Predicting Software Maintainability using Ensemble Techniques and Stacked Generalization

Sara Elmidaoui[1], Laila Cheikhi[1], Ali Idri[1] and Alain Abran[2]

[1] SPM Team, ENSIAS, Mohammed V University in Rabat, Morocco
[2] Department of Software Engineering & Information Technology, ETS, Montréal, Canada
sara.elmidaoui@um5s.net.ma, laila.cheikhi@um5.ac.ma, ali.idri@um5.ac.ma,
alain.abran@etsmtl.ca

Abstract. The prediction of software maintainability has emerged as an important research topic to address industry expectations for reducing costs, in particular maintenance costs. In the last decades, many studies have used single techniques to predict software maintainability but there is no agreement as to which technique can achieve the best prediction. Ensemble techniques, which combine two or more techniques, have been investigated in recent years. This study investigates ensemble techniques (homogeneous as well as heterogeneous) for predicting maintainability in terms of line code changes. To this end, well-known homogeneous ensembles such as Bagging, Boosting, Extra Trees, Gradient Boosting, and Random Forest are investigated first. Then the stacked generalization method is used to construct heterogeneous ensembles by combining the most accurate ones per dataset. The empirical results suggest that Gradient Boosting and Extra Trees are the best ensembles for all datasets, since they ranked first and second, respectively. Moreover, the findings of the evaluation of heterogeneous ensembles constructed using stacked generalization showed that they gave better prediction accuracy compared to all homogeneous ensembles. [1]

Keywords: Software Maintainability Prediction, Machine Learning, Ensemble techniques, Stacked Generalization, Stacking, Homogeneous, Heterogeneous.

1 Introduction

Maintenance of a software product is recognized as a very time-consuming activity, and several attempts have been made to reduce its high cost by improving maintainability [1], which is defined as "the degree of effectiveness and efficiency with which a product or system can be modified by the intended maintainers" [2]. Several attempts have been made to predict software product maintainability (SPM) through empirical studies. However, predicting maintainability remains an open research area since the maintenance behaviors of software systems are complex and difficult to predict [3]. In fact, many systematic literature reviews (SLRs) have been conducted on software product maintainability prediction (SPMP) to provide an up-to-date review of this topic,

[1] Copyright ©2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-2725/paper9.pdf

such as [4], [5], [6], [7], [8]. The results show that most studies have investigated single techniques to identify the most accurate ones, while ensembles have received less attention; more studies are therefore required in order to identify the best ones. Furthermore, researchers have been unable to identify the best guidelines for developing an accurate technique; indeed, all SPMP techniques are prone to error as they depend to some degree on the empirical context, and no single technique can give a correct result in all circumstances.

To tackle this issue, researchers have begun to investigate the use of many single techniques together, known as Ensemble Techniques (ETs). As stated in [9]–[11], these can take one of two forms: Heterogeneous (HT), which combines at least two different techniques, and Homogeneous (HM), in which the single techniques (two at least) are of the same type. Two types of HM technique are proposed: one that combines the same base techniques with at least two configurations, and one that combines one meta-model with one base technique. Since ETs have proved their usefulness in improving accuracy in many areas such as in software effort estimation [11], software fault prediction [12], [13] and face recognition [14], this study investigates whether they can improve the accuracy of software maintainability prediction.

In this context, a set of empirical studies that investigated ETs for SPMP were selected in [7] and summarized in Table 1, with the corresponding techniques used (ML techniques or base learners), the type of ensembles used (HM or HT), the rules used to combine the ML techniques, the datasets, and the accuracy criteria. As seen in the table, HM ensembles were used by means of Bagging [10], [15], [16], RF [17], [18], [16] and Boosting (i.e., AdaBoost, LogitBoost) [19]. HT ensembles were used with different variants, such as combining multilayer perceptron (MLP), radial basis function (RBF), support vector machine (SVM), and M5 for inducing trees of regression models (M5P) using the best in training (BTE) [9], average (AVG) and weighted averages (WT) [10] rules. The ensembles were also evaluated by means of different datasets (e.g., User Interface Management System (UIMS) and QUality Evaluation System (QUES)) and a variety of accuracy criteria such as Mean Magnitude of Relative Error (MMRE), Standard deviation of Magnitude of Relative Error (Std.MRE), Percentage of Relative Error Deviation (Pred), True Positive Rate (TPR), and False Positive Rate (FPR). Moreover, from the above studies, it was found that ensembles provide greater or at least similar prediction accuracy compared to single techniques [9], and one study suggested investigating other combination rules for constructing HT ensembles [10]. None of the studies reported evidence on the best combination rules, or used Stacked Generalization (SG) [20], or investigated HM ensembles such as Extra Trees (ExTrees) and Gradient Boosting (GradBoost) for predicting software maintainability. This study is the first work that applies those variants of HM ensembles and combines the best of them using SG to construct HT ensembles.

The objective of this study is twofold: (1) investigate the use of five variants of HM ensembles: Bagging, Adaptive Boosting (AdaBoost), GradBoost, RF, and ExTrees with their default base techniques, and (2) investigate the use of HT ensembles that are constructed from the most accurate HM ensembles, per dataset, using the SG method. This objective is achieved by addressing the following three research questions (RQs):

- **RQ1:** Among the five HM ensembles, which one generates the best SPMP accuracy?
- **RQ2:** Do the HT ensembles constructed with SG improve SPMP accuracy?
- **RQ3:** Which ensemble gives the best performance regardless of the dataset used?

The rest of the paper is structured as follows: Section 2 gives an overview of the five HM ensembles used. Section 3 presents the method used to construct the HT ensembles. Section 4 presents the empirical design of this study. Section 5 presents and discusses the empirical results obtained for HM as well as HT. Section 6 presents threats to the validity of this study. Section 7 contains the conclusion and suggestions for future work.

Table 1. Related studies on ETs for SPMP

ID	Techniques	ET type	Combination rules	Datasets	Accuracy criteria
[17]	AODE, SVM with linear kernel, Naïve Bayes (NB), Bayesian Networks (BN), RF, K Nearest Neighbor, C4.5, OneR, RBF	HM	AVG	Medical imagining system	Weighted Average Precision (WAP), Weighted Average Recall (WARec)
[15]	DT, Back Propagation Neural Network (BPNN), SVM, Bagging	HM	AVG	Open source system	Precision, Recall, F1 score, TPR, FPR, Area Under Curve (AUC)
[18]	NB, BN, Logistic Regression (LgR), MLP, RF	HM	AVG	Lucence, Hotdraw,JEdit, JTreeview	Recall, Precision, Receiver Operating Characteristic
[16]	LgR, RF, Bagging, AdaBoost, NB, J48, MLP, LogitBoost, BN, Nearest-Neighbor-like that uses non-nested generalized exemplars	HM	AVG	Art-of-Illusion, Sweet-Home-3D	Sensitivity, Specificity, AUC, Cut-off point
[19]	TreeNet	HM	AVG	QUES, UIMS	MMRE, Pred(25), Pred(30)
[10]	SVM, MLP, LgR, Genetic Programming, K-means	HT	BTE, majority voting, decision tree forest	VSSPLUGIN, PeerSim	Correct classification rate (CCR), AUC
	Bagging, Boosting (AdaBoost)	HM	AVG	VSSPLUGIN, PeerSim	CCR, AUC
	MLP, RBF, SVM, M5P	HT	AVG, BTE, WT	QUES, UIMS	MMRE, Pred(30), Std.MRE
[9]	MLP, RBF, SVM, M5P	HT	BTE	QUES, UIMS	MMRE, Std. MRE, Pred(30)

2 Homogeneous Ensemble Techniques

This section provides an overview of the five HM ensembles investigated in this empirical study, namely Bagging, AdaBoost, RF, ExTrees, and GradBoost.

2.1 Bagging

Bagging (also known as Bootstrap aggregation) is a well-known HM ensemble proposed by Breiman [21]. Bagging is based on the bootstrap method for creating a distribution of datasets with replacement from an original dataset [22]. Bagging trains each regression model (i.e., base learner) with the different training sets generated by sampling with replacement from the training data, then it averages the predictions of each constructed regression model to perform the final prediction [23]. To build a model based on bagging, the following steps are performed: "(1) Split the dataset into training set and test set, (2) get a bootstrap sample from the training data and train a predictor using the sample. Repeat the steps a random number of times. The models from the samples are combined (i.e., aggregated) by averaging the output for regression or voting for classification" [24]. Through this process, bagging turns weak learners into strong ones [25], reduces variance, helps to avoid overfitting [24], and improves regression models in terms of stability and accuracy [24]. Moreover, bagging has presented good results whenever the learning technique is unstable [21].

2.2 Adaptive Boosting (AdaBoost)

AdaBoost is one of the first practical boosting methods introduced by Freund and Schapire [26]. In AdaBoost, weak learners are combined and "boosted" to improve ensemble accuracy and produce a strong technique [27]. AdaBoost works as follows: "(…) creates a sample from the training data using the sampling weight vector. The base learner uses this sample to create a hypothesis that links the input to the output data. This hypothesis is applied to all the data to create predictions. The absolute relative error is calculated for each prediction and compared against a threshold, which is used to classify the predicted values as correct or incorrect. The sampling weight of incorrectly predicted samples is increased for the next iteration" [27]. AdaBoost is also "adaptive in that it adapts to the error rates of the individual weak hypotheses" [28] and "tends not to over-fit; on many problems, even after hundreds of rounds of boosting, the generalization error continues to drop, or at least does not increase" [26].

2.3 Random Forest (RF)

RF is an ensemble learner proposed by Breiman in 2001 [29] which "adds an additional layer of randomness to bagging. In addition to constructing each tree using a different bootstrap sample of the data, RFs change how the regression trees are constructed. In standard trees, each node is split using the best split among all variables. In a random forest, each node is split using the best among a subset of predictors randomly chosen at that node" [30]. In other words, RF uses both bagging—a successful approach for combining unstable learners [31] —and random variable selection for tree building

[17]. This strategy performs very well compared to many other ML techniques, including SVR and ANN [30]. RF is robust against overfitting [29], it can achieve both low bias and low variance [32], “it is very user-friendly in the sense that it has only two parameters (the number of variables in the random subset at each node and the number of trees in the forest), and is usually not very sensitive to their values” [30].

2.4 Extra Trees (ExTrees)

ExTrees (short for Extremely Randomized Trees) is a new tree-based ensemble method for supervised classification and regression problems, proposed by Geurts et al. [33]. ExTrees “builds an ensemble of an unpruned decision or regression tree according to the classical top-down procedure. Its two main differences with other tree-based ensemble methods are that it splits nodes by choosing cut-points fully at random and that it uses the whole learning sample to grow the trees” [33]. The technique for splitting “consists of choosing randomly a number of inputs at each node and the minimum sample size for splitting a node with the full original data to generate a number of trees that construct the ensembles. Then, the predictions of the trees are aggregated to yield the final prediction, by arithmetic average in the regression problems” [33]. The advantage of ExTrees is its capability “to reduce variance and minimize bias due to the use of the full original sample rather than bootstrap replicas” [33].

2.5 Gradient Boosting (GradBoost)

GradBoost (or Gradient boosted tree) is an ensemble learner proposed by Friedman [34]. GradBoost constructs “additive regression models by sequentially fitting a simple parameterized function (base learner) to current "pseudo" residuals by least-squares at each iteration. The pseudo residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data, evaluated at the current step. It uses a gradient descent algorithm for the shortcomings of weak learners instead of using a re-weighting mechanism. This algorithm is used to minimize the loss function (also called the error function) by moving in the opposite direction of the gradient and finds (ou “finding”? vérifier le texte source) a local minimum” [35]. The main advantage is that “both the approximation accuracy and execution speed of Gradient Boosting can be substantially improved by incorporating randomization into the procedure” [35].

3 Heterogeneous Ensemble Techniques

As mentioned earlier, none of the SPMP models used the SG method to construct HT ensembles. SG is a different way of combining multiple models that introduces the concept of a meta-learner proposed in 1992 by Wolpert [36]. The learning procedure is illustrated in the following algorithm and consists of the following steps “(1) learn first-level classifiers based on the original training dataset, (2) construct a new dataset based on the output of base classifiers. Here, the output predicted labels of the first-level classifiers are regarded as new features, and the original class labels are kept as the labels

in the new dataset, (3) learn a second-level classifier based on the newly constructed dataset. Any learning method could be applied to learn the second-level classifier" [37].

Algorithm Stacking [37]

Input: Training data $D = \{x_i, y_i\}_{i=1}^{m}\ \{x_i \in R^n, y_i \in Y\}$
Output: An ensemble classifier H
Step 1: Learn first-level classifiers
for $t \leftarrow$ 1 to T do
Learn a base classifier h_t based on D
end for
Step 2: Construct new datasets from D
for $i \leftarrow$ 1 to m do
Construct a new dataset that contains $\{x'_i, y_i\}, where\ x'_i = \{h_1(x_i), h_2(x_i), \dots, h_T(x_i)\}$
end for
Step 3: Learn a second-level classifier
Learn a new classifier h' based on the newly constructed dataset
return $H(x) = h'(h_1(x), h_2(x), \dots, h_T(x))$

In this study, we investigate the use of HT ensembles in SPMP by combining the most accurate HM ensembles using SG. The process adopted for SG consists of taking the predictions of each HM ensembles that appeared in the best cluster generated by the SK test per dataset in the first level and using them as input variables in the second level learning. In the second level, an algorithm (i.e., in our case, we used linear regression since it is a simple technique for modeling the relationship between dependent and independent variables [38], [39]) and is trained to optimally combine the models' predictions to form a new set of predictions [20]. Note that the second level modeling is not restricted to any simple technique. The relationship between the predictions can be more complex, opening the door to other ML techniques [20]. The main advantages of SG are that it correctly classifies the target, thereby correcting any mistakes made by the models constructed in the first level [20], and it offers robust performance [37]. Moreover, "it is flexible and does not require particular expertise in deployment, due to its robust performance" [40].

4 Empirical Design

This section presents the empirical design used. Note that the same strategy has been adopted previously in [11], [41] for software development effort estimation, but using different datasets and ML techniques.

4.1 Datasets

Five publicly available datasets were used in this study, including two popular public datasets (UIMS and QUES [42], provided by researchers from the software engineering

community) and three open-source software (Log4j[2], Xalan[3], and JEdit[4], provided in the PROMISE repository).

The UIMS and QUES datasets are object-oriented (OO) software maintainability datasets from Li and Henry (L&H) [42], developed using Ada. The UIMS contains class-level metrics data collected from 39 classes of a user interface management system, whereas the QUES contains the same metrics collected from 71 classes of a quality evaluation system. Both datasets contain 10 independent variables (five OO metrics from Chidamber and Kemerer(C&K) [43], four OO metrics from L&H [42], and the traditional lines of code). The maintainability is expressed in terms of maintenance effort measured by the number of lines changed per class; a line change could be an addition or a deletion, and a change of the content of a line is counted as a deletion and an addition [42].

The JEdit, Log4j, and Xalan datasets are based on open-source software projects written in Java and of different sizes: 161, 153, and 756, respectively. These three datasets each contain 20 features including the six C&K metrics and other OO metrics. The dependent variable was expressed in terms of bugs in the existing datasets. Since the focus of this study is to evaluate maintainability in terms of code line changes, we calculated the change between the same classes from two versions of the software projects JEdit, Log4j, and Xalan using the Beyond compare tool[5]. It should be noted that the same methodology was used in [18], [44]–[46].

4.2 Accuracy criteria

Accuracy criteria are essential in SPMP techniques. This study focused on the prediction data mining task where the most commonly used techniques (see Table 1) were MMRE and Pred(0.25), which are based on the Magnitude of Relative Error (MRE) (1), (2), and (3).

$$\text{MRE} = \frac{|y_i - \hat{y}_i|}{y_i} \quad (1)$$

$$\text{MMRE} = \frac{1}{n}\sum_{i=0}^{n} MRE_i \quad (2)$$

$$\text{Pred}(0.25) = \frac{100}{N}\sum_{i=1}^{N} \begin{cases} 1 \; if \; MRE_i \leq 0.25 \\ 0 \; otherwise \end{cases} \quad (3)$$

Where: y_i and $\hat{y}_i$ are the actual and predicted change for the i^{th} project. N is the number of data points (examples).

Since the MRE has been criticized for being biased and unbalanced [47], [48], [49], in this study we applied some unbiased accuracy criteria that do not present asymmetric distribution [50], namely: Mean Balanced Relative Error (MBRE) [48], [51], Mean Inverted Balanced Relative Error (MIBRE) [48], [51], and Mean Absolute Error (MAE), which is based on Absolute Error (AE). These accuracy criteria are described in (4), (5), (7), and (6), respectively.

---

[2] Log4J Homepage.https://logging.apache.org/log4j/2.x/

[3] Xalan Homepage. http://xalan.apache.org/index.html

[4] JEdit Homepage.http://www.jedit.org/

[5] https://www.scootersoftware.com/

$$\text{MBRE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i-\hat{y}}{\min(y_i,} \quad (4)$$

$$\text{MIBRE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i-\hat{y}_i|}{\max(y_i,\hat{y}_i)} \quad (5)$$

$$\text{AE} = |y_i-\hat{y}_i| \quad (6)$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} AE_i \quad (7)$$

We also used the logarithmic standard deviation (LSD) to evaluate the accuracy of prediction techniques [48] (8) and Standardized Accuracy (SA) [52] (9) to solve the problem inherent in MRE-based accuracy criteria. To verify if predictions given by a technique are generated by chance and if there is an improvement over random guessing, we used the effect size (Δ) defined by (10). The ratio in SA represents "how much better p_i is than random guessing. Clearly, a value close to zero is discouraging and a negative value would be worrisome" [52]. It is recommended to use the 5% quantile of random guessing: "the interpretation of the 5% quantile for $p_0$is similar to the use of α for conventional statistical inference, that is, any accuracy value that is better than this threshold has a less than one in twenty chance of being a random occurrence" [52].

$$\text{LSD} = \sqrt{\frac{\sum_{i=1}^{n}(\lambda_i+\frac{S^2}{2})^2}{n-1}} \quad (8)$$

$$SA = (1-\frac{MAE_{p_i}}{MAE_{p_0}}) \quad (9)$$

$$\Delta = \frac{MAE_{p_i} - \overline{MAE_{p_0}}}{S_{p_0}} \quad (10)$$

Where: $\lambda_i = \ln(y_i) - \ln(\hat{y}_i)$. S^2 is an estimator of the variance of the residual λ_i. $S_{p_0}$is the sample standard deviation of the random guessing strategy. MAE_{p_i} is defined as the MAE of the prediction method p_i. MAE_{p_0} is the mean of a large number—typically 1000—of runs of random guessing. Note that the above accuracy criteria were used all together previously in [11], [41] for software development effort estimation but have never been used in previously published SPMP studies [6] (see Table 1).

4.3 Scott–Knott significance statistical test

Scott-Knott (SK) [53] is a hierarchical clustering algorithm used as an exploratory data analysis tool for analysis of variance (ANOVA). It was developed by Scott and Knott as a way to find distinct identical comparison groups of treatment means considering type I error [53]. In this study, the SK test was used to statistically investigate whether there is a significant difference between the ensembles based on MAE, i.e., to cluster the techniques that have identical predictions potential. Note that several studies in the software engineering field have used the SK test to rank investigated techniques [11], [41], [54].

5 Results and Discussion

This section evaluates and discusses the prediction performance of HM and HT ensembles used in this study on the five datasets. Fig. 1 presents the process followed to perform this empirical study. Two main steps are identified:

- Step 1: Identify the most accurate HM ensembles (Bagging, AdaBoost, GradBoost, RF, and ExTrees). This step includes:
  - Use of the five historical datasets, namely UIMS, QUES, JEdit, Log4j, and Xalan.
  - Application of the LOOCV method, which consists in using one instance from a dataset as the test set and the remaining N-1 instances as the training set N times, where N is the number of instances in a specific dataset.
  - Evaluate the performance of the HM ensembles in terms of SA and effect size accuracy criteria, and eliminate those that have an SA value less than 5% quartile.
  - Use of the SK test to statistically compare the HM ensembles.
  - Use of the Borda count voting method to rank the HM ensembles that appeared in the best cluster using MAE, Pred(25), MBRE, MIBRE, and LSD.
- Step 2: The most accurate HM ensembles (from Step1) are combined using the SG to construct HT ensembles per dataset.

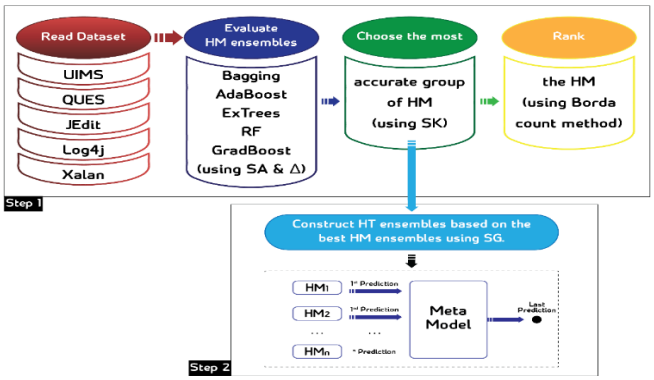

Fig. 1. Process of the empirical study

To perform this empirical study, different tools were used. A software prototype for each ML technique and each HM ensemble based on Scikit-learn API was developed using Python. The statistical tests (SK and Kolmogorov-Smirnov, and the Box-Cox transformation) were performed through R software.

5.1 Results and discussion – HM

Table 2 reports the performance of HM ensembles based on SA and effect size accuracy criteria over the five datasets. The second row of Table 2 shows the 5% quantile SA of random guessing for each dataset ($SA_{5\%}$). It can be seen that all HM ensembles generated better predictions than random guessing since their SA values were higher than $SA_{5\%}$. Moreover, all HM ensembles across all datasets showed a large improvement over guessing since their effect size test results are larger than 0.8. The following summarizes the results for each dataset and then for all datasets: For the UIMS dataset, GradBoost achieved good accuracy compared to other ensembles with an SA value of 0.883, followed by ExTrees with 0.863. The Bagging and AdaBoost also gave good results compared to RF, which ranked last. For the QUES dataset, GradBoost provided

the best prediction results with an SA value of 0.957, followed by ExTrees (SA= 0.935). Other ensembles: Bagging, RF, and AdaBoost were more or less equal in terms of SA values (≈ 0.91). For the JEdit dataset, GradBoost ranked first with an SA value of 0.961, followed by ExTrees (0.959). Bagging and RF gave the same results, 0.943 and 0.942, respectively. The AdaBoost ensemble ranked last with an SA value of 0.890. For the Xalan dataset, GradBoost achieved the best results (SA= 0.981) compared to the other ensembles. Bagging and RF were more or less equal with an SA value of 0.972 and 0.971, respectively, followed by ExTrees (SA= 0.963), while AdaBoost scored lowest among the other ensembles with an SA value of 0.831. For the Log4j dataset, GradBoost came first with an SA value of 0.932, followed by ExTrees (SA= 0.927), then RF and Bagging, which were more or less equal with a SA values of 0.904 and 0.903, respectively. AdaBoost ranked last with an SA value of 0.865.

The results for all datasets are as follows: All HM ensembles across all datasets achieved better predictions than random guessing. GradBoost ranked first in all datasets. Bagging and RF achieved generally the same results in all datasets, while in the UIMS dataset, RF ranked last. AdaBoost ensemble ranked last in four datasets and 4th in one dataset.

Table 2. SA and effect size of the five HM ensembles

Dataset	**UIMS**		**QUES**		**JEdit**		**Xalan**		**Log4j**	
$SA_{5\%}$	0.0200		0.0189		0.0184		0.0003		0.0188	
HM Technique	SA	|Δ|	SA	|Δ|	SA	|Δ|	SA	|Δ|	SA	|Δ|
Bagging	0.80	77	0.91	140	0.94	207	0.97	159	0.90	176
ExTrees	0.86	160	0.93	88	0.95	165	0.96	169	0.92	262
RF	0.78	46	0.91	174	0.94	170	0.97	209	0.90	100
GradBoost	0.88	82	0.95	136	0.96	240	0.98	218	0.93	95
AdaBoost	0.80	80	0.91	149	0.89	115	0.83	203	0.86	117

Furthermore, Fig. 2 shows the plot of SK testing of constructed ensembles per dataset. The *x*-axis represents the selected ensembles and the *y*-axis represents the transformed AEs. Every vertical line shows the variation of transformed AEs for each technique, and the small circle represents the mean of transformed AEs. The farther to the right a technique is positioned, the better its performance, and the brown box indicates the best cluster. Note that we transformed AE values using the Box-Cox method [55] since the data (i.e., AE) do not follow a normal distribution in all cases. As can be seen from Fig. 2, two groups were generated in all datasets, except for the Xalan, in which four groups were generated by the SK test. Note that the green box in the right-hand group for each dataset indicates the best group. ExTrees, GradBoost, Bagging, and RF appear in the best cluster in three datasets: JEdit, Log4j, and UIMS, This means that these ensembles are indifferent. For QUES and Xalan datasets, only GradBoost and ExTrees appear in the best cluster. The AdaBoost technique appears in the last cluster in all datasets.

Table 3 presents the ranks of the HM ensembles that appear in the best cluster calculated by the Borda count method over five accuracy criteria (MAE, Pred(25), MBRE, MIBRE, and LSD). GradBoost ranked first and ExTrees second in all datasets. This

leads to the conclusion that some ensembles are solid and scalable to datasets of different software systems and can be used across software systems for predicting maintainability [3]. As for Bagging and RF, they ranked third or fourth on three datasets, leading us to conclude that these two ensembles generate the same results in predicting software maintainability.

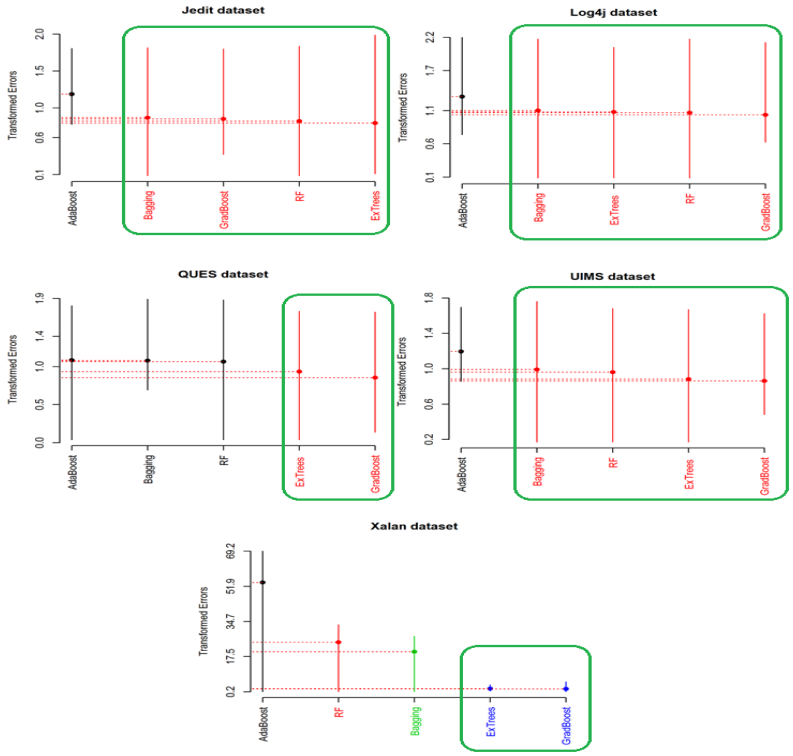

Fig. 2. Plot of SK test of HM ensembles in each dataset

Table 3. Rank of HM ensembles of the best cluster using Borda count

Rank	UIMS	QUES	Log4j	JEdit	Xalan
1	GradBoost	GradBoost	GradBoost	GradBoost	GradBoost
2	ExTrees	ExTrees	ExTrees	ExTrees	ExTrees
3	Bagging		RF	Bagging	
4	RF		Bagging	RF	

5.2 Results and discussion – HT

The HM ensembles generated by SK in the best cluster (Table 3) per dataset were combined using the SG. Table 4 presents the constructed HT ensembles. Bagging, GradBoost, ExTrees, and RF were combined for the UIMS, JEdit, and Log4j datasets, while GradBoost and ExTrees were combined for the Xalan and QUES datasets.

The results of SG in Table 4 indicate that the HT ensembles give the best results when Bagging, GradBoost, ExTrees, and RF are used for UIMS, JEdit, and Log4j datasets. The same results were obtained for QUES and Xalan by combining GradBoost and ExTrees. We can conclude that SG with the proposed combination for each dataset achieves good results with an SA value of 1.00. In fact, such a result was expected, since all the HM ensembles used to construct the HT ones belong to the best cluster of SK (i.e., Bagging, GradBoost, ExTrees, and RF), with SA values ranging from 0.783 to 0.981. The use of SG further improved the results to this good level of accuracy. However, the result obtained in this step is too good to be true, and replicated studies are needed to confirm this finding.

Table 4. Results of HT ensembles over datasets

Dataset	Techniques combined	SA	|Δ|
UIMS	Bagging+GradBoost+ ExTrees+RF	1.00	57
QUES	GradBoost+ ExTrees	1.00	191
JEdit	Bagging+GradBoost+ ExTrees+RF	1.00	218
Log4j	Bagging+GradBoost+ ExTrees+RF	1.00	192
Xalan	GradBoost+ ExTrees	1.00	178

Table 5 compares the constructed HT and HM ensembles. The proposed ETs were ranked using the Borda count method based on five accuracy criteria: Pred(25), MBRE, MIBRE, MAE, and LSD. As shown in Table 5, the constructed HT ensembles using SG ranked first in all datasets compared to HM ensembles, and thus represent a good solution based on this study finding for predicting software maintainability in terms of line code changes.

Table 5. Rank of HM and HT using Borda count

Rank	UIMS	QUES	JEdit	Log4j	Xalan
1	Bagging+GradBoost+ ExTrees+RF	GradBoost+ ExTrees	Bagging+GradBoost+ ExTrees+RF	Bagging+GradBoost+ ExTrees+RF	GradBoost+ ExTrees
2	GradBoost	GradBoost	GradBoost	GradBoost	GradBoost
3	ExTrees	ExTrees	ExTrees	ExTrees	ExTrees
4	Bagging	RF	RF	Bagging	RF
5	RF	Bagging	Bagging	RF	Bagging
6	AdaBoost	AdaBoost	AdaBoost	AdaBoost	AdaBoost

6 Threats to Validity

This section presents the threats to the three aspects of validity of this empirical study (e.g., internal, external, and construct validity) according to [11], [56], [57].

Internal validity is related to the use of a biased validation method to assess the performance of prediction techniques. Many SPMP studies use the whole data or hold-

out method, so that the evaluation focuses only on a unique subset of data, which can result in a biased and unreliable assessment of performance. To overcome this limitation, LOOCV is used as a validation method. It can generate the same results when the empirical study is replicated using a particular dataset, which is not the case for cross-validation [58].

External validity is related to the degree of generalizability of the results. The proposed ensembles were evaluated over five datasets from different sources and different application domains. The datasets used in this study are varied in terms of size, number of features, and application domains, and this makes them adequate for evaluating our techniques. Besides, this study deals only with numerical data; hence, the findings of this study may differ from other studies that use other types of data.

Construct validity is related to the reliability and credibility of the accuracy criteria used to assess performance. This study used SA that is an unbiased accuracy criterion and less vulnerable to asymmetry assumption, proposed by Shepperd and MacDonell [52]. Moreover, five accuracy criteria were used: MAE, Pred(25), MBRE, MIBRE, and LSD, aggregated by means of the Borda count coting method. The most widely used accuracy criterion in SPMP area, MMRE, was not used in this study because many researchers have criticized it for favoring underestimation [48], [52], [59].

7 Conclusion and Future Work

In this empirical study, the performance of five HM ensembles (Bagging, AdaBoost, GradBoost, RF, and ExTrees) for predicting software maintainability was assessed over five datasets using the LOOCV method based on SA and effect size. The SK test and the Borda count were used to statistically compare and rank the HM ensembles based on five accuracy criteria: MAE, Pred(25), MBRE, MIBRE, and LSD. In addition, we took the HM ensembles that appeared in the best cluster in each dataset and combined them using SG to construct HT ensembles. The findings with respect to the research questions are the following:

- **RQ1: Among the five HM ensembles, which one generates the best accuracy of software product maintainability prediction?** The empirical evaluations showed that the GradBoost technique achieved the best results compared to other HM ensembles in all datasets.
- **RQ2: Do the HT ensembles constructed with SG improve SPMP accuracy?** The ensembles combined using SG outperformed the HM ensembles in all datasets and generally gave good results in terms of SA values (SA=1.00 for all combinations).
- **RQ3: Which ensemble gives the best performance regardless of the dataset used?** For all datasets, GradBoost ranked first and ExTrees second. This shows that these two techniques perform better in different datasets.

Ongoing work in SPMP involves using other HM ensembles such as Light Gradient Boosting and eXtreme Gradient Boosting, as well as investigating the HT ensemble by combining base learners with other combination rules such as average, weighted average, best in training, etc. Moreover, replicating this empirical study with other datasets

may help to confirm or refute our conclusions. Generalized results regarding the prediction of software product maintainability are therefore still not available to meet software industry needs and the expectations for software maintainability prediction.

References

1. A. Abran and H. Nguyenkim, “Measurement of the maintenance process from a demand-based perspective,” J. Softw. Maint. Res. Pract., pp. 63–90, 5(2), (1993).
2. “ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models,” Geneva, Switzerland, (2011).
3. A. Kaur and K. Kaur, “Statistical comparison of modelling methods for software maintainability prediction,” Int. J. Softw. Eng. Knowl. Eng., pp. 743–774, 23(06), (2013).
4. M. Riaz, E. Mendes, and E. Tempero, “A systematic review of software maintainability prediction and metrics,” in 3rd International Symposium on Empirical Software Engineering and Measurement, pp. 367–377, Lake Buena Vista, FL, USA (2009).
5. M. Riaz, “Maintainability prediction of relational database-driven applications: a systematic review,” In: International Conference on Evaluation & Assessment in Software Engineering, IET, Ciudad Real, Spain (2012).
6. S. Elmidaoui, L. Cheikhi, A. Idri, and A. Abran, “Empirical Studies on Software Product Maintainability Prediction: A Systematic Mapping and Review,” e-Informatica Softw. Eng. J., pp. 141–202, 13(1), (2019).
7. S. Elmidaoui, L. Cheikhi, A. Idri, and A. Abran, “Machine Learning Techniques for Software Maintainability Prediction: Accuracy Analysis,” J. Comput. Sci. Technol., (2020).
8. H. Alsolai and M. Roper, “A systematic literature review of machine learning techniques for software maintainability prediction,” Inf. Softw. Technol., pp. 106-214, vol. 119, (2020).
9. H. Aljamaan, M. O. Elish, and I. Ahmad, “An ensemble of computational intelligence models for software maintenance effort prediction,” in International Work-Conference on Artificial Neural Networks, pp. 592–603, (2013).
10. M. O. Elish, H. Aljamaan, and I. Ahmad, “Three empirical studies on predicting software maintainability using ensemble methods,” Soft Comput., pp. 2511–2524, 19(9), (2015).
11. M. Hosni, A. Idri, A. Abran, and A. B. Nassif, “On the Value of Parameter Tuning in Heterogeneous Ensembles Effort Estimation,” Soft Comput., pp. 5977–6010, 22(18), (2018).
12. P. L. Braga, A. L. I. Oliveira, G. H. T. Ribeiro, and S. R. L. Meira, “Bagging Predictors for Estimation of Software Project Effort,” in 2007 International Joint Conference on Neural Networks, pp. 1595–1600, (2007).
13. H. I. Aljamaan and M. O. Elish, “An empirical study of bagging and boosting ensembles for identifying faulty classes in object-oriented software,” in IEEE Symposium on Computational Intelligence and Data Mining, pp. 187–194, (2009).
14. S. Gutta and H. Wechsler, “Face recognition using hybrid classifier systems,” in International Conference on Neural Networks (ICNN’96), pp. 1017–1022, vol.2, (1996).
15. F. Ye, X. Zhu, and Y. Wang, “A new software maintainability evaluation model based on multiple classifiers combination,” in International Conference on Quality, Reliability, Risk, Maintenance, and Safety Engineering (QR2MSE), pp. 1588–1591, (2013).
16. M. Ruchika and J. Ravi, “Prediction and Assessment of Change Prone Classes Using Statistical and Machine Learning Techniques,” J. Inf. Process. Syst., pp. 778–804, 13(4), (2017).

17. Y. Tian, C. Chen, and C. Zhang, “Aode for source code metrics for improved software maintainability,” in International Conference on Semantics, Knowledge and Grid, pp. 330–335, Beijing, China (2008).
18. A. Kaur, K. Kaur, and K. Pathak, “Software maintainability prediction by data mining of software code metrics,” in International Conference on Data Mining and Intelligent Computing (ICDMIC), pp. 1–6, New Delhi, India (2014)..
19. M. O. Elish and K. O. Elish, “Application of treenet in predicting object-oriented software maintainability: A comparative study,” in European Conference on Software Maintenance and Reengineering, pp. 69–78, Kaiserslautern, Germany (2009).
20. C. Sammut and G. I. Webb, Eds., “Stacked Generalization,” in Encyclopedia of Machine Learning, Boston, MA: Springer US, (2010).
21. L. Breiman, “Bagging Predictors,” Mach. Learn., pp. 123–140, 24(2), (1996).
22. B. Efron, D. Rogosa, and R. Tibshirani, “Resampling Methods of Estimation,” Int. Encycl. Soc. Behav. Sci., pp. 492–495, (2015).
23. S. B. Kotsiantis, D. Kanellopoulos, and I. D. Zaharakis, “Bagged Averaging of Regression Models,” in Artificial Intelligence Applications and Innovations, pp. 53–60, (2006).
24. G. P. Kumari, “A Study Of Bagging And Boosting Approaches To Develop Meta-Classifier,” Eng. Sci. Technol. An Int. J., 2(5), (2012).
25. L. L. Minku and X. Yao, “A Principled Evaluation of Ensembles of Learning Machines for Software Effort Estimation,” in International Conference on Predictive Models in Software Engineering, pp. 9:1--9:10, Banff, Alberta, Canada (2011).
26. Y. Freund and R. E. Schapire, “A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting,” J. Comput. Syst. Sci., pp. 119–139, 55(1), (1997).
27. N. Kummer and H. Najjaran, “Adaboost. MRT: Boosting regression for multivariate estimation.,” Artif. Intell. Res., pp. 64–76, 3(4), (2014).
28. Y. Freund, R. Schapire, and N. Abe, “A short introduction to boosting,” Journal-Japanese Soc. Artif. Intell., pp. 771–780, 14(1612), (1999).
29. L. Breiman, “Random Forests,” Mach. Learn., pp. 5–32, 45(1), (2001).
30. A. Liaw, M. Wiener, and others, “Classification and regression by random Forest,” R news, pp. 18–22, 2(3), (2002).
31. C. Breiman, L., Friedman, J., Olshen, R., & Stone, “Classification and regression trees,” Wadsworth Int. Gr., pp. 237–251, 37(15), (1984).
32. R. Díaz-Uriarte and S. Alvarez de Andrés, “Gene selection and classification of microarray data using random forest,” BMC Bioinformatics, 7(1), (2006).
33. P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” Mach. Learn., pp. 3–42, 63(1), (2006).
34. J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” Ann. Stat., pp. 1189–1232, vol. 29, (2000).
35. R. Tugay and S. G. Ögüdücü, “Demand Prediction using Machine Learning Methods and Stacked Generalization,” In: International Conference on Data Science, Technology and Applications - Volume 1: DATA, pp. 216-222, Madrid, Spain (2017).
36. D. H. Wolpert, “Stacked generalization,” Neural Networks, pp. 241–259, 5(2), (1992).
37. C. C. Aggarwal, Data classification: algorithms and applications. CRC press, (2014).
38. C. Catal, “Review: Software Fault Prediction: A Literature Review and Current Trends,” Expert Syst. Appl., pp. 4626–4636, 38(4), (2011).
39. C. Catal and B. Diri, “A Systematic Review of Software Fault Prediction Studies,” Expert Syst. Appl., pp. 7346–7354, 36(4), (2009).
40. L. Jonsson, Machine Learning-Based Bug Handling in Large-Scale Software Development, vol. 1936. Linköping University Electronic Press, (2018).

41. M. Azzeh, A. B. Nassif, and L. L. Minku, “An Empirical Evaluation of Ensemble Adjust Methods for Analogy-based Effort Estimation,” J. Syst. Softw., pp. 36–52, 103(C), (2015).
42. W. Li and S. Henry, “Object-oriented metrics that predict maintainability,” J. Syst. Softw., pp. 111–122, 23(2), (1993).
43. S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,” IEEE Trans. Softw. Eng., pp. 476–493, 20(6), (1994).
44. Kumar Lov and S. Ashish, “A Comparative Study of Different Source Code Metrics and Machine Learning Algorithms for Predicting Change Proneness of Object Oriented Systems,” arXiv Prepr. arXiv1712.07944, (2018).
45. L. Kumar and S. K. Rath, “Hybrid Functional Link Artificial Neural Network Approach for Predicting Maintainability of OO Software,” J. Syst. Softw., pp. 170–190, 121(C), (2016).
46. A. Kaur, K. Kaur, and K. Pathak, “A proposed new model for maintainability index of open source software,” in 3rd International Conference on Reliability, Infocom Technologies and Optimization, pp. 1–6, Noida, India (2014).
47. A. Idri, I. Abnane, and A. Abran, “Evaluating Pred(p) and standardized accuracy criteria in software development effort estimation,” J. Softw. Evol. Process, 30(4), (2018).
48. T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit, “A simulation study of the model evaluation criterion MMRE,” IEEE Trans. Softw. Eng., pp. 985–995, 29 (11), (2003).
49. I. Myrtveit, E. Stensrud, and M. Shepperd, “Reliability and Validity in Comparative Studies of Software Prediction Models,” IEEE Trans. Softw. Eng., pp. 380–391, 31(5), (2005).
50. L. L. Minku and X. Yao, “Ensembles and locality: Insight on improving software effort estimation,” Inf. Softw. Technol., pp. 1512–1528, 55(8), (2013).
51. Y. Miyazaki, A. Takanou, H. Nozaki, N. Nakagawa, and K. Okada, “Method to estimate parameter values in software prediction models,” Inf. Softw. Technol., pp. 239–243, 33(3), (1991).
52. M. Shepperd and S. MacDonell, “Evaluating Prediction Systems in Software Project Estimation,” Inf. Softw. Technol., pp. 820–827, 54(8), (2012).
53. A. J. Scott and M. Knott, “A cluster analysis method for grouping means in the analysis of variance,” Biometrics, pp. 507–512, (1974).
54. N. Mittas and L. Angelis, “Ranking & Clustering Software Cost Estimation Models through a Multiple Comparisons Algorithm,” IEEE Trans. Softw. Eng., pp. 537–551, 39, (2013).
55. G. E. P. Box and D. R. Cox, “An analysis of transformations,” J. R. Stat. Soc. Ser. B, pp. 211–252, (1964).
56. S. Elmidaoui, L. Cheikhi, and A. Idri, “The Impact of SMOTE and Grid Search on Maintainability Prediction Models,” In: ACS/IEEE International Conference on Computer Systems and Applications, AICCSA, Abu Dhabi, United Arab Emirates, (2019).
57. M. Hosni, A. Idri, and A. Abran, “Investigating Heterogeneous Ensembles with Filter Feature Selection for Software Effort Estimation,” In: International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, pp. 207–220, (2017).
58. E. Kocaguneli and T. Menzies, “Software effort models should be assessed via leave-one-out validation,” J. Syst. Softw., pp. 1879–1890, 86, (2013).
59. Y. Miyazaki, M. Terakado, K. Ozaki, and H. Nozaki, “Robust regression for developing software estimation models,” *J. Syst. Softw.*, pp. 3–16, 27(1), (1994).