Information and Software Technology 55 (2013) 1512–1528

Contents lists available at SciVerse ScienceDirect

Information and Software Technology

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

Ensembles and locality: Insight on improving software effort estimation

Leandro L. Minku *, Xin Yao

CERCIA, School of Computer Science, The University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

ARTICLE INFO

Article history:
Available online 13 October 2012

Keywords:
Software effort estimation
Ensembles of learning machines
Locality
Empirical validation

ABSTRACT

Context: Ensembles of learning machines and locality are considered two important topics for the next research frontier on Software Effort Estimation (SEE).
Objectives: We aim at (1) evaluating whether existing automated ensembles of learning machines generally improve SEEs given by single learning machines and which of them would be more useful; (2) analysing the adequacy of different locality approaches; and getting insight on (3) how to improve SEE and (4) how to evaluate/choose machine learning (ML) models for SEE.
Method: A principled experimental framework is used for the analysis and to provide insights that are not based simply on intuition or speculation. A comprehensive experimental study of several automated ensembles, single learning machines and locality approaches, which present features potentially beneficial for SEE, is performed. Additionally, an analysis of feature selection and regression trees (RTs), and an investigation of two tailored forms of combining ensembles and locality are performed to provide further insight on improving SEE.
Results: Bagging ensembles of RTs show to perform well, being highly ranked in terms of performance across different data sets, being frequently among the best approaches for each data set and rarely performing considerably worse than the best approach for any data set. They are recommended over other learning machines should an organisation have no resources to perform experiments to chose a model. Even though RTs have been shown to be more reliable locality approaches, other approaches such as *k*-Means and *k*-Nearest Neighbours can also perform well, in particular for more heterogeneous data sets.
Conclusion: Combining the power of automated ensembles and locality can lead to competitive results in SEE. By analysing such approaches, we provide several insights that can be used by future research in the area.

1. Introduction

Estimating the cost of a software project is a task of strategic importance in project management. Both over and underestimations of cost can cause serious problems to a company. For instance, overestimations may result in a company loosing contracts or wasting resources, whereas underestimations may result in poor quality, delayed or unfinished software systems. The major contributing factor for software cost is effort [1]. So, models for estimating software cost/effort can be used as decision support tools, allowing investigation of the impact of certain requirements and development team features on the cost/effort of a project to be developed.

Several different software cost or software effort estimation (SEE) approaches have been proposed [2]. Among them, effort estimators based on machine learning (ML) approaches such as multi-layer perceptrons (MLPs), radial basis function (RBF) networks and regression trees (RTs) [3–9] have been receiving increased attention [2]. The motivation behind the use of such approaches is that they make no or minimal assumptions about the function being modelled and the data used for training. For instance, Tronto et al. [7] showed that MLPs improve SEE over conventional linear models because they are not restricted to linear functions, being able to model observations that lie far from the best straight line.

More recently, ensembles of learning machines have attracted attention of the SEE community for building SEE models [9,8,10,11]. However, existing work on automated[1] ensembles of learning machines for SEE presents contradictory conclusions regarding whether ensembles improve or not performance for SEE. Section 2.1 presents more details on these works. In the current work, we perform a principled and extensive analysis of existing automated ensembles of learning machines to determine whether they generally improve effort estimations given by single learning

* Corresponding author.
E-mail addresses: l.l.minku@cs.bham.ac.uk (L.L. Minku), x.yao@cs.bham.ac.uk (X. Yao).

[1] We refer to an approach as automated when, given the project data, it does not require human intervention and decision-making in order to be used. More details can be found in Section 2.1.

0950-5849
http://dx.doi.org/10.1016/j.infsof.2012.09.012

machines. We build upon previous work and improve on their weaknesses by following a principled framework and doing an extensive analysis.

The methodology used in our work has the following advantages in comparison to previous work using existing automated ensembles:

- Use of principled experimentation, considering both parameter choice, statistical tests and magnitude of improvements.
- Use of a more reliable non-asymmetric performance measure (Mean Absolute Error – MAE) and a measure that facilitates investigation of the magnitude of the differences in performance (Standardised Accuracy – SA), rather than using only measures based on the Magnitude of the Relative Error (MRE) such as Mean MRE (MMRE) and the Percentage of Estimates within N% of the actual values PRED(N).
- Comparison using three different ensembles of learning machines (Bagging [12], Random [13] and Negative Correlation Learning [14]) which present features potentially beneficial to SEE.
- Use of a larger number of data sets (thirteen against five, the highest number of data sets previously used in studies involving automated ensembles), including both PROMISE [15] and ISBSG [16] data sets, rather than just PROMISE data sets.
- Experimental analysis of the behaviour of promising approaches, gaining insight on how to improve SEE.

Another area of research considered as promising in software project estimation is locality [17]. Approaches that perform estimations considering mainly training examples that are similar to the project being estimated can be referred to as based on locality. As SEE data sets tend to be relatively small and very heterogeneous, such approaches are likely to be more adequate. Examples of works considering locality are Cuadrado Gallego et al. [18] and Kocaguneli et al. [19]. Section 2.2 explains locality further. Even though locality is a promising area, it is not yet clear what locality approach would be more adequate for SEE. For instance, RTs are promising due to the hierarchy of features that they create, but it is not known whether this simply provides the same benefit as other locality approaches or a feature selector. Our work investigates different locality approaches for SEE, providing insight for future approaches on improving SEE.

As an additional contribution, our paper builds upon previous work and proposes an experimental framework for evaluation of SEE approaches. The framework joins (1) the power of statistical tests for comparison of multiple learning machines over multiple data sets as recommended in the general ML literature [20], to (2) an analysis of the approaches among the best, and to (3) the use of a standardised measure proposed by Shepperd and Mc Donell [21] for evaluating prediction systems in software project estimation.

In short, our paper addresses the following research questions:

- RQ1: Do existing automated ensembles of learning machines generally improve effort estimations given by single learning machines, including potentially adequate locality learning machines such as RTs? Which of them would be more useful?
- RQ2: What locality approach is more adequate for SEE tasks? In particular, how well does RT locality do in comparison to other locality approaches? On what type of data sets?
- RQ3: What insight on how to further improve SEE can we gain by analysing competitive ensemble and locality approaches?
- RQ4: How to evaluate/choose ML models for SEE?

Our key contribution is not in a new algorithm, but a better understanding/insight. Furthermore, such better understanding/insight is based on experimental studies, not just an intuition or speculation. We show that combining the power of automated ensembles and locality can lead to competitive results in SEE. For instance, when considering the symmetric performance measure MAE, bagging ensembles of RTs perform well. They are highly ranked in terms of performance across different data sets, are frequently among the best approaches for each data set and rarely perform considerably worse than the best approach for any data set. So, they are recommended over other learning machines should an organisation have no resources to perform experiments to chose a model. Moreover, tailored approaches using ensembles at a higher level and locality at a lower level may be particular useful for improving performance on smaller data sets, whereas approaches using locality at a higher level may be particularly useful for improving on larger data sets. Future work on SEE may benefit from exploiting that further. In terms of locality approach, RTs have been shown to be more reliable than other approaches due to their ability to create hierarchies of features. Nevertheless, *k*-means and *k*-nearest neighbours can also perform well, in particular for more heterogeneous data sets.

The rest of this paper is organised as follows: Section 2 presents related work on ensembles, locality and evaluation of models. Section 3 describes the data sets used in our study. Section 4 explains the experimental framework, which represents part of the answer to RQ4. Section 5 presents the evaluation of existing automated ensembles against single learning machines. It mainly aims at answering RQ1, but also partly addresses RQ2 by considering a promising locality approach namely RTs. This section also gives some insights on how to improve SEE (RQ3) by revealing the success of an approach joining the power of locality and ensembles and by showing that bagging ensembles still have room for improvement. As the analysis singles out a comparatively well performing approach, Section 5 also complements the answer to RQ4. Section 6 performs an analysis of locality approaches, answering RQ2 and part of RQ3. Section 7 presents an analysis of RTs and tailored approaches joining the power of ensembles and locality, mainly addressing RQ3. Section 8 explains threats to validity. Section 9 presents conclusions and future work.

2. Related work

2.1. Ensembles of learning machines for SEE

Ensembles of learning machines are sets of learning machines[2] trained to perform the same task and combined with the aim of improving predictive performance [22]. It is commonly agreed that the base learning machines should behave differently from each other. Otherwise, the overall prediction will not be better than the individual predictions [23–25]. So, different ensemble learning approaches can be seen as different ways to generate diversity among the base learning machines.

Ensembles of learning machines have recently attracted the attention of the SEE community, as they can frequently improve performance over single learning machines. For example, Bootstrap Aggregating (bagging) [12], a well known ensemble approach with solid theoretical background, is able to turn weak learning machines into stronger ones. This can be particularly useful for SEE, as the data sets are usually small, leading to typically less accurate learning machines than in other applications of ML.

In this section, we briefly describe previous work on ensembles for SEE. The works presented by Braga et al. [9], Kultur et al. [8] and Kocaguneli et al. [10] represent the starting points of the research

[2] The learning machines used to compose an ensemble are frequently called base learning machines.

on ensembles for SEE. However, besides using only asymmetric performance measures based on MRE, they provide somewhat contradictory results as outlined below.

Braga et al. [9] claims that bagging improves the SEEs produced by several single learning machines, such as RTs and MLPs. However, the study uses only two versions of a single data set and neither tests the statistical significance of the results nor presents the standard deviations of the performance. As the average performances reported for the ensembles are very close to the performances obtained by the single learning machines, it is not possible to conclude that there was an improvement in the estimations [26].

Kultur et al. [8] uses five data sets and shows that an adapted version of bagging provides very large improvements in comparison to single learning machines. However, there is no information about how the parameters of the approaches were chosen. When performing ML experiments, the parameters choice can highly influence the results. Depending on the choice, a certain approach can become better or worse. So, even though Kultur et al.'s work is a significant contribution, the literature still lacks more evidence in support of ensembles.

Kocaguneli et al. [10] uses three data sets to compare the effect of combining different *types* of learning machines to the use of these learning machines separately. Even though this is not exactly a comparison between single learning machines and ensembles, their results suggest that ensembles do not improve the performance of single learning machines. This is somewhat contradictory in relation to Kultur et al. [8]'s work, which achieved very large improvements when using an adapted version of bagging. Kocaguneli et al. do not present information about the parameters choice either.

In addition to the problems explained above, none of these papers compare the results obtained by different existing automated ensemble learning approaches from the ML literature. Kocaguneli et al. [10] present results obtained by some ensembles of learning machines, but the analysis does not perform a statistical comparison among ensembles and single learning machines. Different ensemble approaches can be more or less adequate for SEE and should also be included in the comparisons. The papers do not provide analyses of the reasons for the results obtained either.

It is worth noting that even though some work in the SEE literature suggests that the performance of different models depends significantly on the characteristics of the data set [27], existing work on ensembles of learning machines suggests that they may provide generally better results than single learning machines in terms of MMRE and PRED(N) even when considering different data sets [8].

As mentioned in Section 1, we overcome the problems of existing works on ensembles by using principled experimentation; by using a more reliable symmetric performance measure; by considering the magnitude of the differences in performance; by comparing three different ensembles of learning machines which present different features potentially beneficial to SEE; by using a larger number of data sets (including both PROMISE and ISBSG data sets); and by performing an experimental analysis of the behaviour of promising approaches to gain more insight on how to improve SEE.

We refer to an approach as automated when, given the project data, it does not require human intervention and decision-making in order to be used. This is an algorithmic feature which reduces the complexity and cost of using the SEE tool, as a person operating it would just need to provide the project data and push a button to obtain a SEE. It is worth noting that parameter choice can usually be automated as well. The ensemble approaches discussed in the previous paragraphs [9,8,10] can be fully automated.

An approach that cannot be fully automated has higher tuning complexity and cost, such as manual/visual inspection of experiments to set/reconfigure the tool up. An example of such approach is Kocaguneli et al. [11]'s. Their ensemble of learning machines has very good performance in comparison to single learning machines. However, it requires a large amount of manual work, including an extensive experimentation procedure using several different types of "solo-methods" (combinations of single learning machines and preprocessing techniques) for creating the ensemble. It consists of selecting the "best" solo-methods in terms of losses and stability to compose the ensemble, by manually/visually checking and comparing their stability. The manual/visual checking process is needed because it is necessary not only to determine what solo-methods have the lowest number of losses, but also to check whether these are the same as the ones comparatively more stable and what level of stability should be considered as comparatively superior or not.

As a fully automated approach would be desirable, the area of ensembles for SEE still has room for improvement and can be considered as one of the promising topics of research in the area. In our current work, we investigate fully automated approaches and provide insight for future research on improving SEE in an automated way.

2.2. Locality for SEE

Approaches that perform estimations based mainly on training examples that are similar to the project to be estimated can be referred to as locality approaches. As SEE data sets tend to be relatively small and very heterogeneous, such approaches are likely to be more adequate and perform better than approaches which do not use locality.

An example of approach based on locality is presented by Cuadrado Gallego et al. [18]. They use a clustering approach based on Expectation Maximization (EM) to separate training examples into different sets and construct a regression equation to model each of them separately. This approach manages to achieve improvements in terms of MMRE and PRED(30) over regression equations created using the whole training set together. Another study by Kocaguneli et al. [19] considers locality by using modified RTs. They report improvements over traditional estimation by analogy based on Euclidean distance nearest neighbours, which is another form of locality approach.

There exist several types of RTs and they present many features potentially useful for SEE even when no tailored modification is introduced. RTs can be seen as rules to separate examples based on their feature values. Each leaf node represents a subset of the training examples used to create the tree. In order to make an estimation, the leaf node most similar to the test example in terms of input feature values is determined. So, RTs are based on locality. An example of RT is shown in Fig. 1. As we can see, its rules are easily readable by practitioners.

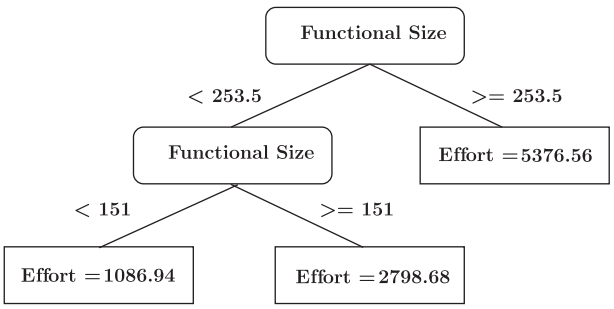

Fig. 1. An example of RT for software effort estimation.

RTs are created by considering not only the existing input features of the training examples, but also the impact of the input feature values on the dependent variable. This is a potential advantage over clustering approaches, which typically separate data according to their input features only. Moreover, the locality approach of determining the leaf node most similar to the example being estimated may operate in such a way to give different importance for different features, based on the levels of the tree in which they are split. For example, if the most important feature for determining the effort is functional size, this feature would be used for the highest level split of the tree. Less important features would be used in lower level splits or even not used at all, as it happened in Fig. 1. This hierarchy of features can be particularly useful for SEE, as data sets frequently have many (more and less relevant) features and few examples.

Nevertheless, it is not clear whether the separation of features provided by the RTs is important only as a correlation-based feature selector or whether the relative importance of features is essential for SEE. Analysis on this matter would provide good insight for future approaches on improving SEE. The literature also lacks comprehensive studies comparing locality approaches such as RTs to clustering methods. So, it is not known whether RTs would be more adequate locality approaches than cluster-based approaches or which of them should be further exploited with the aim of improving SEE. As briefly explained in Section 1, one of the aims of the current paper is to address such questions.

2.3. Evaluation of different approaches for SEE

Current work on frameworks for evaluation of SEE models considers issues such as explicit preprocessing [28], evaluation measures [29,21] and the importance of the magnitude of the differences in performance [21]. There has also been work on statistical approaches for evaluating models across multiple data sets in the general ML literature [20].

Foss et al. [29] show that measures based on MRE are potentially problematic due to their asymmetry, biasing towards prediction models that under-estimate. That includes a performance measure very popular in the SEE literature: MMRE. Another measure, MAE, does not present asymmetry problems and is not biased. However, it is difficult to interpret, since the residuals are not standardised. So, measures such as MMRE have kept being widely used by most researchers in the area. However, Shepperd and Mc Donell [21] very recently proposed a new measure called Standardised Accuracy (SA), defined as follows:

$$SA = 1 - \frac{MAE_{P_i}}{\overline{MAE}_{P_0}},$$

where MAE_{P_i} is the mean absolute error of the prediction model P_i and $\overline{MAE}_{P_0}$ is the mean value of a large number, typically 1000, runs of random guessing. This is defined as predicting $\hat{y}$ for the example t by randomly sampling over the remaining $n-1$ examples and taking $\hat{y}_t = y_r$, where r is drawn randomly with equal probability from $1, \ldots, n \wedge r \neq t$.

Even though this measure is a ratio, such as MMRE, this is not problematic because we are interested in a single direction – better than random [21]. SA can be interpreted as the ratio of how much better P_i is than random guessing, giving a very good idea of how well the approach does.

To judge the effect size, the following measure is suggested:

$$\Delta = \frac{MAE_{P_i} - \overline{MAE}_{P_0}}{s_{P_0}}$$

where s_{P_0} is the sample standard deviation of the random guessing strategy. The values of Δ can be interpreted in terms of the categories proposed by Cohen [30] of small (≈ 0.2), medium (≈ 0.5) and large (≈ 0.8).

Shepperd and Mc Donell also suggested using the 5% quantile of the random guessing to estimate the likelihood of non-random estimation. However, we have found that comparing P_i's SA to P_0's 5% quantile can be a very conservative way to suggest whether a model is or is not better than random. A possible reason for that is the fact that the number of runs is not taken into account in such a comparison. In our experiments, while Wilcoxon sign-rank tests [31] detected significantly different models' MAE in comparison to random guessing with very low p-values (ranging from 1.6492×10^{-4} to 5.0864×10^{-18}) and the effect size Δ against random guess was very high, these models' SAs were no better than the 5% quantile of random guessing.

Nevertheless, the magnitude of the difference in performance among different models is important and SA gives a very good idea of how much better models are from random guessing. So, in our work, we build upon previous work and we propose the use of an evaluation framework that joins the power of statistical tests as suggested by Demšar [20] to the importance of magnitude as suggested by Shepperd and Mc Donell [21]. Our experimental framework also explicitly considers the parameters choice, emphasising its importance.

3. Data sets

The analysis presented in this paper is based on several different data sets from the PRedictOr Models In Software Engineering Software (PROMISE) Repository [15] and from the International Software Benchmarking Standards Group (ISBSG) Repository [16] Release 10. As explained in Section 1, the fact that we use both PROMISE and ISBSG data sets, as well as the high number of data sets, are advantages of our work over previous works evaluating ensembles for SEE. The data sets used in our work were chosen to cover a wide range of features, such as number of projects, type of features, countries and companies. Sections 3.1 and 3.2 provide their description and explanation on how they were processed.

3.1. PROMISE data

The PROMISE data sets used in this study are: cocomo81, nasa93, nasa, sdr and desharnais. Cocomo81 consists of the projects analysed by Boehm to introduce COCOMO [32]. Nasa93 and nasa are two data sets containing Nasa projects from 1970s and 1980s and from 1980s and 1990s, respectively. Sdr contains projects implemented in 2000s and was collected at Bogazici University Software Engineering Research Laboratory from software development organisations in Turkey. Desharnais' projects are dated from late 1980s. Table 1 provides some details about these data sets. The next subsections explain their features, missing values and preprocessing used.

3.1.1. Features

Cocomo81, nasa93 and nasa are based on the COCOMO [32] format, containing as input features 15 cost drivers, the number of lines of code and the development type (except for nasa, which does not contain the latter feature). The actual effort in person-months is the dependent variable. Sdr is based on COCOMO II [33], containing as input features 22 cost drivers and the number of lines of code. The actual effort in person-months is the dependent variable. The data sets were processed to use the COCOMO numeric values for the cost drivers. The development type was transformed into dummy variables.

Desharnais contains as input features the team experience in years, the manager experience in years, the year the project ended,

Table 1
PROMISE data sets. The effort is measured in person-months for all data sets except desharnais, in which it is measured in person-hours.

Data set	# Projects	# Features	Min effort	Max effort	Avg. effort	Std. dev. effort
Cocomo81	63	17	5.9	11,400	683.53	1821.51
Nasa93	93	17	8.4	8211	624.41	1135.93
Nasa	60	16	8.4	3240	406.41	656.95
Sdr	12	23	1	22	5.73	6.84
Desharnais	81	9	546	23,940	5046.31	4418.77

the number of basic logical transactions in function points, the number of entities in the system's data model in function points, the total number of non-adjusted function points, the number of adjusted function points, the adjustment factor and the programming language. The actual effort in person-hours is the dependent variable.

3.1.2. Missing values

The only data set with missing values is desharnais. In total, it contains only four, out of 81 projects, with missing values. As this is a small number of projects (about 5% of the total number of projects) and several previous papers have removed these projects [8,27,34], we decided to remove them from our data set as well.

3.1.3. Preprocessing

Preprocessing data sets for detecting and removing outliers is frequently necessary when building ML models. Works in the SEE literature show that SEE data sets frequently contain a few outliers, which may hinder the SEEs for future projects [35]. Outliers were detected based on k-means. This method was chosen because it has shown to improve performance in the SEE context [35]. K-means is used to divide the projects into clusters. After that, clusters with less than a certain number n of projects or projects with negative silhouette values are considered outliers.

The silhouette [36] value for each project represents the similarity of the project to the other projects in its cluster in comparison to the projects in the other clusters, ranging from -1 (more dissimilar) to 1 (more similar). The silhouette values were calculated as follows: Consider that we use a clustering algorithm to cluster a set of projects into a number of clusters. Repeat the following procedure for each project i. Let $a(i)$ be the average dissimilarity of i with respect to all other projects within A, where A is the cluster to which i belongs. Here, we used Euclidean distance as the measure of dissimilarity. Then, let $d(i,C)$ be the average dissimilarity of i to all projects in C, where C is any of the clusters. After computing $d(i,C)$ for all clusters $C \neq A$, let $b(i) = \text{minimum}_{C \neq A} d(i,C)$. The silhouette value is defined as:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

The average $s(i)$ for all projects is a measure of how appropriately the projects have been clustered. So, it can be used to determine the number of clusters k.

We used $n = 3$, as in [35]. The number of clusters k was chosen among $k = \{2, 3, 4, 5\}$, according to the average silhouette values. The projects considered as outliers are shown in Table 2. The outlier identified for sdr was not eliminated because this data set is very small (only 12 projects), providing not enough evidence to consider the identified project as an outlier. As an additional step of our experiments presented in Section 5, we verify and confirm that such preprocessing is in practice recommendable.

Table 2
PROMISE data sets – outliers. The numbers identifying the outlier projects represent the order in which they appear in the original data set, starting from one.

Data set	K	Outliers
Cocomo81	2	None
Nasa93	2	42, 46, 62
Nasa	2	2, 3
Sdr	2	9
Desharnais	2	9, 39, 54

3.2. ISBSG data

The ISBSG repository contains a large body of data about completed software projects. The release 10 contains 5,052 projects, covering many different companies, several countries, organisation types, application types, etc. The data can be used for several different purposes, such as evaluating the benefits of changing a software or hardware development environment; improving practices and performance; and estimation. In order to produce reasonable SEE using ISBSG data, a set of relevant comparison projects needs to be selected. With that in mind, we preprocessed the data set (resulting in 621 projects) maintaining only projects with:

- Data and function points quality A (assessed as being sound with nothing being identified that might affect their integrity) or B (appears sound but there are some factors which could affect their integrity/ integrity cannot be assured).
- Recorded effort that considers only development team.
- Normalised effort equal to total recorded effort, meaning that the reported effort is the actual effort across the whole life cycle.
- Functional sizing method IFPUG version 4+ or NESMA.
- No missing organisation type field.

After that, with the objective of creating different subsets, the projects were grouped according to organisation type. Only the groups with at least 20 projects were maintained, following ISBSG's data set size guidelines. The resulting organisation types are shown in Table 3.

Table 4 contains additional information about the subsets. As we can see, the productivity rate of different companies varies. A 7-way 1 factor Analysis of Variance (ANOVA) was used to confirm the difference of productivity rates among the subsets, indicating statistically significant difference at the 95% confidence interval (p-value $< 2.2 \times 10^{-16}$).

The next sections explain how the features were selected, how to deal with the missing values and the preprocessing done after dealing with features and missing values.

Table 3
ISBSG data – organisation types used.

Organisation type	Id	# Projects
Financial, property & business services	1	76
Banking	2	32
Communications	3	162
Government	4	122
Manufacturing; transport & storage	5	21
Ordering	6	22
Billing	7	21

Table 4
ISBSG subsets.

Id	Unadjusted function points				Effort				Productivity			
	Min	Max	Avg	Std Dev	Min	Max	Avg	Std Dev	Min	Max	Avg	Std Dev
1	43	2906	215.32	383.72	91	134,211	4081.64	15951.03	1.2	75.2	12.71	12.58
2	53	499	225.44	135.12	737	14,040	3218.50	3114.34	4.5	55.1	15.05	9.94
3	3	893	133.24	154.42	4	20,164	2007.10	2665.93	0.3	43.5	17.37	9.98
4	32	3088	371.41	394.10	360	60,826	5970.32	8141.26	1.4	97.9	18.75	16.69
5	17	13,580	1112.19	2994.62	762	54,620	8842.62	11715.39	2.2	52.5	23.38	14.17
6	50	1278	163.41	255.07	361	28,441	4855.41	6093.45	5.6	60.4	30.52	17.70
7	51	615	160.10	142.88	867	19,888	6960.19	5932.72	14.4	203.8	58.10	61.63

3.2.1. Features

The ISBSG suggests that the most important criteria for estimation purposes are the functional size; the development type (new development, enhancement or re-development); the primary programming language or the language type (e.g., 3GL, 4GL); and the development platform (mainframe, midrange or PC). As development platform is missing in more than 40% of the projects for two organisation types, the following criteria were used as features: functional size; development type; and language type.

The normalised work effort in hours is the dependent variable. Due to the preprocessing, this is the actual development effort across the whole life cycle.

3.2.2. Missing values

The features "functional size" and "development type" have no missing values. The feature "language type" is missing in several subsets. Even though this feature is never missing in more than 40% of the projects of any ISBSG subset, it is missing in 25 of the 76 projects (about 33%) of ISBSG subset org1. We consider this percentage to be too high to eliminate the 25 projects from this subset, as we would loose too many data that are potentially useful in improving a model's performance [37,38]. We decided to adopt an imputation method so that this feature can be kept without having to discard the projects in which it is missing.

The imputation method was based on k-Nearest Neighbours (k-NN). K-NN imputation has shown to be able to improve SEEs [39]. It is particularly beneficial to this area because it is simple and does not require large data sets. Another method, based on the sample mean, also presents these features, but k-NN has shown to outperform it in two SEE case studies [39].

According to Cartwright et al. [39], "k-NN works by finding the k most similar complete cases to the target case to be imputed where similarity is measured by Euclidean distance". When $k > 1$, several different methods can be used to determine the value to be imputed, for example, simple average. For categorical values, vote counting is adopted. Typically, $k = 1$ or 2. As language type is a categorical feature, using $k = 2$ could cause draws. So, we chose $k = 1$. The Euclidean distance considered normalised data sets.

3.2.3. Preprocessing

Similarly to the PROMISE data sets (Section 3.1), the ISBSG data sets were preprocessed to detect and remove outliers through k-means [40]. K was chosen among $k = \{2, 3, 4, 5\}$ based on the average silhouette values. The chosen k was 2 for subsets 1–5 and 3 for subsets 6 and 7. The number of outliers varied from none to 5, as shown in Table 5. None of the data sets were reduced to less than 20 projects after outliers elimination.

Table 5
ISBSG subsets – outliers. The numbers identifying the outlier projects represent the order in which they appear in the original data set, starting from one.

Id	K	Outliers
1	2	38
2	2	None
3	2	80, 91, 103, 160
4	2	4, 10, 75, 89, 104
5	2	20
6	3	4
7	3	None

4. Experimental framework

ML experiments involve three important points besides the choice of data sets to be used: (1) choice of learning machines, (2) choice of evaluation methods and (3) choice of parameters. All these points should be considered carefully based on the aims of the experiments, which in this case are the research questions explained in Section 1. The framework presented in this section concentrates mainly on evaluating and choosing ML models for SEE. This is used at least partly to answer each of the research questions of this work, as explained in more detail in the experimental Sections 5 to 7. RQ4 (how to evaluate/choose a ML model for SEE) is itself partly answered by the framework proposed in the current section.

4.1. Choice of learning machines

As explained in Section 1, RQ1 involves determining whether existing automated ensembles of learning machines generally improve effort estimations given by single learning machines, including potentially adequate locality approaches such as RTs. With that aim, three ensemble and three single learning machines were chosen to be used:

- Single learning machines:
  - REPTree Regression Trees (RTs) [13,41];
  - Radial Basis Function networks (RBFs) [42]; and
  - MultiLayer Perceptrons (MLPs) [42].
- Ensembles of learning machines:
  - Bagging [12] with MLPs (Bag + MLPs), with RBFs (Bag + RBFs) and with RTs (Bag + RTs);
  - Random [13] with MLPs (Rand + MLPs); and
  - Negative Correlation Learning [43,14] with MLPs (NCL + MLPs).

RTs were chosen because they are single learning machines based on locality and present several potential advantages for SEE, as outlined in Section 2.2. RBFs are based on locality as well, due to the radial basis functions (typically Gaussians) used in their hidden nodes. When the inputs are fed into the hidden layer, their distances (e.g., Euclidean distances) to the centre of each neuron are calculated. After that, each node applies a radial basis function to the distance, which will produce lower/higher values for larger/shorter distances. Each hidden neuron i is connected to each output neuron j with weight w_{ji} and output neurons usually compute

a linear function of their inputs when working with regression problems. Due to the use of a radial basis function, changes in the weights connecting a given hidden neuron to the output neurons do not affect input values that are far from the centre of this neuron. In short, RBFs use locality.

Even though the choice of RTs and RBFs was based on their relation to locality, MLPs are not local learning machines and were included as an example of non-local approach. They were chosen for being popular learning machines that can approximate any continuous function [42]. They have also been investigated in the SEE context [5,7,3,4,6,8,9]. For instance, Tronto et al. [7] showed that they can outperform linear regression because they are able to model observations that lie far from the best straight line. MLPs can be easily combined using several different ensemble approaches, such as bagging, random ensembles and NCL. Other non-local approaches that are not restricted to certain function shapes were not chosen because they do not perform well for regression tasks (e.g., Naive–Bayes [44]) or have not been so much used for SEE (e.g., support vector machines).

Each of the chosen ensembles has a potential advantage and disadvantage for SEE. So, they were selected in order to verify which of these advantages is in fact more relevant. Bagging is one of the most well known ensemble learning approaches in the literature. It creates diversity by training each base learning machine with a different training set generated by sampling with replacement from the available training data. By using this scheme, bagging is able to turn weak learning machines into stronger ones [12]. This can be particularly advantageous for SEE, as there are usually very few projects to be used for learning, producing very inaccurate base learning machines that may be considered as weak. For instance, combining bagging with RTs may produce good results, as this could be considered an automated way to join the power of ensembles to locality.

On the other hand, each base learning machine in a bagging ensemble is trained with only about 63.2% of the unique examples from the available original training set [45]. So, another approach called random committees/ensembles was also used. Random ensembles are based on the fact that different initial conditions may cause different neural networks to converge into different local minima of the cost function. They turn an apparent disadvantage, local minima in training neural networks, into something useful by averaging (in the case of regression) the predictions of base learning machines trained using different random seeds. Despite its simplicity, this procedure works surprisingly well in many, but not all, cases [46–48]. If the base learning machines are independent of each other, the squared prediction error of the ensemble is equal to or less than the mean of the squared prediction errors of the base learning machines [49]. The problem of this approach is that there is no guarantee that a good level of diversity will be achieved, as random ensembles do not always lead to uncorrelated base learning machines just by using different random seeds [43].

NCL [43,14] was chosen for having strong theoretical foundations for regression problems, explicitly controlling diversity through the error function of the base learning machines [23]. Its disadvantage is that it is usually used with strong learning machines. Besides, it can only be used with neural networks. Other learning machines such as RTs cannot be currently used.

The choice of base learning machines for the ensembles was based not only on combinations of local learning machines to ensembles, but also on the ensemble's intrinsics. The expected error of a predictor can be decomposed into bias and variance terms [50]. The bias measures how different the expected predictions are in comparison to the actual values. The variance measures how much the predictor varies from one training set to another, drawn from the same distribution. Bagging is an approach that can improve ensemble performance by reducing the variance [12], working well with learning machines such as MLPs, RBFs and RTs. However, learning machines such as *k*-NN already have low variance [51] and cannot be improved through bagging. For that reason, we used MLPs, RBFs and RTs as base learning machines for bagging, but not *k*-NN (or other analogy based learning machines). Learning machines such as deterministic regression trees or *k*-NN do not produce different models with different random seeds, so they cannot be used with a random ensemble, which is typically used with neural networks [49].

RQ2 involves analysing different locality approaches and determining which of them is more adequate for SEE. With that in mind, the following locality approaches were chosen:

- RTs;
- *K*-Nearest Neighbours (*k*-NN) [52] based on normalised features and Euclidean distance; and
- Clustering approaches: Expectation Maximization (EM) [52]; Spectral Clustering (SC) [53,54]; and *K*-Means [52] based on normalised features and Euclidean Distance.

K-NN was chosen because it represents a traditional way to perform SEE by analogy [34]. EM was chosen for having been previously successfully used in the SEE domain [18]. SC was chosen because it is not restricted to hypersphere clusters in the space of input features and there is no evidence that SEE data clusters should have hypersphere shape. *K*-means was chosen for being a very simple approach restricted to hypersphere shape, offering a good comparison to indicate whether SEE data needs or not non-spherical clusters.

In order to use the clustering approaches for predictions, a single RT was created to learn each cluster of training examples. Similarly to Cuadrado Gallego et al.'s work with regression equations [18], whenever required to perform a prediction, the approach determines to which cluster the example belongs and then uses the corresponding RT for estimating the effort.

In order to answer RQ3, additional approaches which are based on rules, similarly to RTs, were also used to compose an ensemble of multiple types of base learning machines. These are Decision Tables, Conjective Rules and M5 Rules [13].

All the learning machines but NCL were based on the Weka implementation [13]. The regression trees were based on the REPTree implementation available from Weka. We recommend the software Weka should the reader wish to get more details about the implementation and parameters. The SC algorithm was the one provided by Dragone [55]. The software used for NCL is available upon request.

4.2. Choice of evaluation methods

The following measures of performance were used in this work [28,8,21]: Mean Magnitude of the Relative Error (MMRE), Percentage of Estimates within 25% of the actual values PRED(25), Mean Absolute Error (MAE) and Standardised Accuracy (SA). MMRE and PRED(25) were chosen because they were used in previous work evaluating ensembles, and one of our aims is to build upon those works to determine through principled experiments whether ensembles are or not generally better than single learning machines (RQ1). The results of the ensembles evaluation using these measures were published at PROMISE'11 [56]. Those experiments are extended in the current work by using MAE and SA, which are more reliable performance measures. Unless stated otherwise, the analysis will always refer to the measures calculated on the test set.

Our previous work [56] shows that no learning machine is consistently the best across different data sets. However, learning

machines may be singled out as being most frequently among the best. So, in the current work, we omit the evaluation done in our previous work to determine whether a certain learning machine is consistently the best based on Menzies et al.'s rejection rules [28]. Those rules were previously used to determine whether a certain approach is the best for a given data set through pairwise comparisons with every other approach. Instead, we use Friedman statistical test to determine whether the performance of the learning machines in terms of MAE is statistically significantly different from each other considering multiple data sets.

Friedman was recommended by Demšar [20] as an adequate test for comparing multiple learning machines across multiple data sets. This statistical test also provides a ranking of algorithms as follows. Let r_i^j be the rank of the jth of k algorithms on the ith of N data sets. The average rank of algorithm j is calculated as $R_j = \frac{1}{N}\sum_i r_i^j$. The rounded average ranks can provide a fair comparison of the algorithms given rejection of the null hypothesis that all the approaches are equivalent. Nevertheless, Wilcoxon sign-rank tests [31] are typically used to compare a particular model to other models across multiple data sets after rejection of the Friedman null hypothesis, where necessary. Holm–Bonferroni corrections can be used to avoid high Type-I error due to the multiple tests performed.

As data sets are very heterogeneous in SEE and the performance of the approaches may vary greatly depending on the data set, it is also important to check what approaches are usually among the best and, when they are among the worst, whether the magnitude of the performance is much worse from the best approach for that data set or not. This type of analysis also helps to identify on what type of data sets certain approaches behave better. It is worth noting that statistical tests such as Friedman and Wilcoxon are based on the relative ranking of approaches, thus not considering the real magnitude of the differences in performance when used for comparison across multiple data sets. So, even if a Friedman test accepts the null hypothesis that all approaches perform similarly across multiple data sets, it is still valid to check what approaches are most often among the best, on what type of data sets, and the magnitude of the differences in performance. In our previous work [56], we only checked what approaches were among the best (in terms of MMRE and PRED(25)), not considering the magnitude of the differences in performance. In the current work, we extend the evaluation framework to (1) check the approaches among the best in terms of MAE and (2) check the magnitude of the differences in performance by determining whether an approach has SA worse than the best approach for each given data set in more than 0.1 units.

So, building upon out previous work [56], recent work on evaluation of models for SEE [21], and Demšar's work on comparison over multiple data sets [20], we use an evaluation framework based on the following three steps:

1. Friedman statistical test and ranking across multiple data sets to determine whether approaches behave statistically significantly differently considering several data sets.
2. Determine which approaches are usually among the first and second highest ranked approaches and, possibly, identify to which type of data sets approaches tend to perform better.
3. Check how much worse each approach is from the best approach for each data set.

A good approach would be highly ranked by Friedman and statistically significantly different from lower ranked approaches, would be more frequently among the best, and would not perform too bad in terms of SA when it is not among the best.

The evaluation was based on 30 rounds of executions for each data set. In each round, for each data set, 10 examples were randomly picked for testing and the remaining were used for the training of all the approaches being compared. Holdout of size 10 was suggested by Menzies et al. [28] and allows the largest possible number of projects to be used for training without hindering the testing. For sdr, half of the examples were used for testing and half for training, due to the small size of this data set.

The experiments use the PROMISE data sets explained in Section 3.1, the ISBSG subsets explained in Section 3.2 and a data set containing the union of all the ISBSG subsets. The union was used in order to create a data set likely to be more heterogeneous than the previous ones.

4.3. Choice of parameters

The choice of parameters is a critical step in ML experiments, as results can vary greatly depending on it. For instance, a learning machine that would have better performance under certain choices could have worse performance under other choices. It is important that the method used for choosing the parameters is made clear in papers using ML, so that differences in the results obtained can be better understood.

In order to choose the parameters, we performed five preliminary rounds of executions using all the combinations of parameters shown in Table 6 for each data set and learning approach. The parameters providing the lowest MMRE for each data set were chosen to perform 30 rounds of final executions, which were used in the comparison analysis. In this way, each approach enters the comparison using the parameters that are most likely to provide the best results for each particular data set. These parameters were omitted due to the large number of combinations of approaches and data sets used. The performance measure MMRE was chosen for being used in all previous work evaluating existing automated ensembles.

When ensembles of multiple types of base learning machines were used, parameter values for conjunctive rules varied among $\{3, 4, 5\}$ for choosing number of folds for pruning. Parameter values for M5 rules varied among $\{2, 4, 8\}$ for choosing the minimum number of instances at a leaf node.

In this work, clustering approaches were used together with RTs, as explained in Section 4.1. The RT parameters were chosen as explained above. However, the parameters of the clustering approaches were selected using a different strategy. The silhouette values were used as a heuristic to automatically tune parameters for each set of training examples. The value (among the values shown in Table 7) that provided the highest average silhouette value was chosen, unless it leads to a single cluster or to cluster sizes smaller than three. In this case, the parameter value generating the second highest average silhouette value was chosen and so forth.

5. Evaluation of existing automated ensemble approaches against single learning machines

This section mainly aims at determining whether existing automated ensembles of learning machines generally improve effort estimations given by single learning machines, including potentially adequate locality approaches such as RTs, and which of them would be more useful (RQ1). This is done by evaluating ensemble learning approaches against single learning machines using the framework presented in Section 4. This section also partly investigates RQ2 by evaluating the performance of RT and RBF, which are locality approaches. All the analyses in terms of MAE, SA and Δ are new to this paper, whereas analyses based on MMRE and PRED(25) were previously published in our PROMISE'11 paper [56].

As a pre-analysis, we can observe that very different performances were obtained for different data sets: The MMRE obtained

Table 6
Parameter values for preliminary executions.

Approach	Parameters
MLP	Learning rate = $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ Momentum = $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ # epochs = $\{100, 500, 1000\}$ # hidden nodes = $\{3, 5, 9\}$
RBF	# clusters = $\{2, 3, 4, 5, 6\}$ Minimum std. deviation for the clusters = $\{0.01, 0.1, 0.2, 0.3, 0.4\}$
REPTree	Minimum total weight for instances in a leaf = $\{1, 2, 3, 4, 5\}$ Minimum proportion of the data variance at a node for splitting to be performed = $\{0.0001, 0.001, 0.01, 0.1\}$
Ensembles	# base learning machines = $\{10, 25, 50\}$ All the possible parameters of the adopted base learning machines, as shown above
NCL	Penalty strength = $\{0.3, 0.4, 0.5\}$
K-NN	Number of neighbours = $\{1, 2, 4, 8, 16\}$

Table 7
Parameter values for clustering approaches.

Approach	Parameters
EM	Minimum standard deviation = $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$ Maximum number of iterations = 1000 Number of clusters automatically determined by cross-validation
SC	Maximum normalised cut value = $\{0.3, 0.5, 0.7, 0.9\}$
K-Means	Cluster size = $\{2, 3, 4, 5\}$ Maximum number of iterations = 500

by the best performing approach for each particular data set varied from 0.37 to 2.00. The MdMRE varied from 0.21 to 0.78. The PRED(25) varied from 0.17 to 0.55. The correlation between estimated and real effort varied from 0.05 to 0.91. The SA varied from 0.26 to 0.64. Table 8 shows the SA and effect size Δ in comparison to random guess obtained by the approach with the highest, second highest and lowest SA for each data set.

5.1. Friedman ranking

As a first step of the evaluation, the Friedman ranking of the ensembles, single learning machines and random guess in terms of MAE was determined and Friedman statistical test was used to check whether these approaches have statistically significantly different MAE. The ranking generated by the test is shown in Table 9. The Friedman test rejects the null hypothesis that all approaches perform similarly (statistic $F_f = 12.124 > F(8, 96) = 2.036$). As we can see from the table, random guess was always ranked last. An additional test was performed without including random guess and the null hypothesis was also rejected (statistic $F_f = 4.887 > F(7, 84) = 2.121$).

All bagging approaches performed comparatively well, being on average ranked third. RTs were ranked on average just below, as fourth. As the null hypothesis that all approaches perform similarly was rejected, Wilcoxon sign-rank tests with Holm–Bonferroni corrections were performed to determine whether the RT's MAE is similar or different from Bag + RT's, Bag + MLP's and Bag + RBF's across multiple data sets. Table 10 shows the *p*-values.

The tests reveal that ensembles in general do not necessarily perform better than an adequate locality approach such as RTs. Bag + MLP and Bag + RBF obtained statistically similar performance to RTs. However, Bag + RT, which is an ensemble of locality learning machines, obtains higher and statistically significantly different Friedman ranking in terms of MAE from single RTs. The number of win/tie/loss of Bag + RT vs RT is 10/1/2, further confirming the good performance of Bag + RTs.

Even though single RBFs use locality, they did not perform so well. A possible reason for that is that, even though locality is used in the hidden nodes, the output nodes are based on linear functions.

Table 8
SA and effect size Δ for approaches ranked as first, second and last in terms of SA for each data set.

	Approach	SA	Δ
Cocomo81	Bag + MLP	0.5968	0.9679
	RT	0.5902	0.9572
	RBF	0.3024	0.4904
Nasa93	RT	0.6205	1.2647
	Bag + RT	0.6167	1.2569
	NCL	0.1207	0.2460
Nasa	Bag + RT	0.6423	1.5401
	RT	0.6409	1.5367
	RBF	0.2618	0.6277
Sdr	RT	0.2626	0.7508
	Bag + RT	0.2066	0.5907
	Rand + MLP	-2.9981×10^8	-1.3140×10^8
Desharnais	Bag + MLP	0.5248	1.7132
	Bag + RT	0.5049	1.6484
	RBF	0.3724	1.2157
Org1	MLP	0.4573	0.6670
	Bag + MLP	0.4441	0.6478
	RT	0.2245	0.3275
Org2	Bag + RBF	0.2719	0.9941
	Bag + MLP	0.2716	0.9930
	NCL	0.0280	0.1025
Org3	Bag + RT	0.5511	1.3237
	RT	0.5347	1.2843
	NCL	0.4331	1.0402
Org4	MLP	0.3276	0.6995
	RBF	0.3175	0.6779
	NCL	0.0248	0.0530
Org5	Bag + RT	0.4259	1.9993
	Bag + MLP	0.4038	1.8957
	NCL	0.2117	0.9939
Org6	Bag + RBF	0.4930	2.2358
	MLP	0.4680	2.1224
	NCL	0.3617	1.6405
Org7	Bag + RBF	0.3017	1.3739
	Bag + MLP	0.2948	1.3425
	RBF	0.0285	0.1300
OrgAll	Bag + RT	0.4416	1.0523
	RT	0.4318	1.0288
	Bag + MLP	0.3319	0.7907

Table 9
Friedman ranking of approaches in terms of MAE.

Rounded avg. rank	Avg. rank	Std. dev. rank	Approach
3	2.77	1.69	Bag + RT
	3.38	2.18	Bag + MLP
	3.46	1.98	Bag + RBF
4	4.15	2.58	RT
5	4.54	2.22	MLP
	5.23	1.54	Rand + MLP
6	5.92	2.02	RBF
7	6.54	1.66	NCL + MLP
9	9.00	0.00	Random Guess

Table 10
Wilcoxon sign-rank tests for comparison of RT's MAE to Bag + RT's, Bag + MLP's and Bag + RBF's across multiple data sets. The *p*-value in italics represents statistically significant difference with Holm-Bonferroni corrections at the overall level of significance of 0.05.

Approaches compared	*p*-Value
RT vs Bag + RT	***0.0134***
RT vs Bag + MLP	0.4143
RT vs Bag + RBF	0.4143

5.2. Approaches most often ranked first or second in terms of MAE, MMRE and PRED(25)

As a second step of the evaluation, the learning machines most often ranked as first or second in terms of MAE, MMRE and PRED(25) were determined. Part of the reason to report results in terms of MMRE and PRED(25) here is to allow comparison of our conclusions to previous works in the area, which are based on these measures. However, verifying whether different results are achieved when using different performance measures can also provide interesting insight on the behaviour of the approaches, as discussed in Section 5.4. Some results in terms of SA are also presented in order to provide more easily interpretable results.

In order to check how valid this analysis is, we first compare the SA and effect size Δ of the approach ranked as first, second and last in terms of MAE for each data set. Table 8 shows these values. As we can see, the approaches ranked first or second frequently achieve SA at least 0.20 higher (better) than the worst approaches. That is reflected in the overall average SA considering all data sets, which improves from 0.2251 (approaches ranked last) to 0.4599 and 0.4712 (approaches ranked first and second, respectively). The effect size Δ in relation to random guess is frequently changed from small/medium to large, emphasising the importance of using higher ranked approaches. Wilcoxon rank-sum statistical tests using Holm–Bonferroni corrections at the overall level of significance of 0.05 for comparing the approaches ranked first and second against random guess detect statistically significant difference for all cases. The *p*-values are very low, ranging from 1.6492×10^{-4} to 5.6823×10^{-19}, confirming that the performance of these approaches is indeed better than random guessing.

Table 11a shows the learning machines ranked most often as first or second in terms of MAE. The results show that Bag + RTs, Bag + MLPs and RTs are singled out, appearing among the first two ranked approaches in 27%, 23% and 23% of the cases, whereas all other approaches together sum up to 27% of the cases. It is worth noting that Bag + RBF, which achieved high Friedman rank, does not appear so often among the best two approaches. One might think that Bag + RBF's rank for each data set could be more median, but more stable. However, the standard deviation of the ranks (Table 9) is similar to Bag + RT's and Bag + MLP's. So, the latter approaches are preferable over Bag + RBFs.

The table also shows that RTs are comparatively higher ranked for PROMISE than for ISBSG, achieving similar ranking to bagging approaches for PROMISE, but lower ranking for ISBSG. Even though RTs perform statistically similarly to Bag + MLPs and Bag + RBFs (Section 5.1), the difference in performance is statistically significant in comparison to Bag + RTs. Further Wilcoxon sign-rank tests to compare RTs and Bag + RTs considering separately PROMISE and ISBSG data show that there is no statistically significant difference considering PROMISE on its own (*p*-value 1), but there is considering ISBSG (*p*-value 0.0078). So, approaches joining the power of bagging ensembles to the locality of RTs may be particularly helpful for more heterogeneous data.

Table 11
Number of data sets in which each learning machine was ranked first or second according to MAE, MMRE and PRED(25). Learning machines never among the first or second are omitted.

PROMISE data	ISBSG data	All data
(a) According to MAE		
RT: 4	Bag + MLP: 4	Bag + RT: 7
Bag + RT: 4	Bag + RT: 3	RT: 6
Bag + MLP: 2	Bag + RBF: 3	Bag + MLP: 6
	MLP: 3	MLP: 3
	RT: 2	Bag + RBF: 3
	RBF: 1	RBF: 1
(b) According to MMRE		
RT: 4	RT: 5	RT: 9
Bag + MLP: 3	Bag + MLP 5	Bag + MLP: 8
Bag + RT: 2	Bag + RBF: 3	Bag + RBF: 3
MLP: 1	MLP: 1	MLP: 2
	Rand + MLP: 1	Bag + RT: 2
	NCL + MLP: 1	Rand + MLP: 1
		NCL + MLP: 1
(c) According to PRED(25)		
Bag + MLP: 3	RT: 5	RT: 6
Rand + MLP: 3	Rand + MLP: 3	Rand + MLP: 6
Bag + RT: 2	Bag + MLP: 2	Bag + MLP: 5
RT: 1	MLP: 2	Bag + RT: 3
MLP: 1	RBF: 2	MLP: 3
	Bag + RBF: 1	RBF: 2
	Bag + RT: 1	Bag + RBF: 1

Table 11b shows the two learning machines most often ranked as first and second in terms of MMRE. Both RTs and Bag + MLPs are very often among the first two ranked learning machines according to MMRE. The trend can be observed both in the PROMISE and ISBSG data sets. For PROMISE, RTs or Bag + MLPs appear among the first two ranked approaches in 70% of the cases, whereas all other learning machines together sum up to 30%. For ISBSG, RTs or Bag + MLPs appear among the first two ranked in 62.5% of the cases, whereas all other learning machines together sum up to 37.5%. Wilcoxon sign-rank tests show that these two approaches perform similarly in terms of MMRE (*p*-value of 0.2439).

The analysis considering PRED(25) shows that both RTs and Bag + MLPs are again frequently among the first two ranked (Table 11c), but Rand + MLP becomes more competitive. If we consider PROMISE data by itself, ensembles such as Bag + MLPs are more frequently ranked higher than single learning machines, even though that is not the case for ISBSG. However, a Wilcoxon sign-rank test across multiple data sets shows that Bag + MLPs and RTs are statistically similar in terms of MMRE (*p*-value of 0.7483). Tests considering PROMISE and ISBSG data separately do not find statistically significant difference either (*p*-value of 0.8125 and 0.3828, respectively).

As we can see, RTs and Bag + MLPs are singled out as more frequently among the best both in terms of MMRE, PRED(25) and MAE and they perform statistically similarly independent of the performance measure. Other approaches such as Rand + MLP and Bag + RTs become more or less competitive depending on the performance measure considered. Bag + RBFs, differently from the Friedman ranking, is not singled out, i.e., it is not frequently among the best. Our study also shows that Bag + RTs outperform RTs in terms of MAE mainly for ISBSG data sets, which are likely to be more heterogeneous.

5.3. Magnitude of performance against the best

In this section, we analyse how frequently an approach is worse than the best MAE approach in more than 0.1 units of SA. There are 34 cases in which approaches are worse than the best MAE

Table 12
Number of times that an approach is worse than the best MAE approach of the data set in more than 0.1 units of SA.

Approaches	Number of times
RT, Bag + RT	1
Bag + MLP, Bag + RBF, MLP	3
Rand + MLP	5
RBF	8
NCL + MLP	10

approach of the data set in more than 0.1 units of SA. p-Values of Wilcoxon tests to compare these approaches to the best MAE are below 0.05 in 28 cases. If we apply Holm–Bonferroni corrections considering all the 34 comparisons, the difference is statistically significant in 20 out of 34 cases (59%). So, it is reasonable to consider the number of times that each approach is worse than the best in order to evaluate its performance.

Table 12 shows how many times each approach is worse than the best MAE approach in more than 0.1 units of SA. As we can see, RTs and Bag + RTs are rarely worse than the best MAE approach in more than 0.1 SA. They are worse in only 1 in 13 data sets. Bag + MLPs, Bag + RBFs and MLPs behave slightly worse, with the difference in SA higher than 0.1 SA in three data sets.

5.4. Discussion

Summarizing, we can see that Bag + RTs present a very good behaviour for SEE. They have higher average (Friedman) rank, are more frequently among the best in terms of MAE and are rarely worse than the best MAE approach of the data set in more than 0.1 units of SA. This is an example of how joining the power of bagging ensembles to a good locality approach can help improving SEE. Bag + RTs were particularly helpful to improve performance in terms of MAE for ISBSG data sets, which are likely to be more heterogeneous. Such a behaviour provides part of the answer to RQ1: adequate ensemble approaches benefiting from locality, such as Bag + RTs, are singled out as performing in general comparatively better than single learning machines for SEE, including locality approaches such as RTs. These results also provide the insight that future research on improving SEE using automated learning machines may benefit from further exploiting the advantages of ensembles and locality together, contributing to answer RQ3.

The analysis also shows that, even though RTs perform worse than Bag + RTs in terms of MAE, they do not perform poorly in comparison to other approaches. Their MAE is statistically similar to Bag + MLPs and Bag + RBFs, showing that the power of ensembles on its own is not enough to generally outperform adequate locality learning machines such as RTs, complementing the answer to RQ1. The comparisons performed in this section are also partly related to RQ2, showing that the locality approach RT performs well in comparison to other approaches and is rarely worse than the best performing approach in more than 0.1 units of SA.

The analysis from this section, together with the framework presented in Section 4, provide an answer to RQ4. As no approach is always the best independent of the data set, ideally, an organisation should perform experiments following a principled framework in order to choose a model for their particular data set. Nevertheless, if an organisation has no resources to perform such experiments, Bag + RTs are more likely to perform comparatively well across different data sets and are rarely worse than the best approach for a particular data set in more than 0.1 SA. Even though single RTs may perform slightly worse than Bag + RTs, they do not perform bad in comparison to other approaches and may be used if the practitioners would like to understand the rules used by the learning machine to perform estimations.

The conclusions above are mainly drawn based on MAE and SA, which are considered as more adequate and reliable performance measures than MMRE and PRED(N). In terms of behaviour independent of the performance measure used, RTs and Bag + MLPs are always singled out as being frequently first or second ranked, whereas Bag + RT is less frequently among the best in terms of MMRE and PRED(25). So, in terms of MMRE and PRED(25), ensembles cannot be considered as generally outperforming adequate single learning machine approaches such as RTs. Such a difference in the evaluation based on MAE and MRE-based measures gives us an interesting insight. It suggests that the slightly worse performance of RTs and Bag + MLPs in comparison to Bag + RTs in terms of MAE may be related to the fact that RTs and Bag + MLPs suffer more from underestimations. This issue should be further analysed as future work and could be helpful for improving the performance of these approaches for SEE.

It is worth noting that previous works on ensembles have used mostly measures based on MRE [9,8,10]. One of the reasons for us to report results using MMRE and PRED(N) in addition to MAE and SA is to compare our conclusions to the conclusions obtained by some of those works.

In terms of MMRE and PRED(N), even though our work provides a different (practically the opposite) conclusion from Braga et al. [9], it does not necessarily contradict their reported results. Considering that the best performances obtained by their ensembles and single learning machines is very similar in their experiments, had statistical tests been done, their conclusion could possibly have been more similar to ours.

Another important point to be mentioned is that the bagging version used here is not the same version used by Kultur et al. [8]. In Kultur et al.'s work, instead of taking the simple average of the outputs of the base learning machines as the output of the ensemble, the outputs of the base learning machines are first clustered using adaptive resonance theory. After that, the simple average of the estimations in the largest cluster is considered as the output of the ensemble.

As Kultur et al.'s approach is not available as open source, we performed the following test to compare Bag + MLPs with the best result that the bagging ensemble could produce should other scheme than the simple average be used as the output of the ensemble. The best result was produced by making the output of the ensemble as the best output produced by any of its base learning machines. This ideal ensemble can improve both MMRE, PRED(25) and MAE for several data sets, according to Wilcoxon sign-rank tests considering the level of significance of 0.05 using Holm–Bonferroni corrections. So, Bag + MLPs still have potential to be improved for SEE, in particular considering the choice of the base models to be used for predictions. This is an additional insight (RQ3) provided by this section.

It is also worth making a note on the preprocessing of the data sets. As explained in Section 3, data sets were preprocessed using k-means for outliers removal. Outliers are examples different from usual and that cannot be well estimated by the models generated. Experiments done using RTs and Bag + MLPs reveal that the error obtained when attempting to use the models to estimate solely outlier projects is worse than the average obtained using non-outlier examples, both in terms of MAE and PRED(25). Interestingly, the error in terms of MMRE is not much affected. As MMRE tends to bias models towards underestimations [21], these outliers are likely to be projects underestimated by the models.

We have also repeated the experiments using Bag + RTs, Bag + MLPs and RTs, but without performing this part of the preprocessing, i.e., allowing the train/test sets to contain outliers. The experiments reveal that the SAs obtained are considerably

reduced. So, in practice, the use of a clustering approach such as *k*-means is recommendable to remove outliers from the training sets and to identify whether a test example is an outlier.

6. Comparison of locality approaches

This section concentrates mainly on answering RQ2: what locality approach is more adequate for SEE tasks? In particular, how well does RT locality do in comparison to other locality approaches? On what type of data sets? This research question is new to the current paper. The analysis presented in Section 5 partly contributes to its answer by evaluating RT's suitability in comparison to other approaches. In the current section, we analyse different locality approaches not only in order to verify which of them is more appropriate for SEE, but also to provide insight on improving SEE (RQ3). The approaches analysed are: RTs, EM, *k*-means, SC and *k*-NN.

6.1. Number of clusters generated

Table 13 shows the average number of clusters generated by EM, *k*-means and SC. As we can see, EM and SC tend to produce a single cluster in 5 and 4 out of 13 data sets, respectively. On the one hand, these approaches can be considered to fail to correctly identify training examples with similar features in the SEE domain. On the other hand, we could consider that they manage to identify when their type of locality is or is not useful. As shown in Table 13, these approaches frequently produce a single cluster when data sets are very small. *K*-means never produced less than two clusters because the number of clusters itself is a parameter which was chosen among 2–5. RTs produce a single node in more than a third of the runs only for sdr, which is a very small data set. The executions also reveal that EM sometimes produces empty clusters, i.e., even though the empty cluster was created using the available training data, no training example is associated to it when determining to which cluster the training examples belong. That happened 1–6 times for data sets Org1, Org2, Org4–7.

6.2. Friedman ranking

In order to evaluate the quality of the separation of data provided by the clustering approaches in comparison to the locality provided by RTs and *k*-NN, the clustering algorithms were associated to RTs as explained in Section 4.1. If this scheme provides better performance than single RTs, it shows that the corresponding clustering algorithm can be more beneficial to SEE than RTs by themselves.

Table 13
Average number of clusters – values in italics represent more than 1/3 of runs with less than two clusters.

Data set	Number of projects	EM	*k*-Means	SC
Cocomo81	63	3.10	2.20	1.90
Nasa93	90	6.13	3.67	1.53
Nasa	58	4.67	4.03	3.27
Sdr	12	***1.00***	2.10	5.00
Desharnais	74	2.90	2.00	2.00
Org1	75	2.70	2.00	1.93
Org2	32	***1.10***	2.83	1.73
Org3	158	2.76	2.03	***1.00***
Org4	117	3.40	3.03	2.00
Org5	20	***1.70***	2.13	***1.26***
Org6	21	***1.06***	2.20	***1.03***
Org7	21	***1.20***	2.00	***1.03***
OrgAll	444	4.03	2.07	2.00

Table 14
Friedman ranking of locality approaches.

Rounded avg. rank	Avg. rank	Std. dev. rank	Approach
3	2.77	1.30	RT
	2.77	1.64	*k*-NN
	3.00	1.15	*k*-Means
	3.23	1.42	SC
	3.23	1.69	EM

The Friedman ranking of the approaches is shown in Table 14. The Friedman test detected no statistically significant difference in the MAE of these approaches (statistic $F_f = 0.2612 < F(4, 48) = 2.565$). SC and EM sometimes perform very similarly to RTs because they generate a single cluster, thus basically making predictions based solely on a single RT. However, *k*-means always produces at least two clusters, obtaining MAE legitimately similar to RT's.

6.3. Approaches most often ranked first or second in terms of MAE

Table 15 shows the approaches most often ranked as first or second in terms of MAE, separated according to PROMISE and ISBSG, and according to data set size. Interestingly, the approaches do not perform so differently from each other depending on the size of the data set. Clustering approaches manage to be among the best in a similar frequency to RTs and *k*-NN, including *k*-means, which always divides the training data in at least two clusters. It is worth noting, though, that *k*-means would be more likely to perform worse for smaller data sets if the learning approaches used for each training data cluster were not based on locality.

When considering either PROMISE or ISBSG data sets, the approaches present a clearer difference in behaviour. Approaches such as *k*-NN or *k*-means tend to be more frequently among the best for ISBSG, which is likely to be more heterogeneous, whereas RTs tend to be more frequently among the best for PROMISE.

6.4. Magnitude of performance against the best

There are 21 cases in which the approaches analysed (RT, EM, *k*-means, SC and *k*-NN) have performance worse than the best MAE approach of the data set in more than 0.1 units of SA. Wilcoxon tests to verify the statistically significance retrieve *p*-value less than 0.05 for 16 out of these 21 cases. Holm–Bonferroni corrections lead to 9 (43%) cases being statistically significantly different. The number of times that each approach is worse than the best in more than 0.1 SA is shown in Table 16.

As we can see, even though *k*-means' and *k*-NN's Friedman ranking in terms of performance is similar to RT's and all these approaches appear overall frequently among the best two in terms of MAE, *k*-means and *k*-NN are less reliable. *K*-means' SA is more than 0.1 worse than the best approach in 6 out of 13 data sets, and *k*-NN's is worse in five data sets. RTs, on the other hand, are worse in more than 0.1 units of SA only in two data sets.

6.5. Discussion

This section shows that EM and SC sometimes fail to separate training examples by identifying their similar features. However, if using them for creating new approaches that are not required to always divide the data into different clusters, we could consider these approaches to be able to detect when their type of locality is helpful. As the approaches analysed in this section are considered all statistically similar across multiple data sets, all of them could be further exploited in future work to improve SEE. The clusters provided by *k*-means and the locality provided by *k*-NN may be

Table 15
Number of data sets in which each locality approach was ranked first or second according to MAE.

PROMISE data	ISBSG data	Less than 50 projects	More than 50 projects	All data
RT: 4	*k*-NN: 5	EM: 3	RT: 4	RT: 6
SC: 2	*k*-means: 4	RT: 2	*k*-NN: 4	*k*-NN: 6
EM: 2	EM: 3	*k*-NN: 2	*k*-means: 3	*k*-means: 5
k-means: 1	RT: 2	*k*-means: 2	SC: 3	EM: 5
k-NN: 1	SC: 2	SC: 1	EM: 2	SC: 4

Table 16
Number of times that an approach is worse than the best MAE approach of the data set in more than 0.1 units of SA.

Approaches	Number of times
RT	2
EM	4
k-NN, SC	5
k-Means	6

particularly helpful for more heterogeneous data sets. However, when these approaches are not beneficial, they can cause SA worse than the best in more than 0.1 units. RTs are more rarely worse than the best in more than 0.1 units, being more reliable than the other approaches analysed.

This analysis provides an answer to RQ2 by analysing the adequacy of different locality approaches to SEE, how well RTs do in comparison to other approaches and on what type of data set the approaches are likely to be more useful. Even though RTs have several features likely to be beneficial to SEE, other approaches are also beneficial. The study also provides insight on how to improve SEE (RQ3). It shows that future research for improving SEE should try to benefit from different types of locality, as all the approaches analysed here can be potentially useful in different ways.

7. Insight based on successful ensemble and locality approaches

Sections 5 and 6 give some insights on how to improve SEE and some directions for future work. For instance, they highlight the importance of both ensembles and locality, showing that future research may particularly benefit from combining the two to improve SEE through automated approaches. They also show that different types of locality can provide different benefits. The current section explores these issues further. Section 7.1 reproduces the analysis from our previous paper [56] on the benefit of the hierarchy of features provided by RTs in comparison to correlation-based feature selection. Section 7.2 is new to this paper and analyses different strategies of combining ensembles and locality, and their potential benefits. As explained in Section 1, the key contribution of this paper is not in a new algorithm, but better understanding/insight based on experimental studies.

7.1. Analysis of RTs and feature selection

As explained in Section 2.2, using the relative importance of features for the predictions, as done by RTs, may be particularly beneficial to SEE. In this section, we analyse whether the separation of features provided by RTs is important only as a correlation-based feature selector or whether the relative importance of features is key to the performance of RTs. If the relative importance is very beneficial, other approaches which currently perform similarly to RTs might be improved by incorporating hierarchy of features.

A correlation-based feature selection (CFS) method [57] with greedy stepwise search [13] was used in the analysis. This method was chosen because it uses a similar idea to information gain in the RTs to check what features are more significant. It additionally checks the correlation among features themselves. Greedy stepwise search was used because it allows ranking features. The main reason for using this CFS is its similarity to the working of RTs, with the key difference that simply using its selected features as inputs for a learning machine does not provide it with a hierarchy of relative importance of the features. So, CFS is particularly helpful for understanding the behaviour of RTs and how beneficial its use of the relative importance of features is to SEE. This filter method was also used instead of a wrapper method so that the same set of features can be used for different models, as explained bellow.

As a first step, we ran all the experiments using the framework presented in Section 4, but after performing feature selection. This study showed that feature selection by itself did not change the fact that RTs and Bag + MLPs were usually among the best in terms of MMRE, PRED(25) and MAE. The approaches that obtained most improvements in performance were Bag + RBF, RBF and NCL. However, the improvements were not large even for these approaches. Bag + RBF's SA average considering all data sets was 0.0522 higher, RBF's was 0.0533 and NCL's was 0.0484 higher than when not using features selection.

As a second step for this analysis, we compared the ranking of features given by feature selection against the features appearing in more than 50% of the RTs until their third level, for each data set. An example is shown in Table 17. The results show that: (1) the RTs do not use all the features selected by CFS, even though they usually use at least one of these; (2) the RTs use some features not selected by CFS; and (3) the RTs put higher ranked features according to CFS in higher levels of the tree, confirming the use of the relative importance of features.

So, feature selection by itself was not able to change the relative performance of different learning approaches. However, instead of simply using a subset with the most important features, RTs gave

Table 17
CFS ranking and RT features relative importance for cocomo81: features ranking, first tree level in which the feature appeared in more than 50% of the trees, and percentage of the trees in which it appears in that level. Features bellow the horizontal line are not selected by CFS.

Features ranking	Tree level	% Of trees
LOC	Level 0	100.00
Development mode	-	-
Required software reliabilit y	Level 1	90.00
Modern programing practices	-	-
Time constraint for CPU	Level 2	73.33
Data base size	Level 2	83.34
Main memory constraint	-	-
Turnaround time	-	-
Programmers capability	-	-
Analysts capability	-	-
Language experience	-	-
Virtual machine experience	-	-
Schedule constraint	-	-
Application experience	Level 2	66.67
Use of software tools	-	-
Machine volatility	-	-
Process complexity	-	-

more importance to more important features as shown by the feature ranking, being able to achieve comparatively good performance and suggesting that hierarchy of features is important when working with ML for SEE. This is an insight on how to improve SEE (RQ3), specially considering improvements in other learning machines than RTs, which might be improved by incorporating hierarchy of features.

It is also interesting to verify which features are usually at the top level of the hierarchy produced by the RTs, as these are considered to be the most influential features for the SEEs. The number of lines of code (LOC) and the functional size are the features that most frequently appear at the top level. Table 17 shows an example for cocomo81, for which 100% of the RTs used LOC at the top level (level 0). It is reasonable that this feature appears at the top of the hierarchy, as larger programs require involvement of many programmers, increasing the communication complexity and effort.

Nasa93 was the only data set where another feature than LOC or functional size appeared at the top level in more than 50% of the RTs. For this data set, 100% of the RTs used the feature CPU time constraint at the top level. Interestingly, even though nasa and nasa93 are two data sets from the same organisation, the RTs did not use the same feature at the top level for these two data sets. For nasa, 96.67% of the RTs used LOC at the top level. In order to better understand why this happened, we analysed the values of the feature CPU time constraint for these two data sets. As shown in Table 18, the standard deviation of this feature for nasa93 is much higher than that for nasa. A Levene test [58] shows that the difference in variance is statistically significant (p-value of 0.0036). This indicates that CPU time constraint varies more for nasa93 than for nasa. Very different CPU time constraints are likely to directly affect the difficulty of the software development, thus considerably influencing its required effort. For example, extra high CPU time constraint should require much more effort than low CPU time constraint. So, it is reasonable that this is considered as an important feature for nasa93.

It is also worth noting that the non-functional system features that are becoming more important in the modern systems could also affect the hierarchy provided by the RTs. For example, as writing secure code becomes more important for some companies, security-related features may raise in the hierarchy.

7.2. Combining ensembles and locality into tailored approaches

Section 5 shows that Bag + RT is a way to join the power of locality and automated ensembles to improve SEE over several other approaches. This is very encouraging for future research in the area of automated ensembles and locality for SEE. The reason is that even though Bag + RTs present several features that motivate their usage for SEE, tailored but automated approaches building upon the behaviour of this and other approaches should be able to improve performance even further. In this section, we analyse the potential of two types of combinations of locality and ensembles tailored for SEE, with the aim of providing more insight on how to improve SEE.

RTs and Bag + MLPs are two approaches that did considerably well in Section 5, even though not so well as Bag + RT. As RT is based on locality and Bag + MLPs is based on ensembles, we investigate the effect of two types of approach:

1. Using ensembles to combine several different learning machines which are based on feature rules, similarly to the locality of the RTs. In this case, ensembles are used at a higher level and locality at a lower level in an attempt to improve RTs. We call this approach "Mult", for multiple types of base learning machines. The base learning machines were RT, decision table, conjunctive rule and M5 Rules [13].
2. Using a clustering approach as a locality approach and creating a Bag + MLP for each training set cluster. In this case, a locality approach (k-means) is used at a higher level and ensembles at a lower level in an attempt to improve Bag + MLP. We call this approach "Clusb", for cluster bagging.

The main issue to be investigated here is when ensembles and locality at a higher or lower level are more likely to provide improvements for SEE, giving directions for future work.

Table 19 shows the improvement in SA obtained by Mult in comparison to RTs in italics. As we can see, the improvements still need to be increased in future work, being usually close to or better than 0.05 SA, but lower than 0.1 SA. The insight provided here is regarding the relationship between the improvement and data set size. The correlation between them is −0.3337. This is a reasonable negative correlation, showing that larger improvements may be achieved for smaller data sets. Several other features of the data sets are likely to also have some influence in the performance of the algorithms, making them better or worse for each specific data set. However, we can see that the size is a feature with considerable influence in this case, suggesting that such tailored ensembles at a higher level can be particularly helpful for smaller data sets.

It is worth noting, though that the effect of Mult can be detrimental when the data set is large. This shows that the base learning machines used in addition to RTs can be particularly good or detrimental depending on the data set, hindering Mult's performance in the latter case.

Table 20 shows the improvements obtained by Clusb in relation to Bag + MLPs (results for sdr were omitted as this data set is too small for clustering). Again, when there are improvements, they are smaller than 0.1 SA, but mostly close to or better than 0.05 SA. The correlation between improvement and data set size is 0.3055. Again, this is a reasonable amount of correlation considering SEE, and indicates that locality at a higher level can be mainly advantageous for larger data sets. As shown in Table 20, Clusb can be even detrimental for smaller data sets.

So, this section shows that approaches using tailored ensembles at a higher level may be particular useful for improving performance on smaller data sets, whereas approaches using locality at a higher level may be particularly useful for improving on larger

Table 18
CPU time constraint information.

Data set	Minimum	Maximum	Avg	Std. dev.
Nasa93	1.00	1.66	1.133	0.203
Nasa	1.00	1.66	1.076	0.138

Table 19
Difference in SA between Mult and RTs. Cells in italics the cases where Mult provides higher SA.

Mult's SA minus RT's SA	Data set size	Data set
−0.090502023	63	Cocomo81
−0.083715479	90	Nasa93
−0.083664355	58	Nasa
−0.025044093	444	OrgAll
−0.025036362	158	Org3
0.0187048882	***21***	***Org6***
0.0463227775	***20***	***Org5***
0.0481770822	***75***	***Org1***
0.0570139663	***21***	***Org7***
0.0593049005	***117***	***Org4***
0.0657496405	***32***	***Org2***
0.0780584515	***74***	***Desharnais***
0.0872040493	***12***	***Sdr***

Table 20
Difference in SA between Clusb and Bag+MLPs. Cells in italics highlight the cases where Clusb provides higher SA.

Clusb's SA minus Bag+MLP's SA	Data set size	Data set
−3.532254646	20	Org5
−1.407062325	32	Org2
−0.096944035	58	Nasa
−0.078153916	21	Org7
−0.062928635	74	Desharnais
−0.062255486	21	Org6
−0.017828149	75	Org1
0.0281382156	***63***	***Cocomo81***
0.0460938455	***158***	***Org3***
0.0684082225	***117***	***Org4***
0.0805119301	***90***	***Nasa93***
0.0911181081	***444***	***OrgAll***

data sets. Future work on joining ensembles and locality may benefit from exploiting that.

8. Threats to validity

Internal validity regards to establishing that a certain observable event was responsible for a change in behaviour. It is related to the question "Is there something other than the treatment that could cause the difference in behaviour?" [59]. In ML, it is essential to use a principled experimental framework. In our study, we followed a framework that joins the power of statistical tests to the importance of the magnitude of the differences in performance, besides considering parameters selection as an explicit step in order to deal with internal validity.

Construct validity regards to accurately naming our measures and manipulations [59]. MMRE and PRED(25) can be biased and were used in part of the work to allow comparison of our conclusions to previous works based on MRE measures. However, we based our conclusions mainly on MAE, which is a non-asymmetric and unbiased performance measure. SA is also an unbiased measure recently proposed by Shepperd and Mc Donell [21] and was used in order to provide more easily interpretable results.

External validity regards to generalising the study's results outside the study to other situations [59]. Typical external validity issues in ML are related to the use of few samples. In the present study, we used thirteen different data sets containing a large variety of projects from different organisations and countries in order to deal with this issue. This number is more than twice the number of data sets used in previous work on automated ensembles for SEE, besides considering both PROMISE and ISBSG data sets.

9. Conclusions

This paper presents a principled and comprehensive evaluation of ensembles of learning machines for SEE, an analysis of different locality approaches, an experimental framework for evaluating SEE models and several insights on improving SEE. In this section, we revisit the research questions and summarise the main content of their answers.

RQ1: Do existing automated ensembles of learning machines generally improve effort estimations given by single learning machines, including potentially adequate locality approaches such as RTs? Which of them would be more useful?

When considering MAE and SA as performance measures, Bag + RTs is shown to perform well. It is highly ranked in terms of performance across multiple data sets, it is frequently among the best approaches for each data set, and rarely performs considerably worse than the best approach for any data set. Overall, it performs better than several other approaches, including RTs. This is an inspiring result for future work on joining the power of ensembles to locality based on automated approaches. Bag + MLPs, which is an ensemble approach not benefiting from locality, performed similarly to single RTs.

RQ2: What locality approach is more adequate for SEE tasks? In particular, how well does RT locality do in comparison to other locality approaches? On what type of data sets?

RTs, EM, *k*-means, SC and *k*-NN were considered all statistically similar across multiple data sets. So, all of them could be further exploited in future work to improve SEE. The clusters provided by *k*-means and the locality provided by *k*-NN may be particularly helpful for more heterogeneous data sets. However, when *k*-means and *k*-NN are not beneficial, they can get very poor performance. RTs are more reliable than the other approaches analysed. EM and SC sometimes fail to separate projects into more than one cluster.

RQ3: What insight on how to further improve software effort estimation can we gain by analysing competitive ensemble and locality approaches?

Our paper provides several insights, which are based on extensive experimentation and analyses:

- Bag + RTs do particularly well in comparison to several other approaches, showing that comparatively well performing fully automated approaches based on ensembles are possible and could probably be further improved in terms of magnitude of performance by being tailored for SEE.
- Single RTs and Bag + MLPs perform slightly worse than Bag + RTs, but still perform well in comparison to other approaches. The analyses show that both locality and ensembles can be beneficial to SEE, and that joining these two types of approach can improve results further. Future work on improving SEE through automated approaches may benefit from exploiting that further.
- Bag + MLP has room for improvements in terms of the choice of base model to be used for performing predictions.
- When considering MMRE and PRED(25), which are based on the asymmetric measure MRE, RTs and Bag + MLPs are singled out as frequently highly ranked. Bag + RTs perform less well in terms of these measures. So, the slightly worse performance of RTs and Bag + MLPs in comparison to Bag + RTs in terms of MAE may be related to the fact that RTs and Bag + MLPs suffer more from underestimations. This issue should be further analysed as future work and could be helpful for proposing improvements of these approaches for SEE.
- Future work on locality may benefit not only from "informed" locality learnt based on the target effort of training examples, such as RT's locality, but also from clustering approaches such as *k*-means and traditional approaches such as *k*-NN. Care must be taken, though, as *k*-means and *k*-NN are less reliable, possibly performing particularly bad for databases to which they are not beneficial.
- RT's locality places more important features in higher levels of the trees, which is part of the reason for their better performance. The benefits of such a hierarchy of features go beyond simply using a correlation-based feature selector. So, feature hierarchy might be used for improving SEE where RTs are not employed.
- Approaches using tailored ensembles at a higher level may be particular useful for improving performance on smaller data sets. Approaches using locality at a higher level may be particularly useful for improving on larger data sets. Future work on joining ensembles and locality may benefit from exploiting that.

RQ4: How to evaluate/choose ML models for SEE?

An experimental framework was proposed, considering the choice of approaches, evaluation procedure and parameters explicitly. Evaluation of different approaches across different data sets is recommended to follow three steps based on (1) Friedman ranking and test, (2) identification of the approaches among the best and the characteristics of the data sets to which they tend to perform better, and (3) how frequently the approaches perform considerably worse than the best. As no approach is always the best for all data sets, ideally, an organisation should perform experiments following a principled framework considering its available data in order to choose a ML model. However, if an organisation has no resources to perform such experiments, Bag + RTs are recommended, as they perform comparatively well across a large range of data sets, and rarely perform worse than the best approach for any data set in more than 0.1 units of SA. Even though RTs perform slightly worse, they could also be used should the software manager wish to easily understand the rules underlying the model's behaviour. In summary, this work provides an extensive analysis and several insights that can be used by future works on improving SEE.

Acknowledgments

The authors would like to thank all the participants of PROMISE'11 and especially Dr. Tim Menzies and Prof. Martin Shepperd for the fruitful discussions and suggestions. We would also like to thank Dr. Peter Coxhead and Dr. Rami Bahsoon for their help and advice, and the anonymous reviewers for their constructive comments, which have helped to improve the quality of this paper significantly. This work was supported by EPSRC Grants (Nos. EP/D052785/1 and EP/J017515/1) and European Commission through its FP7 Grant (No. 270428).

References

[1] R. Agarwal, M. Kumar, Y. Mallick, R. Bharadwaj, D. Anantwar, Estimating software projects, Software Engineering Notes 16 (4) (2001) 60–67.
[2] M. Jorgensen, M. Shepperd, A systematic review of software development cost estimation studies, IEEE Transactions on Software Engineering 33 (1) (2007) 33–53.
[3] K. Srivasan, D. Fisher, Machine learning approaches to estimating software development effort, IEEE Transactions on Software Engineering 21 (2) (1995) 126–137.
[4] G. Wittig, G. Finnie, Estimating software development effort with connectionist models, Information and Software Technology 39 (1997) 469–476.
[5] A. Heiat, Comparison of artificial neural network and regression models for estimating software development effort, Information and Software Technology 44 (2002) 911–922.
[6] B. Baskeles, B. Turhan, A. Bener, Software effort estimation using machine learning methods, in: International Symposium on Computer and Information Sciences, Ankara, 2007, pp. 1–6.
[7] I.F.B. Tronto, J.D.S. Silva, N. Sant'Anna, Comparison of artificial neural network and regression models in software effort estimation, in: International Joint Conference on Neural Networks, Orlando, 2007, pp. 771–776.
[8] Y. Kultur, B. Turhan, A. Bener, Ensemble of neural networks with associative memory (ENNA) for estimating software development costs, Knowledge-Based Systems 22 (2009) 395–402.
[9] P.L. Braga, A. Oliveira, G. Ribeiro, S. Meira, Bagging predictors for estimation of software project effort, in: International Joint Conference on Neural Networks, Orlando, 2007, pp. 1595–1600.
[10] E. Kocaguneli, A. Bener, Y. Kultur, Combining multiple learners induced on multiple datasets for software effort prediction, in: International Symposium on Software Reliability Engineering, Mysuru, India, 2009, p. 6.
[11] E. Kocaguneli, T. Menzies, J. Keung, On the value of ensemble effort estimation, IEEE Transactions on Software Engineering (2011) 14. http://dx.doi.org/10.1109/TSE.2011.111.
[12] L. Breiman, Bagging predictors, Machine Learning 24 (2) (1996) 123–140.
[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, SIGKDD Explorations 11 (1) (2009) 10–18.
[14] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks 12 (1999) 1399–1404.
[15] J. Sayyad Shirabad, T. Menzies, The PROMISE Repository of Software Engineering Databases., School of Information Technology and Engineering, University of Ottawa, Canada 2005 <http://promise.site.uottawa.ca/SERepository>.
[16] ISBSG, The International Software Benchmarking Standards Group., 2011 <http://www.isbsg.org>.
[17] T. Menzies, S.P.A.C.E. exploration for software engineering, 2011 <http://crest.cs.ucl.ac.uk/?id=3695>.
[18] J. Cuadrado Gallego, D. Rodriguez, M. Sicilia, M. Rubio, A. Cresp, Software project effort estimation based on multiple parametric models generated through data clustering, Journal of Computer Science and Technology 22 (3) (2007) 371–378.
[19] E. Kocaguneli, T. Menzies, A. Bener, J.W. Keung, Exploiting the essential assumptions of analogy-based effort estimation, IEEE Transactions on Software Engineering <http://doi.ieeecomputersociety.org/10.1109/TSE.2011.27>.
[20] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.
[21] M. Shepperd, S. McDonell, Evaluating prediction systems in software project estimation, Information and Software Technology (2012) 21. <http://dx.doi.org/10.1016/j.infsof.2011.12.008>.
[22] H. Chen, X. Yao, Regularized negative correlation learning for neural network ensembles, IEEE Transactions on Neural Networks 20 (12) (2009) 1962–1979.
[23] G. Brown, J.L. Wyatt, P. Tiño, Managing diversity in regression ensembles, Journal of Machine Learning Research 6 (2005) 1621–1650.
[24] G. Brown, J. Wyatt, R. Harris, X. Yao, Diversity creation methods: a survey and categorisation, Journal of Information Fusion 6 (2005) 5–20.
[25] L.I. Kuncheva, C.J. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2003) 181–207.
[26] B.A. Kitchenham, E. Mendes, G.H. Travassos, Cross versus within-company cost estimation studies: a systematic review, IEEE Transactions on Software Engineering 33 (5) (2007) 316–329.
[27] M. Shepperd, G. Kadoda, Comparing software prediction techniques using simulation, IEEE Transactions on Software Engineering 27 (11) (2001) 1014–1022.
[28] T. Menzies, Z. Chen, J. Hihn, K. Lum, Selecting best practices for effort estimation, IEEE Transactions on Software Engineering 32 (11) (2006) 883–895.
[29] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, A simulation study of the model evaluation criterion mmre, IEEE Transactions on Software Engineering 29 (11) (2003) 985–995.
[30] J. Cohen, A power primer, Psychological Bulletin 112 (1992) 155–159.
[31] F. Wilcoxon, Individual comparisons by ranking methods, Biometrics 1 (6) (1945) 80–83.
[32] B. Boehm, Software Engineering Economics, Prentice-Hall, Englewood Cliffs, NJ, 1981.
[33] B. Boehm, C. Abts, A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D.J. Reifer, B. Steece, Software Cost Estimation with COCOMO II, Prentice-Hall, Englewood Cliffs, NJ, 2000.
[34] M. Shepperd, C. Schofield, Estimating software project effort using analogies, IEEE Transactions on Software Engineering 23 (12) (1997) 736–743.
[35] Y.-S. Seo, K.-A. Yoon, D.-H. Bae, An empirical analysis of software effort estimation with outlier elimination, in: Predictive Models in Software Engineering, Leipzig, 2008, pp. 25–32.
[36] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Computational and Applied Mathematics 20 (1987) 53–65.
[37] L. Minku, X. Yao, Using unreliable data for creating more reliable online learners, in: International Joint Conference on Neural Networks, Brisbane, Australia, 2012, pp. 2492–2499.
[38] L. Minku, X. Yao, Can cross-company data improve performance in software effort estimation? in: Predictive Models in Software Engineering, Lund, Sweden, 2012, p. 10.
[39] M. Cartwright, M. Shepperd, Q. Song, Dealing with missing software project data, in: International Software Metrics Symposium, Sydney, 2003, pp. 154–165.
[40] J. Hartigan, Clustering Algorithms, John Wiley & Sons, New York, 1975.
[41] Y. Zhao, Y. Zhang, Comparison of decision tree methods for finding active objects, Advances in Space Research 41 (2008) 1955–1959.
[42] C. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, United Kingdom, 2005.
[43] Y. Liu, X. Yao, Simultaneous training of negatively correlated neural networks in an ensemble, IEEE Transactions on Systems, Man and Cybernetics – Part B: Cybernetics 29 (6) (1999) 716–725.
[44] E. Frank, L. Trigg, G. Holmes, I.H. Witten, Technical note: Naive bayes for regression, Machine Learning 41 (2000) 5–25.
[45] E. Bauer, R. Kohavi, An empirical comparison of voting classification algorithms: bagging, boosting, and variants, Machine Learning 36 (1999) 105–139.
[46] V. Zorkadis, D.A. Karras, M. Panayoto, Efficient information theoretic strategies for classifier combination, feature extraction and performance evaluation in improving false positives and false negatives for spam e-mail filtering, Neural Networks 18 (5–6) (2005) 799–807.
[47] V. Zorkadis, D.A. Karras, Efficient information theoretic extraction of higher order features for improving neural network-based spam e-mail categorization, Journal of Experimental and Theoretical Artificial Intelligence 18 (4) (2006) 523–534.
[48] R. de Aquino, A. Ferreira, M. Carvalho, O. Neto, G. Santos, Combining multiple artificial neural networks using random committee to decide upon electrical disturbance classification, in: International Joint Conference on Neural Networks, Orlando, Florida, 2007, pp. 2863–2868.

[49] V. Tresp, Handbook of Neural Network Signal Processing, CRC Press, USA, 2001 (Ch. 5: Committee machines).
[50] M.M. Islan, X. Yao, K. Murase, A constructive algorithm for training cooperative neural network ensembles, IEEE Transactions on Neural Networks 14 (4) (2003) 820–834.
[51] L. Breiman, Heuristics of instability and stabilization in model selection, The Annals of Statistics 24 (6) (1996) 2350–2383.
[52] C. Bishop, Pattern Recognition and Machine Learning, Springer, Singapore, 2006.
[53] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.
[54] R. Kannan, S. Vempala, A. Vetta, On clusterings – good, bad and spectral, Tech. rep., CS Dept., Yale University, 2000.
[55] L. Dragone, Spe ctral clusterer for WEKA, 2010 <http://www.luigidragone.com/software/spectral-clusterer-for-weka/>.
[56] L. Minku, X. Yao, A principled evaluation of ensembles of learning machines for software effort estimation, in: Predictive Models in Software Engineering, Banff, Canada, 2011, p. 10 <http://dx.doi.org/10.1145/2020390.2020399>.
[57] M.A. Hall, L.A. Smith, Practical feature subset selection for machine learning, in: Australasian Computer Science Conference, Perth, Australia, 1998, pp. 181–191.
[58] H. Levene, Contributions to Probability and Statistics: Essays in Honor of Harold Hotelling, first ed., Stanford University Press, USA, 1960.
[59] M.L. Mitchell, J.M. Jolley, Research Design Explained, seventh ed., Cengage Learning, USA, 2010.