An Approach to Adjust Effort Estimation of Function Point Analysis

Huynh Thai Hoc(✉), Vo Van Hai, and Ho Le Thi Kim Nhung

Faculty of Applied Informatics, Tomas Bata University in Zlin, Nad Stranemi 4511, 76001 Zlin, Czech Republic
{huynh_thai,vo_van,lho}@utb.cz

Abstract. This study presents a modified approach to adjust a software development effort estimation. The AdamOptimizer-based regression model is adopted to adjust and enhance the accuracy of effort estimation. This approach is derived into three phases. The first step deals with the logarithmized formula of effort estimation computed by Function Point Analysis and Productivity Delivery Rate. The Adam-Optimizer-based regression model is examined in the second phase, and the ISBSG repository 2020 release R1 is considered as a historical dataset in this paper. Moreover, the K-Fold cross-validation technique is adopted to tunning the training model. In the following phase, all results are evaluated by statistical significance and the goodness of fit measure. Finally, a proposed approach is compared with others: Capers Jones, and the Mean Effort.

Keywords: Software effort estimation · FPA · Capers Jones · Adj-Effort · AdamOptimizer · K-Fold

1 Introduction

Effort estimation of software development is a significant challenge in the process of software project building [1–4]. This process is likely the first phase to estimate the development resources (i.e., cost, time). The accurate estimation might lead to the proper contribution to resources for a project. It is probably an important initial-step of project planning. Underestimating or overestimating might cause less accuracy in allocating the budgets, and then might be probably a significant cause of project failure [5]. This problem might boost researchers to study it, and many techniques have been proposed for the last decade regarding how to improve the effort estimation.

In general, there are two types of techniques used in software project effort estimation, that are, non-algorithmic and algorithmic [1, 3, 6]. Non-algorithmic is a technique that predicting the Effort might be based on either the expert's opinion or analytical comparisons with the historical projects. For example, Expert Judgment, Analogy, Price-to-win, Bottom-up, Top-up, Wideband Delphi, Planning Poker [1, 3, 7]. The algorithmic technique is a mathematical-based approach that mainly focuses on algorithms such as COCOMO, Use Case Points (UCP), Function Point Analysis (FPA).

Most algorithmic methods might mainly focus on optimizing factors such as the proportion of effort estimation known as Productivity Delivery Rate (PDR) or environmental factors that impact the development effort [5, 7–9]. These indicators impact the measurement of software effort. The purpose of these optimations is to make them fitter with the actual data. The better value of these indicators might lead to more accuracy of the estimation. In order to obtain those targets, scientists proposed many approaches to compute the predicted results and try to fit with the actual value. In their researches, some applied regression models, such as least squares regression, multiple linear regression [5, 7–9]. The other integrated regression methods work with fuzzy models, deep learning techniques [9, 10], etc.

Some studies presented other optimizable measures to compute the Effort of the application development. Prokopová et al. [11] worked with the VAF factor to identify its influence on the estimation of the effort size. Šilhavý et al. [12] improved this Effort by proposing a new CVS model, and dataset segmentation was applied to estimate it. Moreover, Artificial neural networks (ANN) were also adopted by Wena et al. [13] to do this estimation.

On the other hand, the term PDR is considered as the proportion of effort size. This term has always been adopted to calculate the required software project effort by multiplying with UCP size or FPA size.

$$\text{Effort} = \text{Size} \times \ PDR \tag{1}$$

Additionally, Nassif et al. applied a fuzzy-based methodology to propose four levels of Productivity based on a summation of environmental factors [14]. Moreover, the relationship between project Productivity and environmental factors was also studied by Azzeh et al. [15, 16]. They concluded that such indicators might be good factors for Productivity prediction when observational projects are available. Besides, Hai et al. [17] have proposed Productivity as a Multiple Regression model with independent variables (VAF, EI, EO, EQ, ILF, EIF) in the measured Effort of function point analysis based on the International Software Benchmarking Standards Group (ISBSG) 2018/release R2. The authors report that the result might be better than the existing method, compared with the Effort measured by Mean Productivity based on Eq. 1, and Capers Jones.

The purpose of this paper is to propose an adjusted technique to predict effort estimation based on the original formula of FPA size (Eq. 1). The formula will be transformed into a logarithm-based equation, and then apply the Regression Model to find a good (best) fitness model for estimating Effort. The proposed technique evaluation will be based on datasets extracted from the ISBSG repository released in August 2020. This technique might be called an Adjust Effort Estimation of Function Point Analysis (Adj-Effort).

The rest of the paper is structured as follows: Sect. 2 shows the Research Questions; Sect. 3 presents the background of Function Point Analysis as well as the variant of Effort Estimation; Sect. 4 illustrates the data pre-processing; Sect. 5 proposes the new approach - the Adjust Effort Estimation by Adam-optimizer method; Sect. 6 discusses the criteria validation; Sect. 7 presents the result and discussion; and conclusion – future work is presented in the last section.

2 Research Questions

In order to conduct the proposed model is the "good-fitness-model" for adjusting effort estimation, two research questions should be answered:

(1) RQ1: how close the real efforts are to the predicted regression model? Answering this question determine a statistical measure, including the Coefficient of determination (R^2), and adjusted R^2.
(2) RQ2: Is this a better-fitness model than Capers Jones and Original Effort method? Answering this question is to determine the SSE, MAE, PRED(0.25), the minimised SSE and MAE, and the maximised $PRED(0.25)$. In addition, the paired t-test based on the MAE measurement is examined [18].

H_o : $MAE_{Adj-Effort} = MAE_{CapersJones/OrginalEffort}$: There is no estimation error difference between these methods.

H_1 : $MAE_{CapersJones/OriginalEffort} > MAE_{Adj-Effort}$: Estimation capability of project size by the Adj-Effort might be more feasible than by Capers Jones and Original Effort method.

3 Background

3.1 Function Point Analysis

Function Point Analysis (FPA) was first introduced in 1979 by Alan J. Albrecht when he was a Program Manager of the Application and Maintenance Measurement Program for IBM [19, 20]. It is a Function Point to count the functional size and complexity of a software-based on user requirements [21]. His approach is to quantify the number of External Inputs (EI), External Inquiries (EQ), External Outputs (EO), Internal Logic Files (ILF), and External Interface Files (EIF) that are delivered by software projects. In 1984, the International Function Point Users Group (IFPUG) was established to define the set of rules for the basic components (BCs), including transaction function types (EI, EO, EQ) and data function types (EIF, ILF) [22]. The complexity weights of each component are shown in Table 1.

Table 1. Complexity weights of components [19, 23]

Basic Components (BCs)	Complexity Weight (CWs)		
	Low	Medium	Large
EI	3	4	6
EO	4	5	7
EQ	3	4	6
EIF	5	7	10
ILF	7	10	15

In the Counting Practices Manual, version 4.3.1 (2010) [24], the FPA is organized by the IFPUG, which is accountable for the development as well as the improvement of its rules. The IFPUG's FPA is known as the original function point analysis and is standardized by ISO/IEC 20926:2010. Additionally, there are four other Functional Software Measurements (FSM) of ISO/IEC, including MarkII, MESMA, COSMIC, FISMA. These methods are out of the scope of this paper; they might be found in [25].

The FPA has a majority of characteristics that it is possible to employ to estimate software development in the early phases of the project [26]. Firstly, function points might be counted based solely on either requirements specifications or design specifications. It seems the initial phase of the projects. Secondly, they are unrelated to language programming, specific technologies for developing, or any other data processing [23]. Moreover, non-technical users of software might be easier to understand the function points because they are built based on the user's external view of the system [27].

Table 2. General systems characteristics [19, 23]

GSC factors	Characteristic	Description
F1	Data communications	Does the system require backup and recovery?
F2	Distributed functions	Are data required for communication?
F3	Performance	Does the system include a distributed processing function?
F4	Heavily used configuration	Is critical performance required?
F5	Transaction rate	Will the system work during heavy loads?
F6	Online data entry	Does the system require direct data input?
F7	End-user efficiency	Do data inputs require multiple screens or operations?
F8	Online update	Are the main files up-to-date?
F9	Complex processing	Are inputs, outputs, files, and queries intricate?
F10	Reusability	Is internal processing complicated and complex?
F11	Installation ease	Is the code designed for reuse?
F12	Operational ease	Are conversions and installation included in design?
F13	Multiple sites	Is the system designed for multiple installations in different locations?
F14	Facilitate change	Is the application designed to make it easy for users to make changes?

To count function points, a linear combination between function types (EI, EO, EO, EIF, ILF) with appropriate three levels of complexity weights are established. This function count is also known as Unadjusted Function Points (UFP). The UFP formula is shown in Eq. 2.

$$UFP = \sum\nolimits_{i=1}^{5} \sum\nolimits_{j=1}^{3} BCs_{ij} \times CWs_{ij} \tag{2}$$

where BCs_{ij} is the count of component i at level j, and CWs_{ij} is an appropriate weight given in Table 1.

The final counting of function points is driven by multiplying UFP with adjustment influent factors, General Systems Characteristics (GSC). These might help the counting of UFP to be more accurate [27]. Additionally, VAF is considered as Value Adjustment Factor described by the formula:

$$\text{VAF} = 0.65 + 0.01 \times \sum\nolimits_{i=1}^{14} F_i \times Rating_{Influence} \tag{3}$$

Where F_i is the influence of the GSC factor. There is a total of 14 factors impacting the system. These factors are illustrated in Table 2, and the influence factors rating is shown in Table 3.

Table 3. Influent factors rating [19, 23]

Influence	Rating
None	0
Insignificant	1
Moderate	2
Average	3
Significant	4
Strong significant	5

The Adjusted Function Points (AFP) is then the following:

$$AFP = \text{UFP} \times VAF \tag{4}$$

As mentioned in Sect. 1, based on the AFP, the measurement of Software Effort Estimation is shown in Eq. 5, where Effort (called Original Effort) is measured in person-hours and AFP is considered as a software size.

$$\text{Effort} = \text{AFP} \times \text{PDR} \tag{5}$$

3.2 The Variant of Sofware Effort Estimation

Capers Jones' Approach

From 1997 to 2007, Capers Jones proposed thumb rules to measure the sizing of software project [28]. The first version was published in 1997, and then it was updated in 2003. In 2007, it continued to edit and become the IFPUG counting rules version 4.1. According to the author [28], there are ten simple rules for software sizing estimation, which the last four rules regarding schedules, resources, and costs. As mentioned in the 10th rule, the estimating software development effort is employed by compiling the 7th and 8th rules. As a result, the work effort estimation is presented as the following [17, 28, 29]:

$$\text{Effort}_{Capers\ Jones} = \frac{AFP}{150} \times AFP^{0.4} \tag{6}$$

where $\text{Effort}_{Capers\ Jones}$ is the Effort measured in person-month; AFP divides 150 that illustrates the staff size (the 7th rule) [28] and AFP raises to power 0.4 present the approximate schedule (the 8th rule) [28].

Original Effort with Mean Productivity – the Mean Effort

The Mean Effort is a predicted size to measure an estimated software development effort based on Mean Productivity. It is computed by multiplying Adjusted Function Points with mean Product Delivery Rate value - PDR_{Mean} (Eq. 8). The PDR_{Mean} is expressed in Eq. 7, where n is the number of observed projects. This is a kind of Effort used as a sample method to compare with other predicted efforts. Prokopova et al. (2018), or Hai et al. (2020) have adopted the Mean Effort to evaluate the accuracy of their approaches compared with it.

$$\text{PDR}_{Mean} = \frac{1}{n} \sum\nolimits_{n}^{1} PDR \tag{7}$$

$$\text{Effort}_{Mean} = \text{AFP} \times \text{PDR}_{Mean} \tag{8}$$

4 Data Pre-processing

In this research, the ISBSG project repository Release 2020 will be adopted as the historical dataset. The ISBSG has 251 recorded attribute values, 9592 projects. These attributes are divided into many groups. The first group of these datasets is Rating. This group contains an ISBSG rating code that is in order from high-quality reviewers to low-quality reviewers, denoted A, B, C, or D. The second group is software age, it records the year of the project, ranges from 1989 to 2019. The information of Development type is also presented, it is categorized into three types, such as new development, enhancement, and re-development. Furthermore, the group of attributes regarding Sizing (UFP, AFP, VAF), Effort (Normalised Work Effort Level 1, Normalised Work Effort, Summary Work Effort), Productivity (Normalised Level 1 PDR - unadjusted function points, Normalised PDR, Pre 2002 PDR), Size Attributes (EI, EO, EQ, EIF, ILF) are recorded in these datasets.

These attributes are considered as a majority of datasets. It is noted that Normalised Level 1 Productivity Delivery Rate is the Productivity currently recommended by ISBSG [30, 31]. Additionally, the other project-relevant information such as Effort Attributes (Team Size Group, Resource Level), Tool Data, Project team (experience BA, IT, project manager), and so on is also recorded in this dataset.

In this study, Summary Work Effort, a total effort in hours recorded against the project, and Normalised Level 1 PDR will be used as the "Real-Effort" and PDR for the training dataset. Moreover, due to Normalised Level 1 PDR is selected as a PDR in Eq. 5 and 7, the value of AFP in the training dataset is also re-calculated by Eq. 4 based on the value of UFP and VAF.

There are many projects in the ISBSG repository released in 2020, it contains all 9592 projects. However, this proposal solely chooses projects that have sufficient good quality factors with the IFPUG-based development type. The following guidelines are adopted to filter the datasets.

(1) As recommended by ISBSG as well as the selection in the publish [10, 17], the research only chooses projects with data quality A and B. The other might be insufficient credibility due to either significant data not being provided or one factor or a combination of factors [30, 31], reducing the size of the dataset to 8619 projects.
(2) This paper mainly focuses on FPA based counting approach as the IFPUG, the size is measured by MarkII, MESMA, COSMIC, FISMA that is ignored, leaving only 6365 records.
(3) New development projects used solely [10], other types of project are considered removed, resulting in the total projects to 1460
(4) The missing data on Summary Work Effort (Effort), Normalised Level 1 Productivity Delivery Rate (PDR), UFP, VAF are removed, and the missed AFP might be recovered by Eq. 4, bring the remaining projects to 583.

5 Regression-Based Adjusting Effort

A Regression Model is the most common method for identifying relationships between factors of effort software estimation [3]. One of the factors is the dependent (response) variable, and the others are independent (preditor) variables [32]. This relationship may be used for predicting the values of the response variable for a given set of independent variables. The outcome in this model can be examined based on the historical data. This process is equivalent to optimizing the model's suitable coefficients, including its functional form and coefficients based on sample observational data.

A genre of a regression model is presented as Eq. 9, a linear regression model, where the dependent variable (Effort) is represented by Y and the predictor variable (AFP $\times$ PDR) by X; β_0 is an Intercept Parameter, and β_1 is called Regression Coefficient, $\boldsymbol{\varepsilon}$ is depicted as the error residuals.

$$Y \approx \beta_0 + \beta_1 \mathrm{X} + \varepsilon \tag{9}$$

Figure 1 and Fig. 2, the histograms of software effort and adjusted function points, illustrate that the selected dataset is not a normal distribution. In pact, this sub-dataset might be adopted to attain a predicted effort. However, it might result in an inaccurate outcome [6]. Nevertheless, after the Effort and the AFP are normalized by logarithmic technique, their histograms are slightly normally distributed. These figures are shown in Fig. 3 and Fig. 4. As a result, the prediction effort based on the historical dataset might be obtained by a log-based regression model instead of Eq. 9.

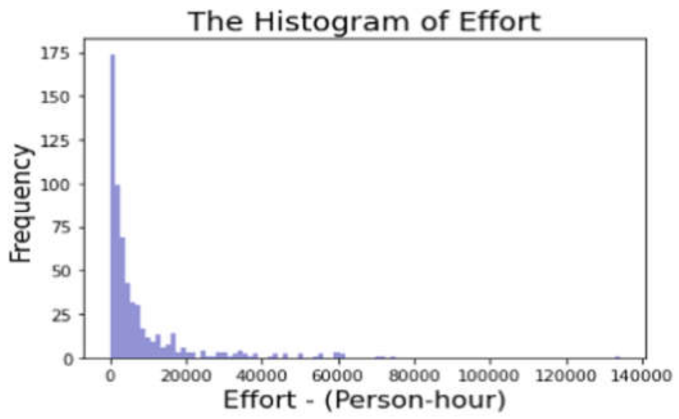

Fig. 1. The histogram of effort

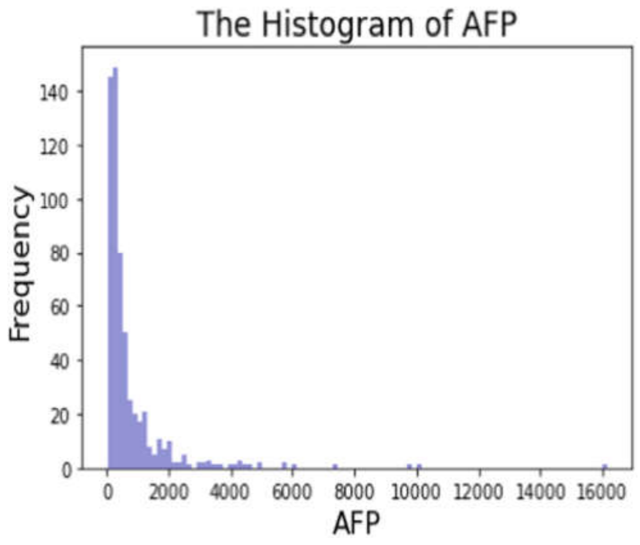

Fig. 2. The histogram of AFP

There are several steps to gain the log-based regression model. First, Eq. 5 is transformed by the logarithmic method, and then the regression model is adopted on it. These equations are given as Eq. 10, and 11, where ln(Effort) as a dependent variable, and $\ln(AFP)$, $\ln(PDR)$ as independent variables. Second, the excellent fitness model in Eq. 11 might be obtained by minimizing the error residual (ε).

$$\ln(\text{Effort}) = \ln(\text{AFP} \times \text{PDR}) = \ln(AFP) + \ln(AFP) \tag{10}$$

$$\ln(\text{Effort}) = \beta_0 + \beta_1 \ln(AFP) + \beta_2 \ln(PDR) + \varepsilon \tag{11}$$

There are some approaches to minimize the error residual. Radek Silhavy et al. (2015) have adopted the Multiple Least Square Regression method, namely AOM, to improve UCP. However, Hoc et al. (2020) have used the Adam-Optimizer module [3] to attain the predictive UCP. It is a kind of open-source, available in the TensorFlow

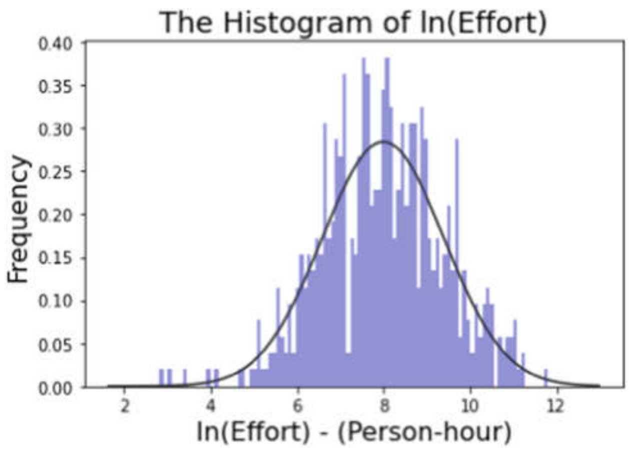

Fig. 3. The histogram of ln(Effort)

package [33], developing by the Google Brain team [33, 35]. Comparing with the AOM on the same historical dataset, they concluded that their approach was better than the AOM.

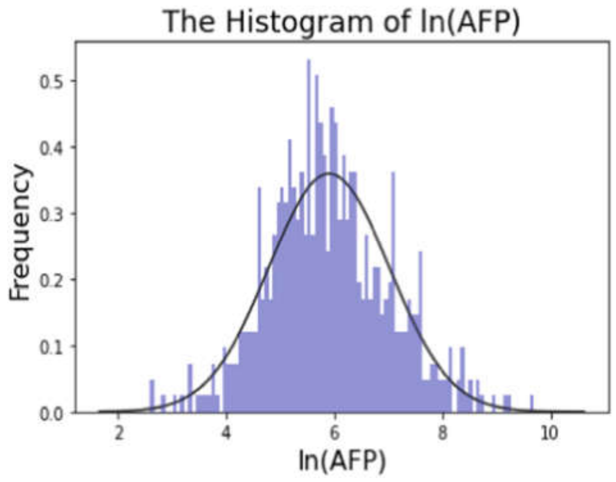

Fig. 4. The histogram of ln(AFP)

Finally, due to the outperformance of the Adam-Optimizer [34, 35], this module is applied to find suitable unknown values ($\beta_0, \ \beta_1, \ \beta_2$) in Eq. 11. It means that the final adjust Effort (Adj-Effort) is computed according to the formula:

$$\ln(\text{Effort}) \approx \ \beta_0 + \ \ln\left(AFP^{\beta_1}\right) + \ln\left(PDR^{\beta_2}\right) \tag{12}$$

$$\text{Effort} \approx \ e^{\beta_0 + \ \ln\left(AFP^{\beta_1}\right) + \ \ln\left(PDR^{\beta_2}\right)} \tag{13}$$

$$\text{Effort} \approx \ e^{\beta_0} \times AFP^{\beta_1} \times PDR^{\beta_2} \tag{14}$$

5.1 Dataset Description

As mentioned in Sect. 4, the Adj-Effort technique will use 583 filtered projects of the ISBSG repository released in 2020 as observational data. Figure 5 presents a boxplot of the log-based dataset before (Fig. 5.a) and after (Fig. 5.b) removing outliers. As can be seen, there are several noticeable outliers in that dataset. According to many researchers [17], removing outliers is suitable for accurate and reliable software effort estimation based on historical datasets. In this study, the interquartile range (IQR) method is adopted to eliminate the outliers. If the value of ln(Effort), ln(PDR), and ln(AFP) out of the range from Q1 − 1.5 × IRQ to Q3 + 1.5 × IRQ, it might be considered to ignore, Q1 is the lower quartile, Q3 is the upper quartile, and IRQ is expressed as the following:

$$\text{IQR} = \text{Q3} - \text{Q1} \tag{15}$$

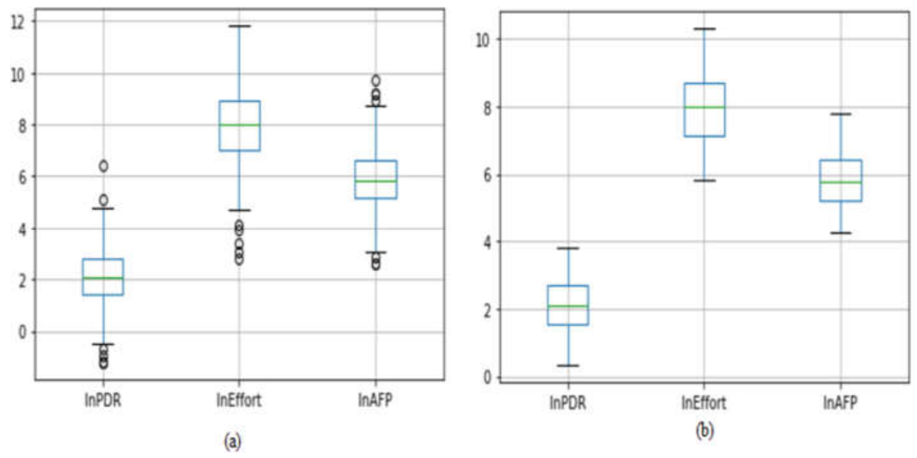

Fig. 5. The Boxplot of the observational dataset before (a) and after (b) remove outliers

Table 4 shows the descriptive characteristics of the logarithm of Summary Work Effort before removing outliers (initial) and after removing outliers, where N stands for the number of observed projects, 583 and 450, respectively. It is noted that the Mean value is slightly equal on both datasets, 7.9793, after adopting outliers. Additionally, the Standard Deviation, the value at 25%, 50%, and 75% are insignificantly different. However, the Minimum and Maximum at Remove Outliers Dataset might be better than the Initial dataset, compared with the others. As a result, the removed outliers-based dataset is used in this research.

Table 4. Descriptive observed datasets – ln(Effort)

Dataset	N	Mean	Standard deviation	Minimum	25%	50%	75%	Maximum
Initial	583	7.9953	1.4042	2.833	7.0148	8.0044	8.9344	11.8072
Remove outliers	450	7.9793	1.0344	5.8200	7.1323	8.0019	8.7073	10.2955

5.2 Adj-Effort Setting

The Adj-Effort technique is processed in three steps - shown in Fig. 6. The first step filters the raw dataset (2020) by adopting Data processing criteria – presented in Sect. 4. Second, the selected sub-dataset, including Effort, PDR, and AFP are logarithmized to make them a normal distribution. In the following phase of the process, the pre-processed projects are randomly divided into two parts. 80% of the projects are used to train the Adj-Effort model. The tested part – 20% of the projects are considered as unknown projects, where the data on Effort, AFP are available in this dataset, they are considered as real values; and the mean of the PDR ($PDR_{Mean} = 11.2569$), gained from the training dataset, is used as a value of the PDR in the testing phase.

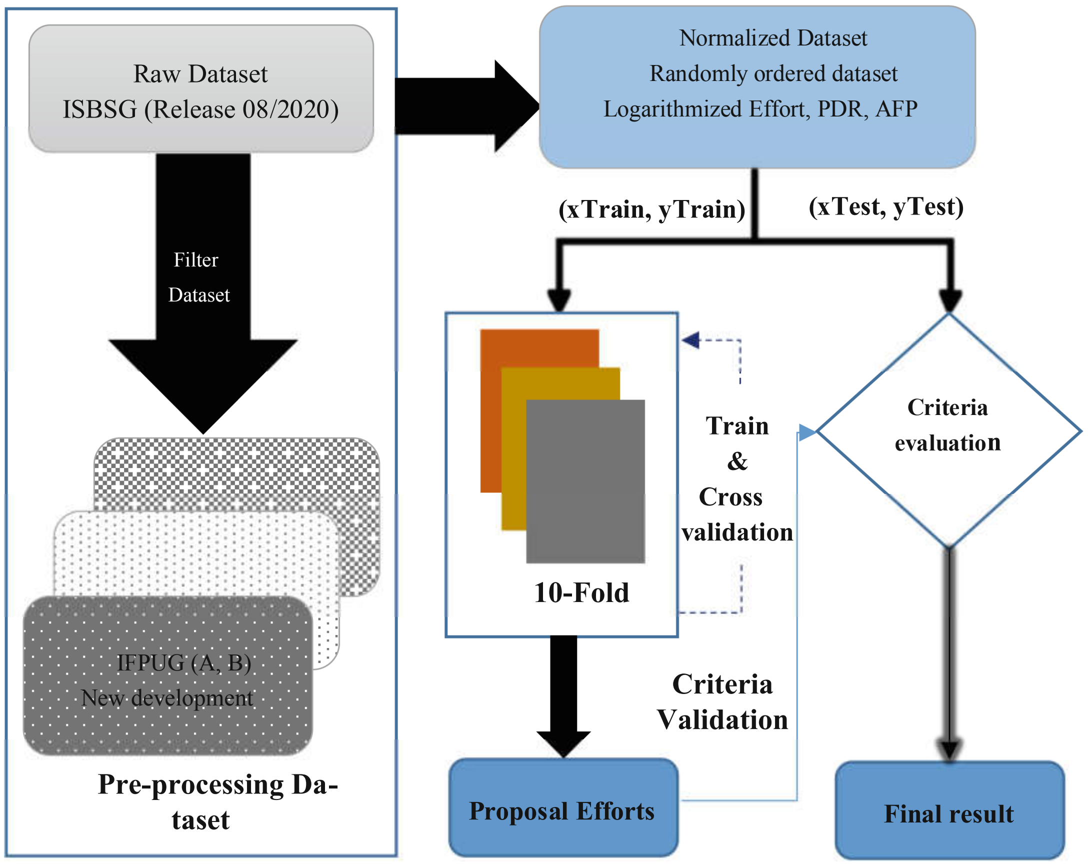

Fig. 6. The Adj-Effort with 10-Fold cross-validation flow diagram

Additionally, the non-cross validation and cross-validation methods are examined in optimizing the Effort. 10-Fold cross-validation on the training set is adopted to examine the performance and reliability of the proposed technique. The average 10-Fold-based Effort is obtained in this phase. Finally, the effort models obtained from non-cross validation and cross-validation are evaluated based on the criteria presented in Sect. 6 and compared with Capers Jones and the Mean Effort method to achieve the final Adj-Effort.

6 Criteria Validation

This section presents the standard criteria to validate the accuracy of the effort estimation of each method. MMRE is the Mean Magnitude of Relative Error, and PRED(x) is the Prediction level. Both are computed based on the Magnitude of Relative Error (MRE). SSE (Sum of Squared Errors) is a metric to compute modeling error variation. MMRE, PRED(x), or SSE are the most well-known measurement of accurate effort estimation [3, 36, 37]. Furthermore, MAE stands for the Mean Absolute Error; it represents the difference between the actual and predicted values. Finally, R^2 *or Adjusted* R^2 are the statistical indicators to present how close the data are to the fitted regression model. Their values might from 0% to 100%, the higher $R^2/Adjusted\ R^2$ means the more accuracy of the proposed approach [38]. The mentioned criteria equations are given as follows:

$$MAE = \frac{\sum_i^N |\hat{y}_i - y_i|}{N} \tag{16}$$

$$MRE = \frac{|\hat{y}_i - y_i|}{y_i} \tag{17}$$

$$MMRE = \frac{\sum_i^N MRE}{N} \tag{18}$$

$$\text{PRED}(x) = \frac{1}{N}\sum_i^N \begin{cases} 1\ if\ MRE_i \le x \\ 0\ otherwise \end{cases} \tag{19}$$

$$SSE = \sum_i^N (y_i - \hat{y}_i)^2 \tag{20}$$

$$R^2 = 1 - \left[\frac{\sum_i^N (y_i - \hat{y}_i)^2}{\sum_i^N (y_i - \bar{y})^2}\right] \tag{21}$$

$$Adjusted\ R^2 = 1 - \frac{N-1}{N-k-1}\left(1 - R^2\right) \tag{22}$$

Where N is the number of observations, k is the number of preditors, y_i the actual value, $\hat{y}_i$ the predicted value, $\bar{y}$ the mean of actual value and the x value is considered to be 0.25 - as recommended in many studies [7, 36, 39]. Some authors also use PRED(0.20) or PRED(0.30) with little differing results [7].

7 Results and Discussion

In this section, the results of proposal efforts are discussed based on the criteria validation in Sect. 6. It is noted that two predictive models are shown in Table 5. The first one is obtained without applying cross-validation (non-cross validation). The second one is attained by dividing the training dataset into 10-folds, one of them for validating and the rest for training. As can be seen, the obtained models might be good fitness models compared with the actual Effort due to both R^2 and *Adjusted* R^2 are strongly high [34]. These values might answer the RQ1 that predicted models might slightly close with the real one.

Table 5. The predictive effort (Adj-Effort)

	Regression models	R-squared	Adj-R-squared
Adj-Effort	Effort $\approx$ 2.45994 $\times AFP^{0.8999} \times PDR^{0.8506}$	0.8357	0.8349
Adj-Effort/10-Fold	Effort $\approx$ 2.4639 $\times AFP^{0.8813} \times PDR^{0.8433}$	0.8866	0.8859

Table 6 shows a summary of the predictive statistics in terms of the effort that was adopted to validate the accuracy between Adj-Efforts (non-cross validation and 10-Fold cross-validation) and other methods (Capers Jones, the Mean Effort). As can be seen, both the regression-based Adj-Efforts perform well and give relatively accurate results with values of PRED(0.25) $\geq$ 21% and MAE $\leq$ 2530. It is noticeable that both Capers Jones and the Mean Effort yields worse results than Adj-Effort models. Although the MMRE of the Adj-Effort is slightly higher than Capers Jones, the results gained from the Adj-Effort are likely more accurate than the ones estimated by Capers Jones and the Mean Effort (i.e. MAE-Adj-Effort < MAE-Capers Jones/ Mean Effort, PRED(0.25)-Adj-Effort > PRED(0.25)-Capers Jones/Mean Effort), and SSE-Adj-Effort < SSE-Capers Jones/Mean Effort as well - matching the RQ2 requirement in Sect. 5.

Table 6. The performance effort estimation comparison methods

	Adj-Effort	Adj-Effort 10-Fold	Capers Jones	Original effort
MMRE	1.17	1.01	0.99	1.26
MAE	2527.89	2447.72	4509.96	2815.93
PRED(0.25)	0.21	0.26	0.0	0.20
SSE	36355.40	36020.93	65488.69	39939.61
N	90	90	90	90

On the other hand, based on MAE, PRED(0.25), and SSE, it is possible to conduct that Adj-Effort with/without 10-Fold might be more accurate than the others. It means that two models might be used as the best-predicted estimation. However, all criteria obtained from Adj-Effort/10-Fold are all more significant than those obtained from Adj-Effort without cross-validation. For example, PRED(0.25) attains the maximum, while MMRE, MAE and SSE reach the minimum when compared with Adj-Effort/without cross-validation. It expresses the meaning that Adj-Effort/10-Fold might be more accurate than the other one.

In addition, the paired t-test result of the hypothesis is given in Table 7. There is a significant difference in MAE between Adj-Effort/10-Fold and Capers Jones $-$(p value $= 0.000004 < 0.05$); the Mean Effort (p value $= 0.0271210 < 0.05$). Based on the paired t-test results, one might reject the null hypothesis and accept the alternative hypothesis. This might reveal that Adj-Effort/10-Fold is the best in the Estimation Accuracy Field.

Table 7. Adj-Effort/10-Fold hypothesis t-test results

	Degree of Freedom	t-value	p-value
Capers Jones	89	4.89	0. 000004
Mean effort	89	2.25	0.0271210

8 Conclusion and Future Work

In this study, FPA-based effort estimation is proposed, namely Adj-Effort, and the AdamOptimizer-based regression model is used to adjust and improve that estimation. The model was trained and tested, employing 583 filtered projects of the ISBSG repository 2020. The training dataset was divided into 10-Fold, and then applied cross-validation to tunning the precise of the accurate predictive model.

As discussed in Sect. 7, the Adj-Effort is more consistent than Capers Jones and the Mean Effort. Moreover, based on the analysed results in that section, we conclude that with $R^2 = 0.8866$, and *Adjusted* $R^2 = 0.8859$, it is noticeable that the adjusted Effort of projects gained from the Adj-Effort/10-Fold, overall, is close to the actual Effort of projects. In other words, the real Effort of the project is considered to slightly fit with the estimated Effort from the Adj-Effort/10-Fold.

However, as presented in Sect. 4, there are many factors in the ISBSG repository. It might influence the quality of the estimation. Therefore, in our future research, a statistical analysis of the different factors affecting the accuracy of effort estimation software will be considered.

Acknowledgment. This work was supported by the Faculty of Applied Informatics, Tomas Bata University in Zlín, under Project IGA/CebiaTech/2021/001.

References

1. Muketha, G.: A review of Agile software effort estimation methods. Int. J. Comput. Appl. Technol. Res. **5**(9), 612–618 (2016)
2. Trendowicz, A., Jeffery, R.: Software Project Effort Estimation. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-03629-8

3. Hoc, H.T., Van Hai, V., Le Thi Kim Nhung, H.: A review of the regression models applicable to software project effort estimation. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) CoMeSySo 2019 2019. AISC, vol. 1047, pp. 399–407. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31362-3_39
4. Van Hai, V., Le Thi Kim Nhung, H., Hoc, H.T.: A review of software effort estimation by using functional points analysis. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) CoMeSySo 2019 2019. AISC, vol. 1047, pp. 408–422. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31362-3_40
5. Seo, Y.-S., Bae, D.-H., Jeffery, R.: AREION: software effort estimation based on multiple regressions with adaptive recursive data partitioning. Inf. Softw. Technol. **55**(10), 1710–1725 (2013)
6. Nassif, A.B., Ho, D., Capretz, L.F.: Towards an early software estimation using log-linear regression and a multilayer perceptron model. J. Syst. Softw. **86**(1), 144–160 (2013)
7. Fedotova, O., Teixeira, L., Alvelos, H.: Software effort estimation with multiple linear regression: review and practical application. J. Inf. Sci. Eng. **29**(5), 925–945 (2013)
8. Silhavy, P., Silhavy, R., Prokopova, Z.: Analysis and selection of a regression model for the use case points method using a stepwise approach. J. Syst. Softw. **125**, 1–14 (2017)
9. Hoc, H.T., Van Hai, V., Le Thi Kim Nhung, H.: AdamOptimizer for the optimisation of use case points estimation. In: Silhavy, R., Silhavy, P., Prokopova, Z. (eds.) CoMeSySo 2020. AISC, vol. 1294, pp. 747–756. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-63322-6_63
10. Nassif, A.B., Azzeh, M., Idri, A., Abran, A.: Software development effort estimation using regression fuzzy models. Comput. Intell. Neurosci. **2019**, 8367214 (2019)
11. Prokopova, Z., Silhavy, P., Silhavy, R.: VAF factor influence on the accuracy of the effort estimation provided by modified function points methods. In: Annals of DAAAM & Proceedings (2018)
12. Silhavy, P., Silhavy, R., Prokopova, Z.: Categorical variable segmentation model for software development effort estimation. IEEE Access **7**, 9618–9626 (2019)
13. Wena, J., Lia, S., Linb, Z., Huc, Y., Huangd, C.: Systematic literature review of machine learning based software development effort estimation models. Inf. Softw. Technol. **54**(1), 41–59 (2012)
14. Azzeh, M., Nassif, A.B.: Project productivity evaluation in early software effort estimation. J. Softw. Evol. Process **30**, e2110 (2018)
15. Symons, C.R.: Function point analysis: difficulties and improvements. IEEE Trans. Softw. Eng. **14**(1), 2–11 (1988)
16. Heiat, A.: Comparison of artificial neural network and regression models for estimating software development effort. Inf. Softw. Technol. **44**(15), 911–922 (2002)
17. Hai V.V., Nhung H.L.T.K., Hoc H.T.: Productivity optimizing model for improving software effort estimation (2020)
18. Ross, A., Willson, V.L.: Paired samples T-test. In: Basic and Advanced Statistical Tests. SensePublishers, Rotterdam (2017)
19. Albrecht, A.J.: Measuring application development productivity. In: Proceedings of the Joint Share/Guide/IBM Application Development Symposium, pp. 83–92 (1979)
20. Behrens, C.A.: Measuring the productivity of computer systems development activities with function points. IEEE Trans. Softw. Eng. **SE-9**(6) (1983)
21. IFPUG: International Function Point Users Group. http://www.ifpug.org/
22. Abran, A., Robillard, P.N.: Function points: a study of their measurement processes and scale transformations. J. Syst. Softw. **25**(2), 171–184 (1994)
23. Albrecht, A.J., Gaffney, J.E.: Software function, source lines of code, and development effort prediction: a software science validation. IEEE Trans. Softw. Eng. **SE-9**(6), 639–648 (1983)