The Journal of Systems & Software 175 (2021) 110904

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Ensemble Effort Estimation using dynamic selection ☆

Jose Thiago H. de A. Cabral, Adriano L.I. Oliveira *

Centro de Informática, Universidade Federal de Pernambuco, Recife, PE, Brazil

ARTICLE INFO

Article history:
Received 29 June 2020
Received in revised form 14 December 2020
Accepted 4 January 2021
Available online 8 January 2021

Keywords:
Machine Learning
Software Effort Estimation
Ensemble Effort Estimation
Dynamic selection
Dynamic ensemble selection

ABSTRACT

The Software Effort Estimation (SEE) process has been approached in different ways in the literature, including models built from Machine Learning (ML). The combination of these models (Ensemble) is an important research topic in ML, and has lead to improvements in accuracy compared to individual models. This paper proposes heterogeneous and dynamic ensemble selection (DES) models, composed by a set of regressors dynamically selected by classifiers to estimate software development effort. In the training phase, a pool of regression algorithms is trained using training data and a validation data set to validate the models. Next, some classifiers are trained to identify the best regression model from the pool for each training instance. In the test phase each trained classifier is used to dynamically select a regressor model from the pool for predicting the effort for each test instance. The final prediction is given by the combination of the predictions of the regressors selected by the classifiers. An experimental analysis considering a relevant set of software effort estimation problems is reported. The experiments demonstrate that the proposed method outperforms individual regressors and some state of the art models of the literature.

1. Introduction

SEE is an important challenge in software project management. This process is defined as *the process of predicting the effort required to develop a software system* (Wen et al., 2012). Some reasons for failures of software projects are: (i) unrealistic or non-articulated project goals, (ii) inaccurate estimates of needed resources, and (iii) inability to handle the project's complexity (Charette, 2005). The research community has proposed a set of models and techniques for achieving high prediction effort accuracy. More recently, the use of ML approaches has attracted the attention of the software effort estimation community. However, the choice of a general and far reaching model for a given problem, has been considered a hard task (Wen et al., 2012). This is because according to the no free lunch theorem there is no model that is the best for all software effort data sets (Bardsiri et al., 2013). Furthermore, the authors in the paper (Shepperd and Kadoda, 2001) suggest that it is more fruitful to determine the best model in a particular context rather than determining the best single model, since estimate models behave differently for different data sets, which makes them unstable. In addition, there is an additional cost to achieve optimal results for SEE using only individual models. This additional cost concerns the need to find more stable individual models. Therefore, an alternative way to overcome such drawback is to use multiple regression or classifier algorithms (ensembles) (Kocaguneli et al., 2012; Idri et al., 2016a; Dietterich, 2000), which are a composition of homogeneous or heterogeneous learning machines. It is worth mentioning that according to Dietterich (2000), an ensemble model is homogeneous if it uses a single ML algorithm, while a heterogeneous model uses two to several ML algorithms. Ensemble models are more trustworthy (present less ranking instability) (Kocaguneli et al., 2012).

There are two main approaches for combining multiple classifiers (Dietterich, 2000): (i) merging (models are combined according to a mixed rule) and (ii) selection (models are combined based on the region of competence (neighborhood) of the data). The advantage of ensembles in comparison to single models has been reported in terms of increased robustness and accuracy (Mendes-Moreira et al., 2012).

The selection could be static or dynamic. In static selection, all the test instances would have the same models subset engaged in the prediction process. The models would be selected prior to the prediction phase of tests instances. Nevertheless, the test instances would be normally associated to a range of prediction difficulty levels. In this regard, the literature has shown that better results could be obtained with a selection of a models subset for each instance (Ko et al., 2008a).

For that matter, the presented systematic reviews (Wen et al., 2012; Idri et al., 2016a; Jorgensen and Shepperd, 2007) involve ML, ensembles and effort, it displays a comprehensive analysis

☆ Editor: [ROMAIN ROBBES].

* Corresponding author.
E-mail addresses: jthac@cin.ufpe.br (J.T.H. de A. Cabral), alio@cin.ufpe.br (A.L.I. Oliveira).

https://doi.org/10.1016/j.jss.2021.110904
0164-1212/

of dynamic selection (Britto et al., 2014), and it finally inspects the effort estimation in agile software development (Usman et al., 2014). Note that the field of SEE is still in its beginnings (Idri et al., 2016a). To the best of our knowledge, the effort estimation literature has not yet considered multiple heterogeneous regressors designed by dynamic approaches. Therefore, in order to solve SEE problems, this work proposes a dynamic ensemble selection of heterogeneous models. This process is composed by a set of regressors dynamically selected by classifiers.

This paper deals with a regression problem, therefore, the models used are generated from regression algorithms. The literature has not yet presented studies that assess effort estimation problems using the selection and the combination of heterogeneous multiple regressors. A preliminary version of the method proposed in this work was presented in de A. Cabral et al. (2017).

In this paper a pool of regression algorithms is trained using training data and a validation data for avoiding overfitting. The three regressors that initially achieved the best accuracy in the validation data sets were chosen to compose the basic ensemble. A new data set for classifiers is made from the training data set. This data set target attribute is the regression model, which belongs to the basic ensemble, with the best performance for a given instance of the validation data set. Subsequently, some classifiers are trained to identify the best regression model from the pool for each training instance. In the test phase each trained classifier is used to dynamically select a regressor model from the pool. The regressor model selected by each classifier is saved for each test instance. The final prediction is given by the combination of the predictions of the regressors selected by the classifiers.

The next items list the main differences from this paper to our preliminary study (de A. Cabral et al., 2017).

(i) In the previous paper, the proposed method defined the best individual regressors using training data set, and this could be argued as to the validity of the model. In this paper, validation sets were separated for each database with the aim of identifying the best three individual methods. Moreover, the validation set was also used to define the classifiers (selectors) in the DES process. Besides that, the databases with less than 25 projects were disregarded.

(ii) The metric used in the statistical tests in de A. Cabral et al. (2017) benefits underestimation, and this was demonstrated in this paper (Shepperd and MacDonell, 2012). The evaluation carried out in this paper used the Absolute Residual (AR) metrics to verify the new model proposed in this paper. So we eliminated the metric bias. We also used the AR in the logarithmic scale and a new version of the Magnitude Relative Error (MRE) to build a rankings among all evaluated methods.

(iii) New individual methods were included in this paper, that is, one regressor and seven classifiers. The experimental section includes many new analysis and comparisons with state-of-the-art methods.

This paper assesses the application of DES models of ML algorithms. Aiming at the solution of SEE, it is proposed the dynamic selection of heterogeneous models, composed of an ensemble of regression models, dynamically selected by classifiers. Along with this purpose, the current research examines the classifiers that had the best performances as the regression models selectors.

An experimental analysis using sixteen relevant databases of software effort estimation is reported in this paper. The AR measure has been used to assess performance and to compare the proposed method to individual models and ensemble methods previously presented in the literature [1].

[1] A Java implementation of the proposed methods is available at https://github.com/jthac/siseect/

This work performs statistical analysis (using the Friedman test (Friedman, 1940) and the Least Significant Difference (LSD) test (Keselman et al., 1991)) to establish a performance rank that evaluates the investigated models based on the performance of pairwise algorithms. The present paper compares the proposed methods to individual models, to static heterogeneous ensembles, to homogeneous ensembles, to dynamic single selection methods, and to dynamic ensemble selection methods.

The following research questions are considered in this paper:

(i) can dynamic selection improve the accuracy of ensemble methods in SEE?

(ii) can dynamic selection using classifiers to select regressors from a set of heterogeneous models improve the accuracy of the effort prediction?

(iii) which criteria of dynamic selection of regression models show better results in the effort estimation?

This paper is organized as follows. Section 2 briefly reviews the basic concepts and definitions of the software development effort estimation and of the dynamic selection methods. Section 3 presents the formal definition of the proposed method. Section 4 discusses the experimental methodology. In Section 5 shows the experiments statistical results, discussion, and threats to validity. Lastly, Section 6 summarizes the study.

2. Theoretical background

Many methods for estimating costs in software development projects are based on building a model based on data from previous projects. A number of person-hours, day or month is implemented for the solution of each project task. Effort represents the major factor in software costs (Argawal et al., 2001). The main problem to predict the effort put into software development is the establishment of a database composed by features of previous software projects, as well as to quantify its attributes in numerical terms (de A. Araujo et al., 2012a). The dependent variable, called effort, is given in hours, which represents the final software development cost (de A. Araujo et al., 2012b). Note that some databases also consider the prediction of the software delivery time (Oliveira et al., 2010). Accordingly, seeking to meet the current demand for software development processes, the project manager has to estimate in advance its duration and its correspondent effort (Oliveira, 2006). de A. Araujo et al. (2012a) sustains that the main risk factor for software development is the cost to finish it. Seen in these terms, the particularities of software projects development can make the process of cost estimation too hard (de A. Araujo et al., 2012b). For this reason, effort estimation is still considered a challenge for the software engineering.

ML has been used to solve many different types of problems. Since the 80's, many articles have suggested ML methods to improve the accuracy in effort estimation. The most common ML techniques used are: Case-Based Reasoning, Neural Networks, Decision Trees, Bayesian Networks, and Support Vector Regression (Wen et al., 2012). Several studies show that ML methods have overcome linear regression methods (Shepperd and Schofield, 1997; Gray and MacDonell, 1997; Walkerden and Jeffery, 1999), however different studies showed that linear regression methods have overcome some learning machines (Mendes et al., 2003; Briand et al., 1999; Stensrud, 2001). There certainly is not a suitable mathematical model to solve all problems, even if it is only software effort estimation problems.

Support Vector Regression (SVR) are ML models widely used in the field of software engineering, especially problems related to increased dimensionality and tendency to contain noises. SVR models have showed relevant results in data sets with these characteristics (Corazza et al., 2011; Oliveira, 2006). Dragicevic et al. (2017) shows a Bayesian network capable of predicting

effort in any method of agile development. A comparative investigation between Multilayer Perceptron (MLP) and radial basis function (RBF) against multiple linear regression models (MLR) was performed in López-Martín and Abran (2015), the prediction accuracy of the neural networks resulted statistically better than that of a statistical regression. In view of these researches, and the large number of works known in the area of effort estimation, it is possible to say that ML models are consolidating in the software engineering community. However, the combination of ML models has grown in the context of effort estimation problems. These combined models are known as ML ensembles and they try to improve the accuracy previously achieved by simple models of learning machines.

The literature on methods for predicting software development effort defines Ensemble Effort Estimation (EEE) as a combination of several single estimation techniques, or base models, under a specific combination rule (Seni and Elder, 2010). The combinations model may be homogeneous or heterogeneous. They are homogeneous when built by models from the same algorithm, and heterogeneous when constructed by models from different algorithms.

Methods that combine ML models are categorized as either selection or fusion methods. Selection methods are those that select (static or dynamic) models from an initial set of models whereas fusion methods are the ones that merge the outputs of the sets of models. Generation and combination methods of homogeneous models have appeared in the literature and are widely used to this day. For example, Bagging (Breiman, 1996), Boosting (Schapire et al., 1998), and Random Subspace (Ho, 1998), among others. As for the combination among heterogeneous models, we highlight the stacking method which is known in the literature as a stacked generalization of models (Wolpert, 1992). Another simple and effective way to combine heterogeneous models is simply join the models to linear operations (mean, median).

Previous works have investigated the use of static ensembles for effort estimation, and the results have shown that ensembles are generally more efficient (Twala and Verner, 2016; Idri et al., 2016b,a). Braga et al. (2007) assert that bagging improves the SEE produced by several single learning machines, such as Regression Trees (RT) and MLP. Kultur et al. (2009) used five databases and have shown that an adapted version of bagging (homogeneous ensemble) provides considerable improvements in comparison to single learning machines. The paper (Hosni et al., 2016) proposed and evaluated a heterogeneous ensemble based on four well-known ML models (K-Nearest Neighbors (KNN), MLP, SVR, and M5Base) using three combiner rules over two well-known data sets. The empirical results of Hosni et al. (2016) showed that the proposed ensemble improved the estimation accuracy of its solo techniques. In Pospieszny et al. (2018) the authors presented a simple combination of a Support Vector Machine, a Multilayer Perceptron and a Generalized Linear Models, the results indicate that ensemble tends to improve the prediction accuracy. Heterogeneous ensemble are less used than homogeneous ensembles in the literature, but they allow us to achieve desirable results (Mendes-Moreira et al., 2012). In this respect, Dietterich (2000) also argues that heterogeneous models perform better when their accuracy and stability are taken into account.

However, the selection strategies commonly adopted to build these models are not dynamic, which can lead to a failing when combining the regressors (Kocaguneli et al., 2012; Kittler et al., 1998). Test instance is generally linked to different association difficulties, and the basic ensemble is applied in all instances. The main reason that makes us choose dynamic rather than static selection (basic ensemble) is that in dynamic selection we can achieve greater precision for an unknown test pattern. The literature shows that better results can be achieved by selecting a subset of a pool of models for each instance (Ko et al., 2008b). The models are selected based on the characteristics or space regions in each instance. The main strategies of classifiers dynamic selections are based on local accuracy.

Five methods from the literature of dynamic selection are employed in our experimental study in this paper to compare to our proposed method: (i) The Dynamic Classifier Selection by Local Accuracy (DCS-LA) (Woods et al., 1997), this method is an approach to the dynamic selection of a classifier, based on the concept of location precision estimation. The idea is to estimate the accuracy of each classifier in local regions of the features space around the sample of unknown tests, therefore, the classifier that achieves the best accuracy in the region is chosen; (ii) The Dynamic Classifier Selection by Local Accuracy Weighted (DCS-LAW) is an approach with weights (Mendes-Moreira et al., 2012); (iii) the method proposed by Di Nucci et al. (2017) which is able to dynamically select from a set of machine learning classifiers the one that best predicts the bug proneness of a class based on its characteristics; (iv) The K-Nearest Oracle Eliminate (KNORA-E) (Ko et al., 2007), the classifiers are only considered competent when they achieve a 100% accuracy in the region of competence and (v) The K-Nearest Oracle Union (KNORA-U) (Ko et al., 2007), this method selects all classifiers that correctly classified at least one sample belonging to the region of competence of the query sample. The KNORA selects a models subset. The two most frequently dynamic selection methods are DCS-LA, and the KNORA. They have been originally developed for the selection of homogeneous models, besides that the five dynamic selection methods (DCS-LA, DCS-LAW, NUCCI, KNORA-E, and KNORA-U) are not specifically used for effort estimation.

The oracle concept is linked to the selection of the best classifier subsets. Its purpose is to find the best performing classifier or classifiers for each instance. The KNORA selects a classifiers subset for each instance. This selection is based on the best ratings of the k-nearest neighbors, from the validation data set to the testing instances data set. For each testing instance, KNORA stores the classifier that matched the classes of neighboring k (Ko et al., 2008a). The base classifier is selected if it correctly estimates the class of the neighbors k. The method uses the topology in parallel, considering that all classifiers may respond to the same problem. The result is directed to a single point. In Cruz et al. (2017), it was reported a method of dynamic selection based on the formal definition of the oracle, named META-DES Oracle. The oracle is a selection based on an abstract method, which selects the ideal classifier for a selection scheme. The problem of this proposal, and of the previous version of this same method (Cruz et al., 2015), is that they are applicable exclusively to classification problems.

Furthermore, recent work on the dynamic selection literature has demonstrated that dynamic selection techniques are effective tools for classification problems that are ill-defined, i.e., for problems where the size of the training data is small and there is not enough data available to model the classifiers (Cavalin et al., 2013). Most of the data sets related to software effort estimation are small, if compared with other data sets in ML. It is also generally accepted that the highest accuracy results that a classifier system can achieve depend on the quality of data and the appropriate selection of a learning algorithm for the data.

A systematic mapping study carried out between 2000 and 2015 which summarized the existing EEE studies was presented in Idri et al. (2016b). Furthermore, a systematic literature review in EEE was done in Idri et al. (2016a), summarizing and gathering the empirical results of the several studies published between 2000 and 2015. The authors concluded that ensembles built from ML models often improve the results achieved by single models and have already been used in the software engineering community. We also performed an automatic and manual search in

the literature between 2016 and 2019 that was similar to the one performed by Idri et al. (2016a). The selected papers did not present dynamic ensemble selection in their structure and the results achieved by the ensembles are better than the results achieved by the base models. A review on dynamic ensembles has been presented (Britto et al., 2014), and no study has investigated problems of effort estimation. In this sense, this paper proposes heterogeneous and dynamic ensemble selection (DES) models, composed by a set of regressors dynamically selected by classifiers to estimate software development effort.

3. The proposed method

The next subsections will show the proposed method.

The ensemble process can be divided into three steps (Cruz et al., 2018), but there is no widely accepted definition for an ensemble learning for regression (Mendes-Moreira et al., 2012).

The first step is the ensemble generation, which consists on generating a set of models. In the second step, named ensemble pruning, the ensemble is pruned by eliminating some of the models generated earlier. Finally, in the ensemble integration step, a strategy to combine the base models is defined. This strategy is then used to obtain the prediction of the ensemble for new cases, based on the predictions of the base models.

Concerning the experimental setup, the most common approach is to split the data into three parts: (1) the training set, used to obtain the base predictors; (2) the validation set, used for assessment of the generalization error of the base predictors; and (3) the test set, used to assess the performance of the final ensemble method.

Fig. 1 shows the new version of the proposed method. The items list with the main changes to our preliminary study was approached in the begin this paper in Section 1.

3.1. Description

Fig. 1 illustrates the proposed method. It can be observed that the process involves the following stages: (i) the stage of the regression models generation, which are used to estimate the effort of software projects; (ii) the stage of the pruning phase, in which only three models are chosen based on their accuracy and diversity — which is easily obtained due to the use heterogeneous models; and (iii) the stage of integration, on what the models are combined and selected for each validation instance.

Initially in the first stage, we started with the training data set being used by the individual regression algorithms, in which a regressor estimation model is generated for each base regression algorithm. In second stage, the three regressors that achieved the best accuracy in the validation data sets are chosen to compose the basic ensemble that will be evaluated through static and dynamic selection. A previous evaluation that combined the regressors in the validation bases indicated that the best basic ensemble would be built from 3 regressors, using the median as a combination method. In third stage, it is made a training data set for classification, which is similar to the first training data set (used for training the regression models). The second data set has the same input features of the first one, yet its target attribute is the regression model (nominal value) with the best performance for a given instance of the training data set. Later, this data set is used for training a classifier to dynamically select the most suitable regression model for each instance of the validation data sets. Then, for each data set we selected the best three classifiers evaluated. They will be used to dynamically select regressors for each test sample in each data set.

In this sense, we choose the best regressors combination (basic ensemble) from a pool of regressors and the best three classifiers that operate as dynamic selectors for each database. We use the best classifier to select one regressor, and we also use on each database the best three classifiers to dynamically select three regressors, respectively. This way, the selection can be determined by a unique or by various classification models; and the combination is defined by linear methods. This combination is possible only if at least two classification models make the selection, that is, when the DES is built. In this paper we use three classification models for each data set. It is important to note that the DES proposed may contain repeated regressors, since a classifier may select the same regressor previously selected by another classifier.

3.2. Challenges to build ML ensemble models

The literature asserts that combinations of ML models tend to increase accuracy prediction (Shunmugapriya and Kanmani, 2013). The underlying idea behind a pool of heterogeneous models is to rely on expert algorithms that are able to differ themselves in terms of properties and concepts (Britto et al., 2014). Defining accurate and diverse models is key to building a quality ensemble. Accurate models can be obtained with a previous evaluation. Beside that, in the heterogeneous ensemble approach is expected to obtain models with higher diversity (Webb and Zheng, 2004). Since the models are generated by different ML algorithms, which have distinct characteristics, the diversity among them is induced. This feature is critical to the achievement of better results (Brown et al., 2005b,a).

Nevertheless, the use of several models could lead to two problems: (i) the suitable selection of the base models that will build the ensemble, and (ii) the appropriate selection of criteria to match the basic models (Shunmugapriya and Kanmani, 2013). Solving these problems is a challenging, but achievable target.

Choosing individual models is a difficult task, so we generate several regressors from widely used and well evaluated algorithms in the ML and software effort estimation literature (Wen et al., 2012). Table 1 presents the population set of regression models evaluated in this study. The KNN and the MLP have been parameterized to engender different and diverse regression models. For all the other algorithms, the default parameters of Weka 3.6.10 have been adopted.

It is essential to define a criterion for measuring the competence level of each base model of DES (Cruz et al., 2014). In the present proposal, several distinct classifiers are used for the dynamic selection. Some of these selectors (classifiers) are based on local accuracy, but not all of them consider this same approach (eg: neural networks). Table 2 presents the set of classifiers used in the dynamic selection process proposed in this work. Thus, the proposed system differs from the current state-of-the-art dynamic selection techniques not only because it uses multiple regressors, but also because the regressor selection rule is learned by various classifiers. This work sustains that the results could be improved with the use of a rigorous criterion to dynamically select a regressor that composes the ensemble. The dynamic selection based on different selection criteria has achieved excellent accuracy (Cruz et al., 2015, 2018, 2017).

It is worth mentioning that some reports in the SEE literature suggest that the performance of different models depends considerably on the characteristics of the data set (Minku and Yao, 2013). This led us to define a specific set of classifiers for each database. Then, a set of classifiers for each database was used to select the regressors that will carry out the estimation. This allow us to use different and specific approaches to dynamic selection and therefore, leads to more accurate results than the ones obtained with a simple approach, such as local accuracy (Cruz et al., 2014). Therefore, we aimed at diversity in our selection of

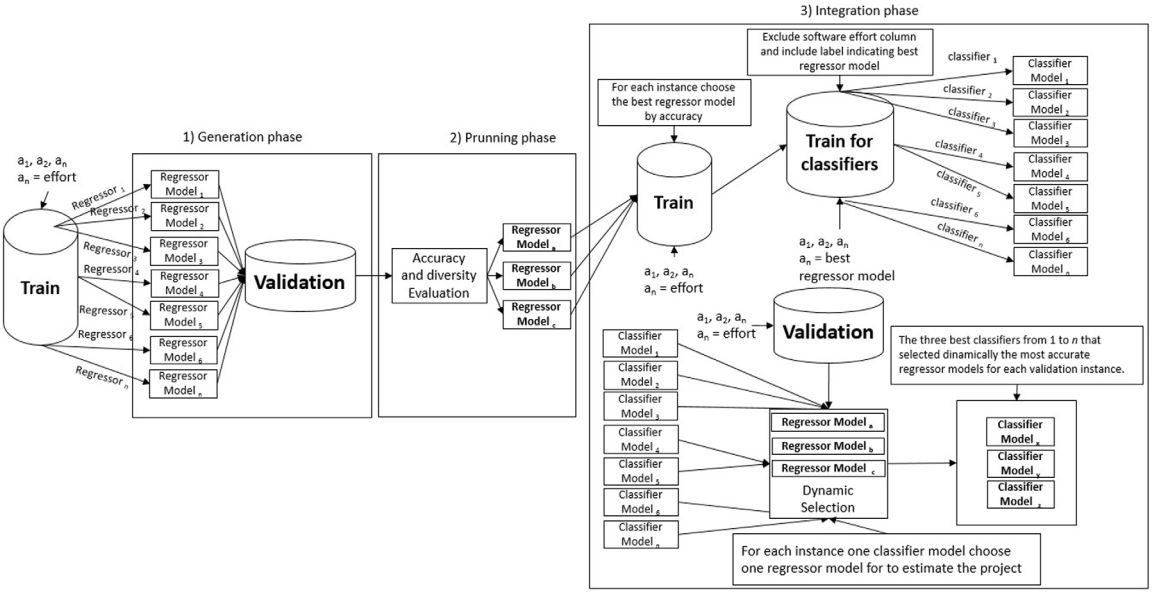

Fig. 1. The proposed method.

Table 1
Regression algorithms used in this work.

Algorithm	Initials
Conjunctive Rules	CR
Decision Stump	DS
Decision Table	DT
Gaussian Process	GP
IBk (K-Nearest-Neighbor, k = 5)	KNN
Least Median Squared	LMS
Linear Regression	LR
Locally Weight Least	LWL
M5Base	M5P
M5 Rules	M5R
Multilayer Perceptron (hiddenlayer = 2, learningRate = 0.01; momentum = 0.02; trainingTime = 1000)	MLP
RBF Network	RBF
Rep Tree	RP
Support Vector Regression	SVR
Zero R	ZEROR

Table 2
Classifiers algorithms used in this work.

Algorithm	Initials
Adaboost (base classifier = RepTree)	AD
Bagging (base classifier = RepTree)	BA
Bayes Net	BN
Best First Tree	BFT
Decision Stump	DS
Decision Table	DT
IBk (K-Nearest-Neighbor, k = 3,5,7)	KNN
J48	J48
JRip	JRip
KStar	KS
LMT	LMT
Locally Weight Least	LWL
Logistic Regression	LoR
Lad Tree	LT
Multilayer Perceptron (hiddenlayer = 1, learningRate = 0.01; momentum = 0.02; trainingTime = 1000)	MLP
Naive Bayes	NB
Random Forest	RF
RBF Network	RBF
Rep Tree	RP
Support Vector Machine	SVM
Oner	OR

classifier models in order to find the most suitable ones for each data set.

Next, we will present the details of the steps of the construction of the proposed method.

3.3. Steps

The proposed method is summarized in the following steps:

1. For datasets with more than 500 instances, we use repeated hold out. In this case, we split the dataset into n training datasets TR_1, TR_2, ..., TR_n, validation datasets VA_1, VA_2, ..., VA_n, and testing datasets TE_1, TE_2, ..., TE_n. Each training dataset contains 50% of original data, randomly selected (with replacement). The remaining data are used for validation (25%) and testing (25%). On the other hand, we use leave-one-out cross validation for datasets with less than $n = 500$ projects. In this case, they are distributed in $n - 2$ instances for training, 1 instance for validation, and 1 instance for testing. In this configuration, n (number of samples) is equal to the size of the data set. In this procedure, it is assured that all data is used for training, validation and testing.
2. Construct n individual regression models RM_1, RM_2, ..., RM_n) for each different training bases TR_1, TR_2, ..., TR_n to obtain n distinct regressors (members of the set) generated by the r algorithms, in this paper $r = 15$. Starting from n training data created in the previous step, n regression models for each regression algorithm (r_i) are generated and validated in the n validation data set. The absolute error of the models in the validation bases defines the accuracy of a regression algorithm (r_i). The resulting accuracy is linked to each of the r_i algorithm.
3. Evaluate all distinct regressors using statistical methods. The results achieved by the models in the validation data set will be compared through non-parametric and post hoc statistical tests. Each regression algorithms obtains the absolute error value according to the models generated and

validated in the n validation data set. Statistical comparison is made in order to identify the best three individual regressors in general.

4. Select the models according to the diversity and accuracy of these models. According to the previous result, we identify the best regressors. These individual methods are chosen based on the accuracy and diversity among them. Thus, the best combination of the models will build the basic ensemble.
5. Define the best regression model for each instance of training TR_1, TR_2, …, TR_n and add a new field for model identification. A data set for classification problem is created from initial training set. Training instances are validated, and a new field identifies the best regressor among the three individual models of the basic ensemble (prior selected models), that is add in the training instance. The tag with the best regressor will replace the label attribute "effort"; therefore, we started the creation of a new training set for classification models.
6. Save the training data sets for training the classifiers TRC_1, TRC_2, …, TRC_n. These data sets do not contain the effort as a target, instead, the target it the ID of the best regressor. Once the previous step is finished, we will have n training sets for classification algorithms, where n is the number of training sets (50 data sets when the set has more than 500 instances, or the total number of instances of the database if it is less than 500 instances). All the data sets are saved for the next step (training the classifiers).
7. Train c individual classifiers using the same training data sets TRC_1, TRC_2, …, TRC_n for a dynamic selection of the regressors. n identifies the number of training data sets that classifiers will use to learn. Thus, n also identifies the number of iterations of the evaluation. For each iteration we have identical training, validation and test data sets for each evaluated classifier. A set of c classifiers are trained, in this work $c = 21$. These classifiers will associate the set of attributes with a selected regressor. The built model will be used to the dynamic selection of the regressor.
8. Validate c individual classifiers by selecting the appropriate regressor for each instance of validation data set VA_1, VA_2, …, VA_n. Considering the classifiers created in the previous step, we start the process that will identify the best classifiers to be used as dynamic selectors of the current data set.
9. Select the three best-performing classifiers in a set of c classifiers for each database. The best classifiers are selected based on the accuracy achieved by the classification model in the validation data set for a specific database. Each database is associated with three classifiers, which will be dynamic selectors for that data set in the testing phase.
10. Use the three selected classifiers and then estimate the testing instance with the regressors suggested by them. At this point, the testing process begins. The classifiers selected in the previous step are used to dynamically select a set of regression models according to the set of input attributes of the instance. Each classifier suggests the ideal regressor that should estimate the test instance effort.
11. Estimate each test instance with the regressor proposed by the best classifier and with the regressors proposed by the best classifiers. In the previous step, the best classifiers identified the best regressors to perform the estimation, and in this step, the regression models chosen by the classifiers estimate the effort of the current project (instance). When we have only one classifier (the best evaluated in the validation data set) being used to select one regressor, we are performing a simple dynamic selection (1 selected); however, when we have the best three classifiers being used to select the regressors, we are performing a dynamic ensemble selection.
12. Use the regressor selected by the best classifier for PEETACO-DS (Process of the Effort Estimation Through Algorithm Combinations-Dynamic Selection) and combine the set of regressors selected by the three best classifiers using different combination strategies for PEETACO-DES (Process of the Effort Estimation Through Algorithm Combinations-Dynamic Ensemble Selection). We identified by PEETACO-DS method that with only one classifier (the best) to select one regressor. While the dynamic selection that uses the three best classifiers is identified by PEETACO-DES. It is possible that only one regressor have been used to make a combined estimate, this happens when the set of classifiers unanimously choose the same regression model. Each regressor selected will provide an estimate. Then, the mean, the median, the mean of the extremes, the highest and the lowest value of the prediction set are the combination used.

The generation phase covers from step 1 to step 3. The pruning phase begins and is completed in step 4. Finally, the integration phase, including the testing is from step 5 to step 12.

3.4. Algorithms to describe the performed process

Next we present the algorithms that describe the suggested model. Algorithm 1 presents the steps in a structured way to obtain the selected regressors at the beginning of the process. Algorithm 2 presents the structure to obtain the set of classifiers that will be used by the proposed model in a specific data set. Finally, algorithm 3 illustrates the testing process.

Algorithm 1 Process of the Effort Estimation Through Algorithm Combinations — set of regressors

Input: *TR*, *VA*;
reg = amount regressors individuals;
Train using *TR* data set and generate *reg* regression models, RM = {rm_1, rm_2...rm_{reg}};
$n = 0$; $i = 1$; $j = 1$;
for each regression model individual rm_i **do**
  for each validation instance VA_j **do**
    $\hat{e}_j$ = prediction for model rm_i in j;
    $\text{error}_j = |\text{effortReal}_j - \hat{e}_j|$;
    $\text{ArrayOfAbsoluteResiduals}_{\text{rm}_i}.add(\text{error}_j)$;
    n++;
    j++;
  end for
  $n = 0$; $j = 1$; i++;
end for
// Evaluate statistically (section 4.3) *reg* models individuals.
Friedman = Friedman's Ranking;
LSD = Test of the Least Significant Difference;
setOfRegressors = TheBest(*Friedman*, *LSD*, $ArrayOfAbsoluteResiduals_{rm_1}$,..., $ArrayOfAbsoluteResiduals_{rm_{reg}}$)
setOfRegressors = {rm_x, rm_y, rm_z} // winners regressors models
return *setOfRegressors*

3.5. Evaluation of the proposed model

The evaluation of the proposed model was carried out through comparisons against several other state of art models. Below, we list the competing approaches.

Algorithm 2 Process of the Effort Estimation Through Algorithm Combinations — set of classifiers

Input: *setOfRegressors*, *TR*, *VA*;
$i = 1$;
for each training instance TR_i **do**
  $regressorModelChoosen = MinorErrorEvaluated\ (TR_i, rm_x, rm_y, rm_z)$;
  $regressorModelChoosen = rm_x$ or rm_y or rm_z
  $newInstanceTrain = replaceTarget(TR_i, regressorModelChosen)$;
  $trainClassifier = add(newInstanceTrain)$;
  i++;
end for
$n_{Classifiers}$ = amount_classifier;
Train using *trainClassifier* data set and generate $n_{classifiers}$ classification models, CM = $\{cm_1, cm_2 ... cm_{n_{classifiers}}\}$;
$i = 1$; $j = 1$;
for each classification model cm_i **do**
  for each validation instance VA_j **do**
    $rmSelected$ = DynamicSelectionRegressor(VA_j, cm_i);
    $\hat{e}_j$ = prediction for model *rmSelected* in j;
    $\text{error}_j = |\text{effortReal}_j - \hat{e}_j|$;
    $\text{ArrayOfAbsoluteResiduals}_{cm_i}.add(\text{error}_j)$;
    j++;
  end for
  $j = 1$; i++;
end for
Evaluate i classifier models and choose ones that obtained the best accuracy for selection using the $ArrayOfAbsoluteResiduals_{cm}$.
$setOfClassifiers = \{cm_a, cm_b, cm_c\}$
return *setOfClassifiers*

Algorithm 3 Process of the Effort Estimation Through Algorithm Combinations — Testing process

Input: *TE*, *setOfClassifiers*, *setOfRegressors*;
for each test instance TE_i **do**
  R_1 = DynamicSelectionRegressor(TE_i, cm_a, *setOfRegressors*);
  R_2 = DynamicSelectionRegressor(TE_i, cm_b, *setOfRegressors*);
  R_3 = DynamicSelectionRegressor(TE_i, cm_c, *setOfRegressors*);
  $PEETA\hat{C}O(DS)$ = estimate R_1
  $PEETA\hat{C}O(DES)$ = combined estimate by R_1, R_2, and R_3
end for

1. Individual models of machine learning (fifteen models evaluated in validation phase);
2. Static selection single (the best three models selected in validation phase);
3. Homogeneous (bagging and boosting) and heterogeneous combinations (static ensemble selection of the best three models);
4. Dynamic selection methods (DCS-LA, DCS-LAW, and NUCCI);
5. Dynamic ensemble selection (KNORA-E and KNORA-U);

4. Experimental setup

This section presents the development of the experimental methodology for this study.

4.1. Motivation and baselines

The experimental section includes many analysis and comparisons with state-of-the-art methods. We selected widely used and well evaluated regressors in the ML and software effort estimation literature, the three most common ML techniques used are: Case Based Reasoning, Neural Networks and Decision Trees (Wen et al., 2012; Idri et al., 2016b). SVR and Linear Regression are also broadly used in effort estimation problems (Hosni et al., 2016; Mendes et al., 2003). However, the combination of ML models has grown in the context of effort estimation problems (Idri et al., 2016b). Generation and combination methods of homogeneous and heterogeneous models have appeared in the literature and are commonly used to this day (Wolpert, 1992; Breiman, 1996; Schapire et al., 1998). Another way to combine heterogeneous models is to join the models via linear operations (mean, median). However, the selection strategies commonly adopted to build these models are not dynamic, which can lead to a failing when combining the regressors (Kocaguneli et al., 2012; Kittler et al., 1998). Five methods from the literature of dynamic selection are employed in our experimental study in this paper to compare to our proposed method: DCS-LA, DCS-LAW, Di Nucci, KNORA-E, KNORA-U. The main motivations for defining all these baseline methods were: (i) the identification of various works on software effort estimation in the literature that use these well-known ML methods, (ii) the progress of the ensembles (homogeneous and heterogeneous) and the selection methods (static and dynamic) found in the literature.

The individual models were generated using the WEKA data mining software package library.[2] As can be observed in Britto et al. (2014), it is impossible to choose the best method of dynamic selection, since there is no evidence that a specific method outperforms the other methods in any classification task. However, it can be noticed that DCS-LA and KNORA achieve similar results in many problems. Britto et al. (2014) also highlight that DCS-LA and KNORA methods are adopted in several works in the literature and dynamic selection methods generally outperform the best individual ensemble methods. The study performed does not lead to the definition of the best dynamic selection method. On the contrary, there is no evidence that one specific method may surpass all the others in every problem.

In addition to the DCS-LA, we implemented the dynamic selection method proposed by Di Nucci et al. (2017), called ASCI (Adaptive Selection of Classifiers in bug predIctiong), which is able to dynamically select from a set of machine learning classifiers the one that best predicts the bug proneness of a class based on its characteristics. Our proposal is similar yet different in several aspects from the model proposed by Di Nucci et al. (2017), as shown in Table 3.

Regression algorithms are used to predict the effort, since the estimates are real values. It is important to notice that in some circumstances the use of regressors are impracticable, such as in the selection strategy used by KNORA (Britto et al., 2014). In this case, the classification technique should be adjusted to predict the variable target over a range of values. We can observe that these methods were proposed for classification problems. However, ensemble methods that are adopted for classification cannot be directly applied to regression problems. We made minor adjustments to the DCS and KNORA methods, in order to enable them to tackle regression problems.

To adapt DCS-LA for regression, we select the regressor that obtains the lowest mean error according to the metric used among the nearest neighbors (where k = 7). In the original approach, the classifier that obtained the highest accuracy was

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

Table 3
Nucci vs. PEETACO.

Nucci	PEETACO
Dynamic selection for bug detection	Dynamic selection for effort estimation
Selection of classifier models	Selection of regression models
Basic ensemble predefined by the knowledge in the area	Basic ensemble defined through a rigorous statistical analysis in the validation data set
Dynamic selection using one algorithm (Random Forest)	Dynamic selection using classifiers distinct according to the data set
Selection of one classification model. It is a single dynamic selection	Selection of a set of regression models. It is a DES.

Table 4
The data sets used in the experimental analysis.

Database	Features	Projects
ISBSG — All	20	1113
ISBSG — Banking	73	73
ISBSG — Communication	19	137
ISBSG — Government	19	102
ISBSG — Insurance	19	162
ISBSG — Manufacturing	19	80
ISBSG — Service Industry	19	29
ISBSG — Remain All	19	530
PROMISE — Cocomonasa V2	23	93
PROMISE — Desharnais	11	81
PROMISE — China	14	499
PROMISE — Cocomonasa V1	17	60
PROMISE — Cocomo81	18	63
PROMISE — MAXWELL	27	62
PROMISE — KITCHENHAM	6	145
PROMISE — MIYAZAKI94	8	48

selected. For KNORA it has been considered that a regression model generalizes well for one of its closest neighborhoods when the estimation error in a neighboring instance is smaller than the means of the errors of the individual algorithms validated at the beginning of the process. The verified model is added in the dynamic ensemble constructed by the KNORA method.

4.2. Data sets

The construction and evaluation of software effort estimation techniques rely mainly on historical software project data and evaluation methods (Idri et al., 2016a; Wen et al., 2012). Thus, the accuracy of an estimation technique depends on the characteristics of the software project data such size, missing values, and outliers, as well as the evaluation method used. In order to evaluate the proposed method, we used a relevant set of software development effort estimation problems from the PROMISE (Shirabad and Menzies, 2005) and International Software Benchmarking Standard Group repositories (ISBSG) (Abran, 2015). Unlike some related surveys on software effort forecasting, this study applies the proposed method in different data sets, since the performance of different models depends on the database features. Table 4 summarizes all the sixteen data sets presently used.

Since many factors vary simultaneously, the statistical effects may be harder to be identified in a more heterogeneous data set than in a more homogeneous one (Gencel and Buglione, 2008). Therefore, this study built a series of homogeneous subsets by considering the factors that affect the effort. The PROMISE repository contains some homogeneous databases, with a content that is not widely diverse. Asymmetrically, the ISBSG data sets are substantially heterogeneous. With the aim of creating data subsets with homogeneous behavior, the software engineering projects were assembled according to the industry sector that provides homogeneous data, similar to Minku and Yao (2013). This work uses most of the databases present in software effort estimation research. We use 7 databases from the 10 most common ones in the literature (Idri et al., 2016a). Some were no considered because they did not reach the minimum amount of the projects established (25 projects, as explained below).

In accordance with the guidelines provided by ISBSG, regarding the size of the data sets, groups with at least twenty five projects have not been selected, in this work we consider 25 projects to be the minimum quantity allowed to use a PROMISE or an ISBSG database. It has been chosen 1468 ISBSG projects in total. The projects to which Java, C++, or Visual Basic have been assigned as the primary programming language, were measured in function points. Projects with the following characteristics have been eliminated: (i) data quality rating other than A (the data submitted was assessed as being sound with nothing that might affect its integrity being identified) or B (the submission appears fundamentally sound but there are some factors which could affect the integrity of the submitted data), and (ii) resource level other than 1 (development team effort, e.g., project team). In this way, 355 projects were eliminated, and we use the remaining 1113 projects. These projects were grouped by industry sector. The project that lacked value in the industry sector attribute, or the group that did not reach the minimum amount defined to have an approved database were grouped into a single database called Remain ALL. All in all, we selected 16 data sets for the experiments; 8 belong to ISBSG, and the other half to PROMISE. Regarding the PROMISE, 2 data sets were eliminated because they did not reach 25 projects, then remained 8 data sets.

The ISBSG databases were distributed as follows:

1. 1 data set containing all the projects of the ISBSG sample (ISBSG ALL);
2. 6 data sets separated by industry sector from ISBSG sample containing more than 25 projects (Banking, Communication, Government, Insurance, Manufacturing, and Service Industry);
3. 1 data set with all remaining data of the ISBSG sample (Remain ALL).

The effort of each project was assigned as a dependent variable. In order to measure the performance of the studied method, the data sets ($n \geq 500$) were randomly split into training, validation, and testing partitions, with the proportion of 50%, 25%, and 25%, respectively. Note that the leave-one-out (Fukunaga and Hummels, 1989) process was applied in this work in data sets with less than 500 software projects.

4.3. Evaluation metrics

This investigation has selected three regression and classification algorithms. With the purpose of increasing the accuracy of the suggested models, the three best different types of regressors in were selected, considering all databases and a statistical test. Moreover, the three best classifiers based on the validation process have been selected for each data set, in order to select the best regressor for each test instance.

The performance of each trial has been measured by the AR, given by

$$\mathrm{AR} = |\mathrm{target}_i - \mathrm{output}_i|. \quad (1)$$

in which target_i and output_i represents the desired output and the predicted value for ith pattern.

Subsequently, we will present the AR in the logarithmic scale

$$\mathrm{AR}_{\mathrm{LOG}} = log_{10}^{|\mathrm{target}_i - \mathrm{output}_i|}. \quad (2)$$

The MRE could be the third metric to evaluate the models, but Shepperd and MacDonell (2012) gives an example of two projects in which the first project is overestimated and the second one is underestimated. Both estimates have identical absolute residuals, yet the MRE values differ by an order of importance. As a consequence, the Magnitude Relative Error (MRE) will be biased towards underestimated prediction systems. Researchers named it by over-optimism. Subsequently, we will present the MRE adjusted to this work, in order to solve the previously mentioned problem. Table 5 shows an similar example with MRE Adjusted. The MRE values are distinct in large scale, while MRE Adjusted maintained the same error proportion of absolute residual.

$$\mathrm{MRE}_{\mathrm{ADJUSTED}} = \frac{\frac{|\mathrm{target}_i - \mathrm{output}_i|}{\mathrm{target}_i} + \frac{|\mathrm{target}_i - \mathrm{output}_i|}{\mathrm{output}_i}}{2}. \quad (3)$$

4.4. Threats to validity

We address four types of threats to validity. An approach to each is presented below.

Conclusion validity: this type of validation verifies whether there is a relation between what we alter or compare (the treatment) and the results we obtain (the answers). In this paper, the estimation techniques are considered the treatment, and the absolute errors measured by the estimation of each technique are considered the answers. With the purpose of suppressing the main threats to the conclusions, especially in terms of the statistical results, we carried out experiments in 16 different databases (3277 projects were distributed among the data sets) to a significance level of 5%. A validity process with a high level of robustness (leave-one-out) was used in the databases containing less than 500 projects (14 to 16), whereas a simpler process (hold-out) was employed in databases containing more than 500 instances (2 to 16). In order to ensure the robustness of these results, this process was randomly resampled 50 times, and the databases were permanent for all the compared techniques.

Internal validity: in software engineering this kind of validation can be greatly affected by the behavior of the subjects who participate in the experiments. Nonetheless, in this work the treatments and the groups are not related to people, but algorithms and software project data. The quality of data collection could be a threat to the internal validity, however, only the projects with A or B level of quality from the ISBSG repository were selected. In terms of the PROMISE databases, recent studies have shown that they are, in tandem with the ISBSG databases, widely used when it comes to estimation problems of software projects (Idri et al., 2016b). According to Shepperd et al. (2013), some threats regarding NASA data set were identified, but that paper focused on defect predictions data sets rather than effort estimation. Altogether that does not affect the data sets used in the present research. Moreover, NASA data is among one of the commonly used in SEE studies (Idri et al., 2016a; Wen et al., 2012; Idri et al., 2016b). Therefore, out of the 16 databases used in this research, two of them belong to NASA.

Construct validity: this kind of validation investigates whether the effect of the results is determined by the causes that were discussed and identified. In Section 5 we present particular characteristics of the proposed method that lead to better results regarding the effort estimation, such as the use of ensembles, different classifiers (selectors), rigid validation process etc. Furthermore, data sets with different natures and attributes and distinct characteristics were taken into consideration in order to represent the population of software projects in the best possible way. Another potential construct bias would be the comparison metric used, since we considered 3 different metrics, employing the absolute error in the statistical tests, because this metric is widely used in the SEE area with the purpose of comparing the accuracy of estimation techniques. In this work we propose a model that uses 3 classifiers to select the regressors; it would be possible to question whether the results would remain the same should we use 4 or 5 classifiers, for instance, instead of 3. At the end of Section 5 we report some experiments to compare the proposed methods using 1, 2, 3, 4 and 5 classifiers. The results of Fig. 10 show that using 3 classifiers is statistically equivalent to using 4 or 5 classifiers.

External validity: this type of validation investigates whether one can generalize the results in other areas. The generalization of the developed method was better than the generalization of the other methods found in the state of the art, thus, we believe that we can achieve progress in terms of accuracy of effort estimation in software engineering projects. However, it is worth mentioning that the suggested methods are not suitable for contexts that require a high performance in terms of time. A method for defining the best selectors (classifiers) according with the characteristics of the data sets could be done through further research.

4.5. Statistical tests

The non-parametric Friedman's test has been applied for multiple comparisons of the experimental results. The tests have been performed on different databases. As recommended by Demšar (2006) and Hollander et al. (2013), the Friedman's test has been used to detect differences of performances across multiple test trials. The null hypothesis being presently tested is that all algorithms perform equally, and the eventual differences are merely random.

The Friedman test indicates whether there has been significant differences in the results. However, it is necessary to know in which methods the differences have been verified. After the null hypothesis rejection, a post hoc test has been performed to evaluate the performance of pairwise algorithms. The post-hoc test has been applied to the LSD statistical distribution for all software projects. The performance measurements were the AR. All experiments have been executed on a single machine by using Matlab R2018a, with a confidence level of 95%.

5. Results and discussion

The accuracy of the models has been evaluated in sixteen databases. Table 1 presents the regression models evaluated in this work. The results of the Friedman's test and the LSD of the experiment with fifteen individual regression models are shown in Fig. 2. LMS, SVR and M5P were the best individual regressors.

The results of all combinations considering the four best regression models of Fig. 2 (LMS, SVR, M5P and M5R) are shown in Fig. 3. The results show that the best combination is composed of LMS, SVR, and M5P; thus they were selected for the other experiments reported in the paper. We used the median to combine the regressors. Those regressors and the median showed the best combination for static ensemble. They achieved the best accuracy in the validation phase as opposed to the other static combinations. The LMS and SVR were well evaluated models because of their ability to handle noise, which is fairly common in the ISBSG data sets. SVR is characterized by a good generalization capability and robustness to large features. These characteristics found in the models may have led to better accuracy. From these 3 individual models we build the basic ensemble used in the experiment.

The basic ensemble was composed of algorithms with different characteristics, widely used in effort estimation, and positioned among the best, according with the results of validation phase, showed in Figs. 2 and 3.

Table 5
MRE adjusted versus MRE.

Projects	Real value	Estimated value	Absolute residual	MRE	MRE adjusted
Project 1	50	200	150	300	375
Project 2	200	50	150	75	375

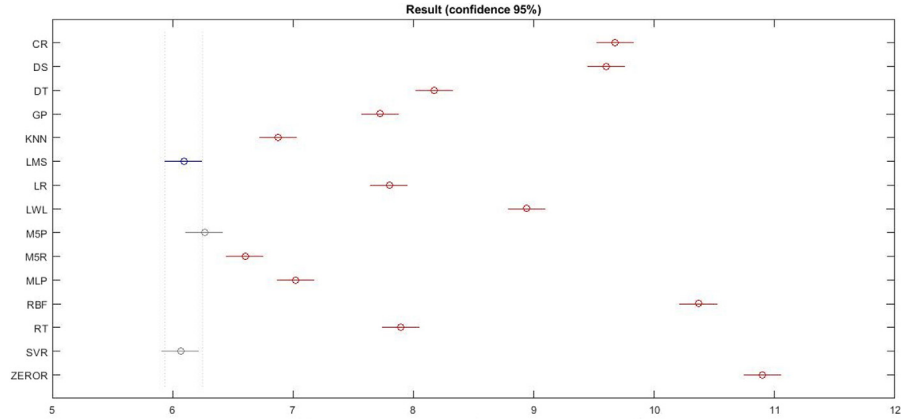

Fig. 2. Statistical results of the Friedman test and LSD for the individual models.

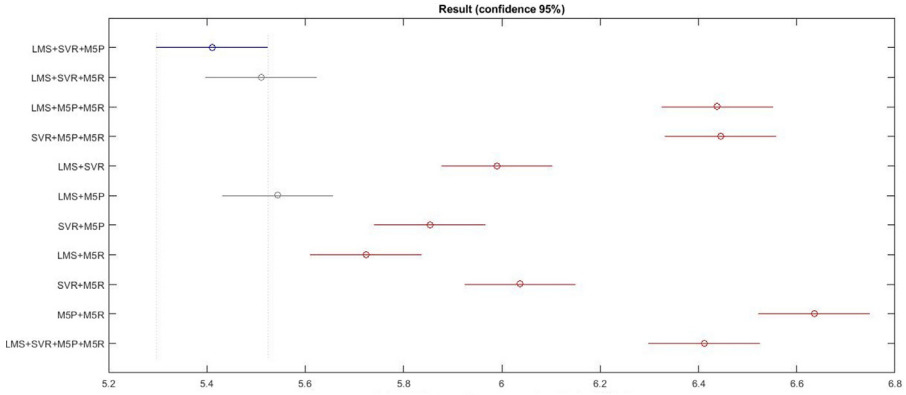

Fig. 3. Statistical results of the Friedman test and LSD for the static ensemble models.

Still in the validation phase, the best classifiers (selectors) of heterogeneous regressors were investigated. Fig. 4 shows that LWL, KSTAR and KNN7 models are the best selectors in the context of this work. However, it is not possible to state that LWL, KSTAR and KNN7 will achieve good results in all the tests, thus, we selected the 3 best classifiers for each data set in the validation phase.

Table 6 presents the classifiers used as dynamic selectors in each data set. The first column identifies the data set and the second column shows the best classifiers in descending order of accuracy in the experiment. Notice clearly in the table that the classifiers based in locality are more present, followed by decision trees (Rep Tree, Random Forest etc.), and models based in rules, which leads us to believe that dynamic selection based on local accuracy tends to improve overall accuracy, but it is not yet sufficient to overcome dynamic selection composed by different criteria. LWL, KSTAR and KNN7 ranked among the best in overall performance, but they only were used 4,5 and 5 times, respectively, considering all 16 databases, indicating that it is a fairly stable method for the dynamic selection of regression models in effort estimation databases.

It is believed that the data size, the number of attributes of the data set, and the regression algorithm used in the ensemble may define the best classifier. In further studies, it is possible to analyze the choice of classifiers that could act as a selector of regression models in a heterogeneous ensemble.

Next, we present a statistical comparison between our proposed method and (i) the best individual models (LMS, SVR, M5P), (ii) the heterogeneous static ensemble models (Mean, Median, Mean of Extremes, Maximum, Minimum and Stacking), (iii) the homogeneous ensemble models (Bagging LMS, Bagging SVR, Bagging M5P, Boosting LMS, Boosting SVR, Boosting M5P), (iv) the single dynamic selection models (NUCCI, DCS-LA, DCS-LAW), and (v) the DES models (KNORA-E, KNORA-U).

Tables 7 and 8 present a summary of the Mean Absolute Residuals (MAR) measurements for the compared methods for each database. The best MAR values have been highlighted in bold. We can note that the six versions of the proposed method (PEETACO) won in 9 of the 16 databases. It is worth saying that there is not a superior method for all databases. The challenge is to make the best generalization, thus, we performed pairwise evaluations of the algorithms.

Figs. 5, 6, 7, 8 and 9 present the comparison of the results of the individual and combined methods against the proposed method. Tables 9–13 summarize the count of competing methods were outperforms. The method analyzed is displayed in one column, and in the other column the number of methods that have been exceeded.

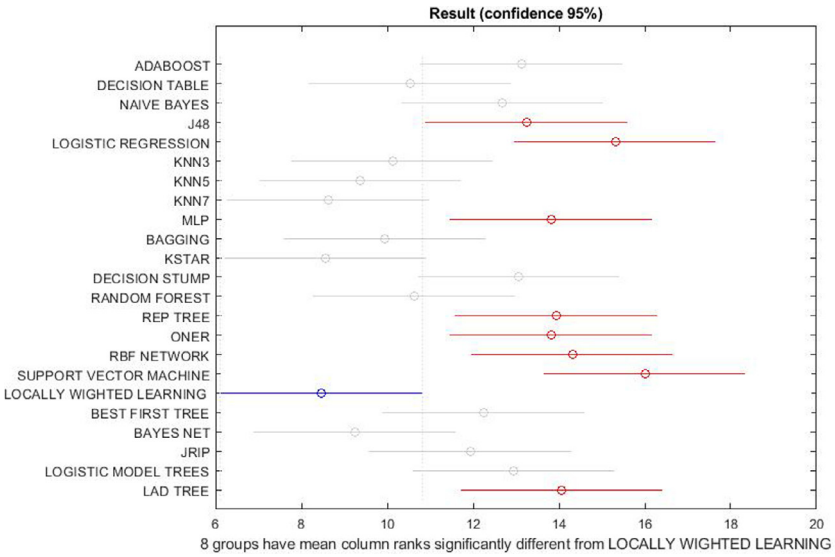

Fig. 4. Friedman test and LSD for comparison of classifiers models for selection of the regressors.

Table 6
Classifiers used in each data set.

Database	Classifiers
ISBSG — All	KNN (k = 3); KSTAR; JRip
ISBSG — Banking	Decision Table; LWL; KNN (k = 7)
ISBSG — Communication	Bagging (Rep Tree); Rep Tree; LMT
ISBSG — Government	Bayes Net; KNN (k = 5); Decision Table
ISBSG — Insurance	Bagging (Rep Tree); KSTAR; LMT
ISBSG — Manufacturing	Lad Tree; KNN (k = 3); Random Forest
ISBSG — Service Industry	LWL; BestFirst Tree; JRip;
ISBSG — Remain All	KSTAR; Decision Table; Bayes Net
PROMISE — Cocomonasa V2	Adaboost (Rep Tree); Decision Table; Rep Tree
PROMISE — Desharnais	Logistic Regression; KNN (k = 5); LMT
PROMISE — Cocomonasa V1	KNN (k = 5); KNN (k = 7); Bayes Net
PROMISE — Cocomo81	LWL; Decision Stump: BestFirst Tree
PROMISE — China	Random Forest; JRip; Adaboost (Rep Tree)
PROMISE — MAXWELL	Naive Bayes; RBF NetWork; Logistic Regression
PROMISE — KITCHENHAM	KSTAR; KNN (k = 7); KNN (k = 5)
PROMISE — MYIAZAKI	KNN (k = 5); KNN (k = 7); KNN (k = 3)

Tables 14–16 present the positions ranking of each method by metric, considering sixteen data sets of Table 4. The positions ranking considers the average of the positions of each method for each database.

Fig. 5 shows that PEETACO, especially PEETACO-DES (Mean), is superior when compared to the individual models. To create ensembles and use combination methods is sufficient to improve the results of the individual methods. This statement can be validated through the rankings presented in Tables 14–16. It is also possible to note that PEETACO-DS did not show, in general, a similar performance when compared to PEETACO-DES (Mean and Mean of Extremes), probably because the former does not use ensembles in the method. In general, the ranks of the PEETACO-DES versions were better than those of PEETACO-DS, but the latter achieved similar results to that of PEETACO-DES (Median, and Minimum) considering the statistical tests and the 3 presented rankings (Tables 14–16). This was possible because PEETACO-DS has fundamental characteristics that are similar to those of PEETACO-DES versions. For example, specific classifiers are used to select the best regressor.

The results shown in Fig. 6 indicate that the performance achieved by heterogeneous combinations may be improved through dynamic ensemble selection using various criteria. The ranking Tables 14, 15, and 16 support this statement, since the PEETACO-DES versions could achieve the best rankings when compared to the heterogeneous ensembles. PEETACO-DS also achieved better results than those of the heterogeneous ensembles, and it was significantly outperformed them. Using classifiers to select regression models according to each data set improve the methods accuracy.

Fig. 7 shows a comparative evaluation considering the homogeneous combination methods and the proposed methods (PEETACO). The superiority of the proposed methods can be easily observed. The heterogeneous combinations and dynamic ensemble selection with different criteria justify the improvement achieved. The homogeneous combinations of the regression models did not show any significant progress in the context of SEE regarding the other models created together. In general, they were better than their respective individual models, as shown in the ranking Tables 14–16. Even though the homogeneous combinations made a small progress regarding the individual models, they for the most part outperform their base models, considering other contexts.

Fig. 8 depicts the superiority of the proposed methods over competing dynamic selection methods (NUCCI, DCS-LA, DCS-LAW). We believe that the main reason for the improvement is the selection of ensembles using different criteria and also the knowledge provided by classifiers used in our method. NUCCI's approach is a method that has similar characteristics regarding the model proposed here, but there are fundamental differences between them that explain PEETACO superiority. Table 3 shows such differences. In the context of this research, the simple

Table 7
MAR values of the methods for each ISBSG database in the experiment.

	ISBSG all	Banking	Communication	Government	Insurance	Manufacturing	Service industry	Remain all
LMS	2537	2691	3274	3490	3050	2902	3675	2243
SVR	2341	2458	2751	3099	2738	2331	4616	2180
M5P	2970	2672	4132	3536	2941	2161	4569	2446
MEAN	2395	**2327**	2984	3025	2625	2231	4041	2055
MEDIAN	2328	2361	2823	2955	2754	2003	3921	1898
MEAN OF EXTREMES	2463	2328	3124	3451	2598	2364	4110	2198
MAXIMUM	2968	2955	4298	3797	3075	3027	5151	3076
MINIMUM	2553	2505	3036	3374	2900	2364	3788	1895
BAGGING LMS	2589	2421	3408	3582	3720	2294	4208	2088
BAGGING SVR	2328	2696	2839	3128	2780	2271	4228	1991
BAGGING M5P	2682	2687	3023	3040	2580	2026	4824	2072
BOOSTING LMS	2544	2689	3283	3273	3026	2815	3761	2253
BOOSTING SVR	2329	2534	2807	**2903**	2748	2760	4615	2227
BOOSTING M5P	3117	2866	4208	3647	2954	2170	4792	2491
STACKING	2343	2481	2880	3333	2654	2105	4037	1988
NUCCI	2399	2374	2777	3372	2634	2244	4948	2051
DCS LA	2375	2556	2804	2908	2752	2516	4603	2181
DCS LAW	2371	2552	2804	2926	2741	2516	4603	2174
KNORA E	2332	2625	2855	2993	2645	2469	4270	2053
KNORA U	2333	2497	2799	2908	2603	2432	4257	2025
PEETACO-DS	2332	2429	2660	3027	2628	1980	3614	1935
PEETACO-DES MEAN	**2269**	2404	2664	2973	2548	2031	3515	1879
PEETACO-DES MEDIAN	2276	2420	**2620**	3052	2627	2016	3512	1883
PEETACO-DES MEAN EX.	2315	2409	2758	2972	**2529**	2091	3531	1943
PEETACO-DES MAXIMUM	2565	2540	2915	3154	2647	2229	4355	2293
PEETACO-DES MINIMUM	2328	2417	2780	2975	2617	2174	**3172**	**1837**

Table 8
MAR values of the methods for each PROMISE database in the experiment.

	Cocomonasa V2	Desharnais	China	Cocomonasa V1	Cocomo 81	Maxwell	kitchenham	Myazaki
LMS	357	1821	122	533	1699	3475	**1099**	39
SVR	684	2005	249	497	1603	4079	1247	47
M5P	416	2147	147	477	930	3763	1274	62
MEAN	402	1902	157	451	1328	3405	1164	45
MEDIAN	373	1926	127	436	1432	3564	1187	41
MEAN OF EXTREMES	433	1907	174	464	1287	3349	1160	47
MAXIMUM	651	2152	235	537	1279	4109	1176	58
MINIMUM	433	1894	155	533	1521	3644	1258	49
BAGGING LMS	333	1921	**121**	543	1604	3596	1335	41
BAGGING SVR	709	2013	172	575	1637	3497	1189	43
BAGGING M5P	**323**	2134	**121**	475	951	3381	1165	53
BOOSTING LMS	356	2049	122	533	1659	3621	1108	**35**
BOOSTING SVR	1003	2090	253	485	1660	3899	1227	48
BOOSTING M5P	373	2118	145	489	**868**	3884	1295	52
STACKING	358	2035	130	469	944	3591	1494	46
NUCCI	563	1869	188	386	919	4274	1138	42
DCS LA	624	1886	226	387	1089	4097	1127	41
DCS LAW	623	1837	228	386	1062	4194	1300	41
KNORA E	561	1843	204	434	1062	3485	1130	38
KNORA U	444	1887	174	448	1182	3685	**1100**	45
PEETACO-DS	459	1813	154	**377**	919	3279	1176	40
PEETACO-DES MEAN	422	1785	145	379	888	3311	1146	39
PEETACO-DES MEDIAN	441	1783	153	**377**	886	3402	1157	40
PEETACO-DES MEAN EX.	419	1795	139	393	906	3275	1140	39
PEETACO-DES MAXIMUM	552	1912	149	396	949	**3269**	1165	39
PEETACO-DES MINIMUM	404	1763	142	419	931	3644	1161	40

Table 9
Number of competing methods outperformed by each method in the experiment showed in Fig. 5.

Methods	Count winner
PEETACO-DES Mean	7
PEETACO-DES Mean of Extremes	6
PEETACO-DS, PEETACO-DES Median, and PEETACO-DES Minimum PEETACO-DES Maximum	3
LMS, SVR, and M5P	0

dynamic selection methods found in the state of the art could not improve the accuracy achieved by the heterogeneous combination. In contrast, the version of the proposed model (PEETACO-DS) is the only one that shows better accuracy than the static ensembles, as shown in Tables 14, 15, and 16. Therefore, we believe that it is essential to use classifiers to carry out dynamic selection of the heterogeneous models using a validation data set to avoid overfitting and improve the accuracy. The method proposed by Di Nucci et al. (2017) uses just one classifier (Random Forest), the same for all data sets, and therefore it does not use the

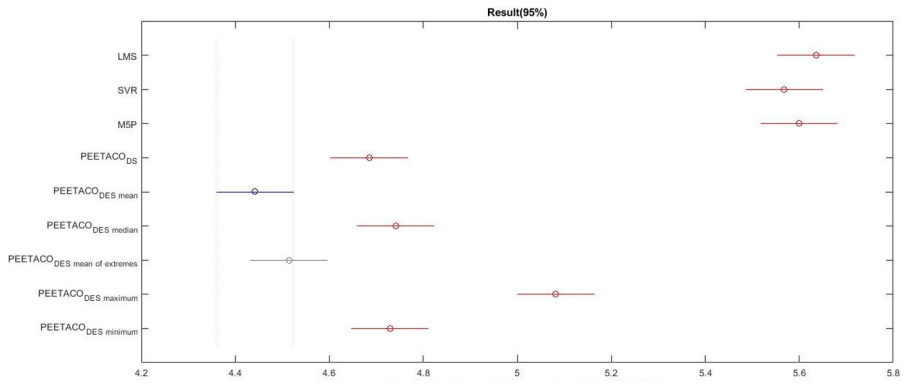

Fig. 5. Statistical results of the experiment using the Friedman test and LSD for comparison of (i) the proposed models (PEETACO), (ii) the best three individual models.

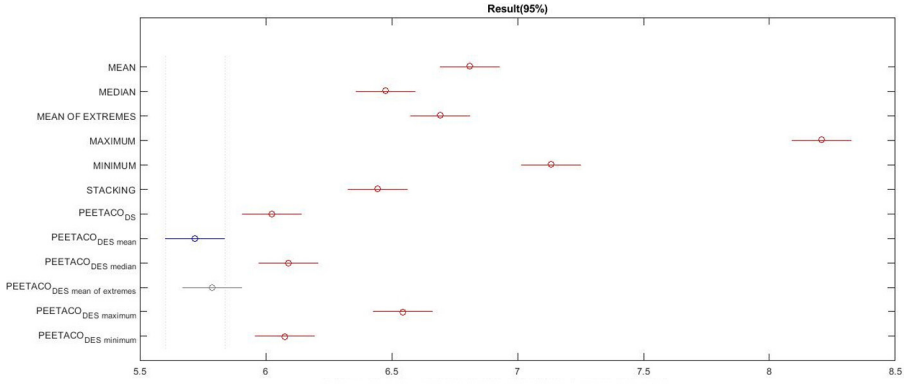

Fig. 6. Statistical results of the experiment using the Friedman test and LSD for comparison of (i) the proposed models (PEETACO), (ii) the heterogeneous combinations of the models.

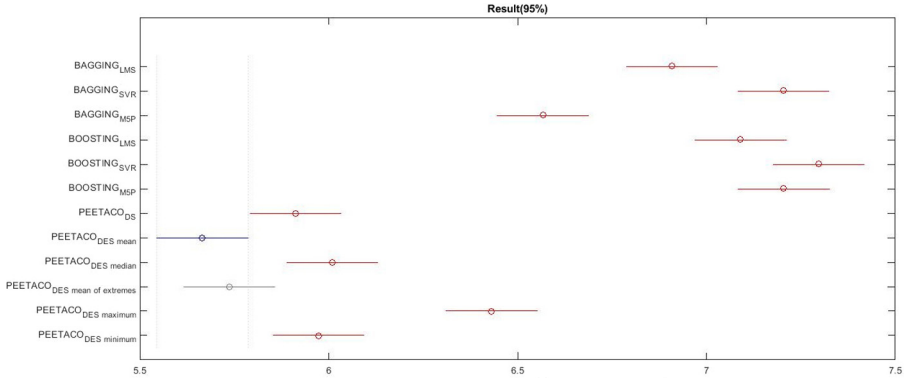

Fig. 7. Statistical results of the experiment using the Friedman test and LSD for comparison of (i) the proposed models (PEETACO), (ii) the homogeneous combinations of the models.

Table 10
Number of competing methods outperformed by each method in the experiment showed in Fig. 6.

Methods	Count winner
PEETACO-DES Mean	10
PEETACO-DES Mean of Extremes	9
PEETACO-DS, PEETACO-DES Minimum, and PEETACO-DES Median	7
STACKING, MEDIAN, and PEETACO-DES Maximum	3
MEAN OF EXTREMES and MEAN	2
MINIMUM	1
MAXIMUM	0

validation data set to select the suitable classifier for a given database.

Finally, Fig. 9 presents the superiority of the proposed methods over the DES methods. The KNORA selects many models, but using only one criterion. Besides that, KNORA is an original method

Table 11
Number of competing methods outperformed by each method in the experiment showed in Fig. 7.

Methods	Count winner
PEETACO-DES Mean	10
PEETACO-DES Mean of Extremes	8
PEETACO-DS, PEETACO-DES Median, and PEETACO-DES Minimum	7
PEETACO-DES Maximum and BAGGING M5P	5
BAGGING LMS	3
BOOSTING LMS, BAGGING SVR, BOOSTING SVR, and BOOSTING M5P	0

Table 12
Number of competing methods outperformed by each method in the experiment showed in Fig. 8.

Methods	Count winner
PEETACO-DES Mean and PEETACO-DES Mean of Extremes	7
PEETACO-DS, PEETACO-DES Minimum, and PEETACO-DES Median	4
NUCCI	1
DCS-LAW, DCS-LA, and PEETACO-DES Maximum	0

Table 13
Number of competing methods outperformed by each method in the experiment showed in Fig. 9.

Methods	Count winner
PEETACO-DES Mean and PEETACO-DES Mean of Extremes	6
PEETACO-DS, PEETACO-DES Minimum, and PEETACO-DES Median	3
KNORA-E, KNORA-U, and PEETACO-DES Maximum	0

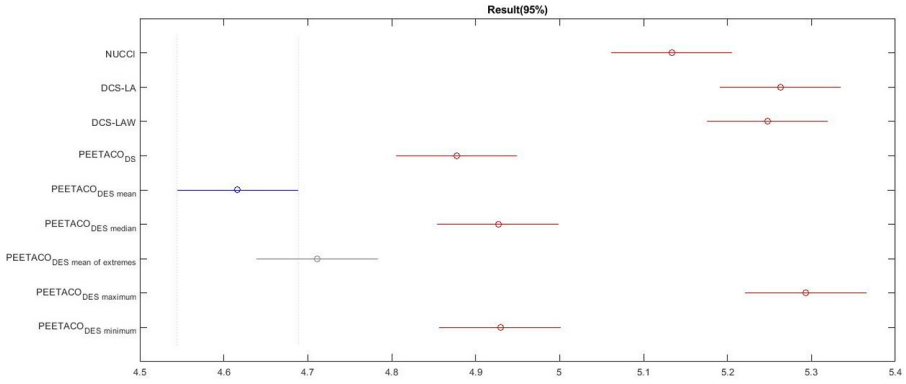

Fig. 8. Statistical result of the experiment using the Friedman test and LSD for comparison of (i) the proposed models (PEETACO), (ii) the competing dynamic selection models.

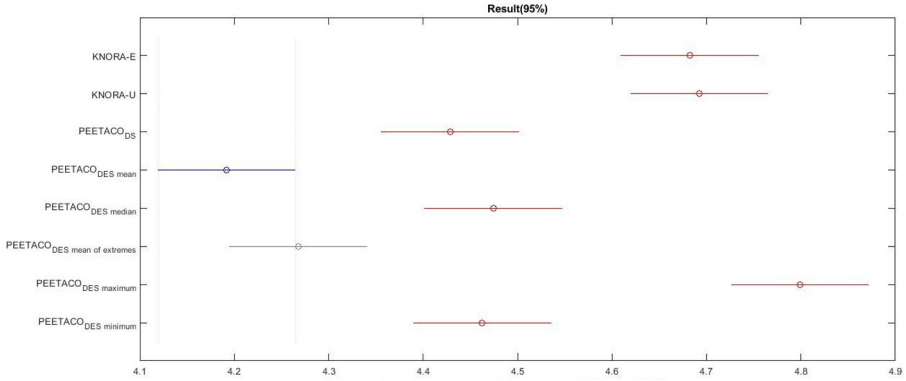

Fig. 9. Statistical results of the experiment using the Friedman test and LSD for comparison of (i) the proposed models (PEETACO), (ii) the dynamic ensemble selection models.

for selecting classifiers, not regressors, as explained in Section 3. If we consider the 3 presented rankings shown in Tables 14, 15, and 16, we can determine that the 2 versions of KNORA are among the 10 best methods analyzed in this paper. We believe that the results achieved using KNORA is due to its capacity of combining individual methods. However, the method could not surpass some heterogeneous combinations, probably because a single dynamic selection criterion was used for all data sets.

Based on these results, it can be noticed that the proposed methods can outperform all competing models. The PEETACO (DES-Mean, DES-Mean of Extremes, DS, DES-Median and DES-

Table 14
Ranking by positions of the experiment using the mean absolute residuals.

Method	N	Mean of ranks
PEETACO-DES Mean	16	4.75
PEETACO-DES Mean of Extremes	16	5.25
PEETACO-DES Median	16	6.06
PEETACO-DES Minimum	16	7.31
PEETACO-DS	16	7.50
MEDIAN	16	9.81
MEAN	16	11.94
KNORA U	16	11.94
KNORA E	16	12.38
STACKING	16	13.00
NUCCI	16	13.13
BAGGING M5P	16	13.50
PEETACO-DES Maximum	16	13.50
MEAN OF EXTREMES	16	13.69
DCS-LA	16	14.38
LMS	16	14.69
DCS-LAW	16	15.13
BOOSTING LMS	16	15.50
MINIMUM	16	16.63
BAGGING LMS	16	16.63
BAGGING SVR	16	16.75
SVR	16	17.38
BOOSTING SVR	16	18.25
M5P	16	18.75
BOOSTING M5P	16	19.00
MAXIMUM	16	23.94

Table 15
Ranking by positions of the experiment using the mean of the absolute residuals logarithmic.

Method	N	Mean of ranks
PEETACO-DS	16	6.50
PEETACO-DES Mean	16	6.63
PEETACO-DES Minimum	16	7.06
PEETACO-DES Mean of Extremes	16	7.44
PEETACO-DES Median	16	7.88
DCS-LAW	16	10.63
MEDIAN	16	10.81
DCS-LA	16	10.94
MINIMUM	16	11.50
NUCCI	16	12.00
BAGGING LMS	16	12.31
BOOSTING LMS	16	12.56
KNORA E	16	13.06
LMS	16	13.44
PEETACO-DES Maximum	16	13.88
KNORA U	16	14.13
STACKING	16	14.88
MEAN	16	14.94
BOOSTING SVR	16	15.44
SVR	16	16.25
MEAN OF EXTREMES	16	15.31
BAGGING M5P	16	17.75
BAGGING SVR	16	18.50
M5P	16	20.13
BOOSTING M5P	16	21.19
MAXIMUM	16	24.75

Minimum) outperformed the individual models, their homogeneous combinations, their heterogeneous combinations, the single dynamic selection methods, and the DES methods.

PEETACO-DES (Maximum) surpass the individual models, some heterogeneous combinations, and many homogeneous combinations, but it achieved similar results to the rest competitive methods, besides that the static ensemble (Maximum) was the worst method evaluated in this paper, that lead us to say that choosing the maximum value it is not good alternative.

We believe that the following factors are the main reasons for the good results obtained by our proposed methods: early evaluated performance on a validation set to select of the best individual models; early evaluated performance to select the best ensemble (basic ensemble); early evaluated performance of classifiers to select regression models according to each data set; commonly used combination methods (mean, median); dynamic ensemble selection; and adoption of different criteria in order to dynamically select the best ensemble.

Table 16
Ranking by positions of the experiment using the mean relative error adjusted.

Method	N	Mean of ranks
PEETACO-DES Mean	16	3.19
PEETACO-DES Mean of Extremes	16	3.31
PEETACO-DES Median	16	5.19
PEETACO-DES Maximum	16	5.38
PEETACO-DS	16	5.63
NUCCI	16	6.63
PEETACO-DES Minimum	16	9.19
MEDIAN	16	9.81
MEAN	16	11.94
KNORA U	16	12.19
KNORA E	16	12.56
BOOSTING LMS	16	13.31
MEAN OF EXTREMES	16	14.56
BAGGING LMS	16	14.75
STACKING	16	15.06
DCS-LAW	16	16.25
LMS	16	16.38
BAGGING M5P	16	16.81
DCS-LA	16	17.31
SVR	16	19.06
BOOSTING SVR	16	19.13
M5P	16	19.25
BOOSTING M5P	16	19.88
BAGGING SVR	16	20.50
MINIMUM	16	21.19
MAXIMUM	16	22.13

In order to evaluate the performance of the proposed model with more classifiers than the quantity used in the experiments so far, we carried out a comparative experiment between the different versions of the PEETACO model. The versions are identified by the quantity of the classifiers used with the purpose of selecting the regressors. This value varied from 1 to 5, and the results are shown in Fig. 10. We used the mean as a combination method for the output of the selected regressors, since the mean had the best performance among the other combination methods. The results show that the performance differences of the dynamic ensemble selection models are not significant. The PEETACO2, PEETACO3, PEETACO4 and PEETACO5 models that used 2, 3, 4, and 5 classifiers, respectively, were statistically equal. Nevertheless, they outperformed the PEETACO model, which uses only 1 classifier that selects 1 regressor. This results ascertain the superiority of the ensembles in relation to the individual models. In the present work we used the PEETACO model with 3 classifiers, since this comparison has shown that its performance is statistically equivalent to the model with 5 classifiers, and it requires less processing time.

It is also possible to ask: Why were the LMS, SVR and M5P models chosen? In the pruning phase, Fig. 2, where 3 algorithms were selected, we used the accuracy and diversity of the models to build basic ensemble. The accuracy was assessed by statistical tests and the diversity was induced by the use of heterogeneous algorithms. Different ensembles built from regression models are shown in Fig. 3. We used the median combination method to obtain the results, which showed that the best three models (LMS, SVR and M5P) achieved higher accuracy in the validation phase than other static combinations.

At the beginning this article, we proposed three research questions:

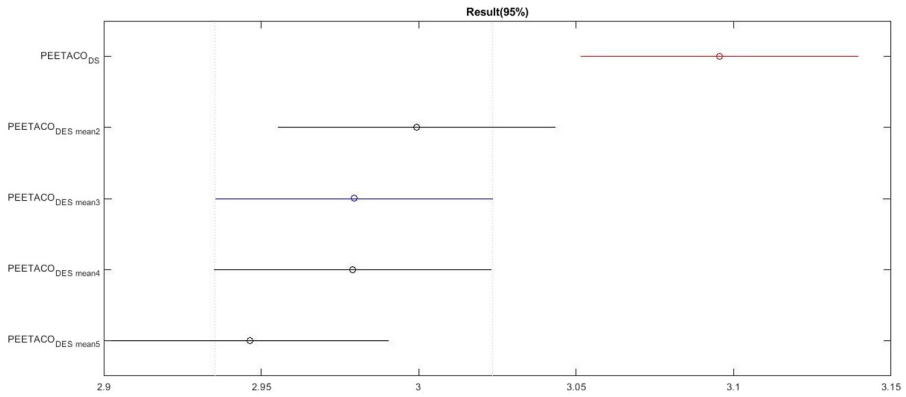

Fig. 10. Statistical results of the experiment using the Friedman test and LSD for comparison among versions of the proposed models (PEETACO with 1, 2, 3, 4 and 5 classifiers).

(i) can dynamic selection improve the accuracy of ensemble methods in SEE?

(ii) can dynamic selection using classifiers to select regressors from a set of heterogeneous models improve the accuracy of the effort prediction?

(iii) which criteria of dynamic selection of regression models show better results in the effort estimation?

In short, we can say that, in general:

(i) dynamic selection has a strong tendency to improve the accuracy of ensemble predictions in effort estimation projects when it is used classifiers to dynamically select. We noticed that traditional dynamic selection methods (DCS-LA, KNORA, DCS-LAW) did not outperform static ensemble selection methods;

(ii) the use of classifiers to dynamically select regression in the data set of effort estimation has led us to achieve significant advances, outperforming even other methods of dynamic and static ensemble selection. We believe that using different algorithms for achieving dynamic selection, such as the ones based on distance, trees and rules, and other strategies in conjunction with the combination of methods and anticipated validation for each data set are key to successful results;

(iii) the proposed model used different classifiers to select the regression models. We used classifiers with different natures based on: functions approximation, Bayesian methods, distance, rules, and decision trees. The quest for the best classifiers was an arduous task (exhaustive search), but these classifiers can improve the accuracy of the dynamic selection. The diversity of selection criteria was fundamental to achieve significant results. Therefore, we identified that LWL, KSTAR and KNN7 algorithms were the most stable classifiers among all the evaluated models. This indicated that local accuracy was the best criterion to select heterogeneous models in SEE.

6. Conclusion

This paper introduced a novel dynamic ensemble selection scheme of heterogeneous models for software effort estimation problems, including a validation phase for selecting the best regressor and classifiers to be used in the model for selecting the regressors. The suggested method construction is sound and the superiority of the proposed method has been confirmed by applying the Friedman's Ranking and LSD post hoc statistical tests to sixteen data sets with 95% confidence. In the process, fifteen regression models have been used, from which the three with the best results were selected. The dynamic selection has been built taken these regressors into account. Additionally, we have used 23 distinct classifiers to select the best regressors for an unknown instance. With respect to the database used, the effectiveness of these classifiers have varied, but overall, LWL, KSTAR and KNN7 have been the most stable selectors in the experiments, these models are based on distance, which leads us to infer that this criterion may be the most stable one among the other selection models.

This research proposes a unique dynamic selection method, which outperforms the currently listed state of the art procedures. The PEETACO-DES proposed method achieves better results than other methods previously presented in the literature. Moreover, the results indicate that the proposed DES methods (Mean and Mean of Extremes) are, in general, notably superior to the proposed PEETACO-DS method.

In order to ensure quality in the present results, this paper presents: (i) clear and well defined objectives, (ii) six solutions presented and discussed in Section 3, (iii) application of the methods evaluated in 16 different databases (iv) accuracy measured in three different types of metrics, (v) comparison of the proposed methods against the individual models and their homogeneous and heterogeneous combinations, besides others dynamic selection, and dynamic ensemble selection using a set of databases widely used for SEE.

One of the suggestions for future works it the investigation of the proposed method applied to other data sets of software effort estimation, including the problems that enable the use of cross project. ML models are promising; In this sense, companies can benefit from this proposed model. From their own historical data, it is possible to find the basic ensemble and apply the DES proposed in this paper, however, ML implementation is still limited in the industry. Another approach that has been used to reach more precise results in the ML is the selection of attributes (feature selection), which tends to improve the accuracy of the individual models and, therefore, the precision of the combined models. In addition to selecting the best features, it is also possible to select the parameters of the classifiers used in our model to improve performance and the choice of classifiers that could act as a selector of regression models in a heterogeneous ensemble. Furthermore, the use of alternative strategies capable of combining the regressors in different ways from the proposed method could lead to more accurate.

CRediT authorship contribution statement

Jose Thiago H. de A. Cabral: Conceptualization, Methodology, Software, Investigation, Writing - original draft. **Adriano L.I. Oliveira:** Writing - review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank CNPq, grant 307892/2018-2, and FACEPE (Brazilian Research Agencies) for their financial support.

References

de A. Araujo, R., Oliveira, A.L., Soares, S., Meira, S., 2012a. An evolutionary morphological approach for software development cost estimation. Neural Netw. 32, 285–291.

de A. Araujo, R., Soares, S., Oliveira, A.L., 2012b. Hybrid morphological methodology for software development cost estimation. Expert Syst. Appl. 39, 6129–6139.

de A. Cabral, J.T.H., de A. Araujo, R., Nobrega, J.P., Oliveira, A.L.I., 2017. Heterogeneous ensemble dynamic selection for software development effort estimation. In: 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). pp. 210–217. http://dx.doi.org/10.1109/ICTAI.2017.00042.

Abran, A., 2015. Data collection and industry standards: The isbsg repository. In: Software Project Estimation. John Wiley and Sons, Inc., pp. 161–184.

Argawal, R., Kumar, Y., Mallick, Y., Bharadwaj, R., Anantwar, D., 2001. Estimating software projects. Softw. Eng. 26, 60–67.

Bardsiri, V.K., Jawawi, D.N.A., Bardsiri, A.K., Khatibi, E., 2013. Lmes: A localized multi-estimator model to estimate software development effort. Eng. Appl. Artif. Intell. 26, 2624–2640.

Braga, P.L., Oliveira, A.L.I., Ribeiro, G.H.T., Meira, S.R.L., 2007. Bagging predictors for estimation of software project effort. In: 2007 International Joint Conference on Neural Networks. pp. 1595–1600.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24, 123–140.

Briand, L.C., Emam, K.E., Surmann, D., Wieczorek, I., Maxwell, K.D., 1999. An assessment and comparison of common software cost estimation modeling techniques. In: Proceedings of the 1999 International Conference on Software Engineering (IEEE Cat. No.99CB37002). pp. 313–323. http://dx.doi.org/10.1145/302405.302647.

Britto, A.S., Sabourin, R., Oliveira, L.E., 2014. Dynamic selection of classifiers—a comprehensive review. Pattern Recognit. 47, 3665–3680.

Brown, G., Wyatt, J., Harris, R., Yao, X., 2005a. Diversity creation methods: a survey and categorisation. Inf. Fusion 6, 5–20, Diversity in Multiple Classifier Systems.

Brown, G., Wyatt, J.L., Tiňo, P., 2005b. Managing diversity in regression ensembles. J. Mach. Learn. Res. 6, 1621–1650.

Cavalin, P.R., Sabourin, R., Suen, C.Y., 2013. Dynamic selection approaches for multiple classifier systems. Neural Comput. Appl. 22, 673–688.

Charette, R.N., 2005. Why software fails [software failure]. IEEE Spectr. 42, 42–49. http://dx.doi.org/10.1109/MSPEC.2005.1502528.

Corazza, A., Di Martino, S., Ferrucci, F., Gravino, C., Mendes, E., 2011. Investigating the use of support vector regression for web effort estimation. Empir. Softw. Eng. 16, 211–243.

Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C., 2014. Analyzing dynamic ensemble selection techniques using dissimilarity analysis. In: El Gayar, N., Schwenker, F., Suen, C. (Eds.), Artificial Neural Networks in Pattern Recognition. Springer International Publishing, Cham, pp. 59–70.

Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C., 2015. Meta-des.h: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach. In: 2015 International Joint Conference on Neural Networks (IJCNN). pp. 1–8.

Cruz, R.M., Sabourin, R., Cavalcanti, G.D., 2017. Meta-des.oracle: Meta-learning and feature selection for dynamic ensemble selection. Inf. Fusion 38, 84–103.

Cruz, R.M., Sabourin, R., Cavalcanti, G.D., 2018. Dynamic classifier selection: Recent advances and perspectives. Inf. Fusion 41, 195–216.

Demšar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Di Nucci, D., Palomba, F., Oliveto, R., De Lucia, A., 2017. Dynamic selection of classifiers in bug prediction: An adaptive method. IEEE Trans. Emerg. Top. Comput. Intell. 1, 202–212.

Dietterich, T.G., 2000. Ensemble methods in machine learning. In: Proceedings of the First International Workshop on Multiple Classifier Systems MCS '00. Springer-Verlag, London, UK, UK, pp. 1–15.

Dragicevic, S., Celar, S., Turic, M., 2017. Bayesian network model for task effort estimation in agile software development. J. Syst. Softw. 127, 109–119. http://dx.doi.org/10.1016/j.jss.2017.01.027, URL: http://www.sciencedirect.com/science/article/pii/S0164121217300171.

Friedman, M., 1940. A comparison of alternative tests of significance for the problem of m rankings. Ann. Math. Stat. 11, 86–92.

Fukunaga, K., Hummels, D.M., 1989. Leave-one-out procedures for nonparametric error estimates. IEEE Trans. Pattern Anal. Mach. Intell. 11, 421–423.

Gencel, C., Buglione, L., 2008. Do base functional component types affect the relationship between software functional size and effort?. In: Cuadrado-Gallego, J.J., Braungarten, R., Dumke, R.R., Abran, A. (Eds.), Software Process and Product Measurement. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 72–85.

Gray, A.R., MacDonell, S.G., 1997. A comparison of techniques for developing predictive models of software metrics. Inf. Softw. Technol. 39, 425–437. http://dx.doi.org/10.1016/S0950-5849(96)00006-7, URL: http://www.sciencedirect.com/science/article/pii/S0950584996000067.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20, 832–844. http://dx.doi.org/10.1109/34.709601.

Hollander, M., Wolfe, D.A., Chicken, E., 2013. Nonparametric Statistical Methods, third ed. Wiley.

Hosni, M., Idri, A., Nassif, A.B., Abran, A., 2016. Heterogeneous ensembles for software development effort estimation. In: 2016 3rd International Conference on Soft Computing Machine Intelligence (ISCMI). pp. 174–178. http://dx.doi.org/10.1109/ISCMI.2016.15.

Idri, A., Hosni, M., Abran, A., 2016a. Systematic literature review of ensemble effort estimation. J. Syst. Softw. 118, 151–175. http://dx.doi.org/10.1016/j.jss.2016.05.016, URL: http://www.sciencedirect.com/science/article/pii/S0164121216300450.

Idri, A., Hosni, M., Abran, A., 2016b. Systematic mapping study of ensemble effort estimation. In: ENASE.

Jorgensen, M., Shepperd, M., 2007. A systematic review of software development cost estimation studies. IEEE Trans. Softw. Eng. 33, 33–53. http://dx.doi.org/10.1109/TSE.2007.256943.

Keselman, H., Keselman, J., Games, A., 1991. Maximum familywise type i error rate: The least significant difference. In: Newman-Keuls. http://dx.doi.org/10.1037/0033-2909.110.1.155.

Kittler, J., Hatef, M., Duin, R.P.W., Matas, J., 1998. On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20, 226–239.

Ko, A.H., Sabourin, R., Alceu Souza Britto, J., 2008a. From dynamic classifier selection to dynamic ensemble selection. Pattern Recognit. 41, 1718–1731.

Ko, A.H.R., Sabourin, R., Britto, Jr., A.S., 2008b. From dynamic classifier selection to dynamic ensemble selection. Pattern Recognit. 41, 1718–1731.

Ko, A.H.-R., Sabourin, R., de Souza Britto, A., 2007. K-nearest oracle for dynamic ensemble selection. In: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007), Volume 1. pp. 422–426. http://dx.doi.org/10.1109/ICDAR.2007.4378744.

Kocaguneli, E., Menzies, T., Keung, J.W., 2012. On the value of ensemble effort estimation. IEEE Trans. Softw. Eng. 38, 1403–1416.

Kultur, Y., Turhan, B., Bener, A., 2009. Ensemble of neural networks with associative memory (enna) for estimating software development costs. Know.-Based Syst. 22, 395–402.

López-Martín, C., Abran, A., 2015. Neural networks for predicting the duration of new software projects. J. Syst. Softw. 101, 127–135. http://dx.doi.org/10.1016/j.jss.2014.12.002, URL: http://www.sciencedirect.com/science/article/pii/S0164121214002805.

Mendes, E., Watson, I., Triggs, C., Mosley, N., Counsell, S., 2003. A comparative study of cost estimation models for web hypermedia applications. Empir. Softw. Eng. 8, 163–196. http://dx.doi.org/10.1023/A:1023062629183.

Mendes-Moreira, J.a., Soares, C., Jorge, A.M., Sousa, J.F.D., 2012. Ensemble approaches for regression: A survey. ACM Comput. Surv. 45, 10:1–10:40.

Minku, L.L., Yao, X., 2013. Ensembles and locality: Insight on improving software effort estimation. Inf. Softw. Technol. 55, 1512–1528.

Oliveira, A.L., 2006. Estimation of software project effort with support vector regression. Neurocomputing 69, 1749–1753.

Oliveira, A.L., Braga, P.L., Lima, R.M., Cornélio, M.L., 2010. Ga-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Inf. Softw. Technol. 52, 1155–1166.

Pospieszny, P., Czarnacka-Chrobot, B., Kobylinski, A., 2018. An effective approach for software project effort and duration estimation with machine learning algorithms. J. Syst. Softw. 137, 184–196. http://dx.doi.org/10.1016/j.jss.2017.11.066, URL: http://www.sciencedirect.com/science/article/pii/S0164121217302947.

Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. Ann. Statist. 26, 1651–1686. http://dx.doi.org/10.1214/aos/1024691352.

Seni, G., Elder, J.F., 2010. Ensemble methods in data mining: Improving accuracy through combining predictions. Morgan and claypool publishers. http://dx.doi.org/10.1109/MSPEC.2005.1502528, URL: https://www.morganclaypool.com/doi/abs/10.2200/S00240ED1V01Y200912DMK002.

Shepperd, M., Kadoda, G., 2001. Comparing software prediction techniques using simulation. IEEE Trans. Softw. Eng. 27, 1014–1022. http://dx.doi.org/10.1109/32.965341.

Shepperd, M., MacDonell, S., 2012. Evaluating prediction systems in software project estimation. Inf. Softw. Technol. 54, 820–827. http://dx.doi.org/10.1016/j.infsof.2011.12.008, URL: http://www.sciencedirect.com/science/article/pii/S095058491200002X. Special Issue: Voice of the Editorial Board.

Shepperd, M., Schofield, C., 1997. Estimating software project effort using analogies. IEEE Trans. Softw. Eng. 23, 736–743. http://dx.doi.org/10.1109/32.637387.

Shepperd, M., Song, Q., Sun, Z., Mair, C., 2013. Data quality: Some comments on the nasa software defect datasets. IEEE Trans. Softw. Eng. 39, 1208–1215. http://dx.doi.org/10.1109/TSE.2013.11.

Shirabad, J.S., Menzies, T.J., 2005. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada.

Shunmugapriya, P., Kanmani, S., 2013. Optimization of stacking ensemble configurations through artificial bee colony algorithm. Swarm Evol. Comput. 12, 24–32.

Stensrud, E., 2001. Alternative approaches to effort prediction of erp projects. Inf. Softw. Technol. 43, 413–423. http://dx.doi.org/10.1016/S0950-5849(01)00147-1, URL: http://www.sciencedirect.com/science/article/pii/S0950584901001471.

Twala, B., Verner, J., 2016. Toward accurate software effort prediction using multiple classifier systems. Comput. Intell. Quant. Softw. Eng. 13, 5–151.

Usman, M., Mendes, E., Weidt, F., Britto, R., 2014. Effort estimation in agile software development: A systematic literature review. In: Proceedings of the 10th International Conference on Predictive Models in Software Engineering PROMISE '14. ACM, New York, NY, USA, pp. 82–91.

Walkerden, F., Jeffery, R., 1999. An empirical study of analogy-based software effort estimation. Empir. Softw. Eng. 4, 135–158. http://dx.doi.org/10.1023/A:1009872202035.

Webb, G.I., Zheng, Z., 2004. Multistrategy ensemble learning: reducing error by combining ensemble learning techniques. IEEE Trans. Knowl. Data Eng. 16, 980–991.

Wen, J., Li, S., Lin, Z., Hu, Y., Huang, C., 2012. Systematic literature review of machine learning based software development effort estimation models. Inf. Softw. Technol. 54, 41–59.

Wolpert, D.H., 1992. Stacked generalization. Neural Netw. 5, 241–259. http://dx.doi.org/10.1016/S0893-6080(05)80023-1, URL: http://www.sciencedirect.com/science/article/pii/S0893608005800231.

Woods, K., Kegelmeyer, W.P., Bowyer, K., 1997. Combination of multiple classifiers using local accuracy estimates. IEEE Trans. Pattern Anal. Mach. Intell. 19, 405–410.

Jose Thiago H. de A. Cabral received his B.S. in Computer Science from Federal University of Paraiba in 2007 and the M.Sc. degree in Computer Science from Federal University of Pernambuco, Recife, Brazil in 2012. Currently, he is a Ph.D. candidate in Computer Science at Federal University of Pernambuco. His research interests include Software Effort Estimation, Data Mining and Computacional Intelligence.

Adriano L.I. Oliveira obtained his B.Sc. degree in Electrical Engineering and M.Sc. and Ph.D. degrees in Computer Science from the Federal University of Pernambuco, Brazil, in 1993, 1997 and 2004, respectively. In 2011 he joined the Center for Informatics at Federal University of Pernambuco as an Associate Professor. He was a Visiting Professor at École de Technologie Supérieure (ÉTS, Université du Québec, Montréal, Canadá) from 2018 to 2019. He was an Assistant Professor at Federal Rural University of Pernambuco from 2009 to 2011 and at the Department of Computing Systems of Pernambuco State University from 2002 to 2009. He has published over 140 articles in scientific journals and conferences and one book. He is a Senior Member of the IEEE. His current research interests include deep neural networks, machine learning, pattern recognition, data mining, and applications of these techniques to time series analysis and forecasting, information systems, software engineering, and biomedicine.