Cluster Computing (2022) 25:2779–2792
https://doi.org/10.1007/s10586-021-03447-5

Software enhancement effort estimation using correlation-based feature selection and stacking ensemble method

Zaineb Sakhrawi[1] · Asma Sellami[2] · Nadia Bouassida[2]

Received: 9 September 2021 / Revised: 9 September 2021 / Accepted: 28 September 2021 / Published online: 23 November 2021

Abstract
Estimating software enhancement efforts became a challenging task in software project management. Recent researches focused on identifying the best machine learning algorithms for software maintenance effort estimation. Most of the research publications investigated the use of ensemble learning for improving software effort estimation. Intending to increase the estimation accuracy over individual models, this paper investigates the use of the stacking ensemble method for estimating the enhancement maintenance effort (EME) of software projects. This paper makes a comparison between two machine learning-based approaches for estimating software EME: The M5P (as an individual model) and the stacking as an ensemble method combining different regression models (GBRegr, LinearSVR, and RFR) using the ISBSG dataset. A correlation-based feature selection (CFS) algorithm is basically used to achieve efficient data reduction. The selected ML techniques-based approaches were trained and tested on a dataset with relevant features leading to the improvement of estimate accuracy. Results show that the software EME estimation using CFS and stacking ensemble method is improved in terms of mean absolute error (MAE) = 0.0383 and root mean square error (RMSE) = 0.1973.

Keywords Enhancement effort estimation · Correlation-based feature selection · M5P ML algorithm · Stacking ensemble method

1 Introduction

Software enhancement effort estimation (also termed prediction) has recognized the growing importance by several software organizations since most software enhancement projects allocated lower cost compared to new development [1]. Software enhancement is considered a critical activity in the software development life cycle. It is defined as "changes made to an existing application where new functionality has been added, or existing functionality has been changed or deleted. This would include adding a module to an existing application, irrespective of whether any of the existing functionality is changed or deleted" [2]. Since changes are frequent throughout the Software Development Life Cycle (SDLC), software project planning should be reviewed frequently. And therefore, the software enhancement effort estimation should be accurate.

The benefits of using enhancement effort estimation models are numerous. For instance, estimation models can help in making decisions about when to restructure or re-engineer a software component to make it more maintainable, know better the underlying reasons about the difficulty of correcting specific kinds of errors [3]. In this area, Machine Learning (ML) techniques are widely used for achieving better estimation. ML techniques are the most suitable for dealing with modeling of high dimensional problems [4]. But there is a lack of consensus among researchers about the technique that can achieve better estimation [5]. Several techniques have been proposed for estimating software enhancement effort, including statistical regressions or machine learning models such as case-

✉ Zaineb Sakhrawi
zeinab.sakhraoui@fsegs.rnu.tn

Asma Sellami
asma.sellami@isims.usf.tn

Nadia Bouassida
nadia.bouassida@isims.usf.tn

[1] Faculty of Economics and Management of Sfax, University of Sfax, Sfax, Tunisia

[2] Higher Institute of Computer Science and Multimedia, University of Sfax, Sfax, Tunisia

based reasoning, neural networks (NN), decision trees (DT), Bayesian networks, support vector machines (SVM), genetic algorithms, genetic programming, and association rules (ARU) [5].

In our previous work [6], we used separately four various ML techniques (M5P, GBRerg, LinearSVR, and RFR) for estimating software enhancement effort. The four selected ML techniques were trained and tested using industrial projects from the International Software Benchmarking Standards Group (ISBSG) Release 12 dataset [7]. The first phase focused on the selection of the optimal features set in the ISBSG dataset using the CFS algorithm, while the second phase focused on estimating the enhancement effort based on the optimal features set obtained from the first phase. The findings of our previous empirical study were as follows:

- The correlation coefficients computed between enhancement functional size and enhancement effort have a value of 0.5 which indicates a good correlation. The enhancement functional size was therefore chosen as the primary independent variable.
- The use of ML techniques without feature selection generated good accuracy. However, the use of ML techniques with the CFS algorithm gives better results.
- The empirical results suggested that M5P is the most accurate model with small MAEs = 0.0612 and with quite good performance that can achieve 99%.

More recently, research publications investigated the use of ensemble learning for improving software effort estimation [8, 9]. Various ensemble methods are considered for estimating software effort such as [10]:

- Bagging: The estimation is based on merging the same type of model.
- Boosting: The estimation is based on the use of sequential method to reduce the bias.
- Stacking: The estimation is done from multiple individuals models to build a novel model.

Based on the obtained results [6] from our previous work, we aim in this paper to build a stacking ensemble method to accurately predict the total enhancement effort for enhancement projects in person-hours. Our constructed Stacking ensemble method combines three different Machine Learning models (GBRegr, LinearSVR, and RFR). Estimation result using staking will be compared to those using a single algorithm (M5P). The M5P is recently used for software estimation [11–14]. M5P is a powerful implementation of Quinlan's M5 algorithm for inducing both Model Trees and Regression Trees [15]. The main motivation for this research study arises from the fact that existing single techniques used for estimating software effort suffer from several limitations [16] while other innovative approaches such as the ensemble method are yet to be adopted in the industry for estimating software effort. This study investigates the use of CFS and stacking ensemble methods for improving enhancement effort estimation. First, the M5P, GBRegr, LinearSVR, and RFR are used separately. Second, the stacking ensemble method that combines GBRegr, LinearSVR, and RFR is used. And finally, comparisons of the experimental results are made. The hypothesizes investigated in this research are the following:

- H1: The enhancement effort estimation accuracy with the stacking ensemble method is statistically better than that obtained with M5P when the functional change Size is used as the independent variable.
- H2: The use of the CFS algorithm improves the accuracy of the selected ML methods.

The rest of the paper is organized as follows: in Sect. 2, we present the background and the related work. The detailed description of our research methodology consisting for achieving better software enhancement effort estimation is presented in Sect. 3. In Sect. 4, we intend to discuss the experimental results. Our evaluation is performed also throughout threats to validity presented in Sect. 5. Finally, we conclude the paper and we give directions for future works in Sect. 6.

2 Background and related work

2.1 Software enhancement effort estimation and machine learning

Enhancement is considered as a type of adaptive maintenance [2]. Regarding the use of ML for software maintenance effort estimation models, we identified 18 studies published between 1995 and 2020. The models in these 18 studies were statistical regressions [17–21], neural networks [22, 23], SVR [24], rule based [23, 24], Bayesian network [25], analogy [26], pattern recognition approach termed optimized set reduction [21], general regression [22], support linear regression models [22], support vector regression [24], and decision trees stochastic gradient boosting [27].

Results showed that there was not a statistically significant difference in the estimation accuracy among the proposed models. A major challenge for the research community is to develop a good theoretical understanding of maintenance and evolution which are scaled to industrial applications [28]. Several studies lack clarity on how the data were prepared and used, which makes it difficult to compare results among studies as well as replicate them [29]. More recently, the use of ensemble method

combining more than one single ML technique has achieved attention in the software engineering research [30]. Hence, a systematic review conducted by Idri et al. [31] and Alsolai et al. [32] have confirmed that the ensemble methods outperformed their constituents (single models). The ensemble method has revealed promising capabilities in ameliorating the accuracy over single models [33]. It contributes to better accurate results even when compared to deep learning models [34]. Indeed, the more diverse the constituents are, the better the ensemble method outputs will be distributed around the desired output [35, 36]. Despite that, in the area of effort enhancement maintenance estimation, ensemble methods are not yet adopted. This is the first study to our knowledge that investigates the use of the ensemble method in software maintenance effort estimation. The main motivation behind using the ensemble method in this work is that it makes the model more reliable and robust due to the advantages of using more than one ML technique for estimation [37]. Hence, with the creation of an ensemble method if any of the used models perform poorly, the ensemble method can reduce the error using many models [38].

2.2 Stacking ensemble method

The stacking model is invented by Wolpert [39]. It is recently used for estimating software effort [30, 34]. The stacking model combines lower-level Machine Learning techniques for achieving more accurate estimation. The constructed linear estimation model consists of two learning levels [40]. The first learning level is called Level-0, where models are trained and tested in independent cross-validation examples from the original input data. Then, the output of Level-0 and the original input data is used as input for level-1, called generalized (i.e. the meta-model). The Level-1 is constructed using the original input data and the output of level-0 generalizers [40].

2.3 Feature selection methods

Feature selection methods or techniques can be classified into three categories: Filters, Wrappers, and Hybrid algorithms [41]. The Filter methods select the features based on the characteristics of the dataset without involving any learning technique. Afterward, this subset of features is presented as input to a classification/regression estimation algorithm [42]. The Wrapper methods select the feature subset based on the performance of given learning techniques according to a performance measure. And Embedded or Hybrid methods perform the selection step and model building simultaneously or combine filter and wrapper techniques. One of the measures used for feature selection is the dependency measure. Many dependency-based algorithms have been proposed. In this study, we will use correlation-based feature selection (CFS) since it can evaluate all the possible combinations [43]. It can also update the subset of the selected features during the evaluation process instead of the greedy forward selection and greedy backward elimination that do not update the subset of features during the evaluation process [44]. CFS employs correlation to evaluate a feature subset that derived from Pearson correlation coefficient [43]. This method is a multivariate Feature Filter. That means that it assesses different feature subsets and chooses the best one. CFS was proposed by Hall [43] to evaluate subsets of features according to the heuristic evaluation function. This study was based on the hypothesis "A good feature subset contains features highly correlated with the class, yet uncorrelated with each other" [43]. Due to the ability of the Feature selection algorithm to produce good subsets of features, its use with the ensemble method will be effective for improving ensemble methods accuracy [45]. This observation was also founded by Hosni et al. [44] in their empirical study, where the CFS ensemble generated better results than the RReliefF ensemble. The choice of feature selection methods differs among various application areas [46]. Table 1 presents the findings that used filter feature selection for software effort estimation.

There are relatively few studies that investigated the use of CFS algorithm in the area of software enhancement effort estimation for both individual and ensemble models. Nevertheless, a number of research studies confirmed the effectiveness of CFS algorithm and ML techniques for software effort estimation [41, 44, 44, 45] .

3 Research process

In this paper, we will extend our previous research methodology [6] by setting up two new models using the CFS algorithm to predict software enhancement maintenance effort. The first model is constructed using four selected regression ML techniques (M5P, LinearSVR, GBRegr, and RFR) separately. While, the second model combines three models (LinearSVR, GBRegr, and RFR) that will construct the stacking ensemble method. Finally, we make a comparison of the estimation accuracy of the two mentioned models. We aim to identify whether the use of the CFS algorithm with the stacking ensemble method improves the performance of the estimation model versus the use of the CFS algorithm with the M5P model.

Table 1 Literature review on CFS algorithm used for software effort estimation

Authors	Estimation techniques	Type of filter feature selection	Type of FS algorithms
Hosni et al. [44]	K-NN, SVR, MLP, and DTs	Correlation based feature selection (CFS) and RReliefF on the estimation accuracy of heterogeneous (HT) ensemble	Correlation based Feature selection (CFS)
Deng et al. [46]	k-NN	Pearson coefficients and the ReliefF index	ReliefF index
Blessie et al. [41]	Without ML	CFS subset evaluation consistency-subset evaluation FCBF algorithm	CFS subset evaluation

Table 2 First selection of data concerning software enhancement projects from the ISBSG dataset

ISBSG data field	Selected values	Discarded values	Projects
Data quality rating	A, B	C,D	1084
Count approach	COSMIC	IFPUG, NESMA, FISMA, etc.	449
Development type	Enhancement	New development and redevelopment	302

3.1 Data preprocessing

The dataset used for training and testing the estimation model is obtained from the ISBSG Release 12 [18]. The ISBSG dataset is widely used for software project estimation [47]. It includes new, enhanced, and re-develop software projects. It has been extensively reviewed for its applicability to building effort estimation models, including the effects of outliers and missing values [48]. The effort expended on the support activities is reported in person-hours. We selected the data regarding "enhancement" as the "development type" where the "count approach" was the COSMIC Functional size measurement method. In addition, we consider only data with soundness and a high level of integrity (i.e., records having "Data Quality Rating" of "A" or "B"). To exclude trivial projects, the following filters were applied:

- Normalized work effort (full life cycle effort for project) equal to or greater than 80 person/hours.
- Development types other than enhancement were excluded.

Table 2 lists the data fields, the corresponding values selected in this study, the discarded values, and the number of projects. After the preprocessing phase, we selected a total of 17 attributes.

3.2 Constructing estimation models

This section presents a series of experiments to investigate the performance of estimation models with the use of the CFS algorithm. We have constructed two estimation models presented in Fig. 1. The first model constructs four ML techniques (M5P, LinearSVR, GBReg, and RFR) for estimating enhancement effort. The chosen models are trained and tested separately on the ISBSG dataset with relevant features using the CFS algorithm. The second model constructs a stacking ensemble method (that combines LinearSVR, GBRegr, and RFR). For this second model, the meta-model provided via the "final_estimator" argument (LinearSVR) is trained to combine the estimation of the chosen regression ML techniques provided via the "estimators" argument (GBReg, RFR). Each regression model is trained on the ISBSG dataset with relevant features filtered using the CFS algorithm allocated for training. Then the outputs of "estimators" are fed into the "final_estimator", which combines each regression estimator model with a weight and delivers the final estimation. For the first set of experiments, the classic approach is to do a simple 70%–30% split. We split data into training and validation/test set. The training set is used to train the model, and the validation/test set is used to validate it on data it has never seen before. The selected ML techniques are trained and tested for various sorts of experiments using features selected from the preprocessing phase. Thereafter, to carry out the experiments, different tools was used. Building the M5P model (tree-based model) has been carried out using Weka software[1]. It is widely used for teaching, research, and industrial applications. It contains a plethora of built-in tools for standard machine learning tasks. For the feature selection methods, 10-fold cross validation and validation test estimation of GBRegr, SVR and RFR models were performed using the Google Colaboratory[2] python programming. Google Colaboratory known as Google Colab is the current inventory tool [49]. It provides GPU for research to the people who do not have

[1] https://www.cs.waikato.ac.nz/ml/weka/.

[2] https://colab.research.google.com/notebooks/intro.ipynbrecent=true.

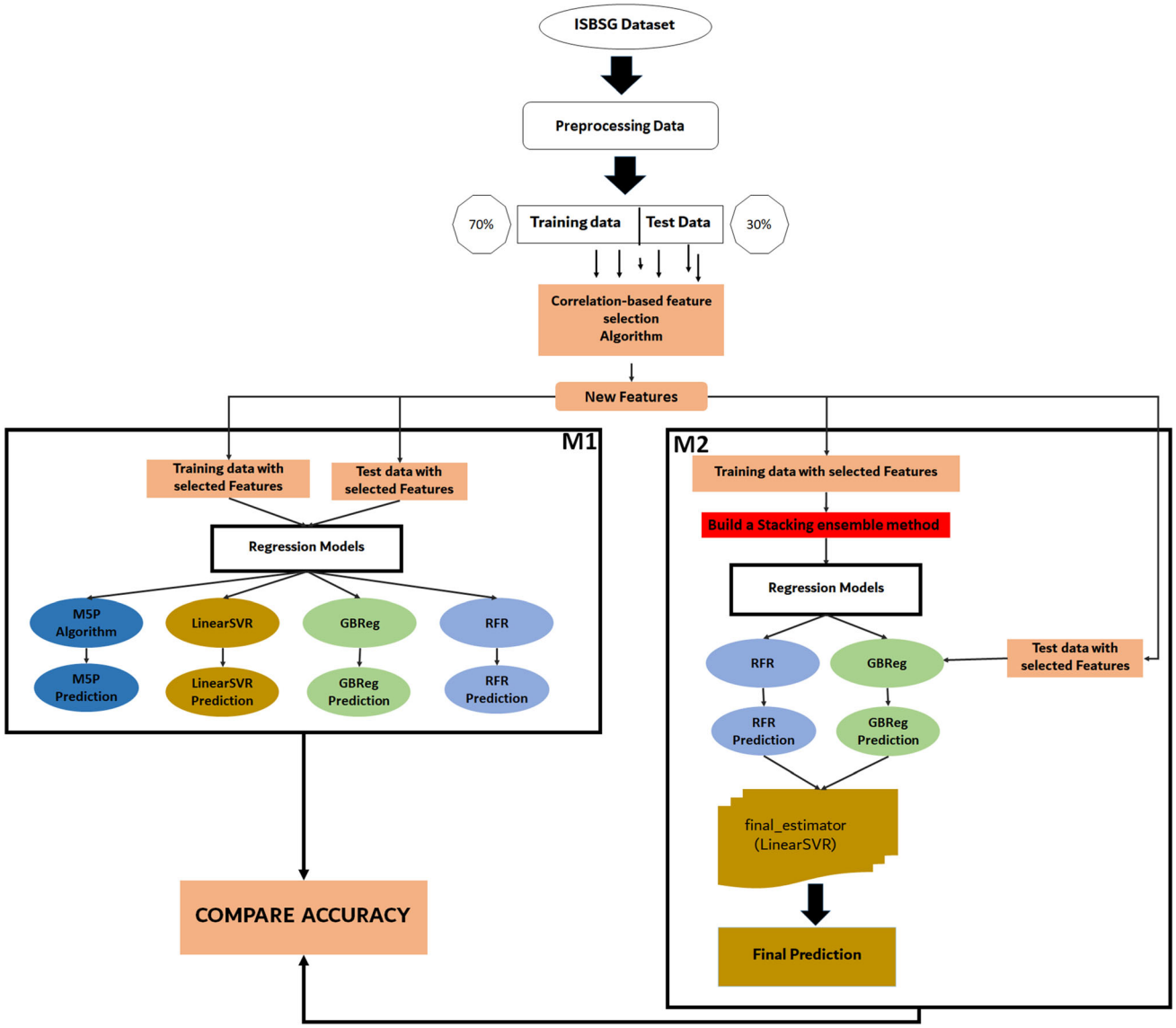


Fig. 1 Research method design

Table 3 Parameters values for grid search

ML techniques	Parameters
M5P	Instances = 5
GBRegr	random_state = 0; min_samples_split = 2
LinearSVR	Kernel = Linear; Complexity = {1,2}; epsilon = {0.2}; Deviation = {0.001, 0.0001}
RFR	random_state = 0; min_samples_leaf = {1,2,3}; Max_depth = {2,4,6}; min_samples_split = {2}

enough resources or cannot afford one. Table 3 lists the selected ML techniques with their corresponding predefined range of parameters values.

3.3 Experiments results

This section evaluates the estimation performance of the two constructed models where two experiments are conducted. In each constructed model, we propose to use the

CFS algorithm. That is after applying the CFS algorithm, we randomly split data with relevant features into two subsets: a training set and a test set. To evaluate the accuracy of the prediction models, we used a wide set of evaluation metrics [47, 48] such as root mean square error (RMSE) and mean absolute error (MAE). We also used the Standardized Accuracy measure (SA) based on MAE proposed by [50]. We also used the cross-validation method [51]. We partitioned the validation size with $K = 10$. It is well-known since the number of the selected ML model fitting to get the estimate now becomes independent of the size of the training sample [52].

3.3.1 Correlation-based feature selection (CFS) algorithm

Once the appropriate projects have been selected (i.e., projects with high quality of data), then we propose to use the CFS algorithm for selecting the features that are relevant for software enhancement effort estimation. The main challenge when using correlation-based Filters is related to the starting points for feature subsets generation [44]. To handle the missing values in a feature, CFS replaces the missing values by taking into account the average value for continuous features and the most common value for discrete features [44]. That is after applying the CFS algorithm, we determine which features globally and consistently appear in the optimal set of features. The filtering here is done by using correlation matrix and Pearson correlation [53].

Pearson correlation Pearson's correlation coefficient is a measure of the strength of the association between two variables [54]. In our research, we will plot the Pearson correlation heat map (see Fig. 2). After the preprocessing phase, we selected a total of 17 attributes where 16 are independent variables and one is the dependent variable (NormalizedWorkEffort). This correlation coefficient is a single number that indicates both the strength and direction of the linear relationship between two continuous variables. Values can range from -1 to $+1$ [54].

- Strength: The greater the absolute value of the correlation coefficient, the stronger the relationship. When the value is in-between 0 and $+1/-1$, there is a relationship, but the points do not all fall on a line.
- Direction: The sign of the correlation coefficient represents the direction of the relationship. Positive coefficients indicate that when the value of one variable increases, the value of the other variable also tends to increase. Negative coefficients represent cases when the value of one variable increases, the value of the other variable tends to decrease.

Since correlation coefficients which magnitude are less than 0.3 have little if any (linear) correlation [54], only the features correlating larger than 0.4 (taking into account absolute value) are selected with the output variable. The use of the CFS algorithm selects 37.5% (6 out of 16) of features (see Table 4). Note that the CFS algorithm is used not only to select features but also to evaluate the impact of the enhancement size (i.e., functional size of the functional change) feature on the accuracy of the software enhancement effort estimation.

It has been observed that COSMIC sizing is an efficient method for measuring not only software size but also the functional size of the functional change that may occur during the Software Life Cycle [55]. Figure 2 shows that the correlation coefficients value between enhancement functional size and enhancement effort is equal to 0.5. This investigation indicates an acceptable correlation when compared with other features (such as CHANGEWorkEffort and UnrecordedWorkEffort). Change functional Size was therefore chosen as the primary independent variable.

M5P algorithm Performance Assessment versus GBRegr, SVR and RFR models Using the CFS algorithm with the selected regression ML techniques separately leads to an accurate enhancement effort estimation when the enhancement functional Size is used as the independent variable (see Table 5). Error metrics (such as MAEs and RMSEs) reveal quite values using M5P (MAE = 0.0612; RMSE = 0.2514). It is evident from the results that M5P method delivers the best performance as compared to other three ML techniques with SA stands at around 99% (see Fig. 3).

3.3.2 Stacking ensemble method based on the use of correlation-based feature selection (CFS) algorithm

Regarding the above estimation results, our stacking ensemble method is based on the hypothesis that "When weak models are rightly aggregated, the strength of the union, therefore, leads to better performance and more accurate estimation of software enhancement effort". (1) Selecting which models to be used as "estimators" and model to be used as a meta-model and (2) making predictions by feeding estimators' predictions into a meta-model.

Selecting estimators and meta-model The main parameters of the stacking ensemble regression model are defined in scikit-learn[3] as follows: StackingRegressor(estimators, final_estimator = None, *) explained in Table 6.

Thus, we try to identify which technique from the three ML techniques can be used as "final_estimator" and which ones should be used as "estimators". In this case, we

[3] https://scikit-learn.org/.../sklearn.ensemble.StackingRegressor.html.

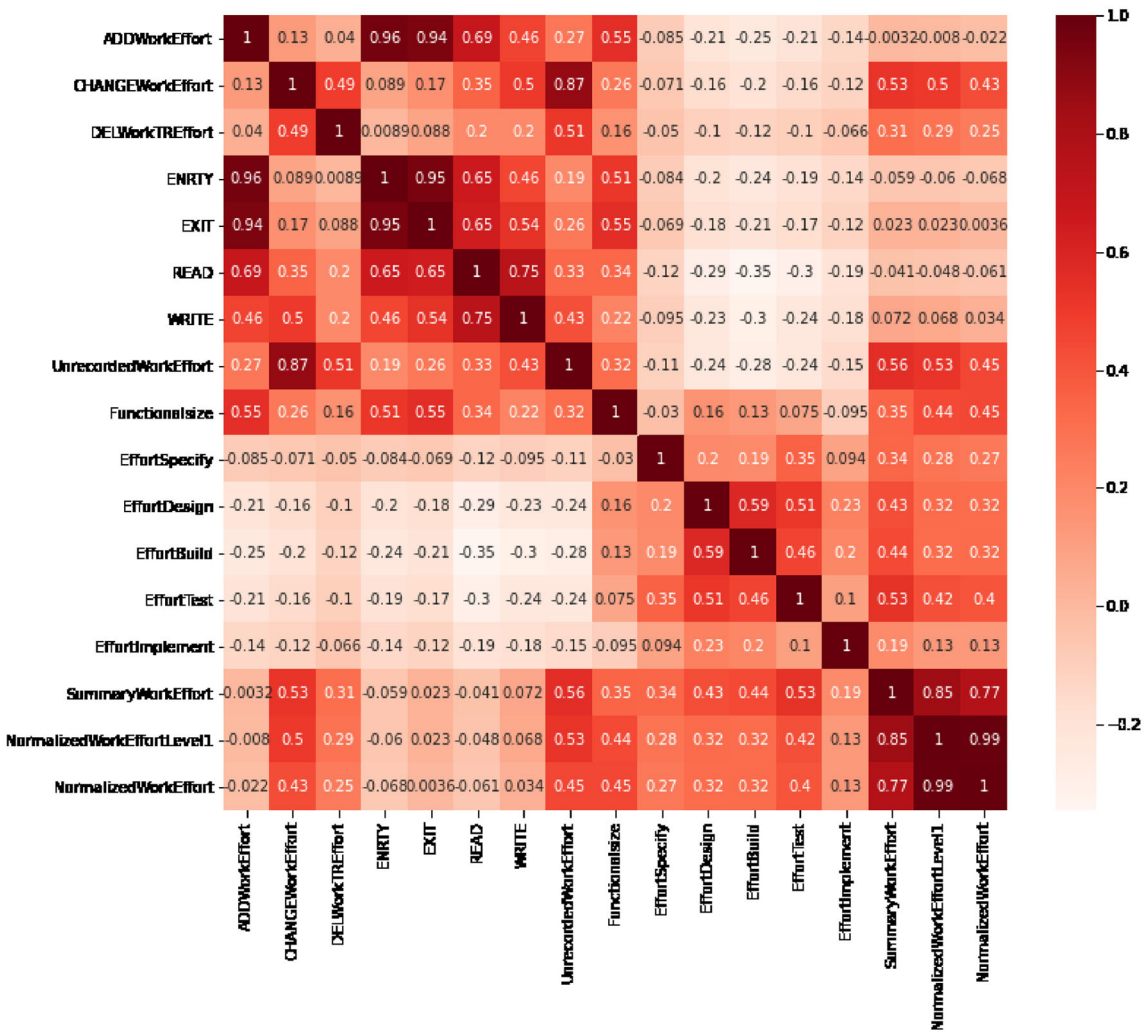


Fig. 2 Pearson correlation heat map

Table 4 Selected feature correlation

Features selection methods	Selected features with value (round(correlation target))
Pearson correlation	CHANGEWorkEffort = 0.4; UnrecordedWorkEffort = 0.5; Functionalsize = 0.5; EffortTest = 0.4; SummaryWorkEffort = 0.8; NormalizedWorkEffortLevel1 = 1

Table 5 Estimation analysis using MAE, RMSE and SA

Method/parameters	MAE	RMSE	SA (%)
M5P	0.0571	0.2514	99.36
GBRegr	0.2625	0.3447	85.43
LinearSVR	0.1110	0.3020	89.69
RFR	0.1665	0.3187	87.54

selected the r2_score evaluation metric[4] to evaluate the overall performance of the selected prediction model to provide an adequate combination. Table 7 illustrates the r2_score results where the best possible score stands at 1.0. Figure 4 shows the ML "estimators" and the average of their predictions.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html.

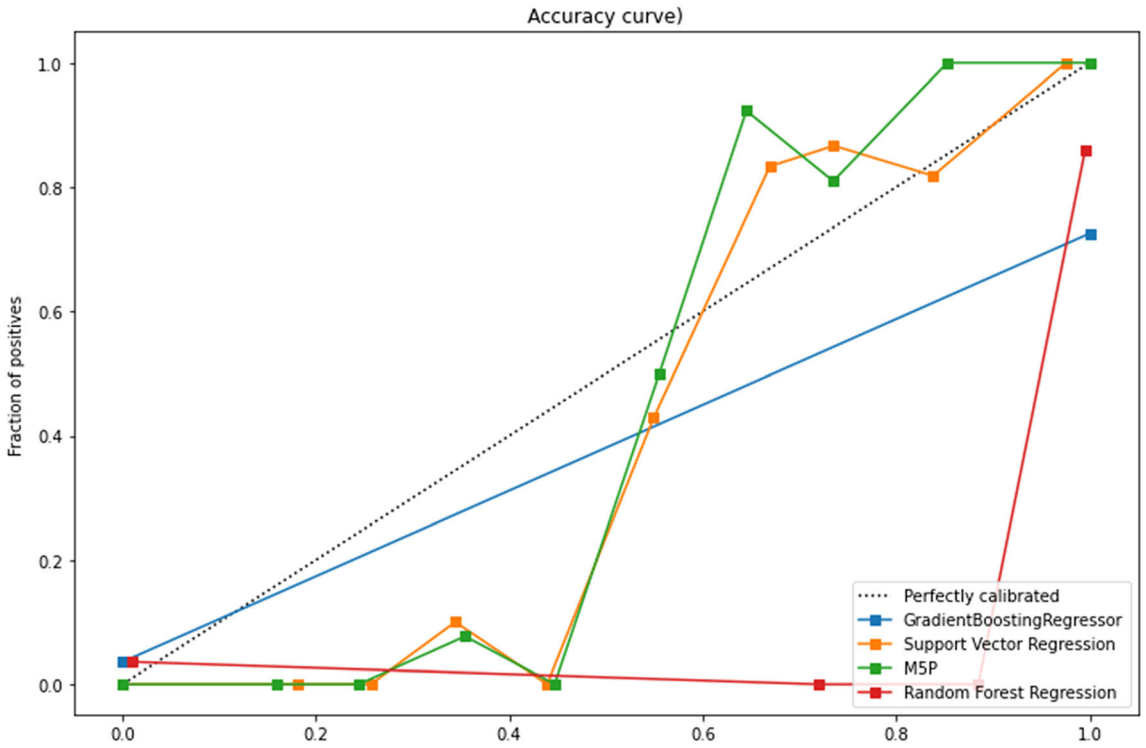


Fig. 3 ML techniques accuracy

Table 6 Stacking ensemble regression model parameters'

Parameters	Description
Estimators	Base estimators which will be stacked together
Final_estimator	An estimator which will be used to combine the base estimators

Table 7 Estimation analysis using R2 score

Method/parameters	R2 score
GBRegr	0.981
LinearSVR	0.956
RFR	0.980

Constructing the estimating software enhancement effort Regarding Table 7, LineanrSVR is selected to be used as the final_estimator. Table 8 shows the stacking ensemble method parameter that defines the best combination.

Using the CFS algorithm with the constructed stacking ensemble method leads to an accurate enhancement effort estimation when the enhancement functional Size is used as the independent variable (see Table 9). It is evident from the results that the stacking ensemble method delivers the best performance when compared with the other three ML techniques. The r2_score arises to 0.987 (see Figs. 3, 5).

4 Discussion and comparison

When comparing the estimation accuracy of the models using the same ISBSG dataset, we can accept the two following hypotheses derived from the one formulated in Sect. 1.

- H1: The enhancement effort estimation accuracy using the stacking ensemble method with an R2score of 0.98 is statistically better than that obtained using M5P the functional change Size is used as the independent variable.
- H2: The use of the CFS algorithm improves the accuracy of the selected ML methods.

The main reason behind selecting the enhancement functional size as a primary independent variable in our study is that the software functional size is correlated to the software project effort. And that affects the sensitivity of the software project [56]. Our previous experiment study was conducted to evaluate the accuracy of four machine learning techniques (M5P, GBRerg, LinearSVR, and RFR) separately. The selected ML techniques are used to provide

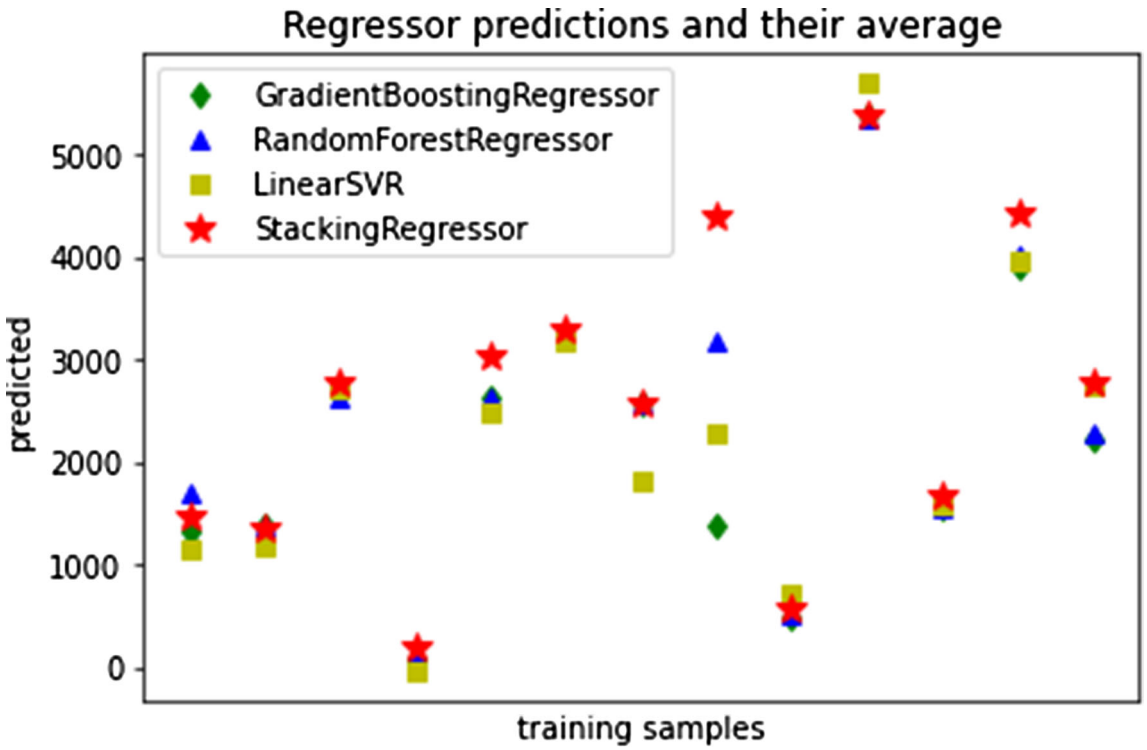


Fig. 4 ML "estimators" and the average of their predictions

Table 8 Parameters values for grid search

ML techniques	Parameters
Stacking model	estimators = [(GBRegr,RFR)], final estimator = LinearSVR()
GBRegr	alpha = 0.9, ccp_alpha = 0.0, criterion = 'friedman_mse', init = None, learning_rate = 0.1, loss = 'ls', max_depth = 3
LinearSVR	C = 1.0, cache_size = 200, coef0 = 0.0, degree = 3, epsilon = 0.1, gamma = 'scale', kernel = 'rbf', max_iter = − 1, shrinking = True, tol = 0.001, verbose = False
RFR	max_depth = 3, max_features = 1, max_leaf_nodes = None, min_samples_leaf = 25, min_samples_split = 2, min_weight_fraction_leaf = n_estimators = 25

Table 9 Estimation analysis using MAE, RMSE and r2_score

Method/parameters	MAE	RMSE	r2_score
M5P	0.0612	0.2514	0.985
Stacking regressor	0.0383	0.1973	0.987

the effort estimation of a new enhancement when software is being developed. Among the selected ML algorithms, M5P is the most effective. This is supported by the results with a minimum MAE of 0.0612. The effectiveness of M5P can be seen from the results obtained when applying a simple method. It has small MAEs and RMSEs values. A good accuracy (SA) of 99% is obtained when using the 10-fold cross-validation.

To identify the effective determinants for enhancement effort estimation, the importance of each feature is computed using the CFS algorithm. Furthermore, the model using the CFS algorithm delivers superior performance when compared to the model that used all the selected features (17 features). Thus, using the M5P ML algorithm improves the accuracy of enhancement estimation.

Furthermore, to ensure the above results, we have investigated the idea of using a stacking ensemble method by combining the weak ML techniques (GBRerg, LinearSVR, and RFR). Experimental results are compared with the M5P algorithm (see Table 9). The effectiveness of

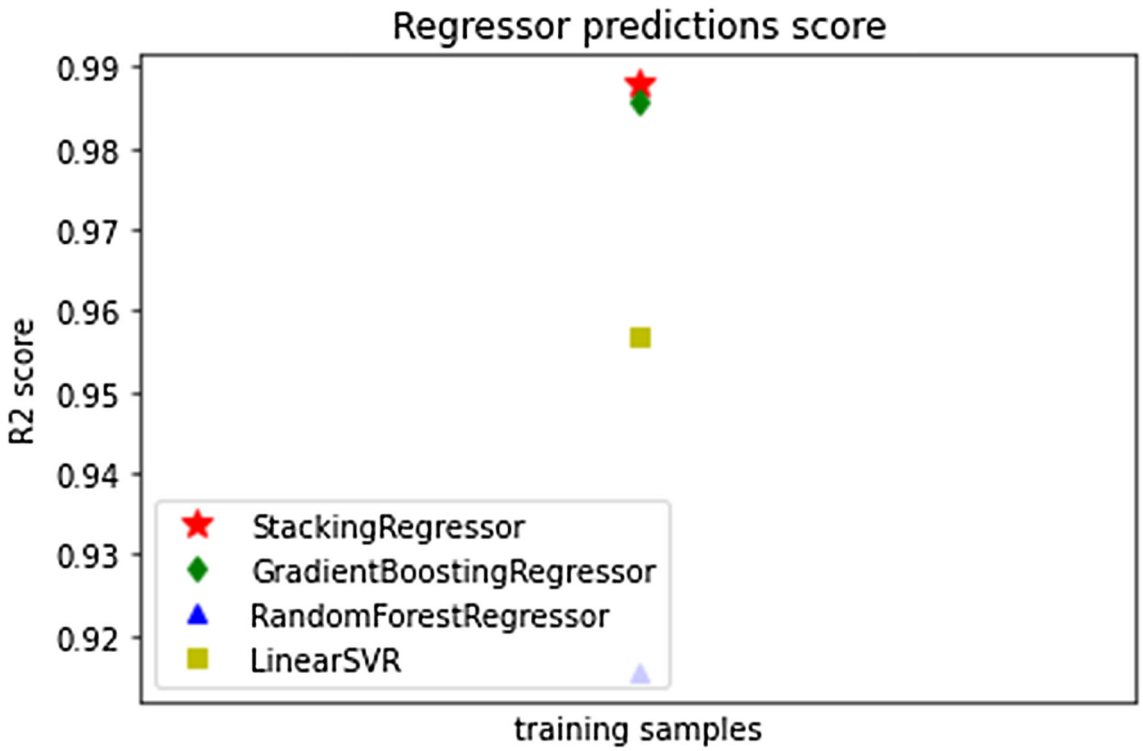


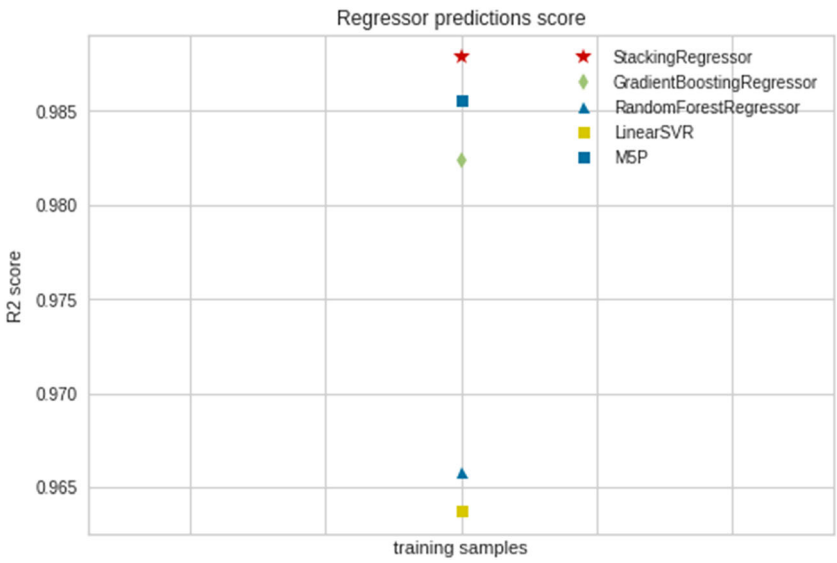
Fig. 5 Regressor estimation score

the stacking ensemble method can be seen in the results (see Figs. 4 and 6). This is supported by the results with the minimum MAE of 0.0383, RMSE of 0.1973, and a good r2_score of 0.987 (Fig. 7).

5 Threads to validity

In this section, we discuss the threats to the validity of this research study according to the guidelines proposed by [57]. The validity of this research results is pertinent to internal validity, external validity, and construct validity.

Fig. 6 ML techniques performance assessment



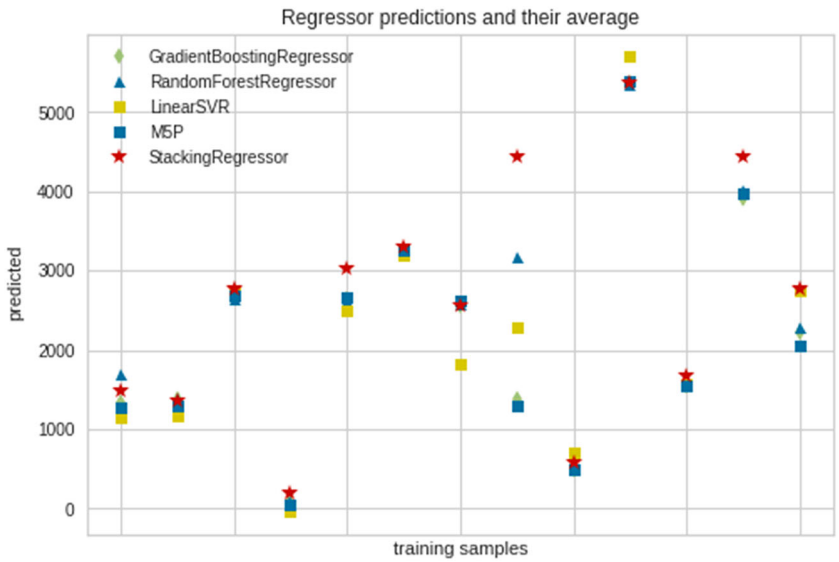


Fig. 7 ML techniques accuracy

5.1 Internal validity

Internal validity is related to (i) the size of the data set where the number of instances in the data set must be more significant, as well as (ii) the number and the nature of attributes used to estimate the software enhancement effort. To overcome this limitation (ii), we have used the CFS algorithm for selecting the attributes from one of the well-known historical software project datasets (the ISBSG dataset that contains many attributes). Since we restricted the study to numerical attributes only 17 features have been selected which constitutes 17% from all the attributes in the ISBSG dataset after the phase of prepossessing data. And, six features have been selected after using the CFS algorithm that constitutes 6% from all the attributes in the ISBSG dataset. This is why the findings of this work may differ from other studies that use other types of data.

5.2 External validity

External validity is related to the degree of the generalization of the results. The results of this study are based only on the use of the ISBSG R12 dataset. Conducting more experiments with other kinds of datasets that present quality characteristics are also required. There are two threats to the external validity of this study: (1) the first threat may come from the CFS algorithm. Although the experiments were performed using CFS, it is still compulsory to test other FS algorithms with different ML techniques. (2) The other threat may come from the selected dataset. We have used a single popular ISBSG dataset containing COSMIC Functional Points measures.

5.3 Threats of construct

Threats of the construct are related to (i) the degree of reliability of the features used to predict enhancement effort and (ii) the accuracy metric used for the analysis. In fact, (i) the estimation of enhancement effort in our study is provided based on the independent variable (i.e., the size of functional change). Even the results about the performance accuracy of the selected ML techniques provide a good accuracy equals to 99%, the correlation coefficients computed between enhancement functional size and enhancement effort is still a moderate value, this is due to the fact that enhancement functional size is identified at a high level of abstraction of the software life-cycle. Regarding the accuracy metric (ii), there has been some criticism of these metrics [47] such as ignoring the importance of the dataset quality. However, we adopted these four evaluation metrics (MAE, RMSE, R2 score, and SA) in our work.

6 Conclusion

The study was based on two main hypotheses (i) "A good feature subset is one that contains features highly correlated with the class, yet uncorrelated with each other" [43] and (ii) "When powerless (/weak) models are rightly combined we can obtain more accurate software enhancement effort estimation models". The constructed models are tested using the ISBSG dataset of historical software projects that take into account the use of software functional size expressed in terms of COSMIC Function Point units (as an independent variable). The findings of the research questions were as follows:

- The correlation coefficient computed between enhancement functional size and enhancement effort has a value of 0.5 which indicates a good correlation. The enhancement functional size was therefore chosen as the primary independent variable.
- The ML techniques without feature selection generated good accuracy. However, ensemble learning techniques with the CFS algorithm give better results.
- The experimental results suggested that:
  - M5P is more accurate with small MAEs = 0.0612 and with quite good performance of 99% compared to GBRerg, LinearSVR, and RFR.
  - The stacking ensemble method (combining GBRerg, LinearSVR, and RFR)is more accurate with small MAEs = 0.0383 and R2 score = 0.987 compared to M5P algorithm.

For future work, several extensions can be made. This work will be extended by exploring other ensemble methods for estimating software enhancement effort, to get more accuracy reaching 100% and other features selection methods including Backward Elimination, Forward Selection, Bidirectional Elimination, and RFE.

References

1. Bourque, P., Fairley, R.E., et al.: Guide to the Software Engineering Body of Knowledge (SWEBOK (R)): Version 3.0. IEEE Computer Society Press, Washington (2014)
2. Group, International Software Benchmarking Standards. Glossary of terms for software project development and enhancement, vol. 113, pp. 188–215 (2018)
3. De Almeida, M.A., Lounis, H., Melo, W.L.: An investigation on the use of machine learned models for estimating software correctability. Int. J. Softw. Eng. Knowl. Eng. **9**, 565–593 (1999)
4. Susto, G.A., Schirru, A., Pampuri, S.: Machine learning for predictive maintenance: a multiple classifier approach. IEEE Trans. Ind. Inform. **11**(3), 812–820 (2015)
5. Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C.: Systematic literature review of machine learning based software development effort estimation models. Inf. Softw. Technol. **54**(1), 41–59 (2012)
6. Sakhrawi, Z., Sellami, A., Bouassida, N.: An improved prediction of software enhancement effort using correlation-based feature selection and M5P ML algorithm. In: 17th IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2020), Antalya, Turkey, 2–5 November 2020, pp. 1–8. IEEE, Piscataway (2020)
7. ISBSG: Repository Data Release 12–Field Descriptions. e.Field Descriptions—Data Release 12. Document Provided as a Part of Data Set. International Software Benchmarking and Standards Group, South Melbourne (2013)
8. Hidmi, O., Sakar, B.E.: Software development effort estimation using ensemble machine learning. Int. J. Comput. Commun. Instrum. Eng. **4**, 1–5 (2017)
9. Minku, L.L., Yao, X.: Ensembles and locality: insight on improving software Effort Estimation. Inf. Softw. Technol. **55**, 1512–1528 (2013)
10. Shukla, S., Kumar, S.: A stacking ensemble-based approach for software effort estimation. In: Proceedings of the 16th International Conference on Evaluation of Novel Approaches to Software Engineering, vol. 1: ENASE, pp. 205–212 (2021)
11. Blaifi, S., Moulahoum, S., Benkercha, R., Taghezouit, B., Saim, A.: M5P model tree based fast fuzzy maximum power point tracker. Sol. Energy **163**, 405–424 (2018)
12. Ali, A., Gravino, C.: Using combinations of bio-inspired feature selection algorithms in software efforts estimation: an empirical study. In: 2019 13th International Conference on Open Source Systems and Technologies (ICOSST), vol. 163, pp. 1–8 (2019)
13. Al Asheeri, M., Hammad, M.: Machine Learning models for software cost estimation. In: 2019 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), vol. 163, pp. 1–6 (2019)
14. Khan, M.Z.: Particle swarm optimisation based feature selection for software effort prediction using supervised machine learning and ensemble methods: A comparative study. Invertis J. Sci. Technol. **13**, 33–50 (2020)
15. Quinlan, J.R., et al.: Learning with continuous classes. In: I5th Australian Joint Conference on Artificial Intelligence, vol. 92, pp. 343–348 (1992)
16. Wang, L., Zhu, Z., Sassoubre, L., Yu, G., Liao, C., Hu, Q., Wang, Y.: Improving the robustness of beach water quality modeling using an ensemble machine learning approach. Sci. Tot. Environ. **765**, 142–760 (2021)
17. Yu, L.: Indirectly predicting the maintenance effort of open-source software. J. Softw. Maintenance Evol. Res. Pract. **18**, 311–332 (2006)
18. Shukla, Ruchi, Misra, A.K.: Software maintenance effort estimation-neural network vs regression modeling approach. Int. J. Comput. Appl. **975**, 157–169 (2010)
19. De Lucia, A., Pompella, E., Stefanucci, S.: Assessing effort estimation models for corrective maintenance through empirical studies. Inf. Softw. Technol. **15**, 3–15 (2005)
20. Ahn, Y., Suh, J., Kim, S., Kim, H.: The software maintenance project effort estimation model based on function points. J. Softw. Maintenance Evol. Res. Pract. **15**, 71–85 (2003)
21. Jorgensen, M.: Experience with the accuracy of software maintenance task effort prediction models. IEEE Trans. Softw. Eng. **21**, 674–681 (1995)
22. López-Martín, C.: Predictive accuracy comparison between neural networks and statistical regression for development effort of software projects. Appl. Soft Comput. **27**, 434–449 (2015)
23. Shukla, R., Shukla, M., Misra, A.K., Marwala, T., Clarke, W.A.: Dynamic software maintenance effort estimation modeling using neural network, rule engine and multi-regression approach. In: International Conference on Computational Science and Its Applications, vol. 15, pp. 157–169 (2012)
24. García-Floriano, A., López-Martín, C., Yáñez-Márquez, C., Abran, A.: Support vector regression for predicting software enhancement effort. Inf. Softw. Technol. **97**, 99–109 (2018)
25. Song, T.-H., Yoon, K.-A., Bae, D.-H.: An approach to probabilistic effort estimation for military avionics software maintenance by considering structural characteristics. In: 14th Asia-Pacific. Software Engineering Conference (APSEC'07), pp. 406–413 (2007)
26. Leung, H.K.N., Emilia, V.: Estimating maintenance effort by analogy. Empir. Softw. Eng. **7**, 157–175 (2002)
27. Cerón-Figueroa, S., López-Martín, C., Yáñez-Márquez, C.: Stochastic gradient boosting for predicting the maintenance effort of software-intensive systems. IET Software, pp. 99–109 (2019)
28. Bennett, K.H., Rajlich, V.T.: Software maintenance and evolution: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering, pp. 73–87 (2000)

29. González-Ladrón-de-Guevara, F., Fernández-Diego, M., Lokan, C.: The usage of ISBSG data fields in software effort estimation: a systematic mapping study. J. Syst. Softw. **113**, 188–215 (2016)
30. Sampath Kumar, P., Venkatesan, R.: Improving accuracy of software estimation using stacking ensemble method. In: Patnaik, S., Yang, X.S., Sethi, I. (eds.) Advances in Machine Learning and Computational Intelligence. Algorithms for Intelligent Systems. Springer, Singapore (2021)
31. Idri, A., Hosni, M., Abran, A.: Systematic literature review of ensemble effort estimation. J. Syst. Softw. **118**, 151–175 (2016)
32. Alsolai, H., Roper, M.: A systematic literature review of machine learning techniques for software maintainability prediction. Inf. Softw. Technol. **119**, 106–214 (2020)
33. Elish, M.O., Aljamaan, H., Ahmad, I.: Three empirical studies on predicting software maintainability using ensemble methods. Soft Comput. **19**, 2511–2524 (2015)
34. Priya Varshini, A.G., Varadarajan, V., et al.: Estimating software development efforts using a random forest-based stacked ensemble approach. Electronics **10**(10), 1195 (2021)
35. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. Inform. Fusion **6**(1), 5–20 (2005)
36. Chandra, A., Yao, X.: DIVACE: Diverse and accurate ensemble learning algorithm. In: Proceedings of 5th International Conference on Intelligent Data Engineering and Automated Learning (LNCS 3177). Springer, Berlin, pp. 619–625 (2004)
37. García-Pedrajas, N., Hervás-Martínez, C., Ortiz-Boyer, D.: Cooperative coevolution of artificial neural network ensembles for pattern classification. IEEE Trans. Evol. Comput. **9**(3), 271–302 (2005)
38. Da Silva, P.M., Lima, M.N.C.A., Soares, W. L., Silva, I.R.R., De Fagundes, R.A., De Souza, F.F.: Ensemble regression models applied to dropout in higher education. In: Proceedings of 2019 Brazilian Conference on Intelligent Systems (BRACIS 2019), pp. 120–125 (2019)
39. Wolpert, D.: Stacked generalization. Neural Netw. **5**, 241–259 (1992)
40. Kraipeerapun, P., Amornsamankul, S.: Using stacked generalization and complementary neural networks to predict Parkinson's disease. In: Proceedings of International Conference on Natural Computing, January 2016, pp. 1290–1294 (2016)
41. Blessie, E.C., Karthikeyan, E.: Sigmis: a feature selection algorithm using correlation based method. J. Algorithms Comput. Technol. **6**, 385–394 (2012)
42. Idri, A., Cherradi, S.: Improving effort estimation of fuzzy analogy using feature subset selection. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), vol. 113, pp.1–8 (2016)
43. Hall, M.A.: Correlation-based feature selection for machine learning. Citeseer 113, 1–8 (1999)
44. Hosni, M., Idri, A., Abran, A.: Investigating heterogeneous ensembles with filter feature selection for software effort estimation. In: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, vol. 113, pp. 207–220 (2017)
45. Oliveira, L.S., Morita, M., Sabourin, R.: Feature selection for ensembles using the multi-objective optimization approach. In: Jin, Y. (ed.) Multi-Objective Machine Learning. Studies in Computational Intelligence, vol 16. Springer, Berlin (2006)
46. Deng, J.D., Purvis, M., Purvis, M.: Software effort estimation: harmonizing algorithms and domain knowledge in an integrated data mining approach. Int. J. Intell. Inf. Technol. (IJIIT) **7**, 41–53 (2011)
47. Idri, A., Azzahra Amazal, F., Abran, A.: Analogy-based software development effort estimation: a systematic mapping and review. Inf. Softw. Technol. **58**, 206–230 (2015)
48. Bala, A., Abran, A.: Use of the multiple imputation strategy to deal with missing data in the ISBSG repository. J. Inf. Technol. Softw. Eng. **6**, 171 (2016)
49. BKarthiga, R., Keerthiga, B., Preethi, S.R.: Analysis on machine learning techniques. i-Manager J. Comput. Sci. **7**, 171 (2019)
50. Shepperd, M., MacDonell, S.: Evaluating prediction systems in software project estimation. Inf. Softw. Technol. **54**, 820–827 (2012)
51. Yadav, S., Shukla, S.: Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In: 2016 IEEE 6th International Conference on Advanced Computing (IACC), vol. 6, pp. 78–83 (2016)
52. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of Fourteenth International Joint Conference on Artificial Intelligence, IJCAI, Montreal, CA, pp. 1137–1143 (1995)
53. Biesiada, J., Duch, W.: Feature selection for high-dimensional data-a Pearson redundancy based filter. Comput. Recogn. Syst. **2**, 242–249 (2007)
54. Fan, J., Lv, J.: Sure independence screening for ultrahigh dimensional feature space. J. R. Stat. Soc. Ser. B (Stat. Methodol.) **70**, 849–911 (2008)
55. Haoues, M., Sellami, A., Ben-Abdallah, H.: Towards functional change decision support based on COSMIC FSM method. Inf. Softw. Technol. **110**, 78–91 (2019)
56. Bhardwaj, M., Ajay, R.: Estimation of testing and rework efforts for software development projects. Asian J. Comput. Sci. Inf. Technol. **5**, 33–37 (2015)
57. Shull, F., Singer, J., Sjøberg, D.I.K.: Guide to Advanced Empirical Software Engineering. Springer, London (2007)

Zaineb Sakhrawi received her Master Thesis degree in Computer Science from Higher Institute of Computer Science and Multimedia, Sfax University, Tunisia in 2018. She is currently a PhD student at Sfax University, Tunisia and a member of Multimedia, InfoRmation systems and Advanced Computing Laboratory (MIRACL). Her research interests include software engineering, ontology, cloud computing and machine Learning techniques.

Asma Sellami is teaching at the University of Sfax in Tunisia. Her current research interest includes broadly measurement in Software Engineering, software quality and software project management. She is also working on ISO standards for measuring the functional size of software, and has been involved in developing case study of ISO 19761 (COSMIC FSM Method). She published more than 40 referred conferences, journals, and technical reports. She is currently member of COSMIC Advisory council in Tunisia.

Nadia Bouassida received a Ph.D. in Computer and Information Science from the University of Science of Tunis, Tunisia. Currently, she is Associate Professor at the Department of Computer Science of the Higher Institute of Computer Science and Multimedia at the University of Sfax, Tunisia. She is a member of the Multimedia, Information systems and Advanced Computing Laboratory, University of Sfax. Her research interests include reuse techniques, such as design patterns, Frameworks and Software Product Lines.