Clustering Dycom: An Online Cross-Company Software Effort Estimation Study

MINKU, L.; HOU, S.

DOI:
10.1145/3127005.3127007

Document Version
Peer reviewed version

Citation for published version (Harvard):
MINKU, L & HOU, S 2017, Clustering Dycom: An Online Cross-Company Software Effort Estimation Study. in *Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE).* ACM/IEEE, Toronto, Canada, pp. 12-21. https://doi.org/10.1145/3127005.3127007

Clustering Dycom

An Online Cross-Company Software Effort Estimation Study

Leandro L. Minku
Department of Informatics, University of Leicester
University Road
Leicester LE1 7RH, UK
leandro.minku@leicester.ac.uk

Siqing Hou
Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology
Thuwal 23955-6900, Saudi Arabia
siqing.hou@kaust.edu.sa

ABSTRACT

Background: Software Effort Estimation (SEE) can be formulated as an online learning problem, where new projects are completed over time and may become available for training. In this scenario, a Cross-Company (CC) SEE approach called Dycom can drastically reduce the number of Within-Company (WC) projects needed for training, saving the high cost of collecting such training projects. However, Dycom relies on splitting CC projects into different subsets in order to create its CC models. Such splitting can have a significant impact on Dycom's predictive performance.
Aims: This paper investigates whether clustering methods can be used to help finding good CC splits for Dycom.
Method: Dycom is extended to use clustering methods for creating the CC subsets. Three different clustering methods are investigated, namely Hierarchical Clustering, K-Means, and Expectation-Maximisation. Clustering Dycom is compared against the original Dycom with CC subsets of different sizes, based on four SEE databases. A baseline WC model is also included in the analysis.
Results: Clustering Dycom with K-Means can potentially help to split the CC projects, managing to achieve similar or better predictive performance than Dycom. However, K-Means still requires the number of CC subsets to be pre-defined, and a poor choice can negatively affect predictive performance. EM enables Dycom to automatically set the number of CC subsets while still maintaining or improving predictive performance with respect to the baseline WC model. Clustering Dycom with Hierarchical Clustering did not offer significant advantage in terms of predictive performance.
Conclusion: Clustering methods can be an effective way to automatically generate Dycom's CC subsets.

CCS CONCEPTS

• **Software and its engineering → Software creation and management**; • **Computing methodologies → Supervised learning by regression**; **Transfer learning**; **Lifelong machine learning**; **Online learning settings**; **Ensemble methods**;

PROMISE , November 8, 2017, Toronto, Canada

ACM ISBN 978-1-4503-5305-2/17/11…$15.00
https://doi.org/10.1145/3127005.3127007

KEYWORDS

Software effort estimation, cross-company learning, concept drift, online learning, ensembles

ACM Reference format:
Leandro L. Minku and Siqing Hou. 2017. Clustering Dycom. In *Proceedings of PROMISE , Toronto, Canada, November 8, 2017,* 10 pages.
https://doi.org/10.1145/3127005.3127007

1 INTRODUCTION

Software Effort Estimation (SEE) is the process of estimating the effort required to develop a software project. The use of machine learning approaches for creating SEE models based on data describing completed projects has been studied for many years [5, 7, 15]. The process of creating such SEE models faces several challenges, such as the heterogeneity of software projects and the high cost associated to collecting Within-Company (WC) projects for training SEE models. These challenges can hinder the resulting SEE model's predictive performance.

However, SEE models could form useful tools to help experts with performing and/or re-thinking their estimations, which in turn can be used to inform many important project decisions, such as project bidding, requirements selection, task allocation, etc. Therefore, many studies have been attempting to tackle such challenges in order to improve the predictive performance of SEE models and facilitate their use. In particular, several studies have investigated the use of Cross-Company (CC) projects to reduce the cost of collecting WC projects for training SEE models [17, 26, 36].

The different processes and environments underlying different companies render CC projects potentially heterogeneous with respect to the projects being estimated [30], requiring solutions to tackle heterogeneity. Therefore, several studies proposed ways to tackle heterogeneity in CC SEE [20, 22, 26, 35, 36]. In particular, the approach Dycom [35] managed to drastically reduce the number of WC projects used for training while maintaining or slightly improving predictive performance in comparison with WC models. Dycom is the only approach able achieve that while treating the online learning nature of the SEE problem, i.e., the fact that new projects arrive over time and that SEE models must be able to adapt to changes that could otherwise affect their suitability.

In order to use CC projects for SEE, Dycom splits the CC projects into different subsets according to their productivity. Specifically, pre-defined productivity thresholds are used to decide the number of CC subsets and their composing projects. As each subset contains projects within the same productivity range, each subset is expected

to contain projects that are more homogeneous to each other. These subsets are thus used to create different CC SEE models. Mapping functions are then learned to dynamically map the estimations given by the CC models to estimations reflecting the current context (in particular, the relationship between project input attributes and effort) of a given company. These mapping functions are learned by using a small number of WC projects received over time.

However, the CC splits can have a significant impact on Dycom's predictive performance [31], and it is unclear how to best split the CC projects into different subsets. This paper aims at investigating whether clustering methods can be used to help finding good CC splits for Dycom. The following research questions are answered:

- RQ1 Can clustering methods help to improve Dycom's predictive performance? Which ones?
- RQ2 Can clustering methods facilitate the creation of CC splits by automatically deciding not only the content, but also the number of CC subsets? Which ones?

In order to answer these questions, Dycom is extended to use clustering methods for splitting the CC projects. The new approach is called *Clustering Dycom*. Three different clustering methods are investigated, namely Hierarchical Clustering, K-Means and Expectation-Maximisation (EM). The analysis reveals that K-Means can potentially help to split the CC projects. Its best and worst case predictive performances were similar or better than those obtained by the original Dycom (RQ1). However, K-Means requires the number of CC subsets to be pre-defined. A poor choice can cause the predictive performance to become worse than that of a baseline WC model. EM enables Dycom to automatically set the number of CC splits, while still maintaining or improving predictive performance with respect to the baseline WC model (RQ2). Clustering Dycom with Hierarchical Clustering did not offer significant advantage in terms of predictive performance over the original Dycom.

This paper is further organised as follows. Section 2 explains related work. Section 3 explains the original Dycom. Section 4 explains how to use Dycom with clustering methods. Section 5 explains the databases used in the study. Section 6 explains the experimental setup used to answer the RQs. Section 9 discusses threats to validity. Sections 7 and 8 present the analysis to answer RQ1 and RQ2, respectively. Section 10 presents the conclusions and future work.

2 RELATED WORK

This section discusses related work on CC SEE and on tackling heterogeneity in SEE. Many studies have investigated the predictive performance of CC versus WC SEE models. A systematic literature review published by Kitchenham et al. [17] found ten studies, seven of which were independent. Three of these concluded that the predictive performance of CC and WC models was similar, whereas the other four concluded that the predictive performance of CC models was worse. McDonell and Shepperd [24] also performed a systematic literature review to investigate whether CC models have similar predictive performance to WC models, and found no strong evidence in support of either type of model. Ordinary least squares regression, stepwise regression, robust regression, regression trees and k-nearest neighbours were among the machine learning approaches investigated.

Several of these studies used CC projects in an attempt to completely eliminate the need for WC projects [17, 24] . They compared WC models against models created only with projects from other companies (e.g., [6, 44]). Some other studies used both CC and WC projects in at least part of the procedure of building SEE models [14, 16, 23]. For example, Lefley and Shepperd [23] trained SEE models with both CC and WC projects, in an attempt to overcome the small number of WC projects [23].

Both systematic reviews mention that the level of heterogeneity of the single-company being estimated may vary and influence the WC models' performance. This has influenced more recent studies that use local learning approaches to deal with the heterogeneity of WC and CC projects. For example, Kocaguneli et al. [20] used a tree-based filtering mechanism, obtaining very encouraging results. Models trained solely with different types of projects (or projects from different centres of a company) from the ones being estimated obtained overall similar performance to models trained on projects of the same type (or from the same centre). Turhan and Mendes [42] investigated the use of a filtering mechanism based on k-nearest neighbours, obtaining very encouraging results in the area of web effort estimation. Specifically, filtered stepwise regression CC models achieved similar performance to WC models in seven out of eight data sets, and worse performance in only one data set.

Menzies et al. [25] proposed to cluster WC+CC projects using a method called WHERE, which splits projects according to the input attributes of highest variability. Prediction rules are then created for each cluster based on a method called WHICH. Given a certain cluster, its neighbouring cluster with the lowest required efforts was referred to as the *envied* cluster. When making predictions for WC projects belonging to a cluster, rules created using only the CC projects from the envied cluster were better than rules created using only the WC projects from the envied cluster. So, the authors recommended to cluster WC+CC projects, but to learn rules using solely the CC projects from the envied cluster. This study was in the context of both SEE and software defect prediction, but this particular conclusion was based only on the defect prediction data.

Other studies also investigated the use of clustering methods to tackle heterogeneity. For example, Huang et al. [12] investigated K-Means and Scheffe's method to cluster CC projects based on their input attributes. An ordinary least squares model was created for each cluster. When estimating a new project, the cluster to which this project belongs is determined and its corresponding SEE model is used for performing the estimation. The predictive performance obtained using clustering was similar to that obtained without clustering. Gallego et al. [10] partitioned CC projects into different subsets based on certain input attributes of interest. Each of the partitions was then further clustered using EM. Linear or exponential regression models were created for each cluster. Predictions were made in a similar way to Huang et al. [12]'s method. The authors concluded that clustering can improve predictive performance, even though no statistical tests were used in this comparison. These methods were evaluated for making predictions for CC projects, i.e., no WC data set was used in these studies.

It is important to note that, even though CC/WC projects could be considered as potentially more/less heterogeneous with respect to the projects being estimated, WC projects may also be heterogeneous. Therefore, the terms CC/WC should not be considered as

synonyms of heterogeneous/homogeneous [30], and the possible heterogeneity of WC projects should be tackled. Menzies et al. [21] and Minku and Yao [34] investigated the use of tree-based SEE models to tackle heterogeneity in general, i.e., not restricted to CC projects. Other local approaches such as k-nearest neighbours [2, 40] could also be seen as tackling heterogeneity.

The studies above did not take into account the fact that SEE is an online learning problem, where new projects need to be predicted over time, and changes suffered by the company may affect the quality of existing SEE models [33, 36]. With that in mind, Dynamic Cross-company Learning (DCL) [33, 36] creates an ensemble of WC and CC models and dynamically identifies which of them is currently beneficial for SEE. The beneficial models are emphasised when the ensemble is asked to estimate a new WC project. This approach managed to improve predictive performance in comparison with a WC model. However, DCL can only benefit from CC projects when they match the current WC context reasonably well. It still requires a fair amount of WC projects to achieve good performance during the periods when the CC models are not beneficial.

Kocaguneli et al.[22] investigated a tree-based filtering mechanism called TEAK [21] to tackle heterogeneity. This mechanism creates trees to represent training projects and provide effort estimations. CC or WC training projects corresponding to sub-trees of high variance are assumed to be detrimental and filtered out. Besides investigating TEAK for CC web effort estimation, Kocaguneli et al. [22] also investigated its ability to transfer knowledge from the past to the present in conventional WC SEE. The approach managed to obtain similar performance when using only training projects from the same time period as the project being estimated, and when using a mix of training projects from the same and different time periods. However, similar to DCL [33, 36], this approach still relies on a good number of projects from the same time period to be available for the process of generating the SEE model.

The approach Dycom [35] has been proposed to overcome this limitation. Similar to DCL, it maintains an ensemble of WC and CC models. However, instead of just identifying which past WC or CC models are more beneficial, it maps the estimations given by the CC models to the WC context. In this way, it can significantly reduce the number of WC projects needed for training while maintaining or slightly improving predictive performance in comparison with a WC model. However, Dycom requires CC projects to be split into different subsets to create its CC models, and the choice of CC split can significantly affect Dycom's predictive performance. Therefore, it would be desirable to have a method to facilitate the splitting process. As explained in section 1, this paper aims to investigate whether clustering methods can help with that. Section 3 explains Dycom in more detail.

3 DYCOM

Dycom (a.k.a. Dynamic Cross-company Mapped Model Learning) is an SEE online learning approach based on the observation that there is a relationship between the SEE context of a certain company and other companies. Minku and Yao [35] formalise the relationship between two companies C_A and C_B as follows:

$$f_A(\mathbf{x}) = g_{BA}(f_B(\mathbf{x})) \quad (1)$$

where C_A is the company in which we are interested; C_B is another company (or a subset of this other company); f_A and f_B are the true functions providing C_A's and C_B's required efforts, respectively; g_{BA} is a function that maps the effort from C_B's context to C_A's context; and $\mathbf{x} = [x_1, x_2, \cdots, x_n]$ are the input attributes describing a software project. The functions f_A, f_B and g_{BA} can be of any type. An illustrative example where $f_A(\mathbf{x}) = g_{BA}(f_B(\mathbf{x})) = 1.2 \cdot f_B(\mathbf{x})$ can be found in [35].

Given equation 1, the task of learning an SEE model for a company C_A can involve the task of learning the relationship between C_A and other companies. Therefore, Dycom uses CC training projects to learn one or more CC models, and uses a very limited number of WC training projects to learn a function that maps the estimations given by each CC model to estimations in the WC context. It is hoped that the task of learning the mapping functions is less difficult than the task of learning a whole WC model based solely on the WC training projects, as this would considerably reduce the amount of WC projects required for learning. As summarised by Minku et al. [32], Dycom works as follows.

CC training projects: Dycom uses CC training projects that are available beforehand. In [35], they were split into M different training subsets C_{Bi}, $1 \leq i \leq M$, of similar size based on productivity thresholds. For example, if the productivity of the CC training projects in terms of effort (in person-months) divided by size varies from 3 to 38, one may wish to separate these projects into three subsets, one containing projects with productivity below 8, one containing projects with productivity between 8 and 16, and one for projects with productivity higher than 16, if these productivity thresholds lead to subsets containing a similar number of projects. Each subset C_{Bi} is considered as a separate CC training set. The reason for splitting CC projects will be explained later in the paragraph on mapping functions. Note that we use the term CC loosely herein. For example, projects from different departments within the same company could be considered as CC projects if such departments employ largely different practices.

CC SEE models: Each of the M CC training sets is used to create a different CC model $\hat{f}_{Bi}$ ($1 \leq i \leq M$).

WC training projects: Dycom considers that the WC projects arrive in order of completion, i.e., online. In particular, it considers that one WC project arrives at each *time step*. WC projects that arrive at every p ($p > 1$) time steps contain both information on their input attributes and actual effort. All remaining WC projects contain only the information on the input attributes, whereas their actual effort, which is more difficult and costly to collect [19], is missing. So, even though an effort estimation is required for all WC projects, only a few of them (those arriving at time steps multiple of p) can be used as training projects.

Mapping functions: Whenever a new WC training project arrives, each model $\hat{f}_{Bi}$ is asked to perform an SEE. Each SEE is then used to create a mapping training example $(\hat{f}_{Bi}(\mathbf{x}), y)$. A mapping function $\hat{g}_{BiA}$ that receives estimations $\hat{f}_{Bi}(\mathbf{x})$ in the context of C_{Bi} as input and maps them to estimations in the context of C_A is trained with the mapping training example. Dycom considers that the relationship formalised in equation 1 can be modelled reasonably well by linear functions of the format $\hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x})) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$ when different subsets containing relatively more similar CC training

projects are considered separately. This is the reason to split CC training projects into different subsets.

Learning a function of the format $\hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x})) = \hat{f}_{Bi}(\mathbf{x}) \cdot b_i$ is equivalent to learning the factor b_i. This is done using equation 2:

$$b_i = \begin{cases} 1, & \text{if no mapping training example has been received yet} \\ \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})}, & \text{if } (\hat{f}_{Bi}(\mathbf{x}), y) \text{ is the first mapping training example} \\ lr \cdot \dfrac{y}{\hat{f}_{Bi}(\mathbf{x})} + (1 - lr) \cdot b_i, & \text{otherwise.} \end{cases} \quad (2)$$

where $(\hat{f}_{Bi}(\mathbf{x}), y)$ is the mapping training example being learnt, lr ($0 < lr < 1$) is a pre-defined smoothing factor and the factor b_i in the right side of the equation is the previous value of b_i. This equation enables the mapping function to dynamically adapt to potential changes affecting software effort over time. For a more detailed explanation of this equation, we refer the reader to [35].

WC SEE model: Whenever a new WC training project arrives, it is used to train a WC model $\hat{f}_{W_A}$. This model is not expected to perform very well, because it will be trained on a limited number of projects. However, its effort estimations may be helpful when used in an ensemble together with the mapped estimations, given that ensembles have been showing to improve SEE considerably [33, 34, 36]. It is also worth noting that, despite being potentially more homogeneus than CC projects, the WC projects could possibly also present a significant level of heterogeneity. In order to tackle this heterogeneity, it is recommended to use Dycom with WC models that are local approaches, such as decision trees [34].

Dycom's SEEs: Both the mapped models $\hat{g}_{BiA}(\hat{f}_{Bi})$ and the WC model $\hat{f}_{W_A}$ can provide an SEE in the WC context when required. The SEE given by Dycom is the weighted average of these $M + 1$ estimations:

$$\hat{f}_A(\mathbf{x}) = \left[\sum_{i=1}^{M} w_{Bi} \cdot \hat{g}_{BiA}(\hat{f}_{Bi}(\mathbf{x}))\right] + w_{W_A}\hat{f}_{W_A}(\mathbf{x}), \quad (3)$$

where the weights w_{Bi} and w_{W_A} represent how much we trust each of the models, are positive and sum to one. So, Dycom uses an ensemble of mapped and WC SEE models.

Weights: The weights are initialised so that they have the same value for all models being used in the ensemble and are updated as as follows: whenever a new WC training project is made available, the model which provided the lowest absolute error is considered to be the *winner* and the others are the *losers*. The losers have their weights multiplied by a pre-defined parameter β ($0 < \beta \leq 1$), and then all weights are normalised in order to sum up to one.

Dycom's pseudocode can be found in [35], and is omitted from here due to space limitations.

4 CLUSTERING DYCOM

CC projects could potentially be split into different CC subsets based on clustering algorithms, rather than pre-defined productivity thresholds. For that, each cluster of CC projects can be considered as a CC subset. We will refer to this approach as *Clustering Dycom*. Clustering Dycom offers potential advantages over pre-defined productivity thresholds. Clustering algorithms could automatically find out which CC projects are most similar to each other and group them together. In this way, project managers using Clustering Dycom would not need to analyse the CC projects and their productivity in order to determine good thresholds to be used. A good clustering algorithm may be able to further boost Dycom's predictive performance, or avoid low predictive performance resulting from poor thresholds. Moreover, certain clustering algorithms may be able to automatically decide the number of CC subsets to be used. This would further facilitate the use of Dycom, as the number of CC subsets would not need to be chosen beforehand.

Different features can be used to describe training projects for clustering. We investigate three different sets of clustering features:

- Productivity, measured by effort divided by size. We investigate productivity because the original Dycom achieved good results when splitting CC projects based on productivity thresholds.
- Size and effort. Together, size and effort represent the productivity associated to a project. However, two projects with similar productivities could still have very different sizes. Given that size is an important factor for estimating effort [36], we investigate the use of clustering based on the size and effort of projects.
- All project input and output attributes. We have investigated the use of all attributes of a project to check whether a more detailed characterisation of projects could lead to better results than using only size and effort or productivity.

Please note that it is ok to use the effort as (part of) a feature for clustering, because (1) clustering is applied only to the CC *training* projects, and (2) Dycom does not require to determine the cluster to which a project being estimated belongs.

Three different clustering methods were investigated: Hierarchical Clustering, K-Means, and EM. These three clustering methods, as well as the reasons for choosing them for the investigation, are explained below.

4.1 Hierarchical Clustering

Hierarchical clustering methods [37] build a hierarchy of clusters by putting together projects that are similar to each other. In this work, we have used an agglomerative hierarchical method. This method uses a bottom-up approach in which each project is considered to be a cluster in the lowest level of the hierarchy. Then, at each step of the algorithm, the two clusters that are closest to each other are merged together. In the end of the procedure, a single cluster containing all the projects is created. In order to use the clusters provided by this method, the level of the hierarchy corresponding to the desired number M of CC subsets is chosen.

Different ways to measure the similarity between clusters can be used. In this work, the distance between two clusters is defined as the Manhattan distance between the two most distant projects, one from each cluster. The Manhattan distance between two projects $\mathbf{x}_1$ and $\mathbf{x}_2$ is defined as $\sum_{i=1}^{F} |x_{1i} - x_{2i}|$, where F is the number of clustering features, and x_{1i} and x_{2i} represent the ith feature of projects $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. If a given feature i is nominal, $|x_{1i} - x_{2i}|$ is set to 1 if x_{1i} and x_{2i} have different values, and to 0 otherwise. As the Manhattan distance is affected by the scale of the features, all numeric features are normalised to be within $[0, 1]$.

This clustering method has been chosen because hierarchical SEE models have obtained promising results for tackling heterogeneity [21, 34]. It also has the advantage of being a deterministic method, i.e., it will always retrieve the same clusters when the same projects and features are used.

4.2 K-Means

K-Means [37] is an iterative clustering method that tries to minimise the distance of the projects to their cluster centres. It first randomly assigns projects to a pre-defined number k of clusters, where k equals to the desired number M of CC subsets to be generated. Then, in each iteration, projects are reassigned to the clusters with the closest centres, and the clusters centres are re-computed. This iterative procedure is typically halted when projects stop moving between clusters, or when a maximum number of iterations is reached. Similar to hierarchical clustering, Manhattan distance has been used as the distance measure.

This clustering method has been chosen because it is one of the most popular and well known clustering methods in the literature.

4.3 EM

EM [4] is a density-based clustering method. It assumes that the projects belonging to each cluster are drawn from a specific distribution. Therefore, this method tries to identify the clusters and their probability distributions. Given a number of clusters M, EM first randomly assigns random values for the parameters θ_j of the probability distributions associated to each cluster C_j. It then performs two steps: expectation and maximisation. In the expectation step, the posterior probability $p(C_j|\mathbf{x})$ that a given project $\mathbf{x}$ belongs to a given cluster C_j is estimated based on the current parameters of the probability distributions. In the maximisation step, the parameters that maximise the log likelihood of the projects $ln\ p(\mathbf{x}|\theta)$ are calculated (note that $p(C_j|\mathbf{x})$ is used to compute that) and used to replace the previous parameters. The algorithm iterates through the expectation and maximisation steps until a stopping criterion is met. This could be when a pre-defined maximum number of iterations is reached, or when the the changes in the log likelihood $ln\ p(\mathbf{x}|\theta)$ become smaller than a threshold. The distributions are assumed to be Gaussian for numeric features, whereas discrete estimators (i.e., frequency counts) are used to characterise the distribution of nominal features.

This algorithm can automatically determine the number of clusters by using 10-fold cross-validation on the CC training projects being clustered. This is done through the following procedure [11]:

(1) The number of clusters (which is equal to the number M of CC subsets) is set to 1.
(2) The CC training projects are divided randomly into 10 folds.
(3) EM is performed 10 times using the 10 folds in the usual cross-validation way. Specifically, in each time, a different fold is used to compute the log likelihood and the remaining folds are used for clustering.
(4) The log likelihood is averaged over all 10 results.
(5) The process above is repeated by increasing the number of clusters by 1 while improvements in the log likelihood are observed. If no improvements are observed, the procedure stops and retrieves the number of clusters with the maximum log likelihood so far.

If there are less than 10 projects, the number of folds is set to the number of projects in order to perform cross-validation.

This algorithm has been chosen because of its potentially useful procedure to automatically determine the number of clusters.

5 DATABASES

The databases used in the experiments to answer the RQs outlined in section 1 are the same as the ones used in Dycom's original paper [35]. Part of their descriptions is transcribed here to make this paper more self-contained. The key difference between the use of these databases in Dycom's original [35] and current paper is that the former fixed the CC subsets based on specific thresholds. The current paper will investigate different CC splits, as explained in sections 4 and 6.

Five different databases were used: KitchenMax, CocNasaCoc81, ISBSG2000, ISBSG2001 and ISBSG. These include both data sets derived from the former PROMISE Repository [27] (now SEACRAFT Repository [28]) and the ISBSG Repository [13]. Each database contains a WC data set and a CC data set.

5.1 KitchenMax

The database KitchenMax is composed of Kitchenham[1] and Maxwell[2], which are two SEE data sets available from the SEACRAFT Repository [28]. Kitchenham's detailed description can be found in [18]. It comprises 145 maintenance and development projects undertaken between 1994 and 1998 by a single software development company. Maxwell's detailed description can be found in [38]. 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. In order to make these data sets compatible, a single input attribute (functional size) was used. Still, Maxwell uses functional size, whereas Kitchenham uses adjusted functional size. An appropriate mapping function should be able to overcome this problem. There were no functional size attribute values missing. The output attribute is the effort in person-hours.

Kitchenham was considered as the WC data, and was sorted according to the actual start date plus the duration. This sorting corresponds to the exact completion order of the projects. Maxwell's projects were considered as the CC projects. Figure 1a shows the productivity of the CC projects in terms of effort divided by size, which is used by the original Dycom to create the CC subsets.

5.2 CocNasaCoc81

The database CocNasaCoc81 is composed of Cocomo Nasa and Cocomo 81, which are two SEE data sets available from the former PROMISE Repository [27]. Cocomo Nasa contains 60 Nasa projects from 1980s-1990s and Cocomo 81 consists of the 63 projects analysed by Boehm to develop the software cost estimation model COCOMO [5] first published in 1981. Both data sets contain 16 input attributes (15 cost drivers [5] and number of lines of code) and one output attribute (software effort in person-months). Cocomo 81 contains an additional input attribute (development type) not

[1] https://doi.org/10.5281/zenodo.268457
[2] https://doi.org/10.5281/zenodo.268461

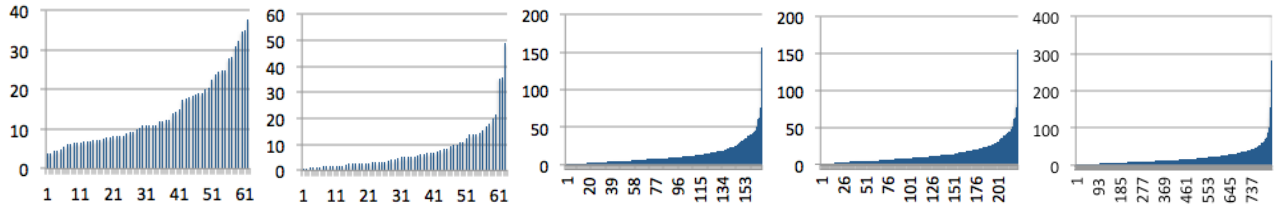

Figure 1: CC projects' productivities (effort divided by size is shown in the y-axis). Projects are sorted from the most productive to the least productive.

present in Cocomo Nasa, which was thus removed. These data sets contain no missing values.

Cocomo Nasa's projects were considered as the WC and Cocomo 81's as the CC projects. The productivity of the CC projects is shown in figure 1b. Cocomo Nasa provides no information on whether the projects are sorted in chronological order. The original order of the Cocomo Nasa projects was preserved in order to simulate a given WC projects chronology scenario. Even though this may not be the true chronological order, it is still useful to evaluate whether approaches are able to make use of mapped CC models when/if they are beneficial. Therefore, this is adequate for the purpose of answering the RQs outlined in section 1.

5.3 ISBSG Databases

Three SEE databases were derived from ISBSG [13] Release 10, which contains software project information from several companies. Information on which projects belong to a single company for composing a WC data set has been provided to us upon request. The databases are:

- ISBSG2000 – 119 WC projects implemented after the year 2000 and 168 CC projects implemented up to the end of 2000.
- ISBSG2001 – 69 WC projects implemented after the year 2001 and 224 CC projects implemented up to the end of 2001.
- ISBSG – no date restriction to the 187 WC and 826 CC projects, meaning that CC projects with implementation date more recent than WC projects are allowed. This database simulates the case where some other companies are likely to be be more evolved than the single company analysed.

Information on how these databases were preprocessed can be found in [33] and is not included here due to space limitations. Four input attributes (development type, language type, development platform and functional size) and one output attribute (software effort in person-hours) were used. The WC projects were sorted based on the implementation date to compose a stream of incoming projects. Figures 1c, 1d and 1e show the productivities of the CC projects of these databases.

6 EXPERIMENTAL SETUP

In order to answer the RQs outlined in section 1, Clustering Dycom was compared against the original Dycom with different CC splits. A baseline WC approach was also used to complement the analysis. The experiments followed the experimental setup from the paper proposing Dycom [35]. In particular, Regression Trees (RTs) were used as Dycom's base learners. As explained in [35], RTs were chosen because they have shown competitive performance in comparison with several other SEE approaches [34]. They are also local approaches, which can help dealing with the heterogeneity within each data set. The approaches used to answer the RQs are explained over the next three paragraphs.

Dycom: Dycom was used with RTs as the CC and WC models with $p = 10$, i.e., a new WC training project was provided only at every 10 time steps. So, Dycom uses only 10% of the WC projects for training, as in [35]. Whenever a new WC training project was made available, the WC RT used by Dycom was rebuilt from scratch using all WC training projects so far. Minku and Yao [35] suggested that the CC splits are created in such a way that different CC subsets have similar size. Therefore, different CC splits were created in the following way [31]. Given a desired number M of CC subsets, productivity thresholds were set to the values that lead to M CC subsets with the same number of projects. Least productive CC subsets may have one project less than the most productive CC subsets if the total number of CC projects is not a multiple of M. The experiments used the RT implementation *REPTree* provided by WEKA [11], where splits are created so as to minimise the variance of the output attributes of the training projects in the nodes.

Clustering Dycom: In order to provide a fair comparison with Dycom, Clustering Dycom also used RTs as the CC and WC models with $p = 10$. The experiments used the WEKA [11] implementation of the clustering methods.

RT: RTs were created to reflect WC online learning, forming a baseline for the analysis, as in [35]. Whenever a new WC training project was provided, the current RT was discarded and a new RT was trained on all projects so far. This approach considers that all WC completed projects have their true effort known, i.e., every time step receives a WC training project. Therefore, it uses ten times more WC training projects than Dycom and Clustering Dycom.

The parameters of all approaches were set to the same values as in previous work [35], except for the number M of CC subsets, and the clustering parameters. Dycom and Clustering Dycom were run with six different values $M = \{1, \cdots, 6\}$ in order to answer RQ1 and RQ2, except for Clustering Dycom with EM, which automatically determines an appropriate value for M. The clustering parameters, which were not needed in [35], were set so as to run the methods described in section 4. The stopping criteria for K-Means and EM were set to the same default value of 500 iterations, and EM's minimum standard deviation was set to the default value of 1.0E-6. The use of default values avoids giving an unfair advantage to Clustering Dycom over Dycom in the experiments. Future work

will investigate whether Clustering Dycom's performance could be improved further by fine tuning the clustering parameters. Dycom's and RT's parameters from [35] were the following. Dycom's parameter β was set to the default value of 0.5. Dycom's parameter lr was set to 0.1 after some preliminary investigation with 0.1 and 0.05. The parameters used with each RT were the ones more likely to obtain good results in previous work [34]: minimum total weight of 1 for the instances in a leaf, and minimum proportion of the variance on all the data that need to be present at a node in order for splitting to be performed 0.0001.

As in [35], at each time step, effort estimations given for the next ten WC projects were performed and evaluated. None of these projects were used for training before being used for evaluation. In this work, Mean Absolute Error (MAE) and Standardised Accuracy (SA) were used to evaluate the estimations. The equations to calculate MAE and SA are the following: $MAE = \frac{1}{T}\sum_{i=1}^{T}|\hat{y}_i - y_i|$, and $SA = (1 - MAE/MAE_{rguess}) \cdot 100$, where T is the number of projects used for evaluating the performance, y_i is the actual effort for the project i, $\hat{y}_i$ is the estimated effort for project i, MAE is the MAE of the approach being evaluated, and MAE_{rguess} is the MAE of 1000 runs of random guess. Random guess is defined as sampling uniformly at random the true effort over all the WC projects received up to the current time step. Therefore, it has access to the same number of WC training projects as RT, which is 10 times higher than the number of WC projects used by Dycom and Clustering Dycom.

MAE has been recommended by Shepperd and McDonell [39] for being unbiased towards under or overestimations. SA is an unbiased measure that allows for interpretability – it is viewed as the ratio of how much better an approach is than random guess [39]. So, it can be used to give a better idea of the magnitude of the differences in performance. Mean Magnitude of the Relative Error (MMRE) and percentage of predictions within 25% of the actual value (PRED(25)) were not used because they are biased towards underestimations [39] and could lead to misleading conclusions. The comparisons involving MAE are supported by Wilcoxon Sign-Rank tests with Holm-Bonferroni corrections at the overall level of significance of 0.05, and A12 [43], which is one of the measures of effect size recommended by Arcuri and Briand [3]. Both Wilcoxon and A12 are non-parametric. As suggested by Vargha and Delaney [43], A12 of 0.50 indicates no difference, up to 0.56 indicates a small difference, up to 0.64 indicates medium difference and over 0.71 indicates large difference. Wilcoxon and A12 have not been used for comparing SA because SA is an interpretable equivalent of MAE.

Dycom and Clustering Dycom with Hierarchical Clustering were run a single time for each data set, as they are deterministic when using deterministic RTs. Clustering Dycom with K-Means and EM were run 30 times, and the MAE obtained at each time step was averaged across these 30 runs.

7 THE IMPACT OF CLUSTERING DYCOM ON PREDICTIVE PERFORMANCE

This section investigates if the use of clustering methods could help to create CC splits that lead to better predictive performance (RQ1). For that, the predictive performance obtained by Clustering Dycom with the 3 different clustering methods explained in section 4 is analysed and compared with Dycom and the baseline RT.

Each clustering method was analysed when using three different sets of project features (productivity, size and effort, and all attributes), as explained in section 4. The MAEs obtained when using size and effort, and when using all attributes, were worse than those obtained by using productivity in most cases. This further supports the fact that Dycom achieved good results when splitting CC projects based on productivity in [35]. Therefore, this section concentrates on the analysis of Clustering Dycom based on productivity. The results obtained with size and effort, and all attributes, are omitted due to space constraints.

Table 1 shows the results achieved by productivity-based Clustering Dycom, Dycom and the baseline RT. For Clustering Dycom with Hierarchical Clustering and K-Means, and for Dycom, the results correspond to the number M^* of CC subsets which achieved the top ranked MAE. For EM, the number of CC subsets was determined automatically (see section 4.3). RT does not use CC projects.

The statistical tests found no significant difference between the MAE of Dycom and Clustering Dycom with Hierarchical Clustering for KitchenMax, CocNasaCoc81 and ISBSG2001. For ISBSG2000 and ISBSG, significant difference was found. Even though Clustering Dycom's MAE was better than Dycom's for ISBSG2000, the A12 of the difference was small (0.53). Clustering Dycom's MAE was worse than Dycom's for ISBSG and A12 was medium-large (-0.69). So, the best MAE achieved by Hierarchical Clustering is similar or worse than that of Dycom. EM's results were also similar or worse than those of Dycom, with A12 varying from small to medium.

According to the MAEs and statistical tests, Clustering Dycom with K-Means obtained better MAE than Dycom for CocNasaCoc81. A12 was medium-large (0.70) and the difference in SA was of 21.55 units. Such difference has considerably large magnitude, and is thus likely to have an effect in practice. For the other databases, no significant difference was found. Therefore, Clustering Dycom with K-Means was able to maintain or improve Dycom's MAE. This suggests that grouping together projects in terms of the distance of their productivities could potentially be a helpful alternative to the original Dycom. Therefore, we investigate the results obtained with K-Means further.

Table 2 shows the results obtained when using K-Means with $k \in \{1, \cdots, 6\}$ and RT. The MAEs and statistical tests show that the top ranked Ms always obtained significantly better MAE than the RTs. The corresponding A12s varied from medium to large and differences in SA varied from 7.07 to 33.75, being very considerable and likely to have an effect in practice. Therefore, Clustering Dycom with K-Means can drastically reduce the number of WC training projects, while significantly improving predictive performance.

However, similar to the original Dycom [31], the number of CC subsets M can significantly affect Clustering Dycom with K-Means. A poor choice of M can lead to significantly worse MAE than the top ranked M, as shown by the statistical tests from table 2. The A12s of such differences varied from small to very large. Moreover, a poor choice of M could lead to worse results than the baseline RT. For instance, Clustering Dycom with K-Means led to significantly worse MAE than RT when using $M = 3$ for CocNasaCoc81 and $M = 1$ for ISBSG (p-values of 2.99E-09 and 6.09E-12, respectively), with very large differences in SA. Therefore, using K-Means involves some risk in terms of predictive performance.

Table 1: Results obtained by productivity-based Clustering Dycom, RT, and Dycom.

Approach / M^*	Database	MAE	SA	P-value	A12
Hierar. / 6	KitchenMax	2163	38.13	6.72E-01	-0.52
Hierar. / 5	CocNasaCoc81	224	53.15	2.04E-01	0.60
Hierar. / 3	ISBSG2000	2146	50.94	3.35E-03	0.53
Hierar. / 6	ISBSG2001	2344	43.01	6.89E-01	-0.52
Hierar. / 6	ISBSG	4137	31.71	1.35E-14	-0.69
K-means / 1	KitchenMax	2176	37.75	1.13E-01	-0.51
K-means / 2	CocNasaCoc81	158	66.89	8.03E-05	0.70
K-means / 6	ISBSG2000	2094	52.12	6.69E-01	0.51
K-means / 6	ISBSG2001	2315	43.70	1.60E-01	0.51
K-means / 3	ISBSG	2826	53.36	9.34E-01	0.51
EM / 2	KitchenMax	2333	33.27	2.26E-06	-0.61
EM / 2	CocNasaCoc81	308	35.57	2.30E-02	-0.64
EM / 3	ISBSG2000	2256	48.42	5.05E-01	-0.51
EM / 4	ISBSG2001	2640	35.82	7.66E-07	-0.58
EM / 5	ISBSG	2937	51.53	2.78E-03	-0.53
RT	KitchenMax	2441	30.18	3.01E-11	0.62
RT	CocNasaCoc81	319	33.14	1.41E-01	0.52
RT	ISBSG2000	2753	37.05	1.11E-05	0.62
RT	ISBSG2001	3622	11.93	1.83E-07	0.76
RT	ISBSG	3253	46.29	6.37E-06	0.58
Dycom / 6	KitchenMax	2165	38.06	–	–
Dycom / 2	CocNasaCoc81	261	45.34	–	–
Dycom / 6	ISBSG2000	2215	49.35	–	–
Dycom / 4	ISBSG2001	2353	42.79	–	–
Dycom / 3	ISBSG	2806	53.69	–	–

M^* is the number of CC subsets with the best ranked MAE, except for EM, where M^* is the median of the number of clusters automatically set by EM across 30 runs. The p-values are the results of the Wilcoxon Sign Rank tests to compare each approach's MAE against Dycom's. P-values highlighted in yellow (light grey) represent statistically significant difference at the overall level of significance of 0.05, when using Holm-Bonferroni corrections considering the 5 comparisons made for a given approach. Positive A12 (or absolute values of negative A12) represent the probability that the corresponding approach has better (worse) MAE than Dycom.

Table 2: Results obtained by Clustering Dycom with K-Means based on productivity, and RT.

	M	MAE	SA	P-value	A12
KitchenMax	1	2176	37.75	–	–
	2	2443	30.13	8.87E-06	0.62
	3	2269	35.11	3.26E-02	0.55
	4	2278	34.85	1.89E-01	0.55
	5	2269	35.10	2.66E-01	0.54
	6	2249	35.68	5.02E-01	0.53
	RT	2441	30.18	3.50E-16	0.60
CocNasaCoc81	1	541	-13.31	3.37E-09	0.81
	2	158	66.89	–	–
	3	818	-71.13	7.56E-10	0.99
	4	675	-41.33	7.56E-10	0.99
	5	598	-25.06	7.56E-10	0.98
	6	612	-28.19	7.56E-10	0.99
	RT	319	33.14	7.16E-07	0.66
ISBSG2000	1	2789	36.24	3.95E-14	0.70
	2	2342	46.44	1.37E-02	0.55
	3	2124	51.43	7.31E-01	0.51
	4	2372	45.76	1.48E-08	0.57
	5	2201	49.67	1.69E-02	0.51
	6	2094	52.12	–	–
	RT	2753	37.05	1.06E-06	0.65
ISBSG2001	1	2435	40.80	8.43E-03	0.55
	2	2663	35.25	1.71E-06	0.61
	3	2749	33.16	7.37E-07	0.58
	4	2483	39.61	8.53E-02	0.53
	5	2353	42.79	2.58E-01	-0.50
	6	2315	43.70	–	–
	RT	3622	11.93	6.76E-09	0.77
ISBSG	1	4961	18.11	2.14E-27	0.72
	2	3695	39.01	1.83E-14	0.64
	3	2826	53.36	–	–
	4	2921	51.78	3.09E-04	0.53
	5	2916	51.87	2.18E-08	0.53
	6	4145	31.58	5.54E-27	0.70
	RT	3254	46.29	2.02E-07	0.59

The column M indicates the number of CC subsets for Clustering Dycom with K-Means, or the baseline RT approach. The top and bottom ranked MAEs for each database are highlighted in lime (light grey) and orange (dark grey), respectively. The p-values are the results of the Wilcoxon Sign Rank tests to compare the MAE of each configuration/approach against the top ranked one. P-values highlighted in yellow (light grey) represent statistically significant difference at the overall level of significance of 0.05, when using Holm-Bonferroni corrections considering the 6 comparisons made for a given database. A12 represents the probability that the corresponding approach has worse MAE than the top ranked one.

Still, the worst MAEs obtained by Clustering Dycom with K-Means were similar or better than those obtained by the original Dycom. The results of the statistical tests and A12s to support this analysis are shown in table 3. When Clustering Dycom obtained significantly better MAE than Dycom (ISBSG2001 and ISBSG), A12 was medium and the differences in SA were of 9.97 and 15.02 units, being of considerable magnitude. Therefore, even though the number of clusters can affect the MAE achieved with K-Means, Clustering Dycom with K-means could be considered as a safer method than the original Dycom in the case of a poor choice of M.

In summary, this section shows that K-Means can help to create CC splits that lead to better predictive performance, giving a positive answer to RQ1.

8 THE EFFECT OF AUTOMATICALLY DETERMINING THE NUMBER M OF CC SUBSETS

K-Means obtained encouraging results in terms of predictive performance, as shown in section 7. However, a poor choice of the number M of CC subsets could lead to worse MAE than the baseline RT. Choosing the right M for Clustering Dycom with K-Means may not be an easy task. Therefore, this section investigates whether clustering methods can facilitate the creation of the CC splits by automatically deciding not only the content, but also the number of CC subsets (RQ2).

The clustering method EM can automatically decide the number of clusters, as explained in section 4. We have seen in section 7 that the MAE obtained by EM was similar or worse than the MAE

Table 3: A12 and p-values of Wilcoxon Sign-Rank tests for comparing the MAEs obtained by Clustering Dycom with K-Means and Dycom, using the bottom ranked M.

Database	P-value	A12
KitchenMax	2.99E-01	0.52
CocNasaCoc81	4.78E-01	0.52
ISBSG2000	6.95E-01	-0.53
ISBSG2001	1.07E-07	0.57
ISBSG	2.50E-05	0.57

P-values in yellow (light grey) indicate statistically significant difference at the overall level of 0.05 with Holm-Bonferroni corrections considering the 5 comparisons made.

Table 4: A12 and p-values of Wilcoxon Sign-Rank tests for comparing the MAEs obtained by RT and Clustering Dycom with EM.

Database	P-value	A12
KitchenMax	1.31E-01	0.52
CocNasaCoc81	9.81E-01	-0.57
ISBSG2000	6.46E-05	0.62
ISBSG2001	7.61E-06	0.70
ISBSG	2.49E-04	0.55

P-values in yellow (light grey) indicate statistically significant difference at the overall level of 0.05 with Holm-Bonferroni corrections considering the 5 comparisons made.

obtained by the original Dycom with its best number of CC subsets. However, this clustering method may still be a good option if it offers less risk than K-Means, i.e., if its MAE is no worse than that of the baseline RT. That is because this would mean that Clustering Dycom with EM can drastically reduce the number of WC training projects needed to be collected, while at least maintaining the predictive performance obtained by a WC model.

Table 4 shows the results of the comparison between the MAE of Clustering Dycom with EM and the MAE of the baseline RT. No significant difference was found for KitchenMax and CocNasaCoc81. For the other databases, Clustering Dycom obtained significantly better MAE than RT. A12s were 0.62 (medium), 0.70 (medium-large) and 0.55 (small-medium) for ISBSG2000, ISBSG2001 and ISBSG, respectively. The differences in SA were 11.37, 23.89 and 5.24, respectively. These differences are of considerable magnitude.

Therefore, when using Clustering Dycom with EM, it is possible to drastically reduce the number of WC training projects, while maintaining or even improving MAE. This means that EM can not only facilitate the creation of the CC subsets by deciding their content, but also their number, giving a positive answer to RQ2.

9 THREATS TO VALIDITY

The experiments used the same databases and use mostly the same setup as Dycom's original paper [35]. Therefore, they have similar threats to validity as follows. When using machine learning approaches, it is important that the approaches being compared use fair parameter choices in comparison to each other in order to address internal validity [29, 41]. In this paper, both the RTs used as WC learners and within Dycom and Clustering Dycom used the same parameters, which were the ones more likely to obtain good results in the literature [34]. (Clustering) Dycom has two extra parameters (β and lr) which were set to the same values for all databases used in this study, i.e., they were not fine tuned for each database and therefore should not lead to an unfair advantage to (Clustering) Dycom. The number M of CC subsets was varied from 1 to 6 to answer the RQs. The other parameters of the clustering methods were set to default values in order not to give Clustering Dycom an unfair advantage over Dycom or the baseline RTs.

ISBSG, ISBSG2000 and ISBSG2001 have an overlap of projects. Therefore, these databases are not independent. However, in online learning, the conclusions that may be obtained with different numbers of preceding and following projects can be very different [36]. For example, as will be shown in section 7, Hierarchical Clustering with 6 clusters appeared to be good for ISBSG2000, but led to poor results for ISBSG. Therefore, it is important to include all versions of the ISBSG database in the analysis, as in previous work [33, 35, 36].

In order to address construct validity, this study used MAE. This is a measure unbiased towards over or underestimations, and has been recommended for SEE studies [39]. SA [39] was also used in order to give a better idea of the magnitude of the differences in performance and the impact that they are likely to have in practice. Wilcoxon Sign-Rank statistical tests with Holm-Bonferroni corrections were used to check the statistical significance of the differences in terms of MAE. Effect size A12 was used to support the comparison, as it is independent of the number of observations in the groups being compared.

Besides never using a WC project for training before using it for testing, we used five databases to handle external validity. Four databases with known WC chronological order were used in the evaluation. Even though the WC chronological order is unknown for the other database (CocNasaCoc81), it can still be used to evaluate whether (Clustering) Dycom is able to successfully make use of CC models, contributing to the generalisation of our results. For ISBSG, some future CC projects were used to simulate the case where it is known that some other companies are likely to be more evolved than the single company being analysed. The use of simulation or synthetic data is common practice in online learning studies [8]. Obtaining additional databases for this study is difficult due to our need for non-proprietary data sets with information on which projects belong to a single-company among the projects of a cross-company data set. However, the databases used in this study can be made available through SEACRAFT [28] and ISBSG [13]. So, researchers and companies willing to use Dycom could use the same CC data sets used in this study. Future work should also investigate Dycom with other base learners, clustering methods, input attributes and clustering features.

As in [35, 36], CC data were considered as fixed CC datasets. As shown in [36], such fixed CC datasets can be useful for prolonged periods of time. So, Clustering Dycom as investigated here is applicable in practice. If the weights of all CC mapped models become too low for a prolonged period of time, this may be an indication that Dycom needs to be re-built with updated CC datasets. Future work should investigate a more streamlined approach to update CC models.

10 CONCLUSIONS

Dycom is a promising approach for SEE. It is able to reduce the amount of WC training projects required for training SEE models

while maintaining or improving the predictive performance of a corresponding WC approach. However, its good predictive performance depends on the choice of a CC split to create its CC models. A poor choice can lead to worse predictive performance than WC approaches. Therefore, this paper investigated whether clustering methods can help to create good CC splits for use with Dycom.

Three different clustering methods were investigated. Among them, K-Means was able to improve predictive performance over the original Dycom strategy for creating the CC splits (RQ1). However, the predictive performance obtained when using K-Means depends on the prior choice of a suitable number M of CC subsets. A poor choice can still lead to worse predictive performance than a baseline WC approach. Even though EM did not achieve so good predictive performance as Clustering Dycom with K-Means with the best M, it can automatically determine the number of CC subsets while maintaining or improving the predictive performance of a corresponding baseline WC approach (RQ2).

Therefore, if the company has the necessary machine learning expertise and an initial set of WC training projects that is large enough to tune Dycom's parameters, we recommend to use Clustering Dycom with K-Means in order to boost Dycom's predictive performance. Otherwise, we recommend to use Clustering Dycom with EM, in order to avoid the risk of obtaining worse results than a WC approach.

This work has several possible future research directions. In particular, other clustering methods, base learners, project input attributes, project features for clustering, parameter values [41] and (automated) tuning procedures [1, 9] could be investigated, besides the proposal of a more streamlined approach to update CC models.

ACKNOWLEDGEMENTS

Part of this work has been conducted during Siqing Hou's internship at the University of Birmingham (UK).

REFERENCES

[1] A. Agrawal and T. Menzies. 2017. "Better Data" is Better than "Better Data Miners" (Benefits of Tuning SMOTE for Defect Prediction). *ArXiv preprint arXiv:1705.03697* (2017).
[2] S. Amasaki, Y. Takahara, and T. Yokogawa. 2011. Performance Evaluation of Windowing Approach on Effort Estimation by Analogy. In *IWSM-MENSURA*. Nara, Japan, 188–195.
[3] A. Arcuri and L. Briand. 2011. A Practical Guide for using statistical tests to assess randomized algorithms in software engineering. In *ICSE*. 1–10.
[4] C.M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.
[5] B. Boehm. 1981. *Software Engineering Economics*. Prentice-Hall, Englewood Cliffs.
[6] L.C. Briand, T. Langley, and I. Wieczorek. 2000. A Replicated Assessment of Common Software Cost Estimation Techniques. In *ICSE*. Como, Italy, 377–386.
[7] K. Dejaeger, W. Verbeke, D. Martens, and B. Baesens. 2012. Data Mining Techniques for Software Effort Estimation: A comparative study. *IEEE TSE* 38, 2 (2012), 375–397.
[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar. 2015. Learning in Nonstationary Environments: A Survey. *CIM* 10, 4 (2015), 12–25.
[9] W. Fu, T. Menzies, and X. Shen. 2016. Tuning for Software Analytics: is it Really Necessary? *ArXiv preprint arXiv:1609.01759* (2016).
[10] J.J.C. Gallego, D. Rodriguez, M.A. Sicilia, M.G. Rubio, and A.G. Crespo. 2007. Software Project Effort Estimation Based on Multiple Parametric Models Generated Through Data Clustering. *JCST* 22, 3 (2007), 371–378.
[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. The WEKA Data Mining Software: An update. *SIGKDD Explorations* 11, 1 (2009), 10–18.
[12] S.-J. Huang, N.-H. Chiu, and Y.-J. Liu. 2008. A comparative evaluation on the accuracies of software effort estimates from clustered data. *IST* 50 (2008), 879–888.
[13] ISBSG. 2011. The International Software Benchmarking Standards Group. (2011). http://www.isbsg.org
[14] R. Jeffery, M Ruhe, and I. Wieczorek. 2010. A Comparative Study of Two Software Development Cost Modeling Techniques Using Multi-Organizational and Company-Specific Data. *IST* 42, 14 (2010), 1009–1016.
[15] M. Jørgensen and M. Shepperd. 2007. A Systematic Review of Software Development Cost Estimation Studies. *IEEE TSE* 33, 1 (2007), 33–53.
[16] B. Kitchenham and E. Mendes. 2004. A Comparison of Cross-Company and Within-Company Effort Estimation Models for Web Applications. In *METRICS*. Chicago, 348–357.
[17] B.A. Kitchenham, E. Mendes, and G.H. Travassos. 2007. Cross versus Within-Company Cost Estimation Studies: A Systematic Review. *IEEE TSE* 33, 5 (2007), 316–329.
[18] B. Kitchenham, S. L.. Pfleeger, B. McColl, and S. Eagan. 2002. An empirical study of maintenance and development estimation accuracy. *JSS* 64 (2002), 57–77.
[19] E. Kocaguneli, B. Cukic, T. Menzies, and H. Lu. 2013. Building a Second Opinion: learning cross-company data. In *PROMISE*. 12.1–10.
[20] E. Kocaguneli, G. Gay, T. Menzies, Y. Yang, and J. W. Keung. 2010. When to Use Data from Other Projects for Effort Estimation. In *ASE*. Antwerp, Belgium, 321–324.
[21] E. Kocaguneli, T. Menzies, A. Bener, and J. W. Keung. 2012. Exploiting the Essential Assumptions of Analogy-Based Effort Estimation. *IEEE TSE* 38, 2 (2012), 425–438.
[22] E. Kocaguneli, T. Menzies, and E. Mendes. 2015. Transfer Learning in Effort Estimation. *Empirical Software Engineering Journal* 20, 3 (2015), 813–843.
[23] M. Lefley and M. Shepperd. 2003. Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets. In *GECCO*, Vol. LNCS 2724. Chicago, 2477–2487.
[24] S. G. McDonell and M.J. Shepperd. 2007. Comparing Local and Global Software Effort Estimation Models – Reflections on a Systematic Review. In *ESEM*. Madrid, 401–409.
[25] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmerman. 2013. Local vs. Global Lessons for Defect Prediction and Effort Estimation. *IEEE TSE* 39, 6 (2013), 822–834.
[26] T. Menzies, A. Butcher, D. Cok, A. Marcus, L. Layman, F. Shull, B. Turhan, and T. Zimmermann. 2013. Local vs. Global Lessons for Defect Prediction and Effort Estimation. *IEEE TSE* 39, 6 (2013), 822–834.
[27] T. Menzies, R. Krishna, and D. Pryor. 2015. The Promise Repository of Empirical Software Engineering Data. (2015). http://openscience.us/repo
[28] T. Menzies, R. Krishna, and D. Pryor. 2017. The SEACRAFT Repository of Empirical Software Engineering Data. (2017). tiny.cc/seacraft
[29] T. Menzies and M. Shepperd. 2012. Special Issue on Repeatable Results in Software Engineering Prediction. *Empirical Software Engineering Journal* 17 (2012), 1–17.
[30] L. Minku. 2016. On the Terms Within- and Cross-Company in Software Effort Estimation. In *PROMISE*. Ciudad Real, Spain, 4.1–4.4.
[31] L.L. Minku. 2017. *An Investigation of Dycom's Sensitivity to Different Cross-Company Splits*. Technical Report. Department of Informatics, University of Leicester. http://www.cs.le.ac.uk/people/llm11/publications/dycom-cc-splits.pdf
[32] L. Minku, F. Sarro, E. Mendes, and F. Ferrucci. 2015. How to Make Best Use of Cross-Company Data for Web Effort Estimation?. In *ESEM*. Bergamo, Italy.
[33] L.L. Minku and X. Yao. 2012. Can Cross-company Data Improve Performance in Software Effort Estimation?. In *PROMISE*. Lund, Sweden, 69–78.
[34] L.L. Minku and X. Yao. 2013. Ensembles and Locality: Insight on Improving Software Effort Estimation. *IST* 55, 8 (2013), 1512–1528.
[35] L. Minku and X. Yao. 2014. How to Make Best Use of Cross-company Data in Software Effort Estimation?. In *ICSE*. Hyderabad, 446–456.
[36] L. Minku and X. Yao. 2017. Which Models of the Past Are Relevant to the Present? A software effort estimation approach to exploiting useful past models. *Automated Software Engineering Journal* 24, 3 (2017), 499–542.
[37] L. Rokach and O. Maimon. 2005. *Clustering Methods*. Springer, 321–352.
[38] P. Sentas, L. Angelis, I. Stamelos, and G. Bleris. 2005. Software Productivity and Effort Prediction with Ordinal Regression. *IST* 47 (2005), 17–29.
[39] M. Shepperd and S. McDonell. 2012. Evaluating Prediction Systems in Software Project Estimation. *IST* 54, 8 (2012), 820–827.
[40] M. Shepperd and C. Schofield. 1997. Estimating Software Project Effort Using Analogies. *IEEE TSE* 23, 12 (1997), 736–743.
[41] L. Song, L.L. Minku, and X. Yao. 2013. The Impact of Parameter Tuning on Software Effort Estimation Using Learning Machines. In *PROMISE*. Baltimore, USA, Article No. 9, 10p., doi: 10.1145/2499393.2499394.
[42] B. Turhan and E. Mendes. 2014. A Comparison of Cross- versus Single- Company Effort Prediction Models for Web Projects. In *SEAA*. Verona, Italy, 285–292.
[43] A. Vargha and H. D. Delaney. 2000. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. In *Journal of Educational and Behavioral Statistics*, Vol. 25. 101–132.
[44] I. Wieczorek and M. Ruhe. 2002. How Valuable Is Company-Specific Data Compared to Multi-Company Data for Software Cost Estimation?. In *METRICS*. Ottawa, 237–246.