The International Journal of Engineering and Information Technology

journal homepage:www.ijeit.misuratau.edu.ly

The Correlation Aspects of Software Development Actual Effort and the Effected Factors

Amarif, M.,
Sebha University of Libya
mab.imaref@sebhau.edu.ly

Awidat, F.,
Sebha University of Libya
fat.awidat@sebhau.edu.ly

Abstract—**estimating the effort of software development is the focus of attention of software engineers, as it allows developers to determine the resources needed for the software project from the beginning until the final delivery. In fact, the actual effort of the software development depends on a set of basic factors that affect it either positively or negatively. These factors vary depending on the type and nature of the software. A few studies have been conducted in order to consider these factors and their correlation to the actual effort. Most of these studies have used methods such as a survey or a questionnaire. However, these methods lack of inaccurate measurement due to the variation of participated experiences and skills. Today, the world have emerged artificial intelligence technologies in most of fields to obtain higher accurate and fast results. This research uses feature selection intelligent algorithms such as Generic Univariate Select, Mutual Information, and RelifF algorithms to determine the correlation of software actual effort with the effected factors through XGBoost machine learning library. The NASA data set and the common effected factors of COCOMO model has been selected to determine the factors with the strongest correlation and influence on the actual effort. Through the results of this research, it has been found that complexity factor has recorded the largest correlation coefficient followed by the size of the software. The relationship between factors has been noticed and this contributes to the importance of the effected factors consideration and perfect software management.**

Index Terms—**software development, actual effort, effected factors, correlation, feature selection.**

I. INTRODUCTION

To develop any software project, it is necessary to estimate the effort and time required to complete it. Estimating the cost of software development is considered to be an essential factor for the success of any software project [1, 2, 3]. it gives both the developer and the customer a clear picture of the project and the needed resources to develop it. Developers follow several models to get an accurate estimate of the cost. One of the most common model is COCOMO model [4, 5].
For estimating the effort, COCOMO relies on a group of factors called effort multipliers (EM).
These multipliers vary in value depending on the type and nature of the project under construction. It also depends on a set of parameters that take estimated values according to the COCOMO model. The software development effort is related to a number of factors that influence it. Each of these factors has certain values according to the regression analysis of the projects in COCOMO I data set [6]. Based on this data, an estimation of the effort has been evaluated using the COCOMO model. In fact, the estimation effort might be inaccurate according to the limitation of the model [5].

Various techniques have been used for software cost estimation with respect to the effected factors. Although these factors have been identified, there is an urgent need to clarify the extent of the relationship and correlation aspects between these factors and the actual effort, especially with the development of technologies and the adoption of artificial intelligence algorithms in this regard.

It is possible to adjust the values of these factors according to the type and nature of the software so that the developers can obtain an optimal estimate of the required effort. This paper aims to identify the correlation between the actual software effort and the effected factors. It also seeks to adapt intelligent feature selection algorithms XGBoost to obtain the most effected features in software development actual effort. The following section describes the related work and section III explains the research methods. Section IV describes the results and discussion and section V gives the conclusion of the paper.

II. RELATED WORK

The effort of software development is defined as the number of human resources required to complete the project in a certain time [1, 5]. This effort is estimated using some existing models which rely on the person-hour unit or person-month units according to the tested dataset that have been used by the model.

In view of previous studies, and with regard to identifying factors influencing effort estimation, a study has been conducted by Bryce [7] in order to find out the

Received 02 May , 2024; revised 08 May, 2024; accepted 15 Mar 2024. Available online 08 Aug, 2024.

estimation issues in software project management. He has developed a questionnaire to conduct a survey to find out the factors that impact the project. He has mentioned that the complexity is the most effected factors in software development effort with 100% agree followed by the staff characteristics with 75%. He has mentioned that the number of line of code is less important issue than the function point of code. He has also identified other issues which might effect in the project effort estimation.

Another study has been conducted by Robert and his colleagues in order to identify the factors that affect the software development cost [8]. The study uses data from 50 projects performed at one of the largest banks in Sweden to identify factors that have an impact on software development cost. Correlation analysis of the relationship between factor and software effort was carried out using ANOVA statistical test and regression analysis. They have found that the most impact factors are duration, consultants and project participants.

The study [9] has conducted a survey to find out the important factors that affect the software development effort. a survey was conducted to consider the techniques for cost estimation. They have been found that function point method is the most common method for calculating the size and project complexity. The statistical tests were used to find out the most effected factors of project cost.
Most studies have been carried out for cost estimation of software development and the effected factors, but uses statistical tests or systematic literature review [10]. Nowadays, an intelligent machine learning algorithms are used for identifying the effected factors and extract the features. These algorithms have the ability to predict the most effected features on the objective variable.

Based on that, this research uses an intelligent machine learning model which is XGBoost model [11]. This model is a robust machine-learning algorithm that can help us understand the data and make better decisions. XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models. By leveraging the capabilities of XGBoost, this research presents a novel approach for understanding the correlation between software development actual effort and the effected factors, moving beyond the limitations of previous studies.

III. RESEARCH METHODS

This research adopts the NASA93 dataset [6] to select the features most closely related to the software development effort. The description of the dataset is described in the following section.

A. The NASA Dataset

The NASA93 dataset was collected by NASA from five of its development centers. It comprises 93 projects carried out between 1971 and 1987. The dataset consists of 24 attributes of which 15 are cost drivers, as the approach is based on that used in COCOMO81. The size attribute was measured in estimated lines of code (KSLOC). The dataset is available online for free as a Software Engineering Repository data set in order to encourage researchers to work for accurate cost estimation and improvable predictive models of software engineering. The NASA datasets have been used to assess how well evolutionary algorithms function. The dependent characteristic of effort is the number of man-months needed to complete the project. The attributes have been divided into two categories according to COCOMO model. The first Category effects positively in the software development effort (target variable) as described in table I.

TABLE I. POSITIVE INCREASE FACTORS

factor	description
acap	analysts capability
pcap	programmers capability
aexp	application experience
modp	modern programming practices
tool	use of software tools
vexp	virtual machine experience
lexp	language experience

The second Category effects negatively in the software development effort (target variable) as described in table II.

TABLE II. NEGATIVE INCREASE FACTORS

factor	description
stor	main memory constraint
data	data base size
time	time constraint for cpu
turn	turnaround time
virt	machine volatility
cplx	process complexity
rely	required software reliability

B. Statistical Analysis

This study adopts the descriptive approach using statistical tests, which are the mean, average, and standard deviation, to obtain the correlation matrix, as shown in Fig. 1.

A correlation matrix is an array table showing correlation coefficients between various variables. Each cell in the table shows the correlation between two variables. A correlation matrix uses Pearson's Product-Moment Correlation (r) to find out the correlations between variables. The reasons for computing a correlation matrix is to summarize a large amount of data where the goal is to see patterns. Moreover, researchers commonly use correlation matrixes as inputs for exploratory factor analysis, confirmatory factor analysis, structural equation models, and linear regression when excluding missing values pairwise. For dealing with missing values when computing correlation matrix, the multiple imputation has been used.

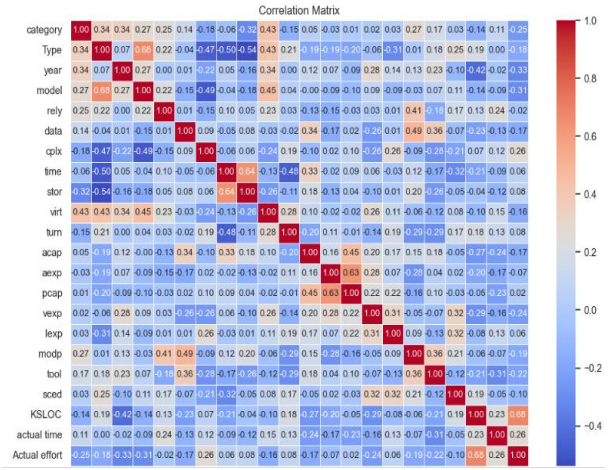


Figure 1. The correlation matrix.

According to the statistical analysis, it has been noticed that the program size was the most highly correlated with effort, at 68%, which makes sense because in most software, the more complex the program size, the more processing effort. It followed by the program complexity and time which reach up to 26%. The other factors' correlations are less than 10% .

C. *Machine Learning XGBoost*

XGBoost is an implementation of gradient-boosting decision trees. It has been used by data scientists and researchers worldwide to optimize their machine-learning models. It presents several advantages over traditional statistical methods. Firstly, it can automatically identify the most relevant features from the dataset, without the need for manual feature engineering or selection. Secondly, it can capture nonlinear dependencies between the factors and the target variable, which may be missed by linear regression models. Thirdly, it's relatively insensitive to noisy or outlier data points, making it more reliable for real-world datasets. Finally, the feature importance scores provided by it offer a clear and interpretable way to understand the relative impact of each factor on the target variable.

XGBoost is based on gradient boosting, a technique where new models are trained to correct the errors made by existing models. It builds an ensemble of trees in a sequential manner. It starts by defining an objective function that needs to be optimized. In regression tasks, this objective function typically measures the difference between the predicted values and the actual target values to calculate the mean squared error or root mean squared error (MSE). It's designed to be highly efficient, both in terms of computation and memory usage. It employs several algorithmic optimizations to speed up the training process. For large datasets that don't fit into memory, XGBoost supports out-of-core computation, which allows for processing data in smaller batches.

XGBoost uses a more advanced tree pruning algorithm compared to traditional gradient boosting. It uses a technique called "max depth" to control the complexity of the trees.XGBoost can handle missing values internally, without needing to pre-process the dataset. It learns the best imputation strategy during the training process. It can automatically handle missing values and zero entries in a more efficient manner to improve robustness and generalization.

Despite regularization, XGBoost might overfit, especially if the number of trees is very large. Tuning the hyper parameters of XGBoost can also be complex and time-consuming.

D. *XGBoost Description*

The objective of this research is to develop a code that performs feature selection using various algorithms and evaluate the performance of an XGBoost regressor trained on the selected features. The code aims to identify the most informative features and assess their impact on the predictive performance of the regression model. The key components of the code are as listed below.

- **Data Preparation**: the dataset is loaded, and features which is denoted by x and target variable which is denoted by y are separated.
- **Feature Selection Methods**: a three feature selection methods are employed. They are Generic Univariate Select, Mutual Information, and ReliefF. Each method is applied to select the most relevant features from the dataset.
- **Model Training and Evaluation**: the selected features using the three pervious methods are used to train the XGBoost model. The predictions are made on a holdout test set. Mean Squared Error (MSE) is computed to evaluate the model's performance. The code generates the importance of selected features and the performance of the XGBoost model.

E. *Generic Univariate Method*

Generic Univariate Select is a feature selection method that evaluates the importance of each feature independently using a univariate statistical test and selects the best features based on their scores. It uses the mathematical equation of Chi-Squared test as in statistical test.

In Generic Univariate method, each feature is scored individually using a statistical test appropriate for the type of data and target variable. Common tests include ANOVA F-value for classification task, Chi-squared test for categorical data, and mutual information for both classification and regression. The features are ranked based on their scores. A predefined number of top-ranked features or those above a certain score threshold are selected.

The advantage of Generic univariate method is effective, simple and fast as it evaluates each feature independently. It helps improving model performance by removing irrelevant features as a preprocessing step in machine learning pipelines to reduce the feature set. Particularly, Generic Univariate is useful in high-dimensional datasets where many features are irrelevant. In the other side, it doesn't consider feature interactions and may not perform well if important interactions between features exist. For these reasons, it's particularly useful in high-dimensional datasets where many features are irrelevant

F. *Mutual Information Method*

Mutual Information (MI) is a measure of the mutual dependence between two variables. In feature selection, it quantifies the amount of information obtained about one

variable through another variable, thereby identifying the relevance of features in predicting the target variable. It measures the reduction in uncertainty of one variable given the knowledge of the other. For feature selection, it calculates the MI between each feature and the target variable. In this method, features are ranked based on their MI scores and the top-ranked features based on MI scores are selected. MI is able to capture any kind of dependency between variables, not just linear correlations, but it involves intensive computation, especially for large datasets.

G. ReliefF Method

ReliefF method is an extension of the Relief algorithm, which is a feature selection method designed to evaluate the quality of features by considering the differences between neighboring instances. It is particularly effective for identifying features that are important for distinguishing between instances that are similar in feature space. It's effective for datasets with many irrelevant features. It also robust to noise and can handle missing data. Commonly, ReliefF method is used in bioinformatics for selecting relevant genes from microarray data. It's also suitable for problems where feature interactions are important and must be considered. Expensive computations are required by ReliefF method, especially with large datasets but it's suitable for problems where feature interactions are important and must be considered.

IV. RESULT AND DISCUSSION

This research uses the NASA93 dataset [6] to extract the features most closely related to the software development effort. It uses the statistical analysis tests to find out the correlation of effort and effected factors. The results indicate that the size of the program is most connected to the software effort. It also show that the complexity of the development software absolutely increases the software effort as described in Fig. 1.

This research also measures the performance of three different feature selection techniques—Generic Univariate Select, Mutual Information, and ReliefF—when used in conjunction with the XGBoost regressor algorithm. The performance is measured using Mean Squared Error (MSE), with lower values indicating better performance.

First of all, the top ten features have been recorded using the three selected methods as described in table 3, 4, and 5. The scores of each feature have been valued as described in the given tables respectively. The data descriptions of each table are visualized in Fig. 3, 4, and Fig. 5 respectively.

TABLE III. TOP TEN FEATURES USING GENERIC UNIVARIATE METHOD

Factor/ Feature	Feature's score
KSLOC	49.85
cplx	18.43
time	15.27
sced	9.58
stor	6.39
data	6.26
rely	5.78
vexp	4.99
acap	4.40
virt	3.42

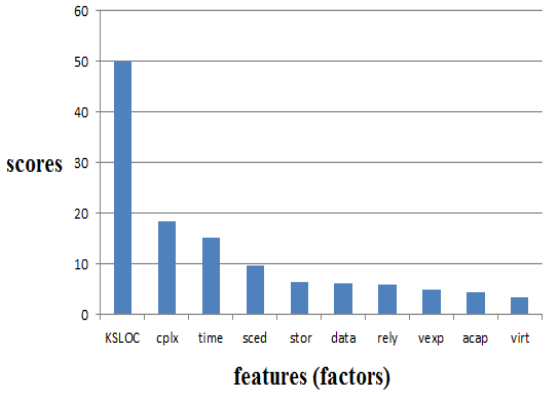


Figure 2. The results of Generic Univariate method

The results of features selection using Generic Univariate method indicate that the size of code (KSLOC) is the strongest related feature with the target variable (actual effort) followed by software complexity and time constraint for CPU. The other features have less relationship with the target variable (actual effort).

TABLE IV. TOP TEN FEATURES USING MUTUAL INFORMATION METHOD

Factor/ Feature	Feature's score
KSLOC	63.83
aexp	21.83
time	15.00
data	13.50
modp	11.41
vexp	10.17
tool	09.82
rely	07.61
turn	06.91
acap	06.82

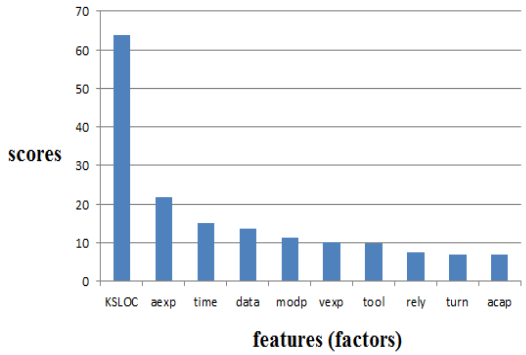


Figure 3. The results of Mutual Information method

The results of features selection using Mutual Information method indicate that the size of code (KSLOC) is also the strongest related feature with the target variable (actual effort) followed by application experience and time constraint for CPU. The other features have less relationship with the target variable (actual effort).

TABLE V. TOP TEN FEATURES USING RELIEFF METHOD

Factor/ Feature	Feature's score
aexp	49.46
KSLOC	45.32
pcap	43.84
time	43.17
lexp	42.93
sced	41.09
modp	37.42
acap	36.51
stor	43.68
turn	34.26

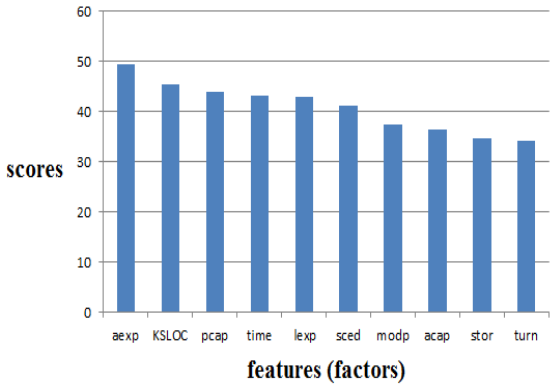


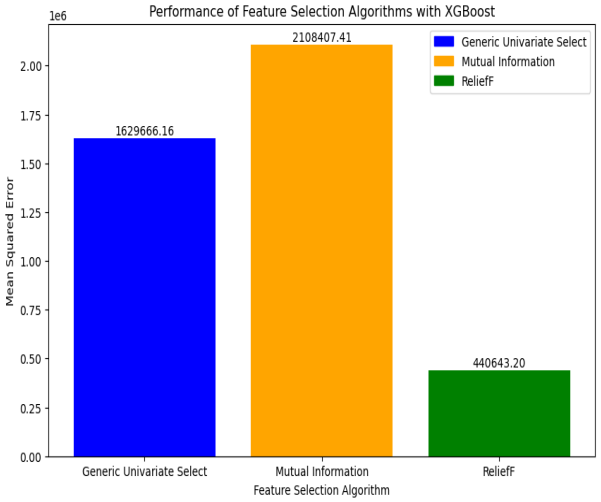
Figure 4. The results of ReliefF method

The results of features selection using ReliefF method indicate that application experience is the strongest related feature with the target variable (actual effort) followed by the size of the software code and programmers capability. It has been noticed that most features are very close effectively in the actual effort using ReliefF method. This suggests that the selected features have a stronger predictive power and that ReliefF is capturing important interactions and dependencies that the other methods miss.

After evaluating the performance of each feature selection algorithm, the script plots a bar chart showing the mean squared error for each algorithm. Each bar is colored according to the corresponding feature selection method. Text labels displaying the error rate are added above each bar. Finally, a legend is created to distinguish between the algorithms as described in Fig. 5.

Figure 5. The comparison of the results of XGBoost model

According to the obtained results, it has been noticed



that the use of the XGBoost algorithm in this research presents a significant advancement over previous studies that have relied on traditional statistical methods. By automatically identifying the most influential features and assessing their relative impact, the XGBoost-based approach offers a more comprehensive and data-driven understanding of these complex relationships.

The findings from the XGBoost analysis, which highlight the importance of factors such as complexity and size, provide valuable insights that can inform more accurate software development effort estimation models. Furthermore, the interpretability of the XGBoost feature importance scores can help software project managers make more informed decisions and prioritize the most critical factors during the development process.

This research demonstrates the value of leveraging advanced machine learning techniques, such as XGBoost, to gain deeper insights into the factors influencing software development effort. By moving beyond traditional statistical methods, the study provides a novel and innovative approach that can contribute to the ongoing efforts to improve software project management and cost estimation practices.

V. CONCLUSION AND FUTURE WORKS

This research aims to find out the correlation of software development actual effort and the effected factors using the statistical tests and the artificial intelligent machine learning model which is XGBoost. This is to show that machine learning are more predictable and fast than the statistical tests.

Comparing the results of algorithms absolutely give insights into the consistency of feature importance rankings and the robustness of selected features.

Features with high importance scores across multiple algorithms are likely to be strong predictors of the target variable and may warrant further investigation or prioritization in model development. These importance scores can guide feature selection, model interpretation, and decision-making in regression analysis, helping to identify the most influential factors affecting the predicted outcome (actual effort in this case).

There are more other factors that might affect the software development effort. The consideration of these factors could be as a future work. Applying the same procedure to various dataset is also another plan.

REFERENCES

[1] Ian Sommerville, Software Engineering, 10th edition, Pearson Education, 2016.

[2] Sai Mohan Reddy Chirra, Hassan Reza, 2019, "A Survey on Software Cost Estimation Techniques", in Journal of Software Engineering and Applications.

[3] Junaid Rashid, Muhammad Wasif Nisar , Toqeer Mahmood, Amjad Rehman , Syed Yasser Arafat ,2020. " A Study of Software Development Cost Estimation Techniques and Models", in Mehran University Research Journal of Engineering and Technology.

[4] Boehm, B. W., C. Abts, A. W. Brown, S. Chulani, B K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. 2000. *Software Cost Estimation with COCOMO II*. Englewood Cliffs, NJ: Prentice-Hall.

[5] Anil Jadhav, Mandeep Kaur, and Farzana Akter , 2022, "Evolution of Software Development Effort and Cost Estimation Techniques: Five Decades Study Using Automated Text Mining Approach" ,Mathematical Problems in Engineering.

[6] nasa93-dem.Available: http://promisedata.googlecode.com/svn/trunk/effort/nasa93dem/nasa93-dem.arff

[7] Schroeder, Bryce G. "Estimation issues in software project management." Project Management Institute, 1991.

[8] Lagerström, Robert, Liv Marcks von Würtemberg, Hannes Holm, and Oscar Luczak. "Identifying factors affecting software development cost." In *The Fourth International Workshop on Software Quality and Maintainability (SQM), Madrid, Spain, March 15-18, 2010*. 2010.

[9] Gangwani, Deepa, and Saurabh Mukherjee. "Analyzing the impact of different factors on software cost estimation in today's scenario." *Journal of Software Engineering and Applications* 8.5 (2015): 245-251.

[10] Khan, J. A., Khan, S. U. R., Iqbal, J., & Rehman, I. U. (2021). Empirical investigation about the factors affecting the cost estimation in global software development context. *IEEE Access*, *9*, 22274-22294.

[11] T Chen and C Guestrin, " Xgboost: A scalable tree boosting system," In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785-794.