALIDITY

With regards to construct validity, a possible threat of emphasis is the verification biasness [18]. To avoid this while performing empirical analysis, we have used a wide variety of unbiased error and accuracy measures. Each of these measures are covering a distinct specific aspect of performance measurement.

C. CONCLUSION VALIDITY

Conclusion validity is related to the degree of variability of the results with different experimental settings. To address this threat, we have worked on sensitivity analysis [70] in this study. Where, we have repeated all the analysis using different cross-validation schemes (RCV and LOOCV). In addition, we have performed the experiments over individual dataset on all 5 ML based SDEE methods using common data splits. Through this notion, we have attempted to rule out any possibility of performance improvement due to randomness.

D. EXTERNAL VALIDITY

The major potential threat concerning external validity is related to the generalization of findings. We have performed all the experiments over a wide variety of datasets including with-in and cross-company data. Therefore, we believe that the results of this study would be helpful in generalizing the findings for homogeneous as well heterogeneous datasets ranging in different sizes and domains.

VII. CONCLUSION AND FUTURE WORK

The accuracy of estimates has significant place in software industry so as to deliver quality software. Since inaccurate estimates cause compromised quality of software at later stage of its development. In this research work, we have proposed a novel approach of adaptive discretization using GS to effectively address outlier detection. We have examined the potential of removing outliers using GSAD in improving the performance of 5 SDEE methods based on SVR, RF, Ridge, KNN, and GBM over 4 benchmark real industrial SDEE datasets comprising with-in and cross-company data using 2 different cross-validation methods (namely 10×3 RCV, and LOOCV). The empirical analysis show that the performance of models is less likely to be sensitive to randomness resulting due to different validation and data splitting approach. In addition, the statistical tests show that performance has been significantly improved for several cases in different validation settings. This signifies that the importance of outlier removal can not be overlooked for SDEE dataset. For the completeness, we have also compared the performance of our GSAD based SDEE methods with IQR and Cooks' distance based SDEE methods. Overall, the proposed GSAD approach is efficient and competitive in enabling a simple yet effective outlier identification and removal procedure to improve the performance of investigated SDEE methods.

The proposed discretization based outlier identification method GSAD works well irrespective of the linearity of data and it does not require class labels prior discretization. Also, it is independent of distributional dependence.

In this study, we have performed extensive empirical analysis of our proposed GSAD based outlier identification and removal along with prominent SDEE methods, yet there are some options which can be explored. The future direction is to consider the correction of identified outliers instead of removing them altogether, specifically in small datasets with high percentage of outliers. We are planning to integrate the finding of the optimum value of outlier frequency threshold threshold for other datasets of different dimensions especially when the dataset has large number of categorical data. We are also determined to further investigate the impact of incorporating multiple nominal variables such as language type, organization type, application type etc. along with numeric variables on estimation performance.

In this paper, we have investigated the applicability of GSAD for SDEE, while this novel outlier identification method can also be adapted in other fields of research to assess its suitability in order to identify optimal number of outliers.

ACKNOWLEDGMENT

The authors are highly grateful to Ministry of Education, Government of India and Indian Institute of Information Technology Allahabad for financial support in this research work. The authors would like to thank the International Software Benchmarking Standards Group (ISBSG) for providing industrial dataset for research purposes.

REFERENCES

[1] Y.-S. Seo and D.-H. Bae, "On the value of outlier elimination on software effort estimation research," *Empirical Softw. Eng.*, vol. 18, no. 4, pp. 659–698, Aug. 2013.

[2] A. Kaushik, P. Kaur, and N. Choudhary, "Stacking regularization in analogy-based software effort estimation," *Soft Comput.*, vol. 26, pp. 1–20, Jan. 2022.

[3] Y.-S. Seo, K.-A. Yoon, and D.-H. Bae, "An empirical analysis of software effort estimation with outlier elimination," in *Proc. 4th Int. Workshop Predictor Models Softw. Eng. (PROMISE)*, 2008, pp. 25–32.

[4] M. M. Rosli, E. Tempero, and A. Luxton-Reilly, "Can we trust our results? A mapping study on data quality," in *Proc. 20th Asia–Pacific Softw. Eng. Conf. (APSEC)*, vol. 1, Dec. 2013, pp. 116–123.

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

[6] M. Ochodek, J. Nawrocki, and K. Kwarciak, "Simplifying effort estimation based on use case points," *Inf. Softw. Technol.*, vol. 53, no. 3, pp. 200–213, Mar. 2011.

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

[8] M. F. Bosu and S. G. MacDonell, "Experience: Quality benchmarking of datasets used in software effort estimation," *J. Data Inf. Qual.*, vol. 11, no. 4, pp. 1–38, 2019.

[9] G. A. Liebchen and M. Shepperd, "Software productivity analysis of a large data set and issues of confidentiality and data quality," in *Proc. 11th IEEE Int. Softw. Metrics Symp. (METRICS)*, Sep. 2005, p. 5.

[10] S. Morasca, "Building statistically significant robust regression models in empirical software engineering," in *Proc. 5th Int. Conf. Predictor Models Softw. Eng. (PROMISE)*, 2009, pp. 1–10.

[11] K.-A. Yoon and D.-H. Bae, "A pattern-based outlier detection method identifying abnormal attributes in software project data," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 137–151, Feb. 2010.

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

[13] S. S. Gautam and V. Singh, "The state-of-the-art in software development effort estimation," *J. Softw., Evol. Process*, vol. 30, no. 12, p. e1983, Dec. 2018.

[14] A. Ali and C. Gravino, "A systematic literature review of software effort prediction using machine learning methods," *J. Softw., Evol. Process*, vol. 31, no. 10, p. e2211, Oct. 2019.

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

[16] M. Jørgensen, "A review of studies on expert estimation of software development effort," *J. Syst. Softw.*, vol. 70, nos. 1–2, pp. 37–60, 2004.

[17] C. Mair and M. Shepperd, "The consistency of empirical comparisons of regression and analogy-based software project cost prediction," in *Proc. Int. Symp. Empirical Softw. Eng.*, 2005, p. 10.

[18] T. Menzies and M. Shepperd, "Special issue on repeatable results in software engineering prediction," *Empirical Softw. Eng.*, vol. 17, nos. 1–2, pp. 1–17, Feb. 2012.

[19] I. Myrtveit and E. Stensrud, "Validity and reliability of evaluation procedures in comparative studies of effort prediction models," *Empirical Softw. Eng.*, vol. 17, nos. 1–2, pp. 23–33, 2012.

[20] I. Myrtveit, E. Stensrud, and M. Shepperd, "Reliability and validity in comparative studies of software prediction models," *IEEE Trans. Softw. Eng.*, vol. 31, no. 5, pp. 380–391, May 2005.

[21] N. Mittas and L. Angelis, "Ranking and clustering software cost estimation models through a multiple comparisons algorithm," *IEEE Trans. Softw. Eng.*, vol. 39, no. 4, pp. 537–551, Apr. 2013.

[22] B. A. Kitchenham, L. M. Pickard, S. G. MacDonell, and M. J. Shepperd, "What accuracy statistics really measure," *IEE Proc.-Softw.*, vol. 148, no. 3, pp. 81–85, 2001.

[23] L. L. Minku and X. Yao, "Software effort estimation as a multiobjective learning problem," *ACM Trans. Softw. Eng. Methodol.*, vol. 22, no. 4, pp. 1–32, Oct. 2013.

[24] F. Sarro and A. Petrozziello, "Linear programming as a baseline for software effort estimation," *ACM Trans. Softw. Eng. Methodol.*, vol. 27, no. 3, pp. 1–28, Oct. 2018.

[25] J. W. Keung, B. A. Kitchenham, and D. R. Jeffery, "Analogy-X: Providing statistical inference to analogy-based software cost estimation," *IEEE Trans. Softw. Eng.*, vol. 34, no. 4, pp. 471–484, Jul. 2008.

[26] V. K. Y. Chan and W. E. Wong, "Outlier elimination in construction of software metric models," in *Proc. ACM Symp. Appl. Comput. (SAC)*, 2007, pp. 1484–1488.

[27] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, "A simulation study of the model evaluation criterion MMRE," *IEEE Trans. Softw. Eng.*, vol. 29, no. 11, pp. 985–995, Nov. 2003.

[28] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," in *Proc. 5th Int. Conf. Predictor Models Softw. Eng. (PROMISE)*, 2009, pp. 1–5.

[29] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," *Inf. Softw. Technol.*, vol. 54, no. 8, pp. 820–827, Aug. 2012.

[30] A. Idri, I. Abnane, and A. Abran, "Evaluating pred (p) and standardized accuracy criteria in software development effort estimation," *J. Softw., Evol. Process*, vol. 30, no. 4, p. e1925, 2018.

[31] B. Kitchenham, S. MacDonell, L. Pickard, and M. Shepperd, *Assessing Prediction Systems* (The Information Science Discussion Paper Series). Univ. Otago, 1999.

[32] P. Silhavy, R. Silhavy, and Z. Prokopova, "Outliners detection method for software effort estimation models," in *Proc. Comput. Sci. On-Line Conf.* Cham, Switzerland: Springer, 2019, pp. 444–455.

[33] A. B. Nassif, M. Azzeh, A. Idri, and A. Abran, "Software development effort estimation using regression fuzzy models," *Comput. Intell. Neurosci.*, vol. 2019, pp. 1–17, Feb. 2019.

[34] K. Ono, M. Tsunoda, A. Monden, and K. Matsumoto, "Influence of outliers on estimation accuracy of software development effort," *IEICE Trans. Inf. Syst.*, vol. 104, no. 1, pp. 91–105, 2021.

[35] S. Kotsiantis and D. Kanellopoulos, "Discretization techniques: A recent survey," *GESTS Int. Trans. Comput. Sci. Eng.*, vol. 32, no. 1, pp. 47–58, 2006.

[36] A. Halder, P. Rakshit, S. Chakraborty, A. Konar, A. Chakraborty, E. Kim, and A. K. Nagar, "Reducing uncertainty in interval type-2 fuzzy sets for qualitative improvement in emotion recognition from facial expressions," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Jun. 2012, pp. 1–8.

[37] J. Gama and C. Pinto, "Discretization from data streams: Applications to histograms and data mining," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 662–667.

[38] R. D. Cook, "Detection of influential observation in linear regression," *Technometrics*, vol. 19, no. 1, pp. 15–18, Feb. 1977.

[39] M. Livio, *The Golden Ratio: The Story of PHI, the World's Most Astonishing Number*. New York, NY, USA: Crown, 2008.

[40] R. S. F. Patron, R. M. Botez, and D. Labour, "Vertical profile optimization for the flight management system CMA-9000 using the golden section search method," in *Proc. 38th Annu. Conf. IEEE Ind. Electron. Soc. (IECON)*, Oct. 2012, pp. 5482–5488.

[41] A. Kheldoun, R. Bradai, R. Boukenoui, and A. Mellit, "A new golden section method-based maximum power point tracking algorithm for photovoltaic systems," *Energy Convers. Manag.*, vol. 111, pp. 125–136, Mar. 2016.

[42] N. López, M. Núñez, I. Rodríguez, and F. Rubio, "Introducing the golden section to computer science," in *Proc. 1st IEEE Int. Conf. Cogn. Inform.*, Aug. 2002, pp. 203–212.

[43] A. Benavoli, L. Chisci, and A. Farina, "Fibonacci sequence, golden section, Kalman filter and optimal control," *Signal Process.*, vol. 89, no. 8, pp. 1483–1488, Aug. 2009.

[44] C. He, Y. F. Zheng, and S. C. Ahalt, "Object tracking using the Gabor wavelet transform and the golden section algorithm," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 528–538, Dec. 2002.

[45] E. Mendes and C. Lokan, "Replicating studies on cross-vs single-company effort models using the ISBSG database," *Empirical Softw. Eng.*, vol. 13, no. 1, pp. 3–37, 2008.

[46] S. Mensah, J. Keung, S. G. MacDonell, M. F. Bosu, and K. E. Bennin, "Investigating the significance of the bellwether effect to improve software effort prediction: Further empirical study," *IEEE Trans. Rel.*, vol. 67, no. 3, pp. 1176–1198, Sep. 2018.

[47] L. L. Minku, "Multi-stream online transfer learning for software effort estimation: Is it necessary?" in *Proc. 17th Int. Conf. Predictive Models Data Anal. Softw. Eng.*, Aug. 2021, pp. 11–20.

[48] (2019). *Kaggle, Kaggle: Your Home for Data Science*. [Online]. Available: https://www.kaggle.com

[49] M. Hosni, A. Idri, A. Abran, and A. B. Nassif, "On the value of parameter tuning in heterogeneous ensembles effort estimation," *Soft Comput.*, vol. 22, no. 18, pp. 5977–6010, Sep. 2018.

[50] L. Villalobos-Arias, C. Quesada-López, J. Guevara-Coto, A. Martínez, and M. Jenkins, "Evaluating hyper-parameter tuning using random search in support vector machines for software effort estimation," in *Proc. 16th ACM Int. Conf. Predictive Models Data Anal. Softw. Eng.*, Nov. 2020, pp. 31–40.

[51] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[52] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Trans. Softw. Eng.*, vol. 23, no. 11, pp. 736–743, Nov. 1997.

[53] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung, "Exploiting the essential assumptions of analogy-based effort estimation," *IEEE Trans. Softw. Eng.*, vol. 38, no. 2, pp. 425–438, Mar. 2012.

[54] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[55] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, and J. Vanderplas, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[57] (2021). *The International Software Benchmarking Standards Group*. [Online]. Available: http://www.isbsg.org

[58] T. Menzies, "NASA93," Zenodo, Switzerland, Tech. Rep., Feb. 2008. Accessed: Nov. 5, 2021, doi: 10.5281/zenodo.268419.

[59] F. H. Yun, "China: Effort estimation dataset," Zenodo, Switzerland, Tech. Rep., Apr. 2010. Accessed: Nov. 5, 2021, doi: 10.5281/zenodo.268446.

[60] R. Silhavy, P. Silhavy, and Z. Prokopova, "Use case points benchmark dataset," Zenodo, Switzerland, Tech. Rep., Mar. 2017. Accessed: Nov. 5, 2021, doi: 10.5281/zenodo.344959.

[61] I. Attarzadeh, A. Mehranzadeh, and A. Barati, "Proposing an enhanced artificial neural network prediction model to improve the accuracy in software effort estimation," in *Proc. 4th Int. Conf. Comput. Intell., Commun. Syst. Netw.*, Jul. 2012, pp. 167–172.

[62] R. Silhavy, P. Silhavy, and Z. Prokopova, "Evaluating subset selection methods for use case points estimation," *Inf. Softw. Technol.*, vol. 97, pp. 1–9, May 2018.

[63] L. Song, L. L. Minku, and X. Yao, "Software effort interval prediction via Bayesian inference and synthetic bootstrap resampling," *ACM Trans. Softw. Eng. Methodol.*, vol. 28, no. 1, pp. 1–46, Feb. 2019.

[64] *Guidelines for Use of the ISBSG Data Release 2021*, ISBSG, Melbourne, VIC, Australia, 2021.

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

[66] K. Korenaga, A. Monden, and Z. Yucel, "Data smoothing for software effort estimation," in *Proc. 20th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jul. 2019, pp. 501–506.

[67] B. W. Boehm, *Software Engineering Economics*, vol. 197. Englewood Cliffs, NJ, USA: Prentice-Hall, 1981.

[68] K. Ono, M. Tsunoda, A. Monden, and K. Matsumoto, "Influence of outliers on analogy based software development effort estimation," in *Proc. IEEE/ACIS 15th Int. Conf. Comput. Inf. Sci. (ICIS)*, Jun. 2016, pp. 1–6.

[69] M. H. Siegel, "Lecture twenty," in *TTC the Joy of Mathematics (Part II)*. Chantilly, VA, USA: Teaching Company, 1999.

[70] F. Campolongo, A. Saltelli, and S. Tarantola, "Sensitivity anaysis as an ingredient of modeling," *Stat. Sci.*, vol. 15, no. 4, pp. 377–395, Nov. 2000.

[71] M. Shepperd and G. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Trans. Softw. Eng.*, vol. 27, no. 11, pp. 1014–1022, Nov. 2001.

[72] W. B. Langdon, J. Dolado, F. Sarro, and M. Harman, "Exact mean absolute error of baseline predictor, MARP0," *Inf. Softw. Technol.*, vol. 73, pp. 16–18, May 2016.

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

[74] M. Choetkiertikul, H. K. Dam, T. Tran, T. Pham, A. Ghose, and T. Menzies, "A deep learning model for estimating story points," *IEEE Trans. Softw. Eng.*, vol. 45, no. 7, pp. 637–656, Jul. 2019.

[75] J. Cohen, "Quantitative methods in psychology: A power primer," in *Psychological Bulletin*, vol. 112, no. I, 1992, pp. 155–159.

[76] J. P. Royston, "An extension of Shapiro and Wilk's *W* test for normality to large samples," *J. Roy. Stat. Soc. C, Appl. Statist.*, vol. 31, no. 2, pp. 115–124, 1982.

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

[78] F. Sarro, A. Petrozziello, and M. Harman, "Multi-objective software effort estimation," in *Proc. IEEE/ACM 38th Int. Conf. Softw. Eng. (ICSE)*, May 2016, pp. 619–630.

[79] P. Phannachitta and K. Matsumoto, "Model-based software effort estimation—A robust comparison of 14 algorithms widely used in the data science community," *Int. J. Innov. Comput. Inf. Control*, vol. 15, no. 2, pp. 569–589, 2019.

[80] P. A. Whigham, C. A. Owen, and S. G. Macdonell, "A baseline model for software effort estimation," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 3, pp. 1–11, May 2015.

SWARNIMA SINGH GAUTAM is currently pursuing the Ph.D. degree with the Department of Information Technology, Indian Institute of Information Technology Allahabad. She has been working in the field of software development effort estimation. Her research interests include empirical software engineering and statistical machine learning.

VRIJENDRA SINGH (Senior Member, IEEE) is currently working as a Professor at the Department of Information Technology, Indian Institute of Information Technology Allahabad, Prayagraj, India. He has more than 20 years of experience in academics (teaching/training), research and development, and consulting. He has developed various intelligent systems/software's for Indian Industries/ Research Organizations like TVS Motors Company, BARC, ISRO, SAIL, IOCL, and NPTEL. He has transferred technology to TVS Motors Company, Hosur, India, for testing of two wheelers engine during assembly line based on acoustics data analytics in the year 2009. He has successfully deployed a system (under Government of India and the National Science Foundation (NSF) USA sponsored joint Project, 2013–2015) for Classification/Protection of Wildlife and Humans, Virtual Fencing, Gunshot Detection and Localization using the IoT devices and Sensor Communication Networks at Panna Tiger Reserve, Madhya Pradesh, India, in collaboration with the Indian Institute of Science Bangalore, the Wildlife Institute of India Dehradun, the Ohio State University, Cornell University, and the University of California, Los Angeles. He has authored more than 100 publications in reputed international journals, conference proceedings/book chapters, and edited three books. His research interests include data science and analytics, software project management, information security, social networks analysis, modeling, and simulations and optimization. He has been conferred with the "50 Fabulous Edutech Leaders Award" from the 8th World Education Congress, Mumbai, India, in 2019. He is currently serving as the President for the IEEE Computational Intelligence Society, UP Section, India.