MINN: A Missing Data Imputation Technique for Analogy-based Effort Estimation

Muhammad Arif Shah[1*], Dayang N. A. Jawawi[2], Mohd Adham Isa[3], Karzan Wakil[4], Muhammad Younas[5*], Ahmed Mustafa[6]

Department of Software Engineering, School of Computing, Faculty of Engineering
Universiti Teknologi Malaysia, Johor Bahru, Malaysia[1,2,3,5,6]
City University of Science and Information Technology, Peshawar, Pakistan[1]
Research Center, Sulaimani Polytechnic University, Sulaimani 46001, Kurdistan Region, Iraq[4]
Department of Computer Science, Government College University Faisalabad, Pakistan[5]

***Abstract*—Success and failure of a complex software project are strongly associated with the accurate estimation of development effort. There are numerous estimation models developed but the most widely used among those is Analogy-Based Estimation (ABE). ABE model follows human nature as it estimates the future project's effort by making analogies with the past project's data. Since ABE relies on the historical datasets, the quality of the datasets affects the accuracy of estimation. Most of the software engineering datasets have missing values. The researchers either delete the projects containing missing values or avoid treating the missing values which reduce the ABE performance. In this study, Numeric Cleansing (NC), K-Nearest Neighbor Imputation (KNNI) and Median Imputation of the Nearest Neighbor (MINN) methods are used to impute the missing values in Desharnais and DesMiss datasets for ABE. MINN technique is introduced in this study. A comparison among these imputation methods is performed to identify the suitable missing data imputation method for ABE. The results suggested that MINN imputes more realistic values in the missing datasets as compared to values imputed through NC and KNNI. It was also found that the imputation treatment method helped in better prediction of the software development effort on ABE model.**

***Keywords*—Analogy-based estimation; effort estimation; missing data imputation; software development**

I. Introduction

Software development effort estimation is an important and complex activity of project management. Be it planning, constructing or development, all aspects are affected by accurate effort estimation of software projects. There are various methods introduced for effort estimation by different researchers, but none could be called as the best method due to its dependency on various factors such as project feature, the available information, and the technique used. The basic aim of all the methods is to accurately estimate the project effort. Larry Putnam, Barry Boehm, and Joe Aron can be considered the pioneers of software effort estimation methods [1]. Barry Bohem introduced COCOMO after IBM's interactive productivity and quality (IPQ) and the manual rule of thumb of estimation [2]. Putnam Life Cycle Management (SLIM) and Software Estimation Model (SEER-SEM) adopted and used the principles of COCOMO. Albrecht and Gaffney [3], introduced Functional Point (FP) as one of the metrics for size estimation. Shepperd and Schofield [4], brought forward Analogy-Based Estimation (ABE) method which became very prominent due to its working based on human manners of problem-solving. Though ABE produced better results it still had to face some constraints such as lack of detailed information, with limited features, and unreal or unnecessary requirements. There are several studies which tried to overcome the issues of ABE through mathematical and statistical solutions [5-8]. Soft computing techniques are widely adopted in ABE by researchers to deal with the complicated nature of software projects and to understand the relationship between features [9-16].

This study focuses on the improvement of ABE through missing data imputation with a modified imputation technique. The deviation in some related studies is shown in Table 1.

A. Estimation by Analogy (ABE)

ABE or EBA was introduced as the non-algorithmic estimation method by Shepperd and Schofield [4]. It estimates the effort of a new project by comparing it with the historical projects. There are usually four parts of ABE,

- Historical Projects
- Similarity Function
- Solution Function
- Associated Retrieval Rule

Each of which can be described as:

- Collecting the data of previous projects to form a historical dataset.
- Selecting the project's appropriate features.
- Retrieving the data of past project to find similarities with the target project. The weighted Manhattan Distance and Euclidean Distance are usually preferred at this stage.
- To estimate the software development effort of the target project.

TABLE I. DEVIATION IN SOME RELATED STUDIES

Source	Numeric Cleansing	KNNI	MINN	ABE
[17]	✗	✓	✗	✗
[18]	✗	✓	✗	✓
[19]	✗	✓	✗	✓
This Study	✓	✓	✓	✓

1) Similarity Function: In ABE, to compare the features of two projects, a similarity function is used. Euclidean Similarity (ES) and Manhattan Similarity (MS) are the two prominent similarity functions used by ABE to find out the similarity between the target and the past projects Shepperd and Schofield [4]. The ES is shown in Equation (1).

$$Sim(p,p') = \frac{1}{\left[\sqrt{\sum_{i=1}^{n} w_i Dis(f_i, f'_i)} + \delta\right]} \quad \delta = 0.0001$$

$$Dis(f_i, f'_i) = \begin{cases} |f_i - f'_i| \text{ if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\ 0 \quad \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 1 \quad \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i =\neq f'_i \end{cases} \quad (1)$$

Where *Sim* stands for similarity and *Dis* stands for distance, $\boldsymbol{p}$ and $\boldsymbol{p'}$ represent the projects to be compared, $\mathbf{w}_i$ is the weight allocated to the features which can range between $\boldsymbol{0}$ to $\boldsymbol{1}$. $\boldsymbol{\delta}$ is used to retrieve a non-zero result. The $\boldsymbol{f_i}$ and $\boldsymbol{f_i'}$ represent the project features while n determines the number of features.

There are many similarities between MS and ES, but MS calculates the absolute difference between features. MS function is shown in Equation (2), whereas the variable description is the same as in Equation (1).

$$Sim(p,p') = \frac{1}{\left[\sum_{i=1}^{n} w_i Dis(f_i, f'_i) + \delta\right]} \quad \delta = 0.0001$$

$$Dis(f_i, f'_i) = \begin{cases} |f_i - f'_i| \text{ if } f_i \text{ and } f'_i \text{ are numeric or ordinal} \\ 0 \quad \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i = f'_i \\ 1 \quad \text{if } f_i \text{ and } f'_i \text{ are nominal and } f_i =\neq f'_i \end{cases} \quad (2)$$

2) Solution Function: Once the *K* most similar projects are chosen, it becomes possible to predict or estimate the effort of target project according to the selected attributes or features. The Closest Analogy [20], the median [21], the average and the inverse weighted mean of the most similar project are the most common solution functions [22]. The median refers to the median or effort for *K>2* similar projects, the mean refers to the average of effort for *K>1*. In estimation, the portion of each project is adjusted by the inverse distance weighted mean by Equation (3).

$$C_P = \sum_{K=1}^{K} \frac{Sim(p,p_k)}{\sum_{i=1}^{n} Sim(p,p_k)} \, Cp_k \quad (3)$$

Where the new project is depicted by $\boldsymbol{p}$, $\boldsymbol{p_k}$ shows the most similar project at ***kth***, $\boldsymbol{Cp_k}$ illustrates the value of effort of $\boldsymbol{k_{th}}$ $\boldsymbol{p_k}$ and the total number of the project is denoted by $\boldsymbol{K}$.

B. Missing Data Concept

In software projects, the prediction may be inaccurate due to incomplete information collected in the initial stages of the project. There are usually more than one technique employed to estimate the effort to be applied to a software project development [23]. However, the missing data in the historical datasets raise an issue for employing the estimation technique as it affects the accuracy (Strike et al. 2001). The missing values in the dataset lead to inaccurate effort prediction (Sentas Angelis, 2006). This section elaborates the missing data mechanism (such as the ways missing data may be confronted in a dataset) and the missing data techniques (i.e., to deal with the missing data).

1) Mechanisms of Missing Data:- Mechanisms of missing data or patterns of missing values are the assumption of the types and distribution missing of missing data [24]. This missingness mechanism identifies the imputation technique to be used [24]. Missing mechanisms helps to identify if the missingness has any impact on the key variable or not, and it determines the difficulty level of the missing data handling. Missing At Random (MAR), Missing Completely At Random (MCAR) and Missing Not At Random (MNAR) are the three mechanisms of missingness [25]. The three mechanisms can formally be presented as, a dataset being collected as B= (b*i*), 1 ≤i ≤N, in which there is not unobserved value. The missing portion if considered to have unobserved values in B, the M= (m_i) indicator is used for donating the observation outcome. When b_i is unobserved, the outcome is zero "0" and in case of observed it returns "1". It can be characterized by probability distribution (conditional) lf M for B, e.g. p(M | B, ψ), where the unknown parameters are represented by ψ. According to Song, et al. [24], in **MCAR** pattern of missingness the distribution of observed and missing values are not different, or it can be stated that in MCAR mechanism, missingness is independent of observed and missing values of B, e.g. p(M| B, ψ) = p(M, ψ). In **MAR,** the missingness pattern is not depended on missing values but dependent on the observed values. It has to be dependent on at least one of the variables as it does not follow the condition of MCAR. **MNAR**, which intends, the missing data is not dependent on any observed variable in the dataset, but it depends on the missing data itself.

2) Techniques for Missing Data: According to Song, et al. [24], there are three methods to deal with the missing data. Missing Data Ignoring, Missing Data Toleration, and Missing Data Imputation.

a) Missing Data Ignoring: In this technique, the missing data cases are deleted. Though this technique is widely employed due to its simplicity, it leads to biasness and does not utilize the dataset. Missing Data Ignoring can be recommended in the case of MCAR found in a dataset or with a low level of missing data [17, 24]

b) Missing Data Toleration: The strategy of this technique is based on the internal treatment where missing data in the dataset is tolerated and analysis is directly performed on the dataset. One such kind of toleration approach is to assign a NULL value to replace the missing piece of data [17, 18, 26].

c) Missing Data Imputation: There are various strategies employed for missing data imputation, in which the missing values found in the dataset are filled, which lets the complete dataset being analyzed. Out of the many imputation techniques, K Nearest Neighbor imputation (KNNI) is utilized in this study. KNNI is population imputation technique, which has successfully produced good results in software development effort estimation [18, 27]. This is quite a practical approach as it has no explicit assumptions for missing data mechanism. The complete cases of a dataset are considered as a donor for imputing the incomplete cases by this technique. KNNI replaces the values of incomplete cases of missing data with its aggregated values. The k nearest neighbors are determined by finding the distance between the complete cases and incomplete cases which measures the similarity between them. There have been used Manhattan Distance and Euclidean Distance to find nearest cases:

Manhattan Distance: It measures the distance by finding the sum of absolute differences between case a and case b with n attributes by the following Equation (4).

$$dis(a,b) = \sum_{i=1}^{n} |x_i - y_i| \quad (4)$$

Euclidean Distance: It calculates the distance between point a and point b by $\boldsymbol{n}$ number of attributes following the Equation (5).

$$dis(a,b) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad (5)$$

Rest of the paper is organized as Section II presents the related studies with ABE and the missing data in software engineering datasets and ABE. Section III Includes the experimental procedure. Section IV explains the results of MINN, NC and KNNI on ABE. Discussion on the results is shown in Section V whereas Section VI concludes the study and discusses some future work.

II. Related Work

ABE relies on past projects to estimate the effort of future projects; therefore, the quality of past project dataset makes a significant difference. In software engineering dataset there are usually some data missing which leads the ABE model for the wrong estimation. There are very few studies, focusing on missing data techniques and ABE together, however, missing data techniques have been studied on the software engineering datasets in general quite extensively. Idri, et al. [28] performed a systematic mapping study on missing data and software engineering datasets. It was found in their study that Missing Data Imputation method was used the most out of the three methods for dealing with missing data. Strike, et al. [29] studied missing data for regression-based estimation for all mechanisms of missingness and concluded that missing data imputation produces favourable outcomes in comparison with the other techniques of missing data. Cartwright, et al. [30] used Toleration technique of dealing with missing data for missing at random and missing not at random mechanisms. Twala and Cartwright [31] ensembled multiple imputation and KNN and concluded that their proposed approach improved the prediction accuracy on industrial software engineering datasets. Sentas and Angelis [32] used Expectation Maximization Regression based Imputation, and Multi-Logistic Regression-based imputation (MLR) methods and claimed that MLR shows higher accuracy than the other methods. Li, et al. [26] concentrated on Toleration technique for dealing with missing data on Missing Completely At Random to validate AQUA an ABE technique used for estimation. Song, et al. [24] studied KNN and Toleration techniques for all mechanisms of missingness and found that the missingness mechanism affects the performance of KNN and toleration. They showed that missing data has a very negative impact on estimation if the missingness is more than 40%. Idri, et al. [18] conducted a study to evaluate the impact of different missing data techniques on ABE using KNN. Huang, et al. [17] performed an empirical study on cross-validation of KNN imputation for software quality dataset, though the study compared KNN imputation and Mean imputation, it was specifically on software quality dataset, they did not focus on estimation or ABE. The related studies indicate the importance of imputing the missing data in past projects, especially for ABE. This motivates the research community to further work on improving the imputation techniques for better predicting the software development effort by ABE model.

III. Experimental Design

This estimation process lets ABE estimate the effort to be applied to a target project by making analogies with the historical projects for datasets with imputed missing values. In this study, there are three techniques used to impute the missing values in the historical projects. However, it focuses to find the effects of missing data imputation on ABE, and to compare the introduced imputation technique, Median Imputation of the Nearest Neighbor (MINN) with the Numeric Cleansing (NC) and KNNI to identify suitable imputation technique for ABE. The experimental procedure is divided into three steps for finding the best estimate. Such as **Step 1:** Imputing missing values in the dataset by the three imputation techniques (MINN, NC and KNNI) one by one**, Step 2:** estimating the effort through ABE by making analogies, **Step 3:** evaluating the estimation performance by MMRE and PRED as shown in Fig. 1.

In Step 1: The three techniques were interchangeably used to impute missing values in the datasets, so that performance of these techniques could be compared to identify the better imputation technique to be used with ABE for software development effort prediction. In Step 2, the algorithmic procedure of ABE was applied to the datasets with imputed values. The datasets (Desharnais and DeshMiss) with imputed values are infused in ABE initially, followed by the similarity function (Euclidian) to select the project's features. After applying similarity function, the solution function (Inverse weighted mean as in Equation (3)) was used to find the related project and calculated the effort with associated retrieval rule. In Step 3, the estimation accuracy was tested to validate to

find out which imputation methods outperforms the other. MMRE and PRED were used as the performance evaluation metrics.

A. Performance Accuracy Metrics

There are several metrics to evaluate the performance of estimation methods, such as Relative Error (RE), Magnitude of Relative Error (MRE), and Mean Magnitude of Relative Error (MMRE). MMRE is the most frequently used out of the discussed performance metrics. In [33], MMRE is defined as:

$$RE = (Estimated - Acutal)/Actual) \quad (6)$$

$$MRE = |Estimated - Actual|/(Actual) \quad (7)$$

$$MMRE = \sum MRE/N \quad (8)$$

$$PRED(X) = \frac{A}{N} \quad (9)$$

In Equation (8) and (9), N represents the number of projects, A represents the projects with MRE >= X. The level of X is usually kept at 0.25 in software development effort estimation. The main aim of all the effort estimation models is to increase Percentage of Prediction (PRED) and decrease MMRE. PRED (X) is another extensively used prediction accuracy indicator. PRED (X) shows the estimates percentage within actual values of X percent. X is usually set to 0.*25* which makes it possible to reveal the number of estimate portion within *25%* tolerance [4].

B. Dataset Description

There are two datasets employed in this study Desharnais [34], and DeshMiss. Desharnais is one of the prominent datasets used for different studies of software engineering. This dataset contains the data of 80 software projects. The data of 4 out of the 80 projects is partially missing (e.g. the values of some of the features are missing). There are 9 features in this dataset, the detail of which can be seen in Table 2.

The feature (effort) is taken as the dependent feature whereas rest of the features are treated as independent features

The TeamExp values of project number 38, 44 and the ManagerExp attribute values of project 38, 66 and 57 are missing in Desharnais dataset. The missing values of Desharnais dataset can be seen in Fig. 1.

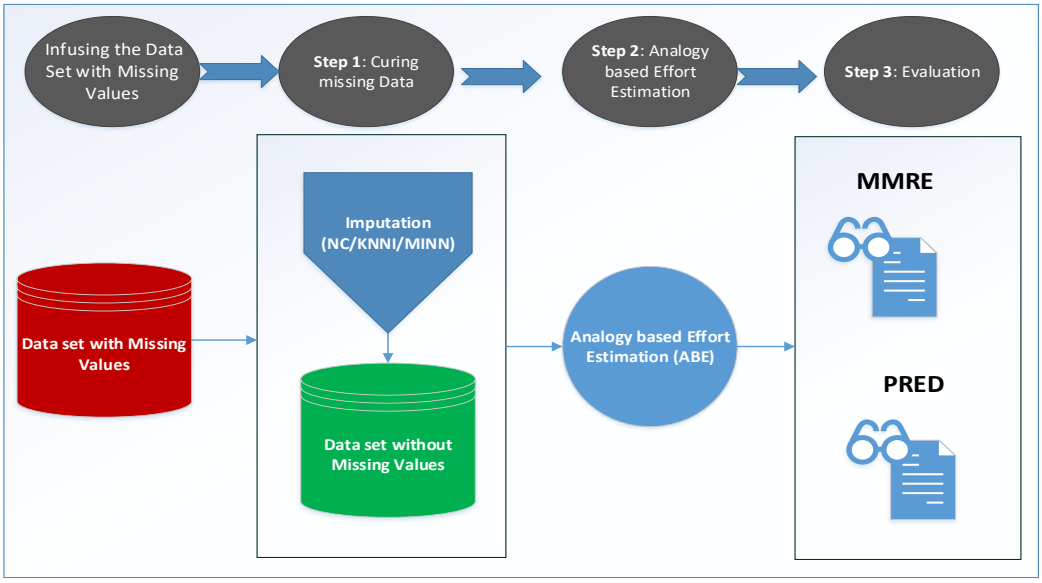


Fig. 1. MINN, Numeric Cleansing based and KNNI based Imputation and the Estimation Process.

TABLE II. DESCRIPTION OF DESHARNAIS DATASET

Feature	Description	Min	Max	Mean	Std Dev
Effort	Development Effort in person-hours	546	23940	4923.516	4646.751
TeamExp	Team Experience in Years	0	4	2.244	1.331
ManagerExp	Manager Experience in Years	0	7	2.803	1.47
Length	Length of Project in months	1	39	11.716	7.4
Transections	Number of Transactions	9	886	179.901	143.315
Entities	Number of Entities	7	387	122.726	86.178
PointsAdjust	Number of Adjusted Function Points	73	1127	311.014	189.185
Envergure	Function Point Complexity Adjustment factor	5	52	27.014	10.851
PointsNonAdjust	Project Size Measured In Unadjusted Function Points. (Entities Plus Transactions)	62	1116	295.765	197.937

Since the Desharnais dataset has a very low number of missing values, an artificial dataset was created similar to Desharnais with the name DeshMiss. In the DeshMiss dataset (Appendix A1), the number and type of features and projects are the same as of Desharnais but 7.22% (52 of the total 720) of the values are deleted with MNAR mechanism to validate the performance of both the imputation methods in the proposed estimation process used in this study. The artificial generation of such missing data has also been performed by studies such as Song, et al. [24], Strike, et al. [29] and Idri, et al. [18], Ali and Omer [35]. Further description of the DeshMiss dataset can be seen in Table 3. The Histogram and pattern of missing data for DeshMiss dataset can be seen in Fig. 2.

Both the imputation methods in the proposed estimation process are used in this study. The artificial generation of such missing data has also been performed by studies such as Song, et al. [24], Strike, et al. [29] and Idri, et al. [18], Ali and Omer [35]**.** Further Description of the DeshMiss dataset can be seen in Table 3. The Histogram and pattern of missing data for DeshMiss dataset can be seen in Fig. 3.

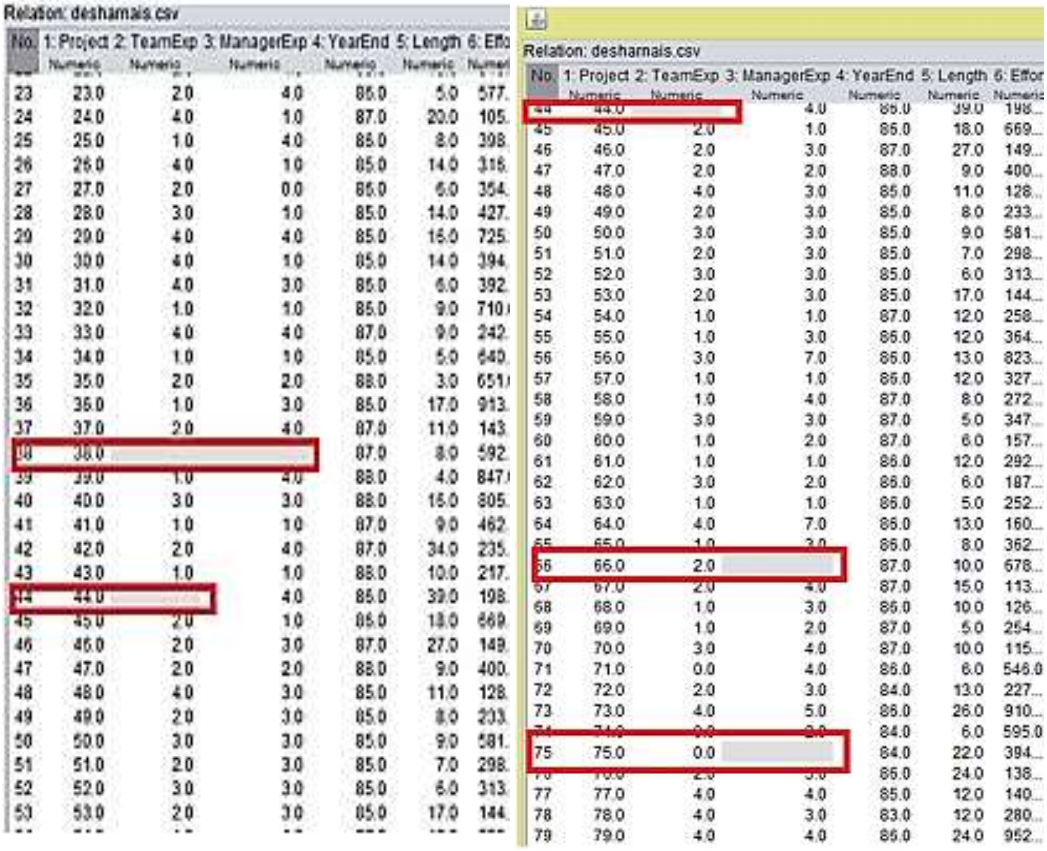


Fig. 2. Desharnais Dataset with Missing Values before Applying the Imputation Technique.

TABLE III. DESCRIPTION OF DESHMISS DATASET

Feature	Description	Min	Max	Mean	Std Dev
Effort	Development Effort in person-hours	546	23940	4924	4446.63
TeamExp	Team Experience in Years	0	4	2.282	1.338
ManagerExp	Manager Experience in Years	0	7	2.563	1.537
Length	Length of Project in months	1	36	10.811	6.188
Transections	Number of Transactions	9	886	183	146.79
Entities	Number of Entities	7	387	123.213	85.046
PointsAdjust	Number of Adjusted Function Points	73	1127	311.125	187.717
Envergure	Complexity Adjustment factor of Function Points	5	52	26.817	10.847
PointsNonAdjust	Project Size Measured In Unadjusted Function Points. (Entities Plus Transactions)	62	1116	200.447	182.676

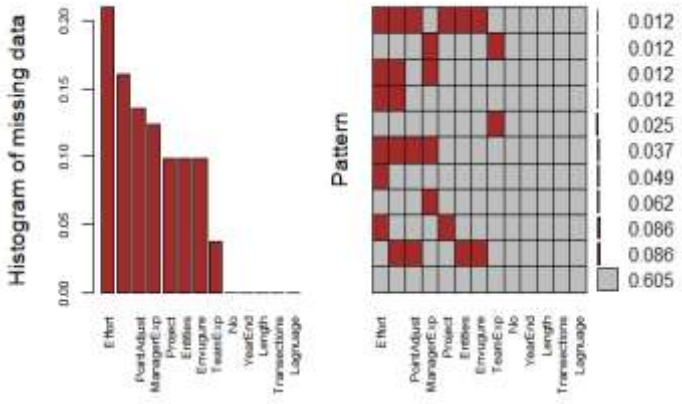


Fig. 3. Histogram and Pattern of Missing Data for DeshMiss dataset.

IV. Experimental Results

A. Median Imputation of the Nearest Neighbor (MINN)

Median Imputation of the Nearest Neighbor (MINN), is a technique introduced in this study for imputing the missing values in software engineering datasets. MINN is the form works the same way as KNNI but with a slight modification. KNNI imputes the missing values based on the neighbors nearest to the value of concern. It works quite productively for MAR or MCAR missingness mechanism but in the case of MNAR, its performance is reduced [17]. The K value for KNN is usually set to 5 or less, in some cases, more than 5 neighbors are chosen to perform imputation. If the missingness pattern or mechanism is MNAR e.g. a number of adjacent values are missing in multiple features, the KNNI imputes some unrealistic values which cause incorrect estimation. A flavor of KNNI also imputes values based on the mean of nearest neighbor but since the unrealistic imputation is continued it imputes slightly irrelevant values. In such a scenario MINN can be very useful, the adjacent missing values are taken in this technique and the median of these neighbors are imputed instead of the mean of all the values, or the random value of nearest neighbors. The procedure of MINN can be seen in Algorithm 1.

Algorithm 1. MINN Algorithm

REQUIRE: Divide the dataset (D) into two sets. D_M is the set with missing values (at least one feature missing). From a set (Dc) and Object O ϵ D_C will complete the feature information by the remaining features.

Step 1: **Begin**

Step 2: For each vector V in D_M:

i) Divide the instance vector into observed and missing parts as $V=[V_0, V_M]$
ii) Calculate $Dist(V_0, O)$, the distance between V_0 and O. Use only those features in O, which are observed in V
iii) Select the K nearest instances vectors (KNN) to V
iv) Replace the missing value using the MEDIAN of attributes

Step 3: **End**

Step 3 of the proposed estimation process shown in Fig. 1, MMRE and PRED were calculated to evaluate the accuracy of estimation in term relative error. As a result, the effects of MINN on ABE while using Desharnais dataset, the MMRE, and PRED values were calculated as 0.1496 and 0.8 respectively as shown in Table 4, and the effects of MINN on ABE while using DeshMiss dataset, the MMRE and PRED values were calculated as 0.0311 and .84 respectively, which can be seen in Table 4. The rate of success by PRED and MMRE value was significantly improved due to the relatively large number of missing data in the DeshMiss dataset.

B. Numeric Cleansing

Numeric cleaning or numeric cleansing is used as operations or filters in different tool for pre-processing the datasets; it also helps to cleanse the dataset with missing values. There are different strategies used by numeric cleansing to deal with the missing data such as using a placeholder, mean or any other values to replace the missing values or completely remove the column with missing values [36].

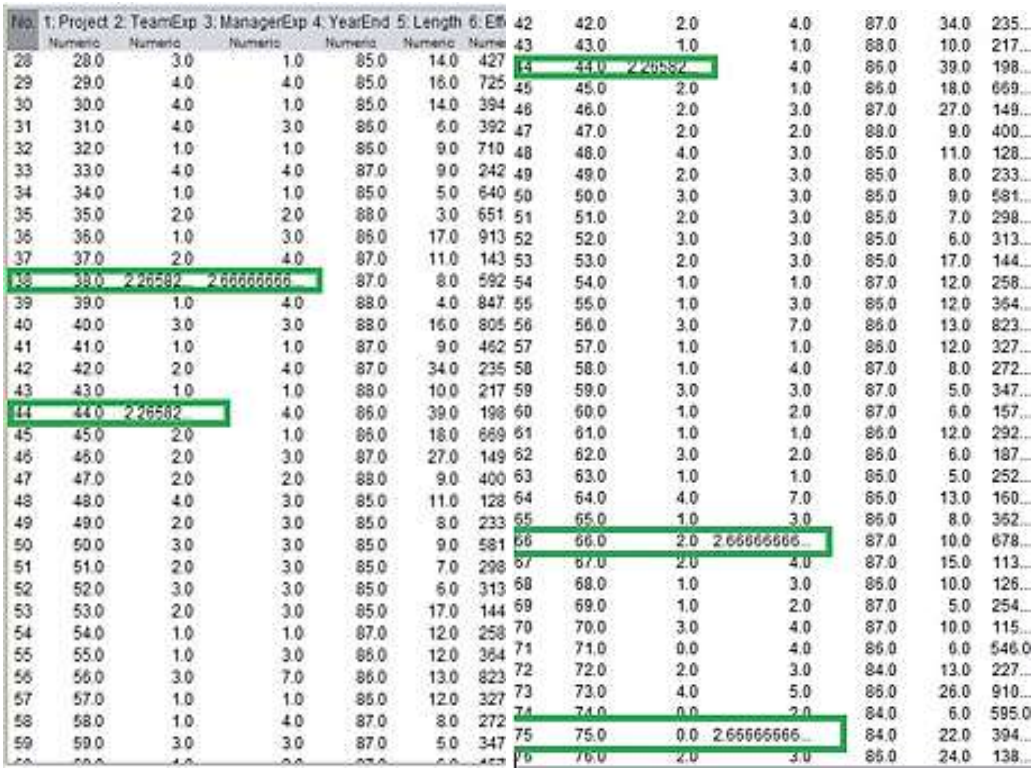
In this section, the numeric cleansing is performed on the datasets (Desharnais and DeshMiss) to increase the accuracy of predicting the development effort using ABE. Initially, in Step 1, the dataset with missing values is treated with numeric cleansing where the mean of the values is imputed in the dataset.

The same procedure when applied on Desharnais it imputed the mean value of the same attribute. It imputed *2.26582* in the *TeamExp* Column of project number *38 and 44* where the data was missing. In the same way, *2.6666666* was imputed in the *ManagerExp* column of project number *38, 66, and 75* through numeric cleansing. Fig. 2 shows the Desharnais dataset with missing values and Fig. 4 shows the dataset with the imputed mean values through numeric cleansing.

In Step 2, the analogy-based effort estimation is performed which is provided with the pre-processed (missing values imputed) Desharnais dataset. The ABE used ES to retrieve similar project based on attribute comparison. The solution function in ABE chose the most related project and calculated effort using the inverse distance weighted mean. In Step 3, the mean magnitude of relative error (MMRE) and PRED are calculated to evaluate the accuracy of estimation in term relative error. The complete estimation process where numeric cleansing was performed for imputation of the missing data is shown in Fig. 3.

TABLE IV. Effects of MINN on ABE for Desharnais and Desh Miss Datasets

Evaluation Metric	Desharnais Dataset	DeshMiss Dataset
MMRE	0.1496	0.0311
PRED (.25)	0.8	.84



No.	1: Project	2: TeamExp	3: ManagerExp	4: YearEnd	5: Length	6: Eff
	Numeric	Numeric	Numeric	Numeric	Numeric	Nume
28	28.0	3.0	1.0	85.0	14.0	427
29	29.0	4.0	4.0	85.0	16.0	725
30	30.0	4.0	1.0	85.0	14.0	394
31	31.0	4.0	3.0	86.0	6.0	392
32	32.0	1.0	1.0	86.0	9.0	710
33	33.0	4.0	4.0	87.0	9.0	242
34	34.0	1.0	1.0	85.0	5.0	640
35	35.0	2.0	2.0	88.0	3.0	651
36	36.0	1.0	3.0	86.0	17.0	913
37	37.0	2.0	4.0	87.0	11.0	143
38	38.0	226582	2.66666666	87.0	8.0	592
39	39.0	1.0	4.0	88.0	4.0	847
40	40.0	3.0	3.0	88.0	16.0	805
41	41.0	1.0	1.0	87.0	9.0	462
42	42.0	2.0	4.0	87.0	34.0	235
43	43.0	1.0	1.0	88.0	10.0	217
44	44.0	226582	4.0	86.0	39.0	198
45	45.0	2.0	1.0	86.0	18.0	669
46	46.0	2.0	3.0	87.0	27.0	149
47	47.0	2.0	2.0	88.0	9.0	400
48	48.0	4.0	3.0	85.0	11.0	128
49	49.0	2.0	3.0	85.0	8.0	233
50	50.0	3.0	3.0	85.0	9.0	581
51	51.0	2.0	3.0	85.0	7.0	298
52	52.0	3.0	3.0	85.0	6.0	313
53	53.0	2.0	3.0	85.0	17.0	144
54	54.0	1.0	1.0	87.0	12.0	258
55	55.0	1.0	3.0	86.0	12.0	364
56	56.0	3.0	7.0	86.0	13.0	823
57	57.0	1.0	1.0	86.0	12.0	327
58	58.0	1.0	4.0	87.0	8.0	272
59	59.0	3.0	3.0	87.0	5.0	347

42	42.0	2.0	4.0	87.0	34.0	235...
43	43.0	1.0	1.0	88.0	10.0	217...
44	44.0	226582	4.0	86.0	39.0	198...
45	45.0	2.0	1.0	86.0	18.0	669...
46	46.0	2.0	3.0	87.0	27.0	149...
47	47.0	2.0	2.0	88.0	9.0	400...
48	48.0	4.0	3.0	85.0	11.0	128...
49	49.0	2.0	3.0	85.0	8.0	233...
50	50.0	3.0	3.0	85.0	9.0	581...
51	51.0	2.0	3.0	85.0	7.0	298...
52	52.0	3.0	3.0	85.0	6.0	313...
53	53.0	2.0	3.0	85.0	17.0	144...
54	54.0	1.0	1.0	87.0	12.0	258...
55	55.0	1.0	3.0	86.0	12.0	364...
56	56.0	3.0	7.0	86.0	13.0	823...
57	57.0	1.0	1.0	86.0	12.0	327...
58	58.0	1.0	4.0	87.0	8.0	272...
59	59.0	3.0	3.0	87.0	5.0	347...
60	60.0	1.0	2.0	87.0	6.0	157...
61	61.0	1.0	1.0	86.0	12.0	292...
62	62.0	3.0	2.0	86.0	6.0	187...
63	63.0	1.0	1.0	86.0	5.0	252...
64	64.0	4.0	7.0	86.0	13.0	160...
65	65.0	1.0	3.0	86.0	8.0	362...
66	66.0	2.0	2.66666666...	87.0	10.0	678...
67	67.0	2.0	4.0	87.0	15.0	113...
68	68.0	1.0	3.0	86.0	10.0	126...
69	69.0	1.0	2.0	87.0	5.0	254...
70	70.0	3.0	4.0	87.0	10.0	115...
71	71.0	0.0	4.0	86.0	6.0	546.0
72	72.0	2.0	3.0	84.0	13.0	227...
73	73.0	4.0	5.0	86.0	26.0	910...
74	74.0	0.0	2.0	84.0	6.0	595.0
75	75.0	0.0	2.66666666...	84.0	22.0	394...
76	76.0	2.0	3.0	86.0	24.0	138...

Fig. 4. Desharnais Dataset with No Missing Values after Applying Numeric Cleansing.

In Step 3 of the proposed estimation process, MMRE and PRED were calculated to evaluate the accuracy of estimation in term relative error. As a result, the effects of NC on ABE while using Desharnais dataset, the MMRE, and PRED values were calculated as 0.1518 and 0.8 respectively which is also shown in Table 5.

As a result, the effects of NC on ABE while using DeshMiss dataset, the MMRE and PRED values were calculated as 0.0596 and 0.8 respectively which can be seen in Table 6. The rate of success by PRED for both the dataset remained the same but the MMRE value was significantly improved due to the relatively large number of missing data.

C. K Nearest Neighbor Imputation (KNNI)

The KNNI, as discussed in section 3.2.3 is used in place of Numerical Cleansing in Step 1 of the estimation process shown in Fig. 1, to impute the missing values in Desharnais (as shown in Fig. 2: dataset without imputed values). The KNNI imputed *2* in TeamExp column and *1* in ManagerExp column of the 38th project. It imputed *3* in TeamExp column of the *44th* project. In *66th* and *75th* project *2* and *7* were imputed respectively in the ManagerExp column. The values imputed through KNNI shows the natural effect due to its dynamic nature, unlike NC which imputes static values for all the missing information. The default value of K was utilized which is 6.

In Step 2, the analogy-based effort estimation is performed as it was performed for NC in section 5.2, which is provided with the pre-processed (missing values imputed) Desharnais and DeshMiss datasets one after another. The ABE used ES to retrieve a similar project based on attribute comparison. The solution function in ABE chose the most related project and calculated effort using the inverse distance weighted mean. In Step 3, the mean magnitude of relative error (MMRE) and PRED are calculated to evaluate the accuracy of estimation in term relative error. The complete estimation process where numeric cleansing was performed for imputation of the missing data is shown in Fig. 1.

The Step 3 of proposed estimation process shown in Fig. 1, MMRE and PRED were calculated to evaluate the accuracy of estimation in term relative error. As a result, the effects of KNNI on ABE while using Desharnais dataset, the MMRE, and PRED values were calculated as 0.1503 and 0.8 respectively as shown in Table 7.

TABLE V. EFFECTS OF NUMERIC CLEANSING ON ABE FOR DESHARNAIS DATASET

Evaluation Metric	**Numeric Cleansing**
MMRE	0.1518
PRED (.25)	0.8

TABLE VI. EFFECTS OF NUMERIC CLEANSING ON ABE FOR DESHMISS DATASET

Evaluation Metric	**Numeric Cleansing**
MMRE	0.0596
PRED (.25)	0.8

TABLE VII. EFFECTS OF KNNI ON ABE FOR DESHARNAIS DATASET

Evaluation Metric	KNNI
MMRE	0.1503
PRED (.25)	0.8

As a result, the effects of KNNI on ABE while using DeshMiss dataset, the MMRE and PRED values were calculated as 0.0323 and .84 respectively, which can be seen in Table 8. The rate of success by PRED and MMRE value was significantly improved due to the relatively large number of missing data in the DeshMiss dataset.

D. Effects of Numeric Cleansing, KNNI and MINN on ABE for Desharnais Dataset

The value of MMRE was calculated as 0.1518 for Numeric Cleansing based ABE whereas 0.1596 for KNNI and 0.1496 for MINN. The PRED (.25) was calculated as 0.8 for all three imputation techniques (Numeric cleansing, KNNI, MINN) based ABE, as shown in Table 9. The results are insignificantly improved due to the MCAR missingness mechanism.

E. Effects of Numeric Cleansing, KNNI and MINN on ABE for DeshMiss Dataset

The missing values in the DeshMiss dataset are high in number, unlike Desharnais dataset. The MMRE values were calculated for NC, KNNI, and MINN as 0.0596, 0.0323 and 0.0311 respectively. Whereas, the PRED values for NC, KNN and MINN were calculated as 0.80, 0.84 and 0.84 respectively, which shows a significant improvement. The results of MINN, KNNI, and NC no ABE for DeshMiss dataset are shown in Table 10.

TABLE VIII. EFFECTS OF NUMERIC CLEANSING ON ABE FOR DESHMISS DATASET

Evaluation Metric	KNNI
MMRE	0.0323
PRED (.25)	.84

TABLE IX. EFFECTS OF NUMERIC CLEANSING, KNNI AND MINN ON ABE FOR DESHARNAIS DATASET

Imputation Method / Evaluation Metric	Numeric Cleansing	KNNI	MINN
MMRE	0.1518	0.1503	0.1496
PRED (.25)	0.8	0.8	0.8

TABLE X. EFFECTS OF NUMERIC CLEANSING, KNNI AND MINN ON ABE FOR DESHMISS DATASET

Imputation Method / Evaluation Metric	Numeric Cleansing	KNNI	MINN
MMRE	0.0596	0.0323	0.0311
PRED (.25)	.80	.84	.84

V. DISCUSSION

The experimental results showed that, imputing the missing data have a positive impact on the overall performance of ABE. Moreover, the MINN showed better results against NC and KNN in imputing the missing data which is verified by the proposed estimation process. Though the proposed approach shows insignificant improvement in the ABE performance on the Desharnais dataset, it is due to the very low number of missing values in it. Therefore, the impact of MINN, KNNI, and NC could not be much differentiated initially. However, when the proposed approach was applied to DeshMiss dataset which was artificially created with 7.22% of missing values following the MNAR missing data mechanism, the results showed significant improvement in the ABE estimation process. Since the DeshMiss dataset contains a considerable number of missing values, the impact of MINN, KNNI, and NC on the ABE estimation process can easily be highlighted. The NC technique imputes feature or attributes wise static values to replace all the missing values, whereas KNNI imputes dynamic values according to the neighboring values which shows the realistic values being imputed.

This study focuses on the ABE model that follows human nature as it estimates the future project's effort by making analogies with the past project's data. Since ABE relies on the historical datasets, the quality of the datasets affects the accuracy of estimation. Since, the majority of the software engineering datasets e.g., Desharnais, ISBSG etc., have missing values, there is the need for a better model to handle such scenarios. Consequently, the researchers either have to remove the projects containing missing values or avoid treating the missing values that reduce the ABE performance. To address this problem, this study is targeting MINN, Numeric Cleansing (NC) and K-Nearest Neighbor Imputation (KNNI) method to impute the missing values in Desharnais dataset for ABE. In this study, a comparison among these imputation methods is performed to identify the suitable missing data imputation method for ABE. The results suggested that MINN imputes more realistic values in the missing datasets as compared to values imputed through KNN

and NC. It was also found that the performance of ABE is reduced with deleted missing values and avoided missing values; the imputation treatment method helped in better prediction of the software development effort.

The results suggested that imputing the missing values to complete the datasets has a positive impact on the performance of ABE. In the comparison of MINN, KNNI, and NC, it was found that, though, all three techniques improve the ABE performance, however, MINN significantly outperforms the results of NC when used with ABE for imputing the missing values in the DeshMiss dataset. The Desharnais dataset has a very low number of values missing, due to which, there could not be observed any significant difference among the three techniques when applied to Desharnais dataset. The dynamic nature of imputing values by MINN shows that it imputes more realistic values as compared to NC and slightly better than KNNI on small datasets.

The concept of missing values is intended to further improve for enhancing the ABE process in the future. The impact of MINN should also be analyzed for large datasets because the dataset used in this study have data of 80 projects only. If MINN loses its performance on large dataset as is predicted, there could be proposed some novel imputation methods which may also deal with the large projects and with a large number of values missing.

VI. Conclusion and Future Work

The successful management of a software project strongly depends upon the accuracy of software development effort estimation as it can substantially affect the planning and scheduling of a project. Analogy-based Estimation (ABE) has been widely adopted for effort estimation right from its genesis until recently. There have been many attempts made but to improve this estimation model from a different perspective but there are a very few studies which really focused on its vital part which is the data quality of the past datasets. It is very much necessary to have a complete dataset for making an analogy to predict the software development effort. There are usually values missing from the software engineering dataset of past projects. There are different treatment methods applied to deal with the missing values in these datasets. Imputing the missing data to replace the missing values is one of the prominent methods. There are different imputation methods used to impute missing values in the software engineering datasets. MINN K Nearest Neighbor Imputation (KNNI) and Numeric Cleansing are two of the imputation techniques. In this study, Numeric Cleansing (NC), K-Nearest Neighbor Imputation (KNNI) and Median Imputation of the Nearest Neighbor (MINN) methods are used to impute the missing values in Desharnais and DesMish datasets for ABE. MINN technique is introduced in this study. A comparison among these imputation methods is performed to identify the suitable missing data imputation method for ABE. The results suggested that MINN imputes more realistic values in the missing datasets as compared to values imputed through NC and KNNI. It was also found that the imputation treatment method helped in better prediction of the software development effort on ABE model. The impact of MINN should also be analyzed for large datasets because the dataset used in this study have the data of 80 projects only. If MINN loses its performance on large dataset as is predicted, there could be proposed some novel imputation methods which may also deal with the large projects and with a large number of values missing.

Appendix A

Table AI The DeshMiss Dataset with Missing Values

ProjectNo	TeamExp	ManagerExp	Length	Effort	Transactions	Entities	PointAdjust	Envergure	PointsNonAdjust
1	0	0	4	5635	197	124	321	33	315
2	4	4	1	805	40	60	100	18	83
3	0	0	5	3829	200	119	319	30	303
4	0	0	4	2149	140	94	234	24	208
5	0	0	4	2821	97	89	186	38	192
6	2	1	9	2569	119	42	161	25	
7	1	2	13	3913	186	52		25	
8	3	1	12	7854	172	88		30	
9	3	4	4	2422	78	38		24	
10	4	1	21	4067	167	99		24	
11	2	1	17	9051	146	112	258		
12	1	1	3	2282	183	72	105		
13	3	4	8	4172	183	61	223		216
14	4	4	9	4977	183	121	344		320
15	3	2	8	1617	183	48	167	26	152
16	4	3	8	3192	183	43	100	43	108
17	4		14	3437	68	316	384	20	326
18	3		14	4494	9	386	395	21	340
19	4		5	840	58	34	92	29	86
20	4		12	14973	318	269	587	34	
21	2		18	5180	88	170	258	34	
22	2		5	5775	306	132	438	37	
23	4	1	20	10577	304	78	382		397
24	1	4	8	3983	89	200	289		283
25	4	1	14	3164	86	230	316		310
26	2	0	6	3542	71	235	306		312
27	3	1	14	4277	148	324	472		491
28	4	4	16	7252	116	170	286	27	263
29	4	1	14	3948	175	277	452	37	
30	4	3	6	3927	79	128	207	27	
31	1	1	9	710	145	38		27	
32	4	4	9	2429	174	78		41	
33	1	1	5	6405	194	91		35	
34	2	2	3	651	126	49		38	

35	1	3	17	9135	137	119	256	34	
36	2	4	11	1435	289	88	377	28	
37			8	5922	260	144	404	24	360
38	1	4	4	847	158	59	217	18	180
39	3	3	16	8050	302	145	447	52	523
40	1	1		4620	451	48	499	28	464
41	2	4		2352	661	132	793	23	698
42	1	1		2174	64	54	118	25	106
43		4		19894	284	230	514	50	591
44	2	1		6699	182	126	308	35	308
45	2	3		14987	173	332	505	19	424
46	2	2	9	4004	252	7	259	28	241
47	4	3	11	12824	131	180	311	51	361
48	2	3	8	2331	106	39	145	6	103
49	3	3	9	5817	96	108	204	29	192
50	2	3	7	2989	116	72	188	18	156
51	3	3	6	3136	86	49	135	32	131
52	2	3	17	14434	221	121	342	35	342
53	1	1	12	2583	61	96	157	18	130
54	1	3	12	3647	132	89	221	5	155
55	3	7	13	8232	45	387	432	16	350
56	1	1	12	3276	55	112	167	12	129
57	1	4	8	2723	124	52	176	14	139
58	3	3	5	3472	120	126	246	15	197
59	1	2	6	1575	47	32	79	14	62
60	1	1	12	2926	126	107	233	23	205
61	3	2	6	1876	101	45	146	15	117
62	1	1	5	2520	78	99	177	14	140
63	4	7	13	1603	69	74	143	14	113
64	1	3	8	3626	194	97	291	35	291
65	2		10	6783	224	110	334	28	311
66	2	4	15	11361	323	184	507	35	507
67	1	3	10	1267	42	31	73	27	67
68	1	2	5	2548	74	43	117	25	105
69	3	4	10	1155	101	57	158	9	117
70	0	4	6	546	97	42	139	6	99
71	2	3	13	2275	134	77	211	13	165
72	4	5	26	9100	482	227	709	26	645
73	0	2	6	595	213	73	286	6	203
74	0		22	3941	139	143	282	22	245
75	2	3	24	13860	473	182	655	40	688
76	4	4	12	1400	229	169	398	39	414
77	4	3	12	2800	227	73	300	34	297
78	4	4	24	9520	395	193	588	40	617
79	4	3	12	5880	469	176	645	43	697
80	4	4	36	23940	886	241	1127	34	1116

REFERENCES

[1] C. Jones, "Estimating Software Costs: Bringing Realism to Estimating, Osborne," ed: McGraw HIll), 2007.

[2] B. Boehm, "Constructive cost model," Software Engineering Economics, 1981.

[3] A. J. Albrecht and J. E. Gaffney, "Software function, source lines of code, and development effort prediction: a software science validation," IEEE transactions on software engineering, pp. 639-648, 1983.

[4] M. Shepperd and C. Schofield, "Estimating software project effort using analogies," IEEE Transactions on software engineering, vol. 23, pp. 736-743, 1997.

[5] J. Li and G. Ruhe, "Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+," Empirical Software Engineering, vol. 13, pp. 63-96, 2008.

[6] J. W. Keung and B. Kitchenham, "Optimising project feature weights for analogy-based software cost estimation using the mantel correlation," in Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific, 2007, pp. 222-229.

[7] J. Wen, S. Li, and L. Tang, "Improve analogy-based software effort estimation using principal components analysis and correlation weighting," in Software Engineering Conference, 2009. APSEC'09. Asia-Pacific, 2009, pp. 179-186.

[8] A. Tosun, B. Turhan, and A. B. Bener, "Feature weighting heuristics for analogy-based effort estimation models," Expert Systems with Applications, vol. 36, pp. 10325-10333, 2009.

[9] N.-H. Chiu and S.-J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," Journal of Systems and Software, vol. 80, pp. 628-640, 2007.

[10] Y.-F. Li, M. Xie, and T. N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation," Journal of Systems and Software, vol. 82, pp. 241-252, 2009.

[11] J. Pahariya, V. Ravi, and M. Carr, "Software cost estimation using computational intelligence techniques," in Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on, 2009, pp. 849-854.

[12] A. L. Oliveira, P. L. Braga, R. M. Lima, and M. L. Cornélio, "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation," information and Software Technology, vol. 52, pp. 1155-1166, 2010.

[13] V. K. Bardsiri, D. N. A. Jawawi, S. Z. M. Hashim, and E. Khatibi, "Increasing the accuracy of software development effort estimation using projects clustering," IET software, vol. 6, pp. 461-473, 2012.

[14] V. K. Bardsiri, D. N. A. Jawawi, A. K. Bardsiri, and E. Khatibi, "LMES: A localized multi-estimator model to estimate software development effort," Engineering Applications of Artificial Intelligence, vol. 26, pp. 2624-2640, 2013.

[15] A. Idri, F. azzahra Amazal, and A. Abran, "Accuracy Comparison of Analogy-Based Software Development Effort Estimation Techniques," International Journal of Intelligent Systems, vol. 31, pp. 128-152, 2016.

[16] T. R. Benala and R. Mall, "DABE: Differential evolution in analogy-based software development effort estimation," Swarm and Evolutionary Computation, vol. 38, pp. 158-172, 2018.

[17] J. Huang, J. W. Keung, F. Sarro, Y.-F. Li, Y.-T. Yu, W. Chan, et al., "Cross-validation based K nearest neighbor imputation for software quality datasets: An empirical study," Journal of Systems and Software, vol. 132, pp. 226-252, 2017.

[18] A. Idri, I. Abnane, and A. Abran, "Missing data techniques in analogy-based software development effort estimation," Journal of Systems and Software, vol. 117, pp. 595-611, 2016.

[19] I. Abnane and A. Idri, "Improved Analogy-Based Effort Estimation with Incomplete Mixed Data," in 2018 Federated Conference on Computer Science and Information Systems (FedCSIS), 2018, pp. 1015-1024.

[20] F. Walkerden and R. Jeffery, "An empirical study of analogy-based software effort estimation," Empirical software engineering, vol. 4, pp. 135-158, 1999.

[21] L. Angelis and I. Stamelos, "A simulation tool for efficient analogy based cost estimation," Empirical software engineering, vol. 5, pp. 35-68, 2000.

[22] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd, "Experiences using case-based reasoning to predict software project effort," in Proceedings of the EASE 2000 conference, Keele, UK, 2000.

[23] J. Magne and S. Grimstad, "Avoiding irrelevant and misleading information when estimating development effort," IEEE software, pp. 78-83, 2008.

[24] Q. Song, M. Shepperd, X. Chen, and J. Liu, "Can k-NN imputation improve the performance of C4. 5 with small software project datasets? A comparative evaluation," Journal of Systems and software, vol. 81, pp. 2361-2370, 2008.

[25] R. J. Little and D. B. Rubin, "Bayes and multiple imputation," Statistical analysis with missing data, pp. 200-220, 2002.

[26] J. Li, A. Al-Emran, and G. Ruhe, "Impact analysis of missing values on the prediction accuracy of analogy-based software effort estimation method AQUA," in Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, 2007, pp. 126-135.

[27] A. Mockus, "Missing data in software engineering," in Guide to advanced empirical software engineering, ed: Springer, 2008, pp. 185-200.

[28] A. Idri, I. Abnane, and A. Abran, "Systematic mapping study of missing values techniques in software engineering data," in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2015 16th IEEE/ACIS International Conference on, 2015, pp. 1-8.

[29] K. Strike, K. El Emam, and N. Madhavji, "Software cost estimation with incomplete data," IEEE Transactions on Software Engineering, vol. 27, pp. 890-908, 2001.

[30] M. H. Cartwright, M. J. Shepperd, and Q. Song, "Dealing with missing software project data," in Software Metrics Symposium, 2003. Proceedings. Ninth International, 2003, pp. 154-165.

[31] B. Twala and M. Cartwright, "Ensemble imputation methods for missing software engineering data," in Software Metrics, 2005. 11th IEEE International Symposium, 2005, pp. 10 pp.-30.

[32] P. Sentas and L. Angelis, "Categorical missing data imputation for software cost estimation by multinomial logistic regression," Journal of Systems and Software, vol. 79, pp. 404-414, 2006.

[33] I. Myrtveit and E. Stensrud, "A controlled experiment to assess the benefits of estimating with analogy and regression models," IEEE transactions on software engineering, vol. 25, pp. 510-525, 1999.

[34] J.-M. Desharnais, "Analyse statistique de la productivitie des projects informatique a partie de la technique des point des function," Masters Thesis University of Montreal, 1989.

[35] N. A. Ali and Z. M. Omer, "Improving accuracy of missing data imputation in data mining," Kurdistan Journal of Applied Research, vol. 2, pp. 66-73, 2017.

[36] A. Fatima, N. Nazir, and M. G. Khan, "Data Cleaning In Data Warehouse: A Survey of Data Pre-processing Techniques and Tools," 2017.