The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines

Liyan Song, Leandro L. Minku, Xin Yao
CERCIA, School of Computer Science
The University of Birmingham
Edgbaston, Birmingham B15 2TT, UK
{lxs189,l.l.Minku,x.yao}@cs.bham.ac.uk

ABSTRACT

Background: The use of machine learning approaches for software effort estimation (SEE) has been studied for more than a decade. Most studies performed comparisons of different learning machines on a number of data sets. However, most learning machines have more than one parameter that needs to be tuned, and it is unknown to what extent parameter settings may affect their performance in SEE. Many works seem to make an implicit assumption that parameter settings would not change the outcomes significantly.
Aims: To investigate to what extent parameter settings affect the performance of learning machines in SEE, and what learning machines are more sensitive to their parameters.
Method: Considering an online learning scenario where learning machines are updated with new projects as they become available, systematic experiments were performed using five learning machines under several different parameter settings on three data sets.
Results: While some learning machines such as bagging using regression trees were not so sensitive to parameter settings, others such as multilayer perceptrons were affected dramatically. Combining learning machines into bagging ensembles helped making them more robust against different parameter settings. The average performance of k-NN across different projects was not so much affected by different parameter settings, but the parameter settings that obtained the best average performance across time steps were not so consistently the best throughout time steps as in the other approaches.
Conclusions: Learning machines that are more/less sensitive to different parameter settings were identified. The different sensitivity obtained by different learning machines shows that sensitivity to parameters should be considered as one of the criteria for evaluation of SEE approaches. A good learning machine for SEE is not only one which is able to achieve superior performance, but also one that is either less dependent on parameter settings or to which good parameter choices are easy to make.

PROMISE '13, October 9, 2013, Baltimore, MD, USA
ACM 978-1-4503-2016-0/13/10.
http://dx.doi.org/10.1145/2499393.2499394

Categories and Subject Descriptors

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

General Terms

Experimentation

Keywords

Software effort estimation, sensitivity to parameters, machine learning, online learning, ensembles

1. INTRODUCTION

Studies using Machine Learning (ML) approaches for Software Effort Estimation (SEE) have been done for many years [29, 13]. Most studies involve comparisons among different approaches. For instance, recent works have been pointing out the relatively good performance achieved by ensembles of learning machines [18, 26, 24, 16] and local methods that make estimations based on completed projects similar to the project being estimated [24, 22, 3].

These studies concentrate on how well a certain approach is able to perform in comparison to others in terms of different performance measures. Several studies use statistical tests to draw conclusions on whether the difference between the performance achieved by different approaches is significant [26, 24, 16]. A recent study also emphasizes the importance of using measures of effect size to quantify the practical impact that differences in performance may incur [28].

Analyses of SEE approaches are affected by many factors, which frequently lead different conclusions to be obtained by different researchers [14, 12, 21]. Examples of factors that may affect the results are the data sets used in the study, the type of preprocessing, the method used to divide data into training and test sets, the performance measures, and the amount of fine tuning of the approaches [23]. Among these factors, the effect that different data sets and performance measures may have in the conclusions is relatively well known [28, 11, 26, 24]. However, even though ML approaches frequently have more than one parameter that needs to be set prior to the training process, there has been little work investigating the effect that different parameter settings may have in ML approaches for SEE.

As explained in [24], the methodology used to chose parameter settings is frequently omitted from the experimental framework reported in SEE papers, which thus seem to make

an implicit assumption that parameter settings would not change the outcomes of the algorithms significantly. Nevertheless, it is not known to what extent different parameter settings affect the performance of several of the ML approaches that have been used for SEE. Such knowledge would be very valuable for the SEE community, as it could guide the choice of ML approaches for SEE. So, sensitivity to parameter settings should be considered as another criterion for the evaluation of SEE approaches.

There are several examples of how analyses of sensitivity could allow for more informed choices of SEE models to be made. For example, if two approaches are able to achieve similar performance, but one of them is much less sensitive to parameter settings than the other, it would be safer to use the less sensitive approach. If a certain approach is fairly sensitive to parameter settings, but is able to achieve much better performance than a less sensitive approach, it may be a good choice to use it if the person responsible for setting it up and maintaining it has some knowledge on how to tune parameters. Approaches that are very sensitive to parameter settings could be considered by certain project managers as undesirable because a wrong choice of parameter settings may cause them to obtain very poor performance.

Another important point to be considered when evaluating SEE models is the chronology of the projects, i.e., the fact that, when a new effort estimation needs to be performed, SEE models can only be created using projects that have already been completed at that time step (point in time). The reason why it is important to consider chronology is that environmental changes may happen with time and affect the performance of SEE models [19, 25]. Sensitivity to parameter settings may affect not only the best choice of parameters in terms of average performance across time steps, but also the best choice of parameters at each time step. So, analyses of sensitivity to parameters should ideally consider not only the effect of parameter settings across time, but also their effect throughout time.

With that in mind, this study aims at answering the following research questions:

- RQ1: Given an approach and a data set, how sensitive is this approach to different parameter settings in terms of its average performance across time steps?
- RQ2: Given an approach and a data set, does the best parameter setting in terms of average performance across time steps perform consistently well throughout time steps in comparison to other parameter settings?
- RQ3: Ensembles of learning machines such as Bagging have been showing to obtain relatively good results for SEE [24]. Could Bagging also help to lessen the base learners' sensitivity to parameter settings?

Answering RQ1 and RQ2 for several different learning machines can also provide insight on which of these learning machines are more sensitive to their parameters.

2. RELATED WORK

2.1 ML in SEE

Algorithmic SEE models have been studied for many years [5, 13]. Among them, ML algorithms have been increasingly investigated [13]. A landmark study using ML for SEE is the work of Shepperd and Schofield (1997) [29], who used a k-Nearest Neighbour (k-NN) algorithm [4] based on normalised attributes and on Euclidean distance as similarity measure. This approach is also known as estimation by analogy. Despite being first used for SEE more than fifteen years ago, it has remarkably been shown to be competitive in terms of how frequently it obtained the best Mean Absolute Error (MAE) over thirteen data sets in comparison to other approaches more recently applied for SEE, such as Regression Trees (RTs) [24]. Nevertheless, when it is not among the best approaches for a certain data set it can perform considerably worse than the best in terms of MAE [24].

Recent works have been emphasising the relatively good performance achieved by ensembles of learning machines [18, 26, 24, 16] and local methods that make estimations based on completed projects similar to the project being estimated [24, 22, 3]. For instance, a study involving a total of eight learning machines and thirteen data sets has shown that RTs and Bagging ensembles of MultiLayer Perceptrons (Bagging+MLPs) were frequently ranked best in terms of Mean Magnitude of the Relative Error (MMRE) and Percentage of predictions within 25% of the actual value (PRED(25)). In terms of MAE, Bagging ensembles of RTs (Bagging+RTs) were the most frequently ranked best, and when they were not the best, they rarely obtained considerably worse performance than the best [24].

There have also been SEE studies paying special attention to the chronology of the projects [20, 19, 25]. SEE tasks operate in so called online environments, where new completed projects arrive with time. Such environments are unlikely to be stationary, as software development companies and their employees evolve with time. For example, new employees can be hired or lost, training can be provided, employees can become more experienced, etc. Changing environments have been shown to affect the performance of SEE models [19, 25], and it is thus important to consider chronology when evaluating SEE models.

2.2 Studies on Parameter Settings in Software Engineering

Despite the existence of several works comparing ML approaches in the SEE literature, there has been little work on quantifying the effect of different parameter settings on the performance of such approaches. Minku and Yao (2012) [24] emphasized the importance of explaining clearly how the parameters were chosen in SEE studies involving comparisons of different approaches, as they may have significant influence in the conclusions obtained. Menzies and Shepperd (2012) [23] also expressed concern regarding the effect that spending more time tuning one approach than another may cause in the conclusions of comparative studies. However, these publications have not provided an analysis of the impact that different parameter settings can have in SEE.

Few studies analyse the impact of different parameter settings in ML for SEE. For example, Dejaeger et al. (2012) [10] performed a comparison of several different ML approaches. Some of them used default parameters, the others were tuned based on a validation procedure, and one of them (k-NN) was directly included in the analysis using four different parameter choices. Their study based on nine data sets revealed that the different values of k did not significantly affect k-NN's performance. The impact of the parameter settings of the other approaches was not analysed.

Kocaguneli, Menzies and Keung (2013) [17] performed an analysis of the impact of kernel types and bandwidth parameters in non-uniform weighting analogy-based effort estimation through kernel methods using nineteen data sets. They concluded that these parameters did not affect the performance of the approach significantly. Corazza et al. [8, 9] pointed out the importance of parameter tuning when using Support Vector Regression (SVR) into SEE. In particular, Tabu Search has been proposed to search for an optimal SVR parameter setting. An evaluation of this approach on 21 data sets showed that it outperformed several others including widely used ones such as case based reasoning.

A comprehensive study of the impact of parameter settings can be found in a software engineering area related to ML in SEE. Arcuri and Fraser (2011) [2] performed a large study of parameter settings in the field of test data generation using genetic algorithms. Their study specifically aimed at answering questions such as how large the potential impact of a wrong choice of parameter settings is, and how a default setting compares to the best and worst achievable performance. Their analysis showed that parameter tuning can have critical impact on algorithmic performance, and that overfitting of parameter tuning is a serious threat to external validity of empirical analysis in search based software engineering.

Nevertheless, the impact of the parameter settings of several ML approaches for SEE, including approaches that have been showing to obtain relatively good performance, is unknown. As explained in section 1, a study analysing the sensitivity of different SEE ML approaches to parameter settings would be important for a more informed choice of what SEE approach to adopt.

3. DATA SETS

The following data sets were used in this work: Kitchenham, Maxwell and SingleISBSG. These data sets were chosen because they provide time information that can be used to sort projects to perform online learning. They are described in sections 3.1, 3.2 and 3.3. For instance, the steps used in the manual preprocessing of the data are explained. No further manual preprocessing has been made, but some of the learning machines employed in this study automatically process the data further (e.g., data normalisation) according to their needs, as explained in section 4.2. A thorough analysis of these data sets is left as future work.

3.1 Kitchenham

This data set was obtained from the PRedictOr Models In Software Engineering Software (PROMISE) Repository[1],and its detailed description can be found in [15]. It comprises 145 maintenance and development projects undertaken between 1994 and 1998 by a single software development company. The following steps were performed to process this data set for use in this work:

1. Sort the projects according to the actual start date plus the duration. This sorting corresponds to the exact completion order of the projects. The Appendix lists the sequence of ids of the sorted projects.

2. Remove the attributes project id, actual start date, actual duration, estimated completion date, first estimate and first estimate method. Project id was removed because it is irrelevant for training a SEE model. Actual start date was removed because the projects are sorted according to their completion date. Completion date together with start date would give the duration of the project, and duration was removed because it is considered as a dependent variable. The other attributes were removed because they are themselves estimations of completion date or effort, or represent the method used for such estimations. This preprocessing resulted in the three input attributes (adjusted function points, project type and client code) and one output attribute (effort in hours).

3. Treat missing values using 1-NN imputation [7]. There were in total ten projects with missing values.

Table 1: Maxwell data set input attributes.

Application size in function points	Database
Efficiency requirements	User interface
Where developed (in-house/outsourced)	Tools use
# different development languages used	Telon use
Customer participation	Methods use
Development environment adequacy	Standards use
Software's logical complexity	Hardware platform
Requirements volatility	Staff availability
Quality requirements	Staff team skills
Installation requirements	Staff tool skills
Staff application knowledge	Staff analysis skills

3.2 Maxwell

This data set was also obtained from the PROMISE Repository, and its detailed description can be found in [27]. It contains 62 projects from one of the biggest commercial banks in Finland, covering the years 1985 to 1993 and both in-house and outsourced development. The following steps were performed to process this data set for use in this work:

1. Sort the projects according to $syear + duration/12$, where *syear* is the start year and *duration* is the duration of the project in months. This sorting corresponds approximately to the completion order of the projects. The reason why the exact completion order is unknown is that the month of the year that the project started is not provided by the data set.

2. Remove the attributes start year, time and duration. Start year and time ($time = syear - 1985 + 1$) were removed because the projects are sorted according to their completion year. Completion year together with start year would give the duration of the project, and duration was removed because it is considered a dependent variable. This preprocessing resulted in the 23 input attributes listed in table 1 and one output attribute (effort measured in hours).

3. There were no missing values in this data set.

3.3 SingleISBSG

This data set is a subset of the International Software Benchmarking Standards Group (ISBSG) Repository[2] Release 10 which has been previously used in [25]. It comprises 69 projects from a single-company. Information on what projects belong to this single anonymous company has been

[1] http://code.google.com/p/promisedata/

[2] http://www.isbsg.org/

provided to us by ISBSG upon request. These projects have the following characteristics:

- 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).
- 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 identified as with addendum to existing standards.
- Implementation date after the year 2001.

The following steps were performed to process the projects of this data set:

1. Sort the projects according to the implementation date.
2. Select development type, language type, development platform and functional size as input attributes, as recommended by ISBSG. The output attribute is the effort in hours. Remove all other attributes.
3. Treat missing values using 1-NN imputation [7]. There were in total only two projects with missing values.

4. EXPERIMENTAL FRAMEWORK

This section presents the experimental framework used to accomplish the goals of this work.

4.1 Online Scenario

As briefly explained in section 1, it is important to consider the chronology of projects when evaluating SEE approaches. SEE operates in an online scenario, where additional projects are completed with time and can be used for training SEE models. Whenever a new effort estimation needs to be provided, only projects that have already been completed can be used for building an SEE model to make the estimation. As the environment where the SEE approaches operate is unlikely to be stationary (new employees can be hired or lost, training can be provided, etc), the characteristics of the projects being completed may change with time, be it a change in the frequency of certain input values or in the effort that would normally be necessary to complete a project. As such changing environment has been shown to affect the performance of SEE models [25], it is important to evaluate models considering not only their overall performance across time steps, but also their performance throughout time.

With that in mind, similarly to [25], we consider an online learning scenario in which a new project completed by a company is received as a training example at each time step, forming a data stream. Different from the strict online learning scenario, we do not discard training examples received in previous time steps. At each time step, the SEE approach is trained on all completed projects received so far, i.e., one training project is used for training at the first time step, two at the second, three at the third, and so on. At each time step, after the training, the next ten projects of the data stream are estimated. The performance of the SEE approach at a certain time step in terms of a certain measure is calculated based on the estimations given to this window of ten projects. Window size of ten has also been used in previous work [25], and we consider it reasonable because not so many projects are produced per year by a company. Since our major aim is to investigate the effect of different parameter settings of the learning machines, other outside factors such as different window sizes used in the evaluation procedure are left as future work.

4.2 Learning Machines and Parameter Settings Investigated

We investigate the following five approaches in this study: k-NN, RTs, MLPs, Bagging+RTs, and Bagging+MLPs. We do not investigate k-NN combined with Bagging because Bagging is known to improve accuracy for unstable procedures[3] such as MLPs and RTs, while it can slightly degrade the performance of stable procedures such as k-NN [6]. An online learning class has been developed so that the WEKA implementations of these approaches could be used. RTs were based on the REPTree implementation without pruning, k-NN was based on IBK with normalised attributes and Euclidean distance, and the other approaches were based on the implementations with the same name in WEKA. All MLPs used a single hidden layer and were set to automatically normalise dependent and independent variables and to use the nominal to binary filter.

RTs, Bagging+RTs and Bagging+MLPs were chosen because they have been shown to perform well in comparison to several other ML approaches in SEE [24], as explained in section 2. Nevertheless, the evaluation of these approaches provided in the literature did not consider their sensitivity to parameter settings. Knowledge on whether these approaches are very sensitive or not to parameter choices would be important for deciding whether to use them, or which one of them to use for SEE. Additionally, ensembles such as bagging have been shown to be able to improve the frequency that their base learners are ranked first in terms of MAE [24]. So, it would be good to know whether they could also make these approaches less sensitive to parameter settings.

K-NN is among the simplest learning machines, and we have included it in the analysis because it can perform frequently very well, but sometimes considerably worse than the best approach depending on the data set [24]. It would be useful to know whether the same sensitivity to the data set also applies to the sensitivity to its parameters, or if the simplicity of this approach could make it more robust to parameter settings.

MLPs have not been shown to be so frequently among the best approaches as the other approaches included in our analysis [24]. However, it is not known whether this approach is performing worst because it is simply frequently not able to achieve better performance than the others, or if it is highly sensitive to parameter settings and thus difficult to tune, or if in fact some guidelines on its parameter choices could improve its performance. So, the main reason to include MLP in the analysis is to provide a better understanding of the behaviour of this approach for SEE.

The parameter values investigated in this paper are shown in table 2. Their default values are emphasized with bold and correspond to the default values from WEKA. For RTs,

[3]Unstable here means when small changes in the training sample can result in large changes in the model learnt.

Table 2: Parameter Values

Approach	Parameters
RTs	M(mim.# instance/leaf)={1,**2**,3,6,12,20} V(mim.varience for split)= {0.0001,**0.001**,0.01,0.1,10} L(max.tree depth)={**-1**,2,6,10,15,20}
k-NN	k(# neighbours)={**1**,3,5,7,9,11,13,15,17,19,21}
MLPs	L(Learning rate)={0.1, 0.2, **0.3**, 0.4, 0.5} M(Momentum)={0.1,**0.2**,0.3,0.4,0.5} N(# epochs)={100,**500**,1000} H(# hidden nodes)={***a***,1,3,5,9}
Bagging	# I(iteration for Bagging)={5,**10**,25,50,75} All the possible parameters of the adopted base learners, as shown above.

the maximum depth of -1 means unlimited depth. For MLPs, the default value a in the number of hidden nodes is calculated as follows: $a = (\#attributes + \#classes)/2$, where $\#attributes$ is the number of input attributes in the data set, and $\#classes$ is the number of outputs, which equals to one in the case of SEE.

It is impossible to use all possible parameter values here, as that would be infinite. We believe that the values shown in table 2 form a good range for each of the parameters. Additional values could be investigated as future work.

4.3 Evaluation Criteria

Given a data set and an approach described in Section 4.2, we run the approach with all the combinations of parameter values shown in table 2, to calculate the performance at each time step as well as the average performances across times steps considering the online scenario from section 4.1. From that we can determine the best/worst parameter settings in terms of the average performances across time steps, as well as the performance of the default parameter setting. The standard deviation of the performances across time steps can also be calculated.

For the non-derteministic approaches (MLPs, Bagging+RTs and Bagging+MLPs), 30 runs were performed to obtain the mean performance at each time step, which was then averaged across time steps. The corresponding mean standard deviation across time steps was calculated by means of the pooled standard deviation as follows:

$$std_{pooled} = \sqrt{\frac{std_1^2 + std_2^2 + ... + std_n^2}{n}},$$

where n is the the number of runs (30 in our case) and std_i is the standard deviation across time steps in the i^{th} run.

The performance at each time step was measured by the Mean Absolute Error (MAE) over the predictions on the next ten projects of the data stream. MAE is defined as $\sum_{i=0}^{n} \frac{|y_i - \widehat{y_i}|}{n}$, where $n = 10$ is the number of samples considered, y_i is the actual value of the variable being predicted and $\widehat{y_i}$ is its estimation. MAE was chosen for being a symmetric measure not biased towards under or overestimations [28]. Lower MAE indicates higher/better performance.

With the aim of investigating to what extent an approach is sensitive to its parameter settings given a certain data set, the performances of the best and worst parameter settings were first compared based on Wilcoxon sign-rank test with Holm-Bonferroni corrections considering the total number of comparisons made for the corresponding learning machine, at the overall level of significance of 0.05. Even when there is statistically significant difference, that does not necessarily mean that the differences are large enough to have significant effect *in practice* [28]. So, the effect size (Cohen's d) was checked. Effect size is simple a way of quantifying the size of the difference between two groups. It was calculated using pooled standard deviation as follows:

$$d_1 = \frac{\overline{MAE_1} - \overline{MAE_2}}{\sqrt{\frac{std_1^2 + std_2^2}{2}}},$$

where $\overline{MAE_i}$ is the average MAE across time steps of the model created using the parameter setting i, $i \in \{best, worst\}$, and std_i is its corresponding standard deviation across time steps. Similarly to the above, Wilcoxon test with Holm-Bonferroni corrections were also computed for the comparisons between the best and the default, and the default and the worst parameter settings, when deemed necessary. The effect sizes corresponding to these comparisons are referred to as d_2 and d_3, respectively.

As will be explained in section 5, in some cases it was not possible to use a pooled standard deviation in the analysis. In these cases, one of the parameter settings being compared was selected as the control setting, and the standard deviation used to calculate the effect size was the standard deviation of the model created using the control parameter setting, instead of the pooled standard deviation.

The effect size was interpreted in terms of the Cohen's categories [28]: small (≈ 0.2), medium (≈ 0.5) and large (≈ 0.8). If d_1 is small (around 0.2) for a certain data set, the performances of the best and the worst parameter settings are considered quite similar for this data set, and we could claim that this approach is not sensitive to different parameter settings for this data set. If this behaviour extends to other data sets as well, then this approach is considered robust to parameter settings.

If d_1 is medium (around 0.5) or large (around 0.8), the approach is somewhat or highly sensitive to its parameter settings. In this case, d_2 will reveal whether a default parameter setting could provide reasonable performance despite the approach's sensitivity to parameter settings. If d_2 is small or small/medium, it means that even though this approach is sensitive to the overall parameter choices, its default parameter setting is fairly good, and we could simply use its default parameter setting. However, if d_2 is not small enough, the performance of default parameter setting is significantly worse than the performance of the best parameter setting, and we should pay attention to tune the parameters of this approach.

For this kind of approach, we could still step further by calculating d_3. If d_3 is large (more than 0.8), that means that even though the performance of the default parameter setting is significantly worse than the best one, it is still significantly better than the worst one and thus somewhat helpful. On the other hand, if d_3 is not large, then this approach is too sensitive to its parameter settings, and a tiny change to its parameter settings could cause a significantly bad effect on its performance. Therefore, one may consider this approach as not recommended for SEE.

5. EXPERIMENTAL ANALYSIS

5.1 Sensitivity in Terms of Average Performance Across Time Steps

This section mainly aims at answering RQ1: Given an approach and a data set, how sensitive is this approach to

different parameter settings in terms of its average performance across time steps?

5.1.1 MLPs and Bagging+MLPs

Table 3 shows the average MAE across time steps of MLPs and Bagging+MLPs in their best, default and worst parameter settings, as well as the statistical tests and effect sizes of the differences in performance between these parameter settings. As we can see from tables 3(a) and 3(b), the performances of the worst parameter settings of MLPs and Bagging+MLPs are so inferior that most of their standard deviation across time steps are infinite, which makes it impossible to compute the effect size with pooled standard deviation. Instead, we will calculate the effect size with the best one as the control approach for both MLPs and Bagging+MLPs, to measure the performance difference between a certain parameter setting with the best one. We found that the performances with the worst parameter settings of MLPs and Bagging+MLPs are sensitive to starting points, e.g., the initial weights of the MLPs. Depending on the starting point, the predictions given to a few examples get extremely high error, causing the average MAE to be also extremely large, and the standard deviation to be infinite.

As shown in tables 3(a) and 3(b), intuitively we can conclude that both MLPs and Bagging+MLPs are extremely sensitive to different parameter settings, since the difference of the average performance between the best and the worst parameter settings are significiantly large both for MLPs and Bagging+MLPs. Such supposition can be further confirmed by the Wilcoxon tests and effect sizes listed in tables 3(c) and 3(d), which show that there is statistically significant difference and the effect sizes between the best and the worst are all extremely large in all the investigated data sets both for MLPs and Bagging+MLPs.

However, both MLPs and Bagging+MLPs in the best and default parameter settings can achieve a fairly good performance, which are competitive to the counterparts of all the other learning machines investigated in this paper shown in tables 4(a), 4(b) and 5(a). Even though the performance of the default parameter settings are statistically significantly different and usually rather worse than the best ones in terms of effect size, they are acceptable compared with the worst ones. Moreover, since their standard deviations across time steps are all finite values (see table 3(a) and 3(b)), we can conclude the default and the best parameter settings are not so sensitive to the starting points. Therefore, we recommend the project manager to use the default parameter settings when he/she has little experience of tuning parameters or he/she does not have time to do so.

Furthermore, when exploring deeper, we found that the best parameter setting for MLPs across all data sets is the simplest one with the number of hidden nodes equal to one, and for Bagging+MLPs their best parameter settings are the ones with the simplest base learners.

Overall, in the SEE literature about MLPs either on their own or ensembled with Bagging, some researchers said they did not achieve a good performance, but others disagreed [10, 24, 30]. However, our experiments show that they can achieve a relatively good performances, but they are very sensitive to parameter choice, and even to the starting point. This can at least partly explain the previous controversial conclusions in the literature. Moreover, the best parameters for different data sets were the same. Therefore, in future work, we will investigate more data sets, and if this parameter setting is still competitive, we could claim that using simple MLP both on its own and combined with Bagging is good for SEE.

5.1.2 RTs and Bagging+RTs

Table 4(a) and 4(b) show the average MAE across time steps of RTs and Bagging+RTs in their best, default and worst parameter settings. As shown in table 4(c) and 4(d), there is statistically significant difference between the best and worse parameter settings for all data sets. The corresponding effect sizes using pooled standard deviation are small (0.126) and small (0.199) in Kitchenham, small (0.310) and medium (0.453) in Maxwell, and large (0.768) and large (0.665) in SingleISBSG for RTs and Bagging+RTs respectively. This means that both RTs and Bagging+RTs are not sensitive to different parameter settings given Kitchenham and Maxwell, but a bit sensitive in SingleISBSG. The relatively small standard deviations in SingleISBSG could be a reason for the large effect size. However we still need to explore further whether SingleISBSG is the real "exception" as well as the cause for this exception, which will be left as our future work. These effect sizes show that RTs and Bagging+RTs are much less sensitive to parameter settings than MLPs and Bagging+MLPs.

Even though RTs are a bit sensitive to different parameter settings in SingleISBSG, the effect size with pooled standard deviation between the best and the default, and between the default and the worst parameter settings are medium (0.419) and medium (0.398) respectively. That means that their default parameter settings can achieve relatively good performance, even though there exist statistically significant improvements if the parameter settings are tuned carefully. For Bagging+RTs, the effect size between the best and the default parameter settings in SingleISBSG is very small (0.105), which indicates that even though Bagging+RTs is slightly sensitive in SingleISBSG across all parameter settings, the performance difference between the best and the default parameter settings is quite tiny.

Overall, RTs and Bagging+RTs are usually not very sensitive to parameter settings in SEE – it will be a good option to simply use the default parameters if tuning parameters is not allowed. However, we still suggest to tune the parameters in order to achieve better performance. In comparison to other learning machines such as MLPs and Bagging+MLPs, the performance of RTs and Bagging+RTs under the worst parameter setting is not so much worse than the best one. So, blind parameter tuning will not cause so much problem for RTs and Bagging+RTs.

5.1.3 K-NN

Experiments indicate that k-NN is not sensitive to parameter choices in SEE. As we can see from table 5(b)), the effect sizes are always small or medium, and for SingleISBSG no statistically significant difference has been found in the comparisons. However, it is worth noting that the default parameter setting ($k = 1$) is always the worst (see table 5(a)). Therefore, we recommend not to use 1-NN in SEE. One of the possible reasons is that there exists much noise in the data sets of SEE, so the performance can be strongly affected only using the nearest neighbour.

Further investigation reveals that the best performances are always achieved when k equals to three or five (see ta-

Table 3: Average Performance, Effect Size and Statistical Tests Across Time Steps for MLPs and Bagging+MLPs. In 3(c) and 3(d), '+/-' indicates whether or not there is significant difference based on Wilcoxon test with Holm-Bonferroni corrections, considering the 6 comparisons; the corresponding p-values are in parentheses. Here, 'std' is short for the standard deviation across time steps in terms of MAE. Effect sizes that are considered as medium/large are in yellow/red (light/dark gray).

(a) Performance of MLPs

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	2046.35	5358.02	2754.78
	std.	2868.96	1979.71	1006.01
Default PS	MAE	2474.78	7893.26	3682.47
	std.	2846.06	3629.54	1254.03
Worst PS	MAE	7.42E+138	1.19E+155	1.07E+153
	std.	4.71E+140	Inf	Inf

(b) Performance of Bagging+MLPs

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	1946.18	5089.75	2705.77
	std.	2883.74	1918.31	790.20
Default PS	MAE	2188.81	5932.99	3025.83
	std.	2892.06	2262.39	860.47
Worst PS	MAE	9.26E+151	1.33E+153	4.52E+153
	std.	Inf	Inf	Inf

(c) Effect size and p-value of MLPs with the best as the control

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	2.5863E+135	6.011E+151	1.0636E+150
	(6.77E-21)+	(6.15E-10)+	(3.51E-11)+
best vs. default	0.149	1.281	0.922
	(2.66E-22)+	(6.15E-10)+	(3.51E-11)+

(d) Effect size and p-value of Bagging+MLPs with the best as the control

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	3.211E+148	6.933E+149	5.720E+150
	(7.52E-21)+	(7.35E-10)+	(2.44E-09)+
best vs. default	0.084	0.440	0.405
	(2.00E-19)+	(2.80E-09)+	(7.15E-09)+

Table 4: Average Performance, Effect Size and Statistical Tests Across Time Steps for RTs and Bagging+RTs. In 4(c) and 4(d), '+/-' indicates whether or not there is significant difference based on Wilcoxon test with Holm-Bonferroni corrections, considering the 7 and 6 comparisons; the corresponding p-values are in parentheses. Here, 'std' is the standard deviation across time steps in terms of MAE. Effect sizes that are considered as medium/large are in yellow/red (light/dark gray).

(a) Performance of RTs

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	2249.14	5629.51	2751.86
	std.	2928.71	2426.86	852.77
Default PS	MAE	2618.38	5930.50	3144.56
	std.	2899.82	2611.51	1016.74
Worst PS	MAE	2618.96	6429.93	3621.96
	std.	2935.13	2725.38	1356.32

(b) Performance of Bagging+RTs

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	2055.24	5110.56	2814.39
	std.	2908.12	2606.10	941.08
Default PS	MAE	2209.14	5260.25	2915.74
	std.	2926.48	2572.87	986.80
Worst PS	MAE	2634.56	6230.56	3356.18
	std.	2926.55	2327.33	665.64

(c) Effect Size and p-value of RTs Using Pooled Std. Dev.

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	0.126	0.310	0.768
	(2.52E-10)+	(0.0104)+	(4.94E-07)+
best vs. default	0.127	0.119	0.419
	(1.35E-11)+	(2.94E-05)+	(7.12E-05)+
default vs worst	-	-	0.398
			(5.77E-05)+

(d) Effect Size and p-value of Bagging+RTs Using Pooled Std. Dev.

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	0.199	0.453	0.665
	(1.89E-20)+	(1.67E-09)+	(2.09E-07)+
best vs. default	0.053	0.058	0.105
	(3.71E-08)+	(1.01E-07)+	(0.0011)+

ble 5(c)), which can be interpreted as follows: though using more neighbours would lessen the effect of noise existed in the data sets, as k growing bigger, more less relevant samples can be involved into predicting, which is not preferred. Considering the small data sets of SEE, three or five (or a bit more like seven) nearest neighbours may be a good choice to avoid both extremes. Figures 1–3 further support that.

In our experiments, we find that the simple learning machine k-NN is quite competitive with all learning machines we are investigating, including RTs, Bagging+RTs, MLPs and Bagging+MLPs. Another advantage is that its average performance across time steps is not very sensitive to different parameter settings. However, as we will see later, the parameter settings that obtain the best average performance across time steps for k-NN are not so consistently the best throughout time steps as in the other approaches.

5.2 Performance of the Best Parameter Setting at Each Time Step

In the previous section, we presented the overall performance across all time steps, and analyzed the sensitivity of each approach to different parameter choices. In this section, we will look into each time step to investigate RQ2: Given an approach and a data set, do the best parameter settings in terms of average performance across time steps perform consistently well throughout time steps in comparison to other parameter settings?

According to figures 4 to 9 (the other figures are omitted due to space restriction), we can conclude that though there are a few time steps in which the default or even the worst parameter settings outperform the best ones, usually the best ones achieve a better performance than the others. For instance, in figure 7, at the time steps between ten and fifteen, the worst parameter setting outperforms the best and the default ones, which hints that there still exists room for improvement in terms of the performance throughout each time step providing proper adaptive parameter settings. Even so, we find that in the majority of the time steps, the best parameter setting is better than the default and the worst ones.

Furthermore, comparing figures 6, 9, and 10 with 4, 5, 7, and 8, we find the frequency that the worst parameter settings for k-NN outperform the best ones is higher than others. It indicates the best parameter setting for k-NN is more dependent on the moment in time than for other approaches. In other words, k-NN is less stable than others

Table 5: Average Performance, Effect Size, and Statistical Tests Across Time Steps plus Parameter Settings for K-NN. In 5(b), '+/-' indicates whether or not there is significant difference based on Wilcoxon test with Holm-Bonferroni corrections, considering the 6 comparisons; the corresponding p-values are in parentheses. Here, 'std' is short for the standard deviation across time steps in terms of MAE. Effect sizes that are considered as medium/large are in yellow/red (light/dark gray).

(a) Performance of k-NN

MAE across time steps		Kitchenham	Maxwell	SingleISBSG
Best PS	MAE	1889.67	4642.61	2937.49
	std.	2770.78	2574.22	688.85
Default PS	MAE	2315.12	5667.09	3394.39
	std.	2838.20	2172.91	1562.69
Worst PS	MAE	2315.12	5667.09	3394.39
	std.	2838.20	2172.91	1562.69

(b) Effect Size and p-value of k-NN Using Pooled Std. Dev.

Effect Size	Kitchenham	Maxwell	SingleISBSG
best vs. worst	0.152	0.430	0.378
	(8.62E-17)+	(4.55E-04)+	(0.0265)-
best vs. default	0.152	0.430	0.378
	(8.62E-17)+	(4.55E-04)+	(0.0265)-

(c) Parameter Settings of k-NN

k	Kitchenham	Maxwell	SingleISBSG
Best PS	k=3	k=3	k=5
Defaul PS	k=1		
Worst PS	k=1		

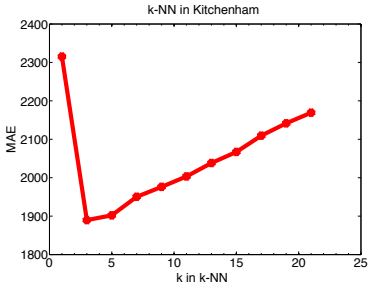

Figure 1: k in k-NN in Kitchenham

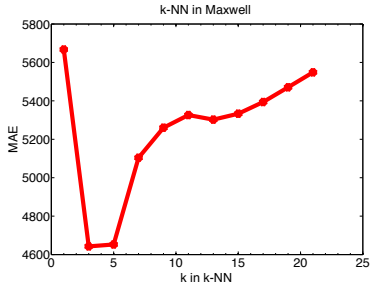

Figure 2: k in k-NN in Maxwell

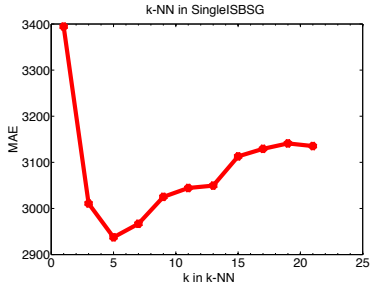

Figure 3: k in k-NN in SingleISBSG

in terms of the consistency of the best parameter settings throughout time steps.

Besides, figure 10 presents the performance of k-NN throughout time steps in Kitchenham, which shows that all parameter settings could perform quite well throughout time steps except the ones between 65 and 74. And such a situation also happens to all the other approaches, which means maybe certain projects in time steps between 65 and 74 are the so-called "bad samples" or "outliers", which are not easy to predict universally. Therefore, the performance of the learning machines in SEE may be improved if the outliers are removed. This could be investigated as future work.

Additionally, figures 4 and 5 are the average performance throughout time steps for thirty runs. We can see that for the worst parameter settings, the performance at most time steps are quite competitive with the best and the default ones, but there are only a few time steps for each, at which the learning machines so extremely bad that they cannot even be shown in the figures (they are not necessarily unlimited, but they are so large that showing them will make other part of the figures much less visualized).

In summary, usually the performance of the best parameter settings outperform the other two in most time steps.

5.3 How Could Ensemble Help?

In this section, we focus on answering RQ3: Could Bagging help to lessen the base learners' sensitivity to parameter settings?

From tables 3(c) and 3(d), we can see that the effect size between the best and the default parameter settings for Bagging+MLP are all smaller than the ones for MLPs in all data sets. The effect sizes decrease from 0.149 (Kitchenham), 1.281 (Maxwell), and 0.922 (SingleISBSG) for MLPs to 0.084 (Kitchenham), 0.440 (Maxwell), and 0.405 (SingleISBSG) for Bagging+MLPs. That means that the performance of the default parameter settings for Bagging+MLPs is closer to the best ones than that for MLPs on their own. Also, the default and the best curves are much closer in figure 5 than the ones in figure 4, which indicates that besides helping to shorten the difference of average performance between the default and the best parameter settings obtained by MLPs, Bagging also helps to shorten the difference in terms of each time step.

From table 4(c) and 4(d), we can see that a similar result is obtained by Bagging+RTs and RTs: the effect size between the best and the default parameter settings for Bagging+RTs are all smaller than the ones for RTs in all data sets, which means the performance of the default ones for Bagging+RTs is closer to the best ones than that for RTs on their own. Furthermore, figure 7 and 8 show that Bagging helps to drag the curves of the default parameter settings closer to the best one throughout time steps.

Overall, our experiments indicate that combining learning machines into bagging can help making the performance of the default parameters to get closer to the best ones.

6. THREATS TO VALIDITY

There exists many learning machines such as Radial Basis Function Networks (RBFs), Bagging with RBFs, Negative Correlation Learning (NCL) with MLPs used in SEE, and some other standard data sets. Due to the limitation of time, in this paper, we only performed experiments on five representative approaches using three standard data sets to investigate whether sensitivity to parameter settings dif-

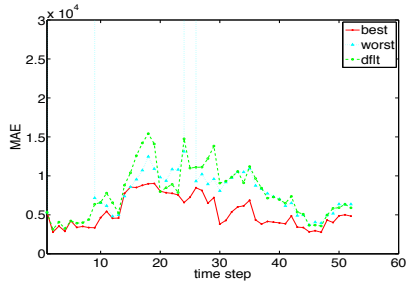

Figure 4: Performance on each time step for MLPs in Maxwell

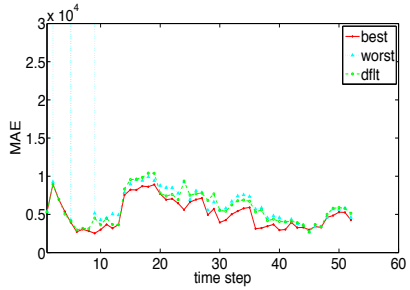

Figure 5: Performance on each time step for Bagging+MLPs in Maxwell

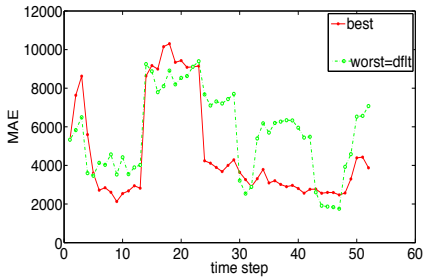

Figure 6: Performance on each time step for K-NN in Maxwell

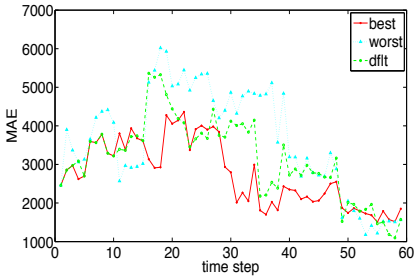

Figure 7: Performance on each time step for RTs in SingleISBSG

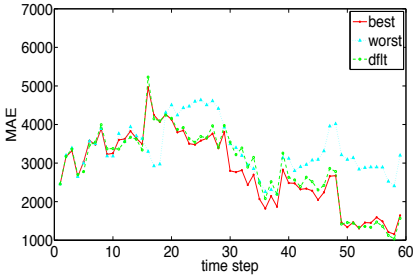

Figure 8: Performance on each time step for Bagging+RTs in SingleISBSG

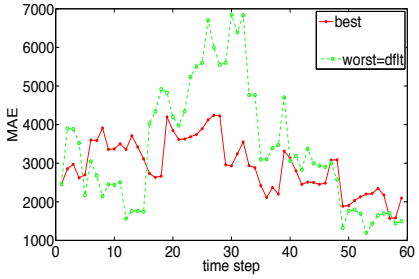

Figure 9: Performance on each time step for K-NN in SingleISBSG

fer accordingly. Additional learning machines and data sets should be investigated in the future.

Another potential threat is the fact that the possible values for a parameter are unlimited when its definition domains belongs to the real number field, and it is impossible to use all possible parameter values here. So, it is impossible for any study to draw the full picture of a learning machine's sensitivity to different parameter settings. We believe that the values shown in table 2 form a good range for each of the parameter considered in the study, but additional values should be investigated in future work.

7. CONCLUSIONS

This paper performs systematic experiments aiming at investigating to what extent parameter settings affect the performance of learning machines in SEE, and whether different learning machines are more/less sensitive to their parameters. It provides answers to the research questions as follows:

RQ1: Given an approach and a data set, how sensitive is this approach to different parameter settings in terms of its average performance across time steps? Different learning machines have different sensitivity to their parameter settings. For instance, RTs and Bagging+RTs are not quite sensitive to different parameter settings in terms of average performance across time steps, but parameter tuning is suggested in order to achieve a better performance. Though MLPs and Bagging+MLPs can achieve very good performance, they are extremely sensitive to their parameter settings, and even to the starting points. K-NN is not very sensitive to its parameter settings, but its performance when k equals to one (1-NN) behaves badly ($k = 1$ was the worst parameter setting in all data sets).

RQ2: Given an approach and a data set, does the best parameter setting in terms of average performance across time steps perform consistently well throughout time steps in comparison to other parameter settings? The best parameter settings commonly achieve a better performance than the default and the worst ones, though there are a few time steps in which the default or even the worst parameter settings outperform the best ones. k-NN is less stable than others in terms of the consistency of the best parameter settings across time steps, since it happens more frequently in k-NN than others that the best parameter settings perform the worst in some time steps.

RQ3: Could Bagging also help to lessen the base learners' sensitivity to parameter settings? Combining learning machines into bagging ensembles can help making the performance of the default parameters closer to the best parameter settings. Therefore, it would be an acceptable choice to combine MLPs and RTs into Bagging when using the default parameter settings, when there is no time to perform parameter tuning.

Among others, future work includes the investigation of other learning machines and data sets; other types of effect size, in particular non-parametric ones [1]; and other window sizes for the evaluation of the online learning procedure.

8. ACKNOWLEDGMENTS

Liyang Song would like to thank Fengzhen Tang for her useful comments and discussion. This work was supported by EPSRC grant EP/J017515/1. Liyan Song was supported by a School-funded PhD studentship. Xin Yao was supported by a Royal Society Wolfson Research Merit Award.

9. APPENDIX

Sequence of project ids from Kitchenham data set sorted according to completion order used in this study: 110, 115, 48, 112, 117, 125, 131, 79, 108, 46, 133, 47, 116, 119, 113, 111, 100, 107, 106, 19, 104, 44, 20, 98, 114, 18, 83, 105, 144, 99, 84, 109, 141, 22, 65, 118, 64, 85, 69, 92, 30, 17, 23, 55, 71, 66, 70, 86, 49, 53, 38,

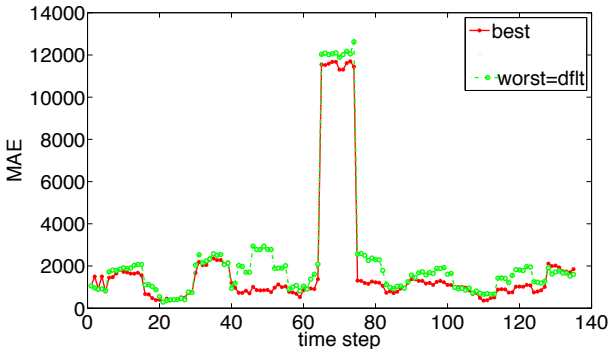

Figure 10: Performance on each time step for K-NN in Kitchenham

97, 34, 41, 77, 82, 51, 88, 140, 43, 132, 58, 60, 50, 81, 93, 31, 54, 57, 75, 76, 29, 1, 101, 102, 90, 16, 39, 67, 61, 24, 103, 35, 87, 59, 62, 68, 121, 134, 32, 56, 137, 26, 91, 120, 10, 40, 27, 45, 36, 89, 139, 2, 13, 135, 136, 5, 73, 138, 11, 15, 143, 63, 28, 96, 74, 4, 42, 127, 78, 12, 14, 52, 123, 25, 8, 124, 72, 37, 129, 33, 80, 94, 95, 6, 142, 21, 3, 9, 122, 126, 7, 128, 145, 130. We are unable to provide the sequence of ids for the other data sets.

10. REFERENCES

[1] A. Arcuri and L. Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *ICSE*, pages 1–10, 2011.

[2] A. Arcuri and G. Fraser. On parameter tuning in search based software engineering. In *SSBSE*, pages 33–47, Szeged, Hungary, 2011.

[3] N. Bettenburg, M. Nagappan, and A. E. Hassan. Think locally, act globally: Improving defect and effort prediction models. In *MSR*, pages 60–69, 2012.

[4] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, Singapore, 2006.

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

[6] L. Breiman and L. Breiman. Bagging predictors. In *Machine Learning*, pages 123–140, 1996.

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

[8] A. Corazza, S. Di Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. How effective is tabu search to configure support vector regression for effort estimation? In *PROMISE*, pages 4:1–4:10, 2010.

[9] A. Corazza, S. Martino, F. Ferrucci, C. Gravino, F. Sarro, and E. Mendes. Using tabu search to configure support vector regression for effort estimation. *ESE*, 18(3):506–546, 2013.

[10] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. Data mining techniques for software effort estimation: A comparative study. *IEEE TSE*, 38(2):375–397, 2012.

[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. Jorgensen. A review of studies on expert estimation of software development effort. *JSS*, 70(1–2):37–60, 2004.

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

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

[15] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. An empirical study of maintenance and development estimation accuracy. *JSS*, 64:57–77, 2002.

[16] E. Kocaguneli, T. Menzies, and J. Keung. On the value of ensemble effort estimation. *IEEE TSE*, 38:1403–1416, 2012.

[17] E. Kocaguneli, T. Menzies, and J. W. Keung. Kernel methods for software effort estimation. *ESE*, 18:1–24, 2013.

[18] 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.

[19] C. Lokan and E. Mendes. Applying moving windows to software effort estimation. In *ESEM*, pages 111–122, Lake Buena Vista, Florida, USA, 2009.

[20] C. Lokan and E. Mendes. Investigating the use of chronological split for software effort estimation. *IET-Software*, 3(5):422–434, 2009.

[21] C. Mair and M. Shepperd. The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In *International symposium on empirical software engineering*, page 10p., Banff, Canada, 2005.

[22] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. Local vs. global lessons for defect prediction and effort estimation. *IEEE TSE*, 39(6):822–834, 2013.

[23] T. Menzies and M. Shepperd. Special issue on repeatable results in software engineering prediction. *ESE*, 17:1–17, 2012.

[24] L. Minku and X. Yao. Ensembles and locality: Insight on improving software effort estimation. *IST*, 55(8):1512–1528, 2013.

[25] L. L. Minku and X. Yao. Can cross-company data improve performance in software effort estimation? In *PROMISE*, pages 69–78, Lund, Sweden, 2012, doi: `10.1145/2365324.2365334`.

[26] L. L. Minku and X. Yao. Software effort estimation as a multi-objective learning problem. *ACM TOSEM*, 2012 (to appear), final author's version available at `http://www.cs.bham.ac.uk/~minkull/publications/MinkuYaoTOSEM12.pdf`.

[27] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. Software productivity and effort prediction with ordinal regression. *IST*, 47:17–29, 2005.

[28] M. Shepperd and S. McDonell. Evaluating prediction systems in software project estimation. *IST*, 54:820–827, 2012.

[29] M. Shepperd and C. Schofield. Estimating software project effort using analogies. *IEEE TSE*, 23(12):736–743, 1997.

[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*, pages 771–776, Orlando, 2007.