A Principled Evaluation of Ensembles of Learning Machines for Software Effort Estimation

Leandro L. Minku, Xin Yao
School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
{l.l.minku, x.yao}@cs.bham.ac.uk

ABSTRACT

Background: Software effort estimation (SEE) is a task of strategic importance in software management. Recently, some studies have attempted to use ensembles of learning machines for this task.
Aims: We aim at (1) evaluating whether readily available ensemble methods generally improve SEE given by single learning machines and which of them would be more useful; getting insight on (2) how to improve SEE; and (3) how to choose machine learning (ML) models for SEE.
Method: A principled and comprehensive statistical comparison of three ensemble methods and three single learners was carried out using thirteen data sets. Feature selection and ensemble diversity analyses were performed to gain insight on how to improve SEE based on the approaches singled out. In addition, a risk analysis was performed to investigate the robustness to outliers. Therefore, the better understanding/insight provided by the paper is based on principled experiments, not just an intuition or speculation.
Results: None of the compared methods is consistently the best, even though regression trees and bagging using multilayer perceptrons (MLPs) are more frequently among the best. These two approaches usually perform similarly. Regression trees place more important features in higher levels of the trees, suggesting that feature weights are important when using ML models for SEE. The analysis of bagging with MLPs suggests that a self-tuning ensemble diversity method may help improving SEE.
Conclusions: Ideally, principled experiments should be done in an individual basis to choose a model. If an organisation has no resources for that, regression trees seem to be a good choice for its simplicity. The analysis also suggests approaches to improve SEE.

Categories and Subject Descriptors

D.2.9 [**Software Engineering**]: Management—*Cost estimation*; I.2.6 [**Artificial Intelligence**]: Learning—*Concept learning, Connectionism and neural nets*

PROMISE '11, September 20–21, 2011, Banff, Canada

General Terms

Experimentation

Keywords

Software cost/effort estimation, machine learning, ensembles of learning machines

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) methods have been proposed. We recommend Jorgensen and Shepperd's work [17] for a systematic review. Among the proposed methods, effort estimators based on machine learning (ML) approaches such as multilayer perceptrons (MLPs), radial basis function (RBF) networks and regression trees (RTs) [29, 33, 15, 2, 30, 21, 6] have been receiving increased attention [17]. 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. [30] 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, Braga et al. [6], Kultur et al. [21] and Kocaguneli et al. [19] investigated the use ensembles of learning machines for SEE. Ensembles of learning machines are sets of learners[1] trained to perform the same task and combined with the aim of improving predictive performance [9]. When combining learners in an attempt to get more accurate predictions, it is commonly agreed that the learners should behave differently from each other. Otherwise, the overall prediction will not be better than the individual predictions.

[1]The learners used to compose an ensemble are frequently called base learners.

So, different ensemble learning approaches can be seen as different ways to generate diversity among the base learners.

The study presented by Braga et al. [6] claims that Bootstrap Aggregating (Bagging) [7], a well known ensemble learning approach, improves the SEEs produced by several single learner methods, 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 learners, it is not possible to conclude that there was an improvement in the estimations [18].

The study presented by Kultur et al. [21] uses five data sets and shows that an adapted version of bagging provides very large improvements in comparison to single learners. 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.

The study presented by Kocaguneli et al. [19] uses three data sets to compare the effect of combining different *types* of learners to the use of these learners separately. Even though it is not exactly a comparison between single learners and ensembles, their results can suggest that ensembles do not improve the performance of single learners. This is somewhat contradictory in relation to Kultur et al. [21]'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 the papers compare the results obtained by different readily available ensemble learning approaches from the ML literature. Kocaguneli et al. [19] present results obtained by some ensemble methods, but the analysis does not perform a statistical comparison among these methods and single learners. 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.

With that in mind, this paper addresses the following research questions, still left unanswered by previous work:

- RQ1: Do readily available ensemble methods generally improve effort estimations given by single learners? Which of them would be more useful?
- RQ2: If a particular method is singled out, what insight on how to improve software effort estimation can we gain by analysing its behaviour and the reasons for its better performance?
- RQ3: How can someone determine what ML model to be used considering a particular data set?

The study performs a principled and comprehensive statistical comparison of three different types of ensembles and three different types of single learners using several data sets (five data sets from the PRedictOr Models In Software Engineering Software (PROMISE) Repository [26] and eight data sets derived from the International Software Benchmarking Standards Group (ISBSG) Repository [16] Release 10). Experiments are also performed to provide insight on how to use machine learning for SEE and how to improve SEE based on machine learning. It is worth noting that our key contribution is not in a new algorithm or methodology, but a better understanding/insight. Furthermore, such a better understanding/insight is based on experimental studies, not just an intuition or speculation.

It is worth noting that there are papers in the literature that investigate issues related to RQ3. An example is Menzies et al.'s work [24], which explicitly considers certain preprocessing as a part of the evaluation framework. However, little importance is usually given to the parameters choice of machine learning methods. A wrong parameters choice can greatly affect model choice, being a serious internal validity problem. The present paper includes the parameters choice as an explicit step in the framework for choosing SEE models, which is part of its contribution to answer RQ3.

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

- Use of principled experimentation, considering both parameter choice and statistical tests.
- Comparison using three different ensemble methods.
- Use of a larger number of data sets (thirteen against five, the highest number of data sets previously used).
- Principled experimental analysis of the behaviour of the methods that are most frequently among the best, gaining more insight on how to improve SEE.
- Risk analysis to evaluate the impact of outlier projects.

Some work in the literature suggests that the performance of different models significantly depend on the characteristics of the data set [28]. However, existing work on ensembles of learning machines suggests that they may provide better results than single learners even when several different data sets are used [21], as mentioned previously. Our initial analysis based on a comparison algorithm provided by Menzies et al. [24] shows that none of the compared approaches is consistently the best over all data sets. It further confirms the dramatic impact that different data sets have on the behaviour of SEE models, even when ensembles of learning machine are used. It shows that it is very unlikely that there is a universally best method. Ideally, principled experiments should be carried out in an individual basis to choose a model.

Nevertheless, two approaches are most frequently ranked as the first two in terms of performance: RTs and bagging using MLPs. So, they are recommended over the others if an organisation has no resources to perform experiments to choose a model. Besides, these approaches usually perform similarly to each other, showing us that ensembles are not generally better than well trained and tuned single learners for SEE.

Further analysis shows that RTs put more important features in higher levels of the trees, which is probably the reason for their better performance. The analysis suggests that feature *weighting* is important when working with ML for SEE, providing an insight on how to improve SEE.

The analysis done with bagging using MLPs shows that ensemble diversity might be linked to its better performance

in comparison to other ensemble approaches. However, the main result from this analysis is that the correlation between ensemble diversity and test accuracy varies dramatically depending on the data set. An approach that can perform self-tuning of diversity depending on its correlation with train accuracy may be able to improve SEE.

The rest of this paper is organised as follows. Section 2 describes the data sets used in the study. Section 3 explains the experimental framework. Section 4 presents the results of the comparisons among different learning machines, aiming at answering RQ1. Section 5 presents further analysis done to understand why certain learning machines were singled out and what insight that provides us to improve SEE. It aims at answering RQ2. RQ3 is answered by a combination of sections 3, 4 and 5. Section 6 provides a risk analysis to check the impact of outliers on the estimations given by the learning machines. Section 7 explains the validity of the work. Section 8 presents conclusions and future work.

2. 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 [26] and from the International Software Benchmarking Standards Group (ISBSG) Repository [16] Release 10. The data sets were chosen to cover a wide range of features, such as number of projects, types of features, countries and companies. Sections 2.1 and 2.2 provide their description and explanation of how they were processed.

2.1 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 [4]. Nasa93 and nasa are two data sets containing Nasa projects from 1970's-1980's and from 1980's-1990's, respectively. Sdr contains projects implemented in 2000's and was collected at Bogazici University Software Engineering Research Laboratory from software development organisations in Turkey. Desharnais' projects are dated from late 1980's. Table 1 provides some details about these data sets. The next subsections explain their features, missing values and outliers.

2.1.1 Features

Cocomo81, nasa93 and nasa are based on the COCOMO [4] 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 [5], 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, 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. Actual effort in person-hours is the dependent variable.

2.1.2 Missing Values

The only data set with missing values is desharnais. In total, it contains only 4 in 81 projects with missing values. So, these projects were eliminated.

2.1.3 Outliers

The literature shows that SEE data sets frequently have a few outliers, which may hinder the SEEs for future projects [27]. In the current work, outliers were detected using k-means. This method was chosen because it has shown to improve performance in the SEE context [27]. K-means is used to divide the projects into clusters. The silhouette value for each project represents the similarity of the project to the other projects of its cluster in comparison to projects of the other clusters, ranging from -1 (more dissimilar) to 1 (more similar). So, the average silhouette value can be used to determine the number of clusters k. After applying k-means to the data, clusters with less than a certain number n of projects or projects with negative silhouette values are considered outliers.

We used $n = 3$, as in [27]. The number of clusters k was chosen among $k = \{2, 3, 4, 5\}$, according to the average silhouette values. The highest average silhouette values were always for $k = 2$ and were very high for all data sets (between 0.8367 and 0.9778), indicating that the clusters are generally homogeneous. The number of outliers was also small (from none to 3). The projects considered as outliers were eliminated from the data sets, apart from the outlier identified for sdr. As this data set is very small (only 11 projects), there is not enough evidence to consider the identified project as an outlier.

2.2 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,

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	1,821.51
Nasa93	93	17	8.4	8,211	624.41	1,135.93
Nasa	60	16	8.4	3,240	406.41	656.95
Sdr	12	23	1	22	5.73	6.84
Desharnais	81	9	546	23,940	5,046.31	4,418.77

Table 2: 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

following ISBSG's data set size guidelines. The resulting organisation types are shown in table 2.

Table 3 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 $<$ 2.2e-16).

The next sections explain how the features were selected, how to deal with the missing values and outliers.

2.2.1 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 has more than 40% missing feature values 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.

2.2.2 Missing Values

The features "functional size" and "development type" have no missing values. The feature "language type" is missing in several subsets, but it is never missing in more than 40% of the projects of any subset.

So, an imputation method based on k-Nearest Neighbours (k-NN) was used so that this feature can be kept without having to discard the projects in which it is missing. K-NN imputation has shown to be able to improve SEEs [8]. It is particularly beneficial for 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 [8].

According to Cartwright et al. [8], "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.

2.2.3 Outliers

Similarly to the PROMISE data sets (section 2.1), outliers were detected through k-means [14] and eliminated. K was chosen among $k = \{2, 3, 4, 5\}$ based on the average silhouette values. The chosen k was 2 for subsets 1 to 5 and 3 for subsets 6 and 7. As with the PROMISE data sets, the silhouette values were high (between 0.8821 and 0.9961), showing that the clusters are homogeneous. The number of outliers varied from none to 5. None of the data sets were reduced to less than 20 projects after outliers elimination.

3. 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 method 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 RQ1 and RQ3, which regard the comparison of several different methods and how to choose a ML model for SEE. The experimental procedure for answering RQ2 is further detailed in section 5.

3.1 Choice of Learning Machines

The following learning machines were used:

- Single learners: MultiLayer Perceptrons (MLPs) [3]; Radial Basis Function networks (RBFs) [3]; and Regression Trees (RTs) [34].
- Ensemble learners: Bagging [7] with MLPs, with RBFs and with RTs; Random [12] with MLPs; and Negative Correlation Learning (NCL) [23, 22] with MLPs.

All the learning machines but NCL were based on the Weka implementation [12]. 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 software used for NCL is available upon request.

MLPs were chosen because they are widely used learning machines that can approximate any continuous function.

Table 3: 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	134211	4081.64	15951.03	1.2	75.2	12.71	12.58
2	53	499	225.44	135.12	737	14040	3218.50	3114.34	4.5	55.1	15.05	9.94
3	3	893	133.24	154.42	4	20164	2007.10	2665.93	0.3	43.5	17.37	9.98
4	32	3088	371.41	394.10	360	60826	5970.32	8141.26	1.4	97.9	18.75	16.69
5	17	13580	1112.19	2994.62	762	54620	8842.62	11715.39	2.2	52.5	23.38	14.17
6	50	1278	163.41	255.07	361	28441	4855.41	6093.45	5.6	60.4	30.52	17.70
7	51	615	160.10	142.88	867	19888	6960.19	5932.72	14.4	203.8	58.10	61.63

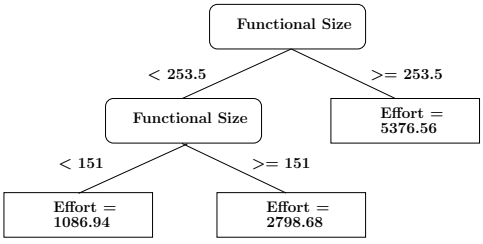

Figure 1: An example of RT for effort estimation.

RBFs perform local learning, which could be particularly useful for SEE, as each data set can be very heterogeneous. RTs were chosen for their simplicity – the rules they use to make estimations can be easily understood by the project manager. An example of RT is shown in figure 1.

Bagging is one of the most well known ensemble learning approaches in the literature. It creates diversity by training each base learner 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 learners into strong ones. This can be particularly advantageous for SEE, as there are usually very few projects to be used for learning, producing very inaccurate base learners that may be considered as weak. On the other hand, each base learner in the ensemble is trained with only about 63.2% of the unique examples from the available original training set. So, another approach called random ensemble was also used. It trains all the base learners using the same training set and diversity is created by simply using a different random seed to create each base learner. The problem of this approach is that there is no guarantee that a good level of diversity will be achieved. NCL was chosen for having strong theoretical foundations for regression problems. Its disadvantage is that it is usually used with strong learners. Besides, NCL can only be used with neural networks. Other learning machines such as RTs cannot be currently used.

3.2 Choice of Evaluation Method

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. [24] 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 the data set.

The experiments use the PROMISE data sets explained in section 2.1, the ISBSG subsets explained in section 2.2 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.

Algorithm 1 Worse(x,y)

```
1: if StatisticallyDifferent(x,y) then
2:    // Rule 1 – compares MMREs
3:    if MMRE(x) < MMRE(y) Return y
4:    if MMRE(y) < MMRE(x) Return x
5: else
6:    // Rule 2 – compares correlations
7:    if correlation(x) < correlation(y) Return x
8:    if correlation(y) < correlation(x) Return y
9:    // Rule 3 – compares std deviation of the estimations
10:   if stdev(x)/mean(x) < stdev(y)/mean(y) Return y
11:   if stdev(y)/mean(y) < stdev(x)/mean(x) Return x
12:   // Rule 4 – compares PRED(N)
13:   if PRED(N)(x) < PRED(N)(y) Return x
14:   if PRED(N)(y) < PRED(N)(x) Return y
15:   // Rule 5 – compares the number of features
16:   // used by the treatment
17:   if #features(x) < #features(y) Return y
18:   if #features(y) < #features(x) Return x
19: end if
20: Return 0
```

The following measures of performance were used in this work [24, 21]: Mean Magnitude of the Relative Error (MMRE), Median of the Magnitude of the Relative Error (MdMRE), Percentage of Estimates within 25% of the actual values (PRED(25)) and correlation between estimated and actual effort (Corr). Unless stated otherwise, the analysis will always refer to the measures calculated on the test set. Despite the MRE's problems identified by Foss et al. [11], Kocaguneli et al. [20] affirm that in practice the probability of obtaining an error in the evaluation due to the fact that MRE divides the error by the actual effort is insignificant and would not have a major impact in the evaluation.

The comparisons followed two steps. The first step was based on Menzies et al.'s work [24], which presents an algorithm that applies a set of heuristic rejection rules to comparatively assess results from different treatments/models in SEE (algorithm 1). The rules are applied to each pair of models x and y and the model considered as "worse" is eliminated from the set of models. In the end of the procedure, the survivor models are printed. The statistical test in the first line compares x's and y's MMREs using a two-tailed t-test at the 95% confidence interval. This procedure could be used for choosing a model to be used (RQ3).

Table 4: 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 learners = $\{10, 25, 50\}$ All the possible parameters of the adopted base learners, as shown above
NCL	Penalty strength = $\{0.3, 0.4, 0.5\}$

In the second step, two-tailed paired t-tests [32] at the 95% confidence interval were used for comparing the two methods that present most often the highest MMRE and PRED(25). Non-parametric Wilcoxon tests [31] were also done to confirm the validity of the analysis. In this way, an assessment of the methods that are not necessarily the best, but are usually among the best, is done. This step build on to answer RQ1 and RQ3.

3.3 Choice of Parameters

The choice of parameters is a critical step in ML experiments. A mistaken parameters choice can affect the results of the experiments in such a way that a learning machine that would have better performance appears to have worse performance. So, it is essential that the method used for choosing the parameters is made clear in papers using ML. When choosing a model for SEE (RQ3), it is also essential to perform a principled parameters choice.

In order to choose the parameters, 5 preliminary rounds of executions were done using all the combinations of parameters shown in table 4 for each data set. 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 space limitations. The performance measure MMRE was chosen for representing the first rule in Menzies et al.'s work [24].

4. COMPARISON AMONG APPROACHES

The experiments performed according to the framework presented in section 3 show that different data sets obtained different performances. The MMRE obtained 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 correlations varied from 0.05 to 0.91. The performance values were omitted due to space limitations. Section 4.1 and 4.2 present the first and second steps of the analysis, as explained in section 3.2.

Table 5: Number of Data Sets in which Each Method Survived According to Algorithm 3.2. Methods that never survived are omitted.

PROMISE Data	ISBSG Data	All Data
RT: 2	MLP: 2	RT: 3
Bag + MLP: 1	Bag + RTs: 2	Bag + MLP: 2
NCL + MLP: 1	Bag + MLP: 1	NCL + MLP: 2
Rand + MLP: 1	RT: 1	Bag + RTs: 2
	Bag + RBF: 1	MLP: 2
	NCL + MLP: 1	Rand + MLP: 1
		Bag + RBF: 1

4.1 Step 1: Survivors

Table 5 shows the number of data sets in which each method survived according to Menzies et al.'s algorithm [24]. There is no indication that ensembles are generally better than single learners: both of them are among the methods that survived more times. It is not possible to note general trends presented by either PROMISE or ISBSG data either. The results show that different methods survive depending on the data set. The same situation happens if the statistical tests used for comparing MMRE are based on non-parametric Wilcoxon tests [31], instead of t-tests. This section provides part of the answer to RQ3: ideally, experiments using a principled framework need to be done in an individual basis using the set of projects to which the manager has access for choosing a model to be used for SEE.

4.2 Step 2: Approaches usually among the best

In this section, we first determine which methods are most often ranked as the first or second in terms of MMRE and PRED(25). We focus on finding out the methods that are usually among the best, instead of the method that is the best for each data set. In this way, it also helps the identification of which ensemble method would be generally preferable, which is part of the RQ1 to be answered by this paper. As shown in this section, the methods singled out are RTs and bagging using MLPs. After that, we compare RTs and bagging using MLPs. As they usually perform similarly, we can conclude that readily available ensemble methods do not generally improve the SEEs given by single learners. This provides the remaining of the answer to RQ1.

In order to check how valid this analysis is, it is important to note the statistically significance of outlying the first and second ranked methods. Two-tailed paired t-tests at the 95% confidence interval show that the difference between the average MMRE of the first ranked method and each other method is statistically significant in 35.16% of the cases. The difference between the second ranked method and the lower ranked methods is statistically significant in 16.67% of the cases. That means that the first and second ranked approaches are similar to the other approaches in terms of MMRE and PRED(25) in most cases, but are sometimes better. So, in general, it is preferable to use them.

Table 6(a) shows the two methods most often ranked as first and second in terms of MMRE. The results show that both RTs and bagging using MLPs are very often among the first two ranked methods according to MMRE. The trend can be observed both in the PROMISE and ISBSG data sets. For PROMISE, RTs or bagging using MLPs appear among the first two ranked approaches in 70% of the cases,

Table 6: Number of Data Sets in which Each Method Was Ranked First or Second According to MMRE and PRED(25). Methods never among the first and second are omitted.

(a) Accoding to MMRE

PROMISE Data		ISBSG Data		All Data	
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

(b) Acording to PRED(25)

PROMISE Data		ISBSG Data		All Data	
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 7: P-values of the two-tailed paired t-tests comparing MMRE and PRED(25) for RTs and bagging using MLPs. P-values less than 0.05 indicate statistically significant difference at the 95% confidence interval and are marked with "t" or "b" if RT or bagging using MLPs has the lower/higher MMRE/PRED(25).

Data Set	P-value	
	MMRE	PRED(25)
Cocomo81	0.8723	0.0003 b
Nasa93	0.0001 t	0.0000 t
Nasa	0.0073 t	0.0001 b
Sdr	0.3256	0.2339
Desharnais	0.1014	0.5829
Org1	0.7634	0.7170
Org2	0.3820	0.3449
Org3	0.8400	0.6004
Org4	0.3552	0.0325 t
Org5	0.9604	0.1423
Org6	0.0059 b	0.2807
Org7	0.9040	0.3331
OrgAll	0.6522	0.0031 t

whereas all other methods together sum up to 30%. For ISBSG, RTs or bagging using MLPs appear among the first two ranked in 62.5% of the cases, whereas all other methods together sum up to 37.5%. So, in general, RTs and bagging using MLPs are preferable over the other methods.

The analysis considering the first two ranked approaches according to PRED(25) shows that both RTs and bagging using MLPs are still frequently among the first two ranked (table 6(b)). If we consider PROMISE data by itself, ensembles such as bagging using MLPs are more frequently ranked higher than single learners. A comparison between bagging using MLPs and RTs would also be helpful to check whether this difference in the ranking is significant.

Two-tailed paired t-tests at the 95% confidence interval were done to compare both MMRE and PRED(25) for RTs and bagging using MLPs. Table 7 shows the results of the comparison. As we can see, the two approaches have statistically equal MMRE in most data sets, with RTs winning twice and bagging winning once. There are more statistically significant differences considering PRED(25). However, even so, RTs win three times and bagging wins twice. It is not possible to identify any tendency for either PROMISE or ISBSG data sets, suggesting that the difference in the PRED(25) ranking for RTs in PROMISE and ISBSG data is not very relevant. A visual comparison of the standard deviations of the two approaches (omitted due to space limitations) does not favour any of them either.

Summarizing, we can see that (1) RTs and bagging with MLPs are singled out as most frequently among the best and (2) these two approaches usually perform similarly. So, we cannot conclude that RTs generally provide benefits in terms of MMRE or PRED(25) in comparison to bagging using MLPs or vice-versa. Besides, ensembles are not generally better than well tuned single learners (RQ1).

Nevertheless, RTs have faster training time and are more transparent in terms of allowing engineers to understand the rules learnt than bagging with MLPs. An example of rule generated for ISBSG subset 2 is shown in figure 1. RTs are thus recommended over the other methods here analysed based on these grounds. Non-parametric Wilcoxon statistical tests [31] were also done considering the 95% confidence interval and they do not change these conclusions. This analysis also complements RQ3: if a company has no resources to perform experiments for choosing a model, RTs are more likely to perform comparatively well and can be used for being comprehensive and having faster training.

It is worth to note that, even though this work provides a different (practically the opposite) conclusion from Braga et al. [6], it does not necessarily contradict their reported results. Considering that the best performances obtained by their ensembles and single learners 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. [21]. In Kultur et al.'s work, instead of taking the simple average of the outputs of the base learners as the output of the ensemble, the outputs of the base learners 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 bagging using 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 learners. This ideal ensemble can improve both MMRE and PRED(25) for several data sets, according to two-tailed paired t-tests considering the 95% confidence interval. So, bagging still has potential to be improved for SEE, but it has to be carefully tailored.

Kocaguneli et al.'s work [19] does not actually provide a comparison between ensembles and single learners. How-

ever, our study can be used to provide further evidence in support of their work. As we show that it is very unlikely that there is a single universally good approach, it is probably also very unlikely that combining several different types of learners (including combining ensembles) will provide generally better results than the learners themselves.

5. APPROACHES SINGLED OUT

The previous section shows that, even though no approach was consistently the best, RTs and bagging with MLPs were more frequently among the best. More important than checking what approaches are usually among the best is to gain insight on how to improve SEE further based on an analysis of these approaches. None of the previous papers involving ensembles [6, 21, 19] provide an analysis of the reasons for the obtained results. Differently from the literature, this section provides insight on how to improve SEE using ML techniques based on experimental studies, not just an intuition or speculation, being a key contribution of this paper. Section 5.1 provides an analysis of RTs and section 5.2 provides an analysis of bagging with MLPs.

5.1 Analysis of RTs

RTs use information gain to determine which feature to split the tree in a certain level. In this way, this approach may be giving more importance for more important features. For example, if the most important feature for determining the effort is the functional size, it 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 figure 1. Using the relative importance of features for the predictions would be particularly beneficial for SEE, as the training sets are usually very small. In this section, we analyse whether RTs achieved comparatively good performance for that reason.

Correlation-based feature selection (CFS) method [13] with greedy stepwise search [12] was used to aid 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, probably being helpful for understanding the behaviour of RTs. It also checks the correlation among features themselves. Greedy stepwise search allows ranking features. This filter method was 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 3, but after performing feature selection. This study showed that feature selection by itself did not change the fact that RTs and bagging with MLPs were usually among the best in terms of MMRE and PRED(25). It managed to improve MMRE significantly in 25% of the cases and it worsened it in 6% of the cases, being helpful mainly for RBFs and bagging with RBFs. PRED(25) was less improved/worsened than MMRE. So, feature selection did not affect PRED(25) much. That is reasonable considering that more methods were ranked as among the best considering PRED(25) in section 4.2.

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 8. 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, making use of the relative importance of features.

Table 8: 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 reliability	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		

In summary, 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 more importance to more important features, being able to achieve comparatively good performance and suggesting that feature weighting is important when working with ML for SEE. This is an insight on how to improve SEE (RQ2), specially considering improvements in other methods than RTs.

5.2 Analysis of Bagging with MLPs

There are several measures of ensemble diversity for regression problems, e.g., correlation, covariance and chisquare [10]. We performed an initial analysis of ensemble diversity using correlation, as its values are restricted to the interval [-1, +1], being comparable across data sets. This analysis has shown that bagging with MLPs usually generates a more moderate level of diversity when it is ranked first or second in terms of MMRE in comparison to other ensemble methods and to when it was not ranked first or second. However, ensemble correlation itself is not highly correlated with MMRE. So, the analysis is not very conclusive.

Considering the three measures of diversity mentioned above, covariance is the most highly correlated with MMRE (the average of absolute values of the correlation is 0.35 considering all the data sets). So, in order to understand better the behaviour of bagging with MLPs, we analysed the correlation (table 9) between (1) ensemble covariance and test MMRE and (2) ensemble covariance and train MMRE. Four data sets were considered: the two in which bagging with MLPs achieved the lowest (best) MMRE among the ones in which it was ranked first or second; and the two in which

Table 9: Correlations between ensemble covariance (diversity) and train/test MMRE for the data sets in which bag+MLP obtained the best MMREs and was ranked 1st or 2nd against the data sets in which it obtained the worst MMREs.

	Cov. vs Test MMRE	Cov. vs Train MMRE
Best MMRE (desharnais)	0.24	0.14
2nd best MMRE (org2)	0.70	0.38
2nd worst MMRE (org7)	-0.42	-0.37
Worst MMRE (cocomo2)	-0.99	-0.99

bagging with MLPs obtained the highest (worst) MMRE.

As we can see, the correlation between covariance and test MMRE is positive for the best cases, but negative for the worst cases. Higher/lower covariance means that the diversity is lower/higher. So, a positive correlation between ensemble covariance and test MMRE means that diversity is helping to reduce MMRE. On the other hand, a negative correlation means that diversity is hindering, instead of helping to get a good performance. As the data sets themselves are also a source of diversity in addition to the ensemble approach being used, we can see that SEE data sets are *very* different from each other. Complementing the results presented in section 4.1, this shows us that it is indeed very unlikely that there is a universally best method for SEE.

Table 9 also shows us that the correlation between covariance and train MMRE followed a similar tendency to the correlation between covariance and test MMRE. So, it may be worth developing an approach which automatically tunes the level of diversity depending on the correlation between covariance and train MMRE. If the correlation is negative, that means diversity should be reduced. Such an approach may be able to improve SEEs (RQ2).

6. RISK ANALYSIS

The learning machines singled out in section 4.2 (RTs and bagging with MLPs) were tested using the outlier projects identified in section 2, so that we can check how sensitive/robust they are to outliers. The MMREs of these approaches for the outlier sets were similar or lower (better), usually better than the ones for the outliers-free test sets from section 4. The PRED(25) were similar or lower (worse), usually worse. No statistical tests were done because the outlier data sets are very small (at most 5 projects).

We can see that, even though the outliers are projects to which the learning machines have more difficulties in predicting within 25% of the actual effort, they are not the projects to which the learning machines give the worst estimates, as the average MMRE was better than the average MMRE of the outliers-free sets. So, the learning machines as robust to these outliers. As future work, the impact of including outliers in the training set should be investigated.

7. THREATS TO VALIDITY

Internal validity regards 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?" [25]. In ML, it is essential to use a principled procedure for selecting the parameters to be used by the learning machines. It is also important not to use the test projects to perform training. In our study, we followed a principled experimental framework, as described in section 3, which provided a careful parameters choice and ensured that no test project was used for training.

Construct validity regards accurately naming our measures and manipulations [25]. We used several measures of accuracy (section 3.2) for the analysis of survivors presented in section 4.1. These measures combined into a set of heuristic rules (algorithm 1) have shown to be able to rank SEE models by considering both the errors and standard deviations of the models [24]. The other analyses used mainly MMRE and PRED(25), which are the measures most widely used in SEE studies. Despite the critics to these measures [11], their problems can be resolved by using statistical evaluation [20]. In the present work, we used both t-tests and Wilcoxon tests to ensure construct validity.

External validity regards generalizing the study's results outside the study to other situations [25]. 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 the previous work involving ensembles for SEE. Besides, we have also performed a risk analysis that showed that the ML approaches singled out in the experiments are robust to outliers. No previous work has performed such an analysis.

8. CONCLUSIONS

This paper presents a principled and comprehensive evaluation of ensembles of learning machines for SEE, providing answers to the research questions introduced in section 1.

The answer to RQ1 is no – readily available ensembles do not provide generally better SEEs. Our study has shown that different learning machines are the best ones for different data sets. Even though bagging with MLPs was singled out as one of the approaches most frequently among the best, RTs were also singled out and usually perform similarly to bagging with MLPs. In addition, RTs are more comprehensive and have faster training.

As for RQ2, RTs usually put more important features in higher levels of the trees, being able to achieve relatively good accuracy and thus suggesting that attributing weights to the features may help improving SEE when using ML. Considering bagging with MLPs, even though a more moderate level of diversity might be helping them to achieve relatively good accuracy, the correlation between diversity and performance can vary dramatically depending on the data set. So, an approach to automatically tune the amount of diversity based on how correlated it is with the train accuracy may help improving SEE.

For RQ3, our analysis has shown that it is very unlikely that there is a universally best model, even when considering ensembles. Ideally, the software manager should run experiments with different models using a principled framework considering all the 3 points outlined in section 3 and using the projects to which s/he has access. In this way, the model likely to provide the best behaviour for the manager's needs can be identified. If the organisation has no resources to perform such experiments, RTs are a good choice for being fast, comprehensive and more frequently among the best.

Different from other papers involving studies of ensembles

for SEE, our work is able to provide these answers thanks to (1) handling validity issues not tackled or not made clear by previous studies, (2) including three different ensemble methods in the comparison, and (3) providing analyses of the behaviour of approaches singled out. In this way, this paper provides not only a comparison of existing machine learning methods, but also insight on how to choose a ML model and how to improve SEE. As future work, we mainly propose the use of feature weights for SEE using ML and the use of self-tuning diversity.

9. ACKNOWLEDGMENTS

The authors would like to thank the SEBASE project members and especially Dr. Rami Bahsoon for the useful comments and discussion. This work was supported by EPSRC grant EP/D052785/1.

10. REFERENCES

[1] R. Agarwal, M. Kumar, Y. Mallick, R. Bharadwaj, and D. Anantwar. Estimating software projects. *Software Engineering Notes*, 16(4):60–67, 2001.

[2] B. Baskeles, B. Turhan, and A. Bener. Software effort estimation using machine learning methods. In *ISCIS'07*, pages 1–6, Ankara, 2007.

[3] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, United Kingdom, 2005.

[4] B. Boehm. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs, NJ, 1981.

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

[6] P. L. Braga, A. Oliveira, G. Ribeiro, and S. Meira. Bagging predictors for estimation of software project effort. In *IJCNN'07*, pages 1595–1600, Orlando, 2007.

[7] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[8] M. Cartwright, M. Shepperd, and Q. Song. Dealing with missing software project data. In *METRICS'03*, pages 154–165, Sydney, 2003.

[9] H. Chen and X. Yao. Regularized negative correlation learning for neural network ensembles. *IEEE TNN*, 20(12):1962–1979, 2009.

[10] H. Dutta. Measuring diversity in regression ensembles. In *IICAI'09*, page 17p, Bangalore, India, 2009.

[11] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE TSE*, 29(11):985–995, 2003.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[13] M. A. Hall and L. A. Smith. Practical feature subset selection for machine learning. In *ACSC'98*, pages 181–191, Perth, Australia, 1998.

[14] J. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, 1975.

[15] A. Heiat. Comparison of artificial neural network and regression models for estimating software development effort. *Information and Software Technology*, 44:911–922, 2002.

[16] ISBSG. The International Software Benchmarking Standards Group. `http://www.isbsg.org`, 2011.

[17] M. Jorgensen and M. Shepperd. A systematic review of software development cost estimation studies. *IEEE TSE*, 33(1):33–53, 2007.

[18] B. A. Kitchenham, E. Mendes, and G. H. Travassos. Cross versus within-company cost estimation studies: A systematic review. *IEEE TSE*, 33(5):316–329, 2007.

[19] E. Kocaguneli, A. Bener, and Y. Kultur. Combining multiple learners induced on multiple datasets for software effort prediction. In *ISSRE'07*, Mysuru, India, 2009.

[20] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. Exploiting the essential assumptions of analogy-based effort estimation. *IEEE TSE*, 2011 (preprint).

[21] Y. Kultur, B. Turhan, and A. Bener. Ensemble of neural networks with associative memory (ENNA) for estimating software development costs. *Knowledge-Based Systems*, 22:395–402, 2009.

[22] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12:1399–1404, 1999.

[23] Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE TSMC - Part B: Cybernetics*, 29(6):716–725, 1999.

[24] T. Menzies, Z. Chen, J. Hihn, and K. Lum. Selecting best practices for effort estimation. *IEEE TSE*, 32(11):883–895, 2006.

[25] M. L. Mitchell and J. M. Jolley. *Research Design Explained*. Cengage Learning, USA, 7th edition, 2010.

[26] J. Sayyad Shirabad and T. Menzies. The PROMISE Repository of Software Engineering Databases. School of Information Technology and Engineering, University of Ottawa, Canada, http://promise.site.uottawa.ca/SERepository, 2005.

[27] Y.-S. Seo, K.-A. Yoon, and D.-H. Bae. An empirical analysis of software effort estimation with outlier elimination. In *PROMISE*, pages 25–32, Leipzig, 2008.

[28] M. Shepperd and G. Kadoda. Comparing software prediction techniques using simulation. *IEEE TSE*, 27(11):1014–1022, 2001.

[29] K. Srivasan and D. Fisher. Machine learning approaches to estimating software development effort. *IEEE TSE*, 21(2):126–137, 1995.

[30] I. F. B. Tronto, J. D. S. Silva, and N. Sant'Anna. Comparison of artificial neural network and regression models in software effort estimation. In *IJCNN'07*, pages 771–776, Orlando, 2007.

[31] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics*, 1(6):80–83, 1945.

[32] I. H. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, USA, 2000.

[33] G. Wittig and G. Finnie. Estimating software development effort with connectionist models. *Information and Software Technology*, 39:469–476, 1997.

[34] Y. Zhao and Y. Zhang. Comparison of decision tree methods for finding active objects. *Advances in Space Research*, 41:1955–1959, 2008.