Noname manuscript No.
(will be inserted by the editor)

Towards Improving Statistical Modeling of Software Engineering Data: Think Locally, Act Globally!

Nicolas Bettenburg · Meiyappan Nagappan · Ahmed E. Hassan

Received: date / Accepted: date

Abstract Much research in software engineering (SE) is focused on modeling data collected from software repositories. Insights gained over the last decade suggests that such datasets contain a high amount of variability in the data. Such variability has a detrimental effect on model quality, as suggested by recent research. In this paper, we propose to split the data into smaller homogeneous subsets and learn sets of individual statistical models, one for each subset, as a way around the high variability in such data. Our case study on a variety of SE datasets demonstrates that such local models can significantly outperform traditional models with respect to model fit and predictive performance. However, we find that analysts need to be aware of potential pitfalls when building local models: firstly, the choice of clustering algorithm and its parameters can have a substantial impact on model quality. Secondly, the data being modeled needs to have enough variability to take full advantage of local modeling. For example, our case study on social data shows no advantage of local over global modeling, as clustering fails to derive appropriate subsets. Lastly, the interpretation of local models can become very complex when there is a large number of variables or data subsets. Overall, we find that a hybrid approach between local and traditional global modeling, such as Multivariate Adaptive Regression Splines (MARS) combines the best of both worlds. MARS models are non-parametric and thus do not require prior calibration of parameters, are easily interpretable by analysts and outperform local, as well as traditional models out of the box in four out of five datasets in our case study.

Keywords Software metrics · Statistical modeling · Clustering

Nicolas Bettenburg, Meiyappan Nagappan, Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University, School of Computing
Kingston, Ontario K1N 3L6 Canada
{nicbet, mei, ahmed}@cs.queensu.ca

1 Introduction

Data modeling is an important technique that is frequently employed in empirical software engineering research for studying observations from software projects. In the past, the foremost application of modeling was to create prediction models [8]. These models describe how a set of product and process measures can be used to predict another measure (such as the amount of defects in a module, or the development effort needed for a software component) in a quantitative manner. Modeling has also gained use as a means for understanding. Instead of using models solely as a black box, which is fed a set of measures and outputs a prediction, closer inspection of a model can provide insights into the underlying processes [6, 8, 23, 24].

Regardless of the final goal (understanding or prediction), models are derived from data about the artifact under study. This data is commonly extracted by mining software repositories. Recent research has shown that several software engineering datasets contain a great amount of variability [16,30]. However, we have been using software engineering datasets for model building as is, without further considering such variability. *In this work, we call models that are learned on a complete dataset, a "global model", as they take the complete dataset without taking into account all its inherent variance.*

A recent study by Menzies et al. demonstrates, that there lies potential benefit in partitioning software engineering datasets into smaller subsets of data with similar properties [20]. Their study showed that training a specialized machine learner using subsets of data leads to better fits.

In this study, we investigate whether the findings of Menzies et al. [20] hold true for commonly used statistical modeling techniques. *In particular, we cluster the complete dataset into smaller subsets with similar data properties, and build statistical models by training them on these subsets. We call these models "local models".* Local models might thus have an advantage over global models (i.e., the traditional way of building models).

We find that local models built through data clustering outperform global models with respect to both goodness of fit, and prediction performance. However, we observe that the advantage of such local models is dependent on the choice of clustering algorithm, tuning of parameters, and the properties of the underlying software engineering data itself. Our study also demonstrates that a hybrid approach between global and local models, *global models with local considerations,* outperforms both global, and local models, without the need of tuning or calibration. In addition, global models with local considerations simplify the interpretation of the predictive model, when compared to their global and local counterparts.

1.1 Research Questions and Key Findings

Overall, our empirical study contains six research questions, which we have grouped into three categories as follows:

1. How can we build and evaluate local statistical models?
We present the use of two categories of evaluation criteria for model evaluation. First, *goodness of fit* criteria describe how well the model represents the data from which it was learned from. Second, *model performance* criteria describe how well the model is able to predict new (unseen) data.

> *RQ1 (Model Fit): Is there an advantage of using local models over global models, with respect to goodness of fit?*

An increased fit will be beneficial for empirical researchers and practitioners, when they are using models for understanding. A more granular model that better describes the data at hand might lead to more insights into the relationships between the metrics in the dataset and the dependent variable. *We find that a local approach produces considerably better fits for statistical models.*

> *RQ2 (Prediction Performance): Is there an advantage of using local models over global models, with respect to prediction performance?*

Models with better prediction performance are of great value to practitioners since they allow them to take better informed decisions and actions. *We find that the improved local fits do not lead to over-fitted models. The local approach considerably increases the predictive performance of statistical models, leading to up to three times lower prediction errors.*

2. Are results dependent on the clustering algorithm or the modeled data?
Any benefit that we might observe with respect to evaluation criteria (RQ1 and RQ2) may be due to the inherent design of local models: a saying among data scientists states that one sure way to build a better model is to use less data [14]. We also investigate how the choice of clustering approach and their individual parameters impacts the fit and prediction performance of local models.

> *RQ3 (Role of Data Partitioning in Local Modeling): What is the role of clustering of datasets when building local models, as compared to randomly partitioning the data into smaller chunks?*

Increased performance of local models may or may not be due to the partitioning of datasets into smaller chunks, which in turn lead to better model

fits. *We observe that in all of our datasets, local models built through random clustering perform considerably worse with respect to prediction performance than the corresponding global models, even though they show improved goodness of fit. Thus, the superior performance of local models over global models stems from a smart partitioning of the datasets, i.e., partitioning and grouping of the data with the goal of finding homogeneous regions in the data.*

> *RQ4 (Impact of Clustering Approach): What is the impact of choice of clustering algorithm and parameters on the performance of the resulting local models?*

Machine learning literature has provided a plethora of clustering algorithms over the past two decades. However, it is not known how the choice of clustering algorithms and their parameters impact local modeling. *We find that certain clustering algorithms yield better performing models than others, provided that the analyst first carries out a careful analysis of parameters for that particular algorithm. In environments where resources are limited or a preliminary calibration of parameters is unfeasible, non-parametric clustering offers a competitive performance for the resulting local models.*

> *RQ5 (Impact of Data): For the same clustering method, modeling techniques, and predicted outcome, how do different software engineering metrics respond to local modeling?*

We conduct experiments with one additional dataset that we used in the past to predict post-release failures in software based on code metrics and social metrics [6]. We aim to investigate to what extent the nature of metrics used to build models for the same outcome lends itself to the approach of local modeling. *We find that different kinds of metrics, social compared to product/process metrics, are not equally suitable for local modeling. Our findings provide evidence that local modeling of software engineering data relies on certain distributional properties of the collected metrics, i.e., enough variance in the data to discern clear cluster boundaries.*

3. What are the practical considerations of local modeling?

While we show that local modeling can lead to considerably improved model fits and prediction performance, the interpretation of local models becomes more complicated.

> *RQ6 (Local Models in Practice): What are the considerations in the use of local models over global models for practitioners?*

An increase in choices may not necessarily be beneficial to practitioners. In particular, conflicting recommendations could be potentially misleading and

prevent models from carrying out the task that they are designed for: to aid practitioners in decision making. *We find that while local models successfully separate relationships in each local region of the data, interpretation of the local can quickly become complex. Global models with local considerations are among the best performing models across our datasets and combine the best of both worlds (simplicity of interpretation, while still capturing local relationships in the data.)*

1.2 Organization of the Paper

This paper extends our work [7] initially published at the 2012 Working Conference on Mining Software Repositories (MSR'12)[1]. In particular, our extensions centre around the additional three research questions (RQ3, RQ4, and RQ5) that expand our understanding of how and why local modeling works. We discuss the impact of different clustering approaches on local modeling, and demonstrate that local modeling has a number of requirements on the data, as well as the analyst, which determine its success and performance.

The rest of this paper is organized as follows: Section II discusses the background of our study, including related work within empirical software engineering. Section III, illustrates the design of our case study, including data collection and pre-processing, as well as a discussion of how we derived the global and local models used in our case study. Section IV, discusses the findings of our case studies along our six research questions. Section V presents our conclusions, as well as recommendations for future research efforts. Furthermore, Appendix A presents a summary of the metrics used in all case study datasets. Appendix B presents the full results of our RQ4 and RQ5 experiments.

To allow for easier replication of our work and encourage future research, we provide all datasets, scripts and analysis steps used to generate the results presented in this paper in a replication package at `http://sailhome.cs.queens.ca/replication/local-vs-global-emse/`.

2 Background and Related Work

Modeling Techniques
The majority of modeling techniques used in empirical software engineering are borrowed from two research areas: Statistics and Machine Learning. For instance, Andreou et al. use a machine learning technique called decision trees, to model and predict the costs involved in software development [3]. Elish et al. use a machine learning technique called Support Vector Machines (SVM),

[1] http://www.msrconf.org

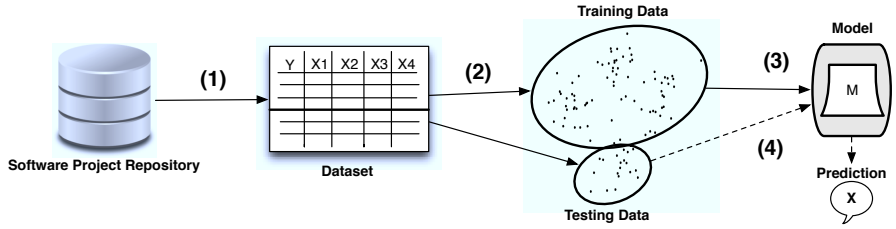


Fig. 1: Example of Statistical modeling of Software Engineering Data.

to predict defect-prone modules at NASA [9]. They find that machine learning techniques exhibit similar performance as statistical approaches for defect prediction.

One of the most popular statistical modeling techniques is regression modeling. Figure 1 presents a general overview of the modeling process. First, the data scientist extracts a set of metrics from the software project repository at hand (Step 1 in Figure 1). These metrics commonly include the objective that is to be modeled (called the predicted variable, or in statistical terms the dependent variable), as well as a set of metrics that are believed to stand in relationship with the objective (these metrics are called the prediction variables, or in statistical terms, independent variables). For instance, Nagappan et al. use regression models to predict the defect density of modules (their dependent variable) based on code churn (their independent variable), a metric of changeability [26].

To prepare the data for training and evaluating the performance of the model, the data is commonly split into a training set (often 90% of the data) and a testing set (10% of the data), as illustrated in Figure 1, Step 2. The model is learned from the training data and the testing data is treated as the "unseen" data, and in turn used to carry out predictions. Since for each point in the testing data, the real outcome is already known, analysts can compare the predicted outcome with the true outcome to obtain a notion of how well the model predicted.

Next, the data scientist derives a statistical model in the form of a linear or non-linear equation that describes a plane through the data, which minimizes the distances between the plane and all data points (Step 3 in Figure 1). There exist off-the-shelf toolkits, such as the `R` statistical environment, which provide readily available algorithms for the derivation of these models from input data. The analyst can then investigate the model and the variables used (Step 3 in Figure 1). For instance, Zimmermann et al. built and analyzed models to study which metrics best describe software defects in the Eclipse project [40].

However, the model can also be used to perform predictions of unseen data (Step 4 in Figure 1), i.e., data from a different release or different project [36, 39], or as a form of apparent validation, from part of the original data (i.e., the testing set), as illustrated in Figure 1.

Modeling Goals

Models have been widely used in the area of empirical software engineering research. Analysts build models mainly for two purposes: prediction and understanding. For instance, Nagappan et al. computed source code complexity metrics for individual modules of 5 commercial software projects and used combinations of these complexity metrics to predict the future failure probability of modules [27]. They were able to find a suitable prediction model for each of the projects, but could not find a common set of metrics that worked across all projects. Similarly, Mockus et al. studied the use of linear regression models to predict the risk of source code changes [23].

Prediction models can also be fine-tuned with respect to additional constraints. For example, in recent work, Kamei et al. [16] present the use of product and process metrics with the goal of effort-aware modeling of software defects for the Eclipse project.

At the same time, models can be used to gain an understanding of why certain outcomes occur. For instance, Mockus et al. use statistical models to understand the relationships between observations on the software process and how customers perceive the quality of the product [25]. A similar approach was used in a study by Mockus et al. to investigate how changes impact the overall software development effort [24].

Modeling Approaches

Researchers in empirical software engineering have typically built global models to predict development effort and software defects with high accuracy [23], as well as for understanding software engineering phenomena [6, 24]. However, recent research suggests that empirical software engineering should focus more on context specific principles. In particular Menzies et al. advise that future work in empirical software engineering should explore lessons learned from individual subsets in contrast to lessons learned across the whole data [20].

A similar issue was also recently raised in the works of Kamei et al. and Nguyen et al. [16, 28], who identify potential bias in performance metrics of defect prediction models, depending on the aggregation granularity of the data used for building the prediction model. Similarly, Posnett et al. [30] confirmed the existence of granularity bias and demonstrate that decomposition of datasets at the wrong granularity level can lead to misleading and fallacious results.

Research Context

The research most closely related to this paper, is the work by Menzies et al., who demonstrate that for their WHICH treatment machine learner, a partitioning of data along the axis of highest data variability using their custom built WHERE clustering method, led to a better fit of WHICH to the underlying datasets [20]. Based on their findings, the authors recommended further investigation of local vs. global approaches in empirical software engineering research. We follow this call with the study at hand.

We identify key differences between the work by Menzies et al. [20] and our study below:

First, we follow the idea of data partitioning into local regions within the context of building statistical linear regression models. Our study investigates, whether off-the-shelf technology that is readily available for practitioners and researchers can experience the same benefits of data partitioning as specialized methods, such as the WHERE clustering method.

Second, in addition to evaluating goodness of fit (how well a model captures the data from which it was trained on), we also perform predictions in an experimental setup that closely resembles how practitioners would use the approach. Furthermore, we evaluate our results through a multiple run cross-validation setup and investigate the effect of data partitioning on prediction performance along multiple performance criteria.

Relative to prior work, our paper makes the following contributions:

First, we build two types of models: global and local, and perform comparison between both types. In addition, our study introduces a third approach: global models with local considerations, which can be considered as a hybrid between global and local models. In particular we use the well-studied implementation of **M**ultivariate **A**daptive **R**egression **S**plines [13] (MARS). MARS models have found extensive and successful use in modeling problems in outside of the empirical software engineering field, such as economics [29], and genetics [38].

Second, we investigate the impact of different clustering approaches for learning local models on model fit and performance. Furthermore, we study how different clustering algorithms, and parameters affect local model performance. In contrast to previous research in the area of building prediction models using less data [17], we work under the premise of using a full data set such that intelligent subdivision of that data through clustering creates meaningful subsets.

Third, our study provides evidence that not all datasets can equally benefit from local modeling. In particular we compare and discuss post-release defect modeling across open-source projects under the umbrella of the Eclipse platform through local models based on code complexity metrics, as well as social metrics. We find that the lack of variance in the social metrics leads to weak clustering performance which in turn leads to local models having no performance advantages over their global model counterparts.

Fourth, we discuss practical considerations of local modeling and interpretation of models. In particular, our work makes a strong case for the benefits of using a hybrid approach, global models with local considerations, which combines the advantages of local and global modeling: easy interpretation of global models, and capturing of datasets localities, like local models.

3 Case Study Design

In this section we discuss the design of our case study on six different datasets. We begin with a general discussion of the data, followed by a detailed description of our experimental setup, including the used experimental design and modeling approaches.

3.1 Data Collection and Preparation

Four of the datasets used in our case study, Xalan 2.6, Lucene 2.4, CHINA and NasaCoc, have been obtained from the PROMISE [2] repository, and have been reported to be as diverse of datasets as can be found in this repository [20]. Since we have not collected the data ourselves, we cannot guarantee perfect data quality. However, the PROMISE repository is a source of curated data sets that are widely used in a plethora of empirical software engineering research [21], and as such the obtained data is as close to a benchmark as we could find in this research domain. Furthermore, the same datasets have been used in the previous study by Menzies et al. when investigating the benefit of data partitioning into local regions for their machine learning approach. In particular, we obtained two datasets concerned with defect prediction (Xalan 2.6 and Lucene 2.4), as well as two datasets concerned with effort prediction (CHINA, and NasaCoc). For a complete set of descriptions of the variables in each of the four PROMISE datasets, we refer the interested reader to [20].

The Eclipse datasets containing code metrics and social metrics was reused from our earlier work [6]. The Eclipse data was collected for a period of 6 months surrounding the release of version 3.0. It contains observations at file-level granularity and the model objective is post-release defects that were reported by users and developers for a period of 6 months following the 3.0 release. A summary of the obtained data is shown in Table 1. Additionally, Appendix A provides a complete list of metrics contained in each dataset.

All datasets are complete, i.e., we have checked that they do not suffer from missing data. However, a threat to the validity of the study at hand is the presence of noise and outliers. We have not removed noise or performed any other kind of outlier analysis and removal. Thus, all the statistical modeling presented in this paper fits the whole data, including possible noise and outliers, and clustering techniques will also be impacted by these irregularities.

Correlation Analysis

The goal of this work is to study local and global modeling under the framework of regression modeling. However, multi-collinearity between predictor variables is one of the major factors impacting regression model building and model quality. Thus, to prepare each of the datasets for use in our statistical modeling experiments, we first carry out a correlation analysis. For each

[2] `http://promisedata.org`

Table 1: Summary of datasets used in our case study.

Dataset	Modeling Of	Metrics	Datapoints
Xalan 2.6	Defects	23	885
Lucene 2.4	Defects	23	340
CHINA	Development Effort	19	499
NasaCoc	Development Effort	27	154
Eclipse Code Metrics	Defects	37	8,696
Eclipse Social Metrics	Defects	21	8,696

dataset, we start from a complete dataset, and carry out an analysis to detect potential multi-collinearity between the metrics (columns) in the datasets. Previous research [5,35] has demonstrated that many process and source code metrics are correlated, both with each other, and with lines of code (LOC). Ignoring such correlations would lead to increased errors in the estimates of model performances, and to increased standard errors of predictions [14]. For all datasets, we observed moderate to high correlation between two or more predictor variables.

VIF Analysis

Within the same vein of prior research, we handle multi-collinearity through analysis of **V**ariance **I**nflation **F**actors (VIF) for each dataset [10]. The goal of VIF analysis is not to obtain a best set of predictors, but to reduce correlations between predictor variables before attempting any statistical modeling. We iteratively compute VIF measures for all variables and then remove the variable with the highest VIF value, until no variable has a VIF measure higher than 5 [10]. The VIF analysis leaves us with a reduced set of variables in each dataset. For instance, the VIF analysis removed the variables `CBO`, `wmc`, `rfc`, and `amc` from the Xalan 2.6 dataset, and the variables `CBO`, `wmc`, `rfc`, and `loc` from the Lucene 2.4 dataset. Furthermore, the VIF analysis removed the variables `NPDU_UFP`, `AFP`, `PDF_AFP`, `NPDR_AFP`, and `Added` in the CHINA dataset, and the variables `defects`, and `tool` from the NasaCoc dataset. A complete list of variables left after VIF analysis is provided in Appendix A.

3.2 Model Building Approach

An overview of our approach to build the models used in our case study is illustrated in Figure 2. For all three types of models (global, local, global with local considerations), we start the modeling process by first splitting the dataset into training and testing data. These splits contain 90% and 10% of the dataset respectively. The model is learned on the training data, while predictions using the model are carried out on and compared to the testing data. We detail below how we derive each type of model from the training data.

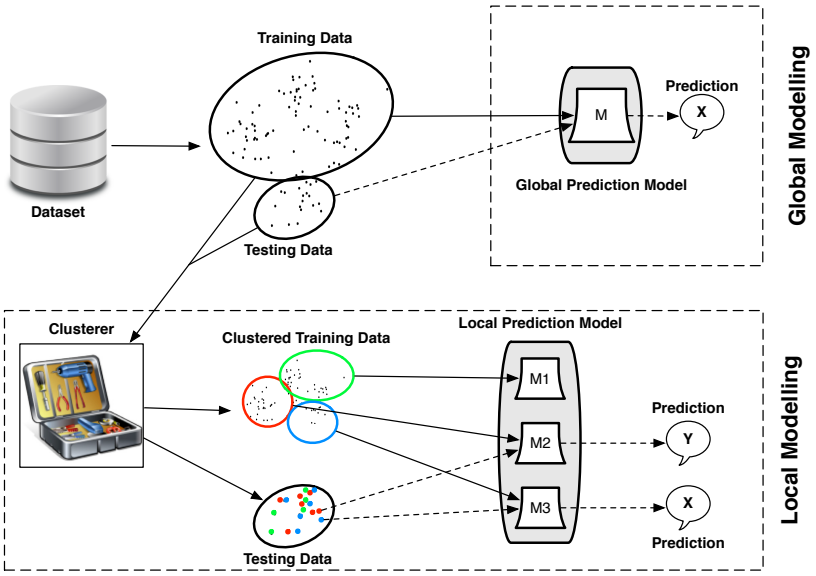


Fig. 2: Overview of our approach for building global and local regression models. This process is repeated 10 times.

Global Models

We use statistical modeling, in particular linear regression, to build global models. In general, linear regression models attempt to find the best fit of a multi-dimensional line through the data, and they are of the form $Y = \epsilon_0 + \alpha_1 * X_1 + \cdots + \alpha_n * X_n$, with Y the dependent variable, ϵ_0 called the *intercept* of the model, α_i the i-th regression coefficient, and X_i the i-th independent variable. In particular, Y denotes the measure that we want to predict (number of bugs for each file in the case of the defect prediction datasets, total development effort for CHINA, and the number of months required to develop the software component for NasaCoc), and X_i denotes the metrics on which we base our predictions.

Local Models

To build local models, we use linear regression, similarly to building global models. However, the main difference is that the data is first partitioned into regions with similar local properties using a clustering approach, before the data is split into training and testing sets. For each data point, we record a unique identifier that corresponds to the cluster that data point is associated with.

The final local model is obtained by creating individual regression models of the form $Y = \epsilon_0 + \alpha_1 * X_1 + \cdots + \alpha_n * X_n$ for each local region of the training data (clusters). To carry out predictions on the testing data using local models, we use the previously recorded cluster association of each testing data point.

For each entry in the testing data, we use the local model that has been fitted to that particular cluster, and carry out the individual predictions.

For the experiments discussed in RQ1 and RQ2, we use a state-of-the-art distribution-based non-parametric clustering technique called MCLUST [12]. This technique automatically derives all necessary parameters within the technique itself, and partitions a dataset into an approximately optimal number of subsets based on the variability in the data [11]. The choice of MCLUST was based on the clustering technique being a non-parametric technique, i.e., there is no prior need for selection and calibrations of parameters.

However, non-parametric clustering might not unlock the full potential of local modeling. Hence, for the experiments discussed in RQ3 and RQ4, which aim to explore how different clustering techniques impact local modeling, we add two additional clustering techniques, called k-means [15] and hierarchical clustering [18]. Both are parametric clustering techniques, which allow the analyst to further refine the clusters through parameters. The k-means clustering technique partitions the dataset into k clusters (the analyst provides the parameter k), such that each point of the data is associated with the cluster to which the distances between the centre of that cluster and the observation is minimized. K-Means is a heuristic clustering algorithm and finds a locally optimal solution. For this reason we repeat the experiments that involve k-means clustering multiple times to account for randomization bias. For k-means clustering, we use the `R` language implementation, named `kmeans`, which is provided in the `stats` package.

For hierarchical clustering, we use an agglomerative (bottom up) algorithm, where each observation in the dataset starts in its own cluster and at each iterative step of execution, pairs of clusters get grouped together based on the distance of their centres in a hierarchical manner. The output of the hierarchical clustering algorithm is a tree (similar to a dendrogram). The tree can then be cut at any level in the hierarchy and the analysts commonly provides the parameter h, which denotes that the tree should be cut at a particular level of the hierarchy, such that h separate clusters are obtained. For hierarchical clustering we use the `R` language implementations of `hclust` to build the tree, and the `cutree` method to cut the tree; both are provided in the `stats` package.

When building local models through clustering, we cannot guarantee that every predictor variable in each cluster can be used in a local regression model. That is, we run into the problems that two predictors might exhibit perfect correlation, which in turn leads to singularities when building the statistical model, i.e., the correlation coefficients for that predictor cannot be computed and is undefined . To solve this problem, we use a predictor selection approach based on the Bayesian Information Criterion (BIC) [34], called Bayesian Model Averaging (BMA) [31]. In particular, we apply BMA to each data cluster to determine the set of predictor variables that we used when building the local model for that cluster. The result of BMA is a set of variables, for which a metric of BIC on the data has an optimum value. We want to note that BMA is not used to select a "best" set of prediction variables (Menzies et al. have demonstrated that there are different sets of best predictors in different

datasets [22]), but to select a set of valid variables for each cluster, such that modeling would not run into singularities.

Global Models with Local Considerations

We use Multivariate Adaptive Regression Splines [13], or MARS, models, as an example of a global model that takes local considerations of the dataset into account. MARS models have become increasingly popular in medical, and social sciences, as well as in economical sciences, where they are used with great success [4, 29, 38].

A MARS model has the form $Y = \epsilon_o + c_1 * H(X_1) + \cdots + c_i * H(X_n)$, where Y is the dependent variable (that is to be predicted), c_i is the i-th hinge coefficient, and $H(X_i)$ the i-th "hinge function". Hinge functions are an integral part of MARS models, as they allow to describe non-linear relationships in the data. In particular, they cluster the data into disjoint regions that can be then described separately (our notion of local considerations). In general, hinge functions used in MARS models take on the form of either $H(X_i) = max(c, X_i - c)$, or $H(X_i) = max(c, c - X_i)$, with c being some constant real value, and X_i an independent (predictor) variable.

A MARS model is built in two separate phases. In the forward phase, MARS starts with a model which consists of just the intercept term (which is the mean of the independent variables). It then repeatedly adds hinge functions in pairs to the model. At each step it finds the pair of functions that gives the maximum reduction in residual error. This process of adding terms continues until the change in residual error is too small to continue or until a maximum number of terms is reached. In our case study, the maximum number of terms is automatically determined by the implementation, and is based on the amount of independent variables we give as input. For MARS models, we use all independent variables in a dataset after VIF analysis (similar to the other modeling approaches).

The first phase often builds a model that suffers from overfitting. As a result, the second phase, called the backward phase, prunes the model, to increase the generalization ability of the model. The backward phase removes individual terms, deleting the least effective term at each step until it finds the best sub-model. Model subsets are compared using a performance criterion specific to MARS models, and the best model is selected and returned as the final model.

MARS models have the advantage that a model pruning phase is already built-in by design, so we do not need to carry out a model pruning step similar to BIC, as we do with global and local models. Second, the hinge functions in MARS models do already model disjoint regions of the dataset separately, such that there is no need for prior clustering of the dataset with a clustering algorithm. Both advantages make this type of modeling approach considerably simpler to use in practice.

3.3 Countering Randomization Bias

For better generalizability of our results, and to counter random observation bias, the experiments described in RQ1 and RQ2 are repeated 10 times on stratified random sub-samples of the data into training (90% of the data) and testing (10% of the data) sets. The stratification is carried out on the metric that is being modeled. For example, for dataset *Xalan 2.6*, this would correspond to the *defects* metric. We then learn the model from the training data, evaluate the fit of the model to the training data, and evaluate the prediction performance on the testing data. We repeat that process ten times and evaluate all our findings based on the average over these 10 repetitions. This practice of evaluating results based on multiple runs when randomness is involved, is a common approach in Machine Learning, and is often referred to as "10-times cross validation" [37]. As an important factor to the comparability of local and global modeling, we want to note that regardless of the approach (local, or global), we always obtain a random sample of 90% of the overall dataset for training and 10% of the data for testing, i.e., the size of training and testing data is always the same.

For the experiments described in RQ3, RQ4 and RQ5, we have more control points that allow for random bias (e.g., parameters passed to the parametric clustering techniques). To counter random effects, we repeat our experiments such that we obtain a confidence interval of less than 3% with a confidence level of 99%, i.e., we can be 99% sure that the results are within a range of plus/minus 3% of the reported values. More details on the repeated execution of these experiments is discussed under the respective research questions.

In all cases where we performed multiple statistical significance tests, we used the Bonferroni correction [33] to avoid the spurious discovery of significant results due to multiple repeated tests.

4 Results

In this section, we present the result of our case study. This presentation is carried out in individual subsections that follow our six research questions. For each part, we first describe our evaluation approach, and discuss and interpret our findings.

RQ1. Is there an advantage of using local models over global models, with respect to goodness of fit?

Our aim in this question is to evaluate the goodness of fit of the models produced by each of the three modeling approaches. In general, a goodness of fit measure describes how well the model describes the observations in the data from which it was learned. The goodness of fit measure is of specific importance for software engineering research that wants to use models as a means of understanding underlying processes.

For example, if one were to investigate the relationships between post release defects and source code complexity, a regression model with post release defects as the dependent variable, and complexity measures as independent variables could be used. However, if the corresponding statistical model showed a low goodness of fit measure, insights derived from the investigations of the model are in the best case misleading (and in the worst case wrong).

Approach
We divide each dataset into training data and testing data, and learn a global model, a local model, and a global model with local considerations from the training data, as outlined in Section 3.2. A commonly used goodness of fit measure for linear regression models is the coefficient of determination, R^2. In general, R^2 measures the amount of variability in the data described by the linear regression model, or in other words, how close the fitted regression model is to the actual values in the dataset.

However, past research has demonstrated that R^2 should not be used to compare different regression models, as the R^2 measure is highly biased towards the number of independent variables in a model [14]. The more independent variables the regression model contains, the higher its R^2 measure will be, even if the independent variables do not describe the dependent variable in any meaningful way. Instead of using R^2, we use two different goodness of fit measures that have been proposed in literature as our evaluation criteria.

Evaluation Criteria
1. Akaike Information Criterion ***(AIC)*** To judge the goodness of fit between global models and local models, we use the Akaike information criterion [2] (AIC), which is a fit measure based on information entropy. One of the main advantages over the traditional R^2 measure for goodness of fit of a regression model, is the robustness of AIC against bias due to using more independent variables in a model. AIC is widely used to judge the relative goodness of fit between different regression models [31].

In general, a lower AIC measure corresponds to a better model fit to the data. Results corresponding to this measure are presented under the name `AIC` throughout this paper. We want to note, that for MARS models, the Akaike information criterion is not available as a relative measure of goodness of fit, so we cannot compare MARS models directly. Hence, we use the other criterion described below to compare across all three types of models.

2. Correlation between predicted and actual values on trained data ***(FitCor)*** In addition to the AIC measure for goodness of fit, we measure how well a model was able to learn from the training data. For this purpose, we feed the same training dataset into the model again to predict values, and finally measure the (Pearson) correlation between actual values and predicted values. This measure of correlation is of particular importance for prediction models.

For example, in the case of defect prediction, models are often not concerned with the absolute number of post-release defects, but rather in a rank-

Table 2: Case study results: goodness of fit for global models. For AIC, smaller is better, for Correlation (FitCor), higher is better. We find that local modeling provides better model fits over global modeling. Global models with local considerations (MARS) provide the best fits of all three modeling approaches. Best values are marked in bold.

	Global		Local		MARS
Dataset	**AIC**	**FitCor**	**AIC**	**FitCor**	**FitCor**
Xalan 2.6	2,190.30	0.33	**352.98**	0.52	**0.69**
Lucene 2.4	1,380.35	0.32	**462.43**	0.60	**0.83**
CHINA	8,696.17	0.83	**1,805.06**	**0.89**	**0.89**
NasaCoc	858.95	0.93	**158.37**	0.97	**0.99**
Eclipse 3.0	19,575.91	0.62	**5,701.08**	0.37	**0.67**

ing of source code entities from "most buggy" to "least buggy" [26]. Resources and testing effort are then allocated according to that ranking. Results corresponding to this measure are presented under the name `FitCor` throughout this paper.

For both goodness of fit criteria, we need to normalize values for local models across clusters. For instance, consider the following case. Imagine, MCLUST would cluster a dataset into six clusters, of which one cluster C_1 contained 90% of the data and the other five clusters C_2 to C_6 each contained 2% of the data.

Now, suppose a hypothetical goodness of fit measure for C_2 to C_6 of 0.9, and 0.05 for C_1. The median goodness of fit in this case would turn out to be 0.9, greatly underestimating the contribution of cluster C_1 which contains the majority of the data. To counter this bias, we normalize by the size of each cluster relative to the size of the complete (training) dataset.

Findings

Table 2 summarizes the results of our experiment. Overall, we observe that local models exhibit a better relative goodness of fit measure (AIC) than global models on corresponding datasets.

The same observation holds true for the analysis of fit correlation (FitCor). Our analysis of the correlation suggests, that MARS models produce very good fits to the underlying data, outperforming both, global models, as well as local models. This further strengthens our conjecture of the advantages of local over global modeling.

To test for the statistical significance of the differences between measured fit correlations, we performed a Fisher's Z-test with the null hypothesis H_0: *the means of the distributions of measured fit correlations over our 10 repetitions are the same.* We reject that hypotheses at $p < 0.01$ and found that all differences were statistically significant, except in one case the fit correlation of the local model and the MARS model for the CHINA dataset (in both cases the fit correlation means were 0.89).

Overall, the results of our case study confirm that in the context of regression models for modeling defect and effort data, local models lead to considerably better fits to the underlying data.

RQ2. Is there an advantage of using local models over global models, with respect to prediction performance?

In the second part of our evaluation, we aim to investigate the actual performance of the models when applied on unseen data, i.e., prediction models. Better performing models are of great value to practitioners since they allow them to take better informed decisions.

Approach

To evaluate the prediction performance of each of the three modeling approaches, we follow the same steps of model building described in Section 4-RQ1. We divide each dataset into training data and testing data, and learn a global model, a local model, and a global model with local considerations from the training data.

Next, we use the testing data as input to these models to obtain predictions. For global models and global models with local considerations, we can directly take each row in the testing data as an input to the linear function that describes the model. For local models, we know for each datapoint in the testing set the corresponding cluster that point was assigned to initially, and then use the corresponding local model from that cluster to carry out the prediction (ref. Section 3.1). We compare these predicted values to the actual values recorded in the testing data, and evaluate performance based on three different criteria. These criteria are discussed in detail below.

Evaluation Criteria

1. Absolute sum of prediction error ***(ErrorSum)***

The sum of all prediction errors $\sum abs(Y_{actual} - Y_{predicted})$ tells us how good the model performed overall in predicting the testing data. The closer the sum of prediction errors is to 0, the better the predictive performance of the model. This performance criterion is of importance to practitioners as it gives an indication of how good a model captures reality. Results corresponding to this measure are presented under the name `ErrorSum` throughout this paper.

2. Median prediction error ***(MedianError)***

This performance criterion tells us about the central tendency of prediction errors, i.e., how far off predictions were from the actual value across all predictions. The closer this measure is to 0, the closer predictions of the model are to the true value recorded in the dataset. In general, this criterion can be seen as the predictive accuracy of the model. Results corresponding to this

Table 3: Summary of experimental results on models' predictive performance. The best observations in each column are marked in bold font face. Stars denote that the best value is statistically significant from the others at $p < 0.01$. Local Models outperform Global models in three datasets. Global Models with local considerations provide best performance across all datasets.

Global Models			
Dataset	**ErrorSum**	**MedianError**	**RankCor**
Xalan 2.6	61.07	0.64	0.36
Lucene 2.4	49.72	1.15	0.71
CHINA	91,592.52	765.00	0.82
NasaCoc	48.75	3.26	0.95
Eclipse 3.0	**340.89**	**0.10**	**0.59***
Local Models			
Dataset	**ErrorSum**	**MedianError**	**RankCor**
Xalan 2.6	57.35	0.52	0.50
Lucene 2.4	55.15	1.15	0.67
CHINA	83,420.53	552.85	0.85
NasaCoc	41.49	2.14	0.95
Eclipse 3.0	389.77	0.18	0.57
Global Models with Local Considerations			
Dataset	**ErrorSum**	**MedianError**	**RankCor**
Xalan 2.6	**50.90***	**0.40***	**0.56***
Lucene 2.4	**43.61***	**0.94***	**0.72**
CHINA	**25,106.00***	**234.43***	**0.99***
NasaCoc	**26.95***	**1.63***	**0.97***
Eclipse 3.0	342.85	**0.10**	0.56

measure are presented under the name `MedianError` throughout this paper.

3. The correlation between predicted and actual values ***(RankCor)***
This performance criterion is of particular interest for defect prediction, as it measures the extent to which the model is able to discern a correct ranking of defective files (from most risky to least risky), which for example, in practical applications of defect prediction models is often used for resource allocation. Similarly to the *FitCor* metric, we use the Pearson correlation coefficient. Results corresponding to this measure are presented under the name `RankCor` throughout this paper.

Findings
The results of our prediction experiments are summarized in Table 3. Overall, we observe from Table 3 that the local models outperform the global models in three out of five datasets, Lucene 2.4 and Eclipse 3.0 being the exceptions. However, the global model with local considerations outperforms both the global model and the local model in all five datasets.

Figure 3 illustrates a comparison of the distributions of prediction errors across all 10 runs on the example of two effort prediction (China, NasaCoc) and two defect prediction (Xalan, Lucene) datasets. We observe similar results for the Eclipse 3.0 dataset. Overall, we find that the local model outperforms

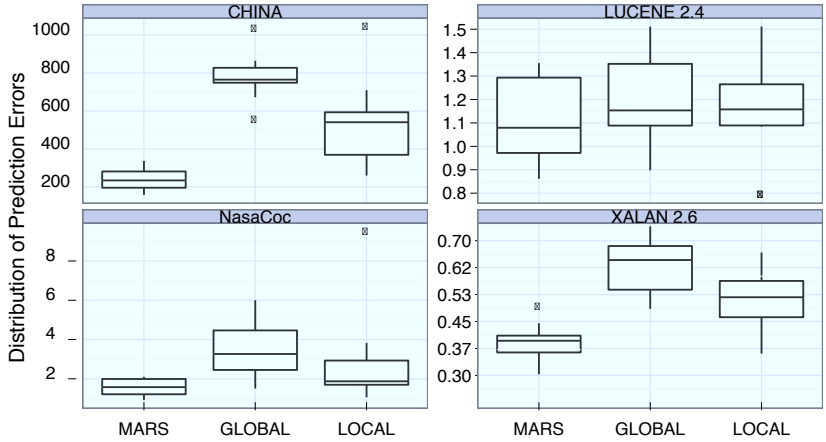


Fig. 3: Distributions of predictions errors for each dataset and modeling approach. Local modeling provides significant improvements of global modeling in three cases and comparable performance in one case (Lucene dataset). Furthermore, global models with local considerations (MARS) shows the lowest maximal prediction error.

the global model in three out of five cases, and in one case (Lucene 2.4) shows comparable performance.

At the same time, the global model with local considerations (denoted as `MARS`) demonstrates a considerably lower median prediction error distribution, which in three out of four cases has a shorter quartile range than both other approaches, i.e., predicted values are closer to the true values recorded in the testing data, and additionally, the worst predictions are less far off from the true values than in the other two modeling approaches.

We use the Mann-Whitney-U test, a two-sided non-parametric statistical test to compare two distributions, to confirm that the differences in the distributions of prediction errors are statistically significant from each other (at $p \leq 0.01$). The only difference that was not deemed statistically significant is in the case GLOBAL vs. LOCAL in the Lucene 2.4 dataset.

Table 3 also shows that local models provide a better ranking than global models in two out five datasets, and as good a ranking in one instance, and slightly worse ranking in two instances. The global model with local considerations is able to provide the best ranking of predictions in four out of five instances, with the Eclipse 3.0 dataset being the only exception.

Our results demonstrate, that local modeling lead to significant improvements in prediction errors over global modeling. Global Models with local considerations demonstrated the best performance in terms of overall prediction error, as well as greatly improved worst case predictions.

RQ3. What is the role of clustering of datasets when building local models?

So far we have demonstrated that splitting larger software engineering datasets into smaller chunks of homogeneous data and learning separate models for each individual chunk leads to an increase in goodness of fit and prediction performance over traditional models that act on the whole dataset.

However, the observed increase in fit and performance of local models may or may not be due solely to the clustering of datasets into smaller chunks, which in turn leads to better model fits. For instance, Harell et al. [14] note that using smaller datasets is often directly connected to improved model fit.

In this question we demonstrate that the observed performance increase is not due to arbitrary splitting of the data with the goal to produce smaller subsets of the data to learn from, but due to chunking with the goal of finding and grouping those observations that have similar properties, by the use of a clustering algorithm.

Approach

To investigate the role clustering plays when building local models, we devised a particular experimental setup that centres around randomly assigning observations (rows) in our datasets to clusters. The reasoning here is to study, whether the improved performance of local models is due to the reduction of the number of data points to which models need to be fitted. In particular, our experiment follows the steps below.

1. We select a number, max_k, between 1 and 10. This number max_k denotes the number of clusters that we will subdivide a dataset into during this single run of the experiment. A choice of $max\_k = 1$ is equivalent to no clustering (i.e., building a global model).
2. For each observation i in our dataset, we randomly select a number k_i between 1 and max_k with equal probability, and record this number as an additional column in the dataset. Every k_i records a random assignment of a particular observation in the data to a cluster, for a single run of the experiment.
3. We randomly subdivide the data into a *training dataset*, which contains 90% of all observations in the data, and a *testing dataset*, which contains the remaining 10% of observations that are not part of the training dataset. For each single run of the experiment, we randomly select a different division into training and testing data.

4. We group all observations in the training dataset by the associated value $k_i \in [1..max\_k]$ to which they were assigned earlier. For each cluster ($j \in [1..k]$) we learn a linear regression model m_j and record the model in a mapping from j to m_j.
5. For each model m_j we calculate the goodness of fit metrics on the model through measures of *AIC* and *fit correlation*, in the same approach as presented during the discussion of RQ1 in Section 4.1.
6. For each observation in the testing dataset, based on the associated value k_i, we know to which cluster that observation was assigned, initially. We use the data of this observation to predict the outcome with the corresponding local model for that cluster m_j, where $j = k_i$. We then record the difference between the predicted outcome $\hat{Y}$ and the actual outcome Y as it is recorded in the dataset to find the *absolute prediction error*.
7. We calculate the *prediction performance criteria* similar to those presented in the discussion of RQ2 in Section 4.2. In particular, we record the absolute prediction sum across all made predictions, the median prediction error, and the correlation in ranking between predicted and actual values.

We repeat the above experiment (Steps 1-7) a total of n=10,000 times, which corresponds to 1,000 repetitions for each choice of $j \in [1..max_k]$, with $max_k = 10$. Since our experiment is carried out in a random fashion, the results of the experiment concerning goodness of fit and prediction performance may vary depending on the selection of k and the subdivision of data into training and testing sets.

To counter the effect of such randomness, it is not sufficient to perform a 10-fold cross validation as we did earlier. To get accurate experimental results, we estimate means and confidence intervals of the results through the use of the Wilcoxon signed rank test.

We have performed an initial calibration of the experiment with varying values for n to determine a choice of n that would yield a confidence interval smaller than 3% and found that for $n = 1,000$ repetitions of the experiment for each k fulfills this criterion.

We recorded the average size of the confidence interval as 1.6% at a 99% confidence level, i.e., for each result that we report in the following, 99% of values observed in random experiments lie in a range of plus/minus 1.6% of the reported result.

Evaluation Criteria

We use the same evaluation criteria presented in RQ1 for judging goodness of fit, and the same evaluation criteria presented in RQ2 for judging prediction performance of a model.

Findings

The results of this experiment are summarized in Tables 4 and 5. In the following we discuss our findings in two parts: the first part details our findings related to the goodness of fit of the models to the training data, and the second

Dataset	Metric	k=1	k=2	k=3	k=4	k=5
Lucene	AIC	1,254.54	622.90	332.27	307.14	**277.59**
	FitCor	0.44	0.45	0.47	0.50	**0.52**
	Error Sum	**46.84**	49.59	53.28	56.15	60.07
	MedianError	**1.04**	1.07	1.11	1.15	1.23
	RankCor	**0.42**	0.38	0.37	0.34	0.31
Xalan 2.6	AIC	2,003.14	1,004.61	531.34	498.81	**455.97**
	FitCor	0.40	0.41	0.42	0.42	**0.43**
	Error Sum	58.74	59.41	60.60	61.45	63.01
	MedianError	0.48	0.48	0.49	0.49	0.50
	RankCor	0.38	0.36	0.35	0.34	0.32
CHINA	AIC	8,492.95	4,222.85	2,207.91	2,072.71	**1,913.38**
	FitCor	**0.70**	0.67	0.66	0.66	0.65
	Error Sum	**121,096.20**	130,887.50	138,033.50	141,708.60	147,038.50
	MedianError	**893.30**	1,015.36	1,096.41	1,150.72	1,182.77
	RankCor	**0.69**	0.67	0.65	0.63	0.62
NASACOC	AIC	826.25	398.77	216.23	186.91	**160.27**
	FitCor	0.76	0.78	0.80	0.83	**0.85**
	Error Sum	**71.73**	77.58	87.57	118.31	272.22
	MedianError	**3.04**	3.16	3.31	3.69	4.22
	RankCor	**0.74**	0.70	0.67	0.60	0.52
Eclipse 3.0	AIC	19,198.89	9,458.64	4,900.63	4,595.93	**4,263.68**
	FitCor	0.29	0.29	0.29	0.29	0.29
	Error Sum	**333.44**	338.97	342.43	345.96	350.62
	MedianError	0.10	0.10	0.10	0.10	0.11
	RankCor	**0.29**	0.28	0.27	0.27	0.27

Table 4: Building local models through random cluster assignment. Increasing k (number of clusters) leads to better model fits, however, prediction performance becomes increasingly worse.

part details our findings related to the prediction performance of models.

1. Goodness of Fit
We observe that the AIC measure decreases when building local models through random clustering, for increasing choices of k. Again, a lower value of AIC denotes a better fit of the model. This holds true for all datasets. Furthermore, in all datasets, except CHINA, we observe an increase in fit correlation with increasing k.

Overall, we find that clustering of data into smaller chunks leads to a better measure of goodness of fit of models irrespective of the used clustering approach (random clustering). This result is not surprising, as the smaller chunks contain fewer data points through which the linear model needs to be fitted through, leading to less occurrences for outliers that are far off the regression line and thus we see a better fit to the underlying training data.

2. Prediction Performance
We observe in Tables 4 and 5 that in no single case does a local model ($k \geq 2$) built through random cluster assignment outperform the corresponding global model ($k = 1$).

Furthermore, for all datasets, we observe a decrease in prediction performance with regards to all three performance measures, with increasing number of clusters k. Figure 4 illustrates this property on the example of the NASACOC dataset for local models built through random clustering with increasing choices of k.

Discussion
Both findings together are striking: first, our results demonstrate that when

Dataset	Metric	k=6	k=7	k=8	k=9	k=10
Lucene	AIC	224.27	190.93	164.05	145.04	127.35
	FitCor	0.54	0.56	0.59	0.61	0.64
	ErrorSum	64.98	70.65	81.86	97.32	113.05
	MedianError	1.28	1.35	1.42	1.54	1.64
	RankCor	0.29	0.27	0.25	0.22	0.21
Xalan 2.6	AIC	370.61	313.19	269.72	237.03	211.33
	FitCor	0.44	0.45	0.46	0.47	0.48
	ErrorSum	64.46	65.63	67.65	69.74	71.05
	MedianError	0.50	0.50	0.51	0.52	0.52
	RankCor	0.31	0.30	0.29	0.28	0.27
CHINA	AIC	1,542.37	1,296.49	1,111.36	975.12	866.50
	FitCor	0.65	0.65	0.65	0.66	0.66
	ErrorSum	151,430.90	154,368.20	158,393.70	161,217.10	166,324.10
	MedianError	1,215.17	1,231.59	1,231.30	1,281.43	1,312.92
	RankCor	0.61	0.60	0.59	0.58	0.57
NASACOC	AIC	125.27	98.05	63.57	26.31	3.41
	FitCor	0.87	0.90	0.92	0.95	0.97
	ErrorSum	389.98	537.98	1,017.08	1,552.96	2,085.86
	MedianError	5.03	6.20	9.45	14.79	21.30
	RankCor	0.46	0.38	0.29	0.22	0.17
Eclipse 3.0	AIC	3,407.48	2,833.42	2,416.07	2,107.10	1,866.15
	FitCor	0.29	0.29	0.29	0.29	0.29
	ErrorSum	355.53	359.46	363.67	368.83	373.26
	MedianError	0.11	0.11	0.11	0.11	0.12
	RankCor	0.26	0.26	0.25	0.25	0.25

Table 5: Building local models through random cluster assignment (continued). Any local model built through random clustering performs worse than their global model counterpart.

building local models, apparent evaluation of model fit can be misleading. For any choice of k we found that the local model has a better fit than the global model counterpart. However, the prediction performance of the better fitting local model is considerably worse than that of the corresponding global model.

This result emphasizes the need for evaluation of models based on their performance of predicting unseen data (RQ2) – an evaluation based on model fit alone (RQ1) is not sufficient to fully evaluate model performance on the one hand, and to compare global versus local models on the other hand.

Second, our experiment demonstrates that "smart" clustering is an integral part to building local models. One key factor that lends local models performance advantages over global models, is the exploitation of homogeneity in the data. Our finding thus provides a strong support for the initial assumptions by Menzies et al. [20].

In particular, empirical software engineering datasets contain observations with similar properties. The advantage of building local models stems from grouping similar observations together and learning from these groups (which is what local models do), as opposed to learning from all data and trying to learn a compromise between all data points (i.e., which is what a global model does).

Our results show that the improved performance of local modeling over global modeling is rooted in taking advantage of the homogeneity in the data, and not in the reduction of the data points to which the models are fitted. Furthermore, we find that both, goodness of fit metrics, and prediction performance criteria together are paramount to fully evaluate and compare local modeling approaches.

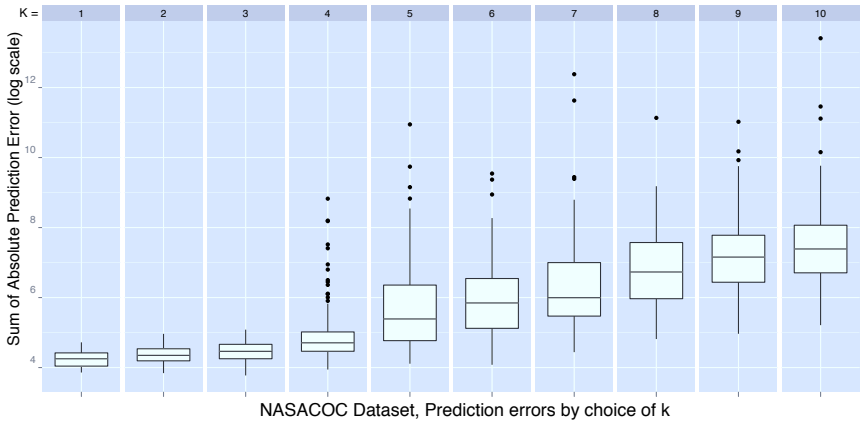


Fig. 4: Sum of absolute Prediction Errors when building local models through random clustering in the NASACOC dataset. We observe that the prediction error increases with k. This observation holds true for all datasets.

RQ4. What is the impact of choice of clustering algorithm on the performance of the resulting local models?

In the discussion of the previous research question, we demonstrated that clustering the data by the inherent homogeneity is an integral part of building local models. Machine learning literature has proposed a plethora of clustering algorithms, each of which with a slightly different angle on how to find and group similar observations in data.

Through these differences it is quite likely that different clustering approaches might be more or less suitable for building local models. In particular, the clustering algorithm we used for our previous experiments, MCLUST, is a non-parametric clustering approach, i.e., MCLUST automatically determines the number of clusters to which the dataset should be divided into. This can be a limiting factor since the choice of k happens irrespective of what the clusters are used for, in subsequent analysis and modeling steps.

With parametric clustering algorithms, the choice of number of clusters is given into the hands of the analyst. This has the advantage that the analyst can control the number of clusters based on historical data of the project, organizational knowledge and other empirical evidence. Hence, with this research question we want to find out if we can get better performance of local models when using parametric clustering approaches.

Approach

To study the impact of the choice of clustering algorithms on building local models, we add two additional well-known parametric clustering algorithms: Hierarchical Clustering [18] and K-Means [15] Clustering. Both clustering approaches depend on the choice of a single parameter k, which denotes the

desired number of clusters that the input data should be divided into. However, it is left to the analyst's expert knowledge to determine the appropriate k for the dataset under study.

In the domain of empirical software engineering, an appropriate choice of k requires expert knowledge. The number of homogeneous regions (clusters) in the data depends on the size of the dataset, the length of the data collection period (for example, the collection of the data might cover several periods with distinct team reorganizations and changes in the software process, where each would almost certainly shape the data being recorded) and other project and domain specific factors that are unique to each dataset.

Since we as researchers lack such expert knowledge for the selected case study subjects, we do not know a priori, which choice of k would be the most appropriate. As a result, we set up our experiment such that we carry out all analysis based on different choices of k between one and ten.

Overall we follow an experimental setup that is similar to the one presented in research question 3. In particular, our experiment consists of the following steps:

1. We select a number, max_k, between 1 and 10. This number max_k denotes the choice of the input parameter for the clustering algorithms for a single run of the experiment. A choice of $k = 1$ is equivalent to no clustering (i.e., building a global model).
2. We cluster the whole dataset through the use of a clustering algorithm. For each observation i in our dataset, we record the cluster to which this observations was assigned membership to by the clustering algorithm.
3. We randomly subdivide the data into a *training dataset*, which contains 90% of all observations in the data, and a *testing dataset*, which contains the remaining 10% of observations that are not part of the training dataset. For each single run of the experiment, we select a different division into training and testing data by random.
4. We group all observations in the training dataset by the cluster $j \in [1..max\_k]$, to which they were assigned earlier. For each j, we learn a linear regression model m_j and record the model in a mapping from j to m_j.
5. For each model m_j we calculate goodness of fit metrics on the model through measures of *AIC* and *fit correlation*, in the same way that we presented during the discussion of RQ1 in Section 4.1.
6. For each observation in the testing dataset, we know by the record of k_i, to which cluster that observation was assigned to initially. We use this information to predict the outcome of the observation with the corresponding local model for that cluster m_j, where $j = k_i$. We then record the difference between the predicted outcome $\hat{Y}$ and the actual outcome Y as it is recorded in the dataset to find the *absolute prediction error*.
7. We calculate *prediction performance criteria* similar to those presented in the discussion of RQ2 in Section 4.2. In particular, we record the absolute prediction sum across all predictions made, the median prediction error, and the correlation in ranking between predicted and actual values.

The same statistical analysis for countering randomization bias that we presented in the previous research question (RQ3) applies to this experimental setup, as well. Hence we repeat the above steps 10,000 times.

Results
The results of this experiment are summarized in Table 6. Again to increase readability, we have marked the "best values" in each column in bold font face, and omitted rows without best values. We provide the full results in Tables 9, 10, 11, which are part of Appendix B. Overall we make a number of separate important observations that we discuss in detail below.

1. Different datasets benefit from different choices of clustering approaches when building local models.
Table 6 presents a summary of the experiment with the optimum choice of k for different evaluation criteria. We find that no single clustering approach performs the best across all datasets. With respect to absolute sum of prediction errors, parametric clustering approaches outperform non-parametric clustering approaches three out of five times (Lucene 2.4, CHINA, Eclipse 3.0 datasets).

With respect to median prediction error, parametric clustering outperforms non-parametric clustering two out of five times (Lucene 2.4 and Eclipse 3.0). With respect to rank correlation between predicted and actual values, non-parametric clustering outperforms both parametric clustering approaches in all cases.

2. A careful choice of parameter k is paramount for building high-performance local models when using k-means clustering. Surprisingly however, local model performance when using hierarchical clustering is robust against any particular choice of k.

For the two parametric clustering algorithms we present the results for each value of k individually in Tables 6 B., and 6 C. As we can see from Table 6 B. and 6 C. in comparison to Table 6, a poor choice of k can have a significant detrimental effect on the model's prediction performance. For example, in the case of the Lucene dataset, the optimal value of k for K-means clustering is 3. At this value of $k = 3$, the sum of absolute prediction error (ErrSum) is 42.83, while at a value of $k = 10$ ErrSum is 131.79.

Overall, we observe that the performance of local models built through k-means clustering, model varies greatly with respect to the choice of parameter k. As opposed to k-means clustering, we observe that for hiearchical clustering, the absolute prediction error varies within a very small interval: the maximum variation is 1.9% in the case of the CHINA dataset. Similarly, the predicted rank correlation ("PredCor") and fit correlation ("FitCor") do not appear to be impacted by choice of k. However, we observe an impact by choice of k on the median prediction error (column "MedErr").

Table 6: Using different clustering techniques for building local models. Best values are marked in bold.

A. Results Summary

	K	AIC	FitCor	ErrSum	MedErr	PredCor
Lucene						
K-Means	3	1,172.89	0.39	**42.83**	**0.91**	0.36
MCLUST	na	**462.43**	0.60	55.15	1.15	0.67
MARS	na	na	**0.83**	43.61	0.94	**0.72**
Xalan						
MCLUST	na	**352.98**	0.52	57.35	0.52	0.50
MARS	na	na	**0.69**	**50.90**	**0.40**	**0.56**
CHINA						
K-Means	8	**624.07**	0.85	51,568.75	276.00	0.82
MCLUST	na	1,805.06	**0.89**	83,420.53	552.85	0.85
MARS	na	na	**0.89**	**25,106.00**	**234.43**	**0.99**
NASACOC						
MCLUST	na	**158.37**	0.97	41.49	2.14	0.95
MARS	na	na	**0.99**	**26.95**	**1.63**	**0.97**
ECLIPSE 3.0						
Random	2	9,458.64	0.29	338.97	**0.10**	0.28
Hierarchical	9	19,205.36	0.29	332.08	**0.10**	0.28
K-Means	3	**3,317.39**	0.31	**324.47**	**0.10**	0.30
Global	na	19,575.91	**0.62**	340.89	**0.10**	**0.59**
MARS	na	na	**0.67**	342.85	**0.10**	0.56

B. Hierarchical Clustering

	K	AIC	FitCor	ErrSum	MedErr	PredCor
Lucene	**2**	1,255.18	0.44	**46.48**	0.98	0.41
	6	**1,254.79**	0.44	46.82	0.84	**0.42**
	9	1,255.47	0.44	46.51	0.72	**0.42**
	10	1,255.26	0.44	46.57	**0.70**	0.41
Xalan	3	**2,003.66**	0.40	58.86	0.47	0.38
	5	2006.17	0.40	**58.28**	0.46	0.38
	9	2,004.58	0.40	58.43	**0.44**	0.38
	10	2,004.82	0.40	58.61	**0.44**	0.38
China	3	**8,492.78**	0.70	119,672.40	850.48	**0.70**
	4	8,496.50	0.70	**117833.30**	839.56	0.69
	6	8,492.83	0.70	119,665.00	791.26	**0.70**
	10	8,495.96	0.70	118,249.60	**726.14**	**0.70**
NASACOC	**3**	826.07	0.76	**72.04**	2.74	0.73
	10	**824.96**	0.76	74.33	**1.63**	0.73
Eclipse 3.0	4	**19,193.89**	0.29	333.38	0.10	0.29
	9	19,205.36	0.29	**332.08**	0.10	0.28

C. K-Means Clustering

K	AIC	FitCor	ErrSum	MedErr	PredCor	
2	1,179.73	0.40	**43.23**	0.95	**0.37**	
Lucene	3	1,172.89	0.39	42.83	**0.91**	0.36
	5	**92.99**	0.41	98.14	1.04	0.26
	10	113.88	**0.53**	131.79	1.11	0.23
Xalan	**3**	371.59	0.44	**58.50**	0.44	0.38
	5	251.52	0.47	60.61	0.41	**0.39**
	6	174.66	0.47	72.49	**0.40**	0.35
	8	**102.88**	0.51	131.57	0.41	0.32
	10	-501.45	**0.52**	156.59	0.43	0.28
China	**8**	625.82	0.85	**51,568.75**	276.00	0.82
	9	673.66	**0.87**	52,944.17	237.54	**0.84**
	10	**529.11**	**0.87**	60,944.49	**234.40**	0.82
NASACOC	**2**	699.19	0.82	**63.84**	2.57	**0.71**
	6	88.23	**0.83**	179.08	1.33	0.56
	7	**15.46**	0.78	119.61	1.31	0.56
	8	40.67	0.80	83.19	**1.07**	0.60
Eclipse 3.0	2	8,625.81	0.31	329.54	**0.10**	0.29
	3	3317.39	0.31	**324.47**	**0.10**	**0.30**
	5	1,533.78	0.31	483.02	**0.10**	**0.30**
	10	**845.23**	**0.32**	497.34	0.11	0.27

In summary we find that for a particular k, the performance of local models built through parametric clustering approaches can be much worse than the global model counterpart, as well as just assigning data points to random clusters. For example, in the XALAN dataset, any choice of $k \geq 4$ for k-means clustering produces a larger absolute prediction error than random clustering (Table 6), and seven out of ten possible choices of k lead to a larger absolute prediction error than a traditional global model (Table 6 C).

The above observation has a significant impact on building local models in practice: we need to perform a careful selection of parameter k for the chosen parametric clustering approach as a calibration step, which is paramount for benefitting from local modeling. A table, such as Table 11 can serve as a first starting point for carrying out such a calibration to the dataset at hand.

3. Depending on the objective (best fit, least prediction error, closest rank correlation), different choices of k provide optimal results when using parametric clustering approaches for building local models.

Tables 6 B. and 6 C. show that *depending on the expected use of the model, a different choice of k may be more appropriate.* This is evidenced by no single row in either Table containing all bold (best) values. For example, practitioners might be more interested in maximizing predicted rank correlation, such as to allocate precious testing resources in decreasing order of predicted relative risk – the absolute number of bugs that the model predicts is not of interest to them. For hierarchical clustering, four out of seven times (57%) selecting k according to minimal prediction errors does not yield the optimal ordering (i.e., value for rank correlation between predicted and actual values). For k-means clustering the same is true five out of seven times (71%).

In addition, Tables 6 B. and 6 C. provide additional evidence to our earlier point, that judging local models by goodness of fit metrics alone is risky. Only a single time does "best goodness of fit" align with an optimal prediction performance measure when using hierarchical clustering (in the case of the CHINA dataset, when considering predicted rank correlation). Similarly, for k-means clustering this is true for the CHINA dataset, with respect to median error.

Discussion

Based on our findings, we cannot recommend a single clustering technique as the optimal technique for building local models. While we observe that MCLUST generally performs very well, other clustering techniques with a careful selection of parameters have the potential for building local models that outperform any other approach we have experimented with in this paper.

However, the advantage of MCLUST being a non-parametric approach: analysts need not perform a preliminary calibration for the parameter k when building local models, but can use the clustering technique out of the box. MCLUST discovers clusters based on analysis of data distributions: an overly simplistic description would be that it creates elliptic regions through the

multidimensional space and grows these regions until they contain normally distributed data (with respect to each metric). However, software engineering data is rarely normally distributed and local modeling may greatly benefit from a non-parametric clustering approach that would work based on geometric and power-law distributions. We leave this as an open challenge for future research.

Furthermore, since our findings support the notion of the importance of clustering algorithm for local modeling, we encourage collaboration between researchers in the fields of empirical software engineering, data mining and machine learning. It would be especially viable for the community if such a collaboration produced clustering algorithms that take into account the special properties of software engineering datasets. One example of a particularly impressive first step into this direction is the WHERE clustering algorithm by Menzies et al. [19].

Parametric clustering approaches have the potential to outperform non-parametric clustering, but only through a careful calibration of parameters by considering both, the dataset and the modeling goal at hand.

RQ5. For the same datasets and predicted outcome, how do different software engineering metrics respond to local modeling?

So far we have demonstrated that software engineering datasets can be used to train local models that outperform global models. We have also demonstrated that the clustering algorithm used to build local models has a major impact on the performance of these models.

However, clustering approaches depend on the existence of observations in the data that are similar to each other, and different from other groups of similar observations. In particular, based on our study on local modeling, we conjecture that local modeling is most viable when the dataset has homogeneous regions with enough variation in the dataset to discern individual clusters of data and clear boundaries between these clusters.

Yet, different sets of software engineering metrics might fulfill these properties to different extent, and as such might respond differently to local modeling. In the following we investigate this conjecture by studying how different sets of metrics for the same datasets and the same predicted outcome (post release defects) respond to local modeling.

Approach

We conduct experiments with an additional dataset that was used in the past to predict post-release defects across all open-source projects under the Eclipse foundation umbrella. The dataset is based on social metrics, as well as code metrics. We have shown in past work [6] that global models based on these social metrics a) have as much explanatory power as models based on traditional code metrics, and b) are orthogonal to traditional metrics, i.e., they

Table 7: Comparison of different sets metrics (code metrics, social metrics) for the same predicted outcome (defect), when using different clustering techniques for building local models in comparison to global models and global models with local considerations (denoted with MARS). Best values are marked in bold face font (for k these correspond to lowest prediction error sum).

ECLIPSE (Social)						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	**23,268.31**	0.30	1,106.88	0.71	0.30
Hierarchical	5	50,001.05	0.30	**1,078.36**	0.69	0.30
K-Means	2	23,402.13	0.30	1,092.13	0.72	0.30
MCLUST	na	50,013.07	0.25	1,099.32	0.66	0.34
Global	na	50,055.34	0.25	1,085.18	0.69	0.37
MARS	na	na	**0.47**	1,152.82	**0.57**	**0.45**
ECLIPSE (Code)						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	9,458.64	0.29	338.97	**0.10**	0.28
Hierarchical	9	19,205.36	0.29	332.08	**0.10**	0.28
K-Means	3	**3,317.39**	0.31	**324.47**	**0.10**	0.30
MCLUST	na	5,701.08	0.37	389.77	0.18	0.57
Global	na	19,575.91	**0.62**	340.89	**0.10**	**0.59**
MARS	na	na	**0.67**	342.85	**0.10**	0.56

capture different aspects of the project. The predicted outcome is the number of post-release defects at a file-level granularity.

In order to investigate to what extent the nature of metrics used to build models for the same outcome lends itself to local modeling, we follow the same experimental setup presented in RQ4. In addition, we use the same evaluation criteria presented in RQ1 and RQ2 in the context of our experiments with the social metrics dataset.

Results

The results of our experiment are presented in Table 7. First, we find that local models based on social metrics do not show a performance advantage over the global model counterparts. If we look at the performance metrics reported in Table 7, we see that only a single local model (Eclipse, Hierarchical) outperforms the global model with respect to prediction error (sum and median error) metrics, and only marginally so. We note that during the execution of the experiment, the MCLUST clustering algorithm reported that it failed to recognize clusters in the Eclipse datasets when using social metrics. Similarly, we see that best performance with k-means clustering is achieved for a choice of $k = 2$.

Furthermore, we observe from Table 8 that the choice of k has a very small impact on the goodness of fit and prediction performance, when using social metrics as the underlying datasets. We carried out a calibration experiment with parametric clustering approaches for local modeling of the social metrics datasets, and found that the reported performance metrics show very little variation with respect to the choice of k, regardless of the used approach (hierarchical, k-means).

Table 8: Performance of local models built through different clustering approaches, by different values of parameter k (number of clusters). Best values marked in bold.

Eclipse 3.0 - Social Metrics - Hierarchical Clustering

K	AIC	FitCor	ErrSum	MedErr	PredCor
2	49995.01	0.30	1082.34	**0.70**	0.30
3	49987.16	0.30	1091.66	0.69	0.30
4	49992.45	0.30	1081.51	0.69	0.30
5	50001.05	0.30	**1078.36**	0.69	0.30
6	49991.83	0.30	1085.68	0.69	0.30
7	49993.21	0.30	1081.43	0.69	0.30
8	**49986.83**	0.30	1089.52	0.69	0.30
9	49996.04	0.30	1079.93	0.69	0.30
10	49990.93	0.30	1084.20	0.68	0.30

Eclipse 3.0 - Social Metrics - K-Means Clustering

K	AIC	FitCor	ErrSum	MedErr	PredCor
2	23402.13	0.30	**1092.13**	0.72	0.30
3	3750.82	0.29	1119.81	0.73	0.29
4	2059.00	0.29	1116.28	0.73	0.29
5	1255.01	0.29	1124.47	0.75	0.29
6	2580.13	**0.32**	1108.37	0.66	**0.31**
7	2685.18	**0.32**	1138.17	**0.64**	**0.31**
8	1696.99	**0.32**	1137.42	**0.64**	**0.31**
9	**968.83**	**0.32**	1156.76	0.65	**0.31**
10	1555.38	**0.32**	1155.50	**0.64**	**0.31**

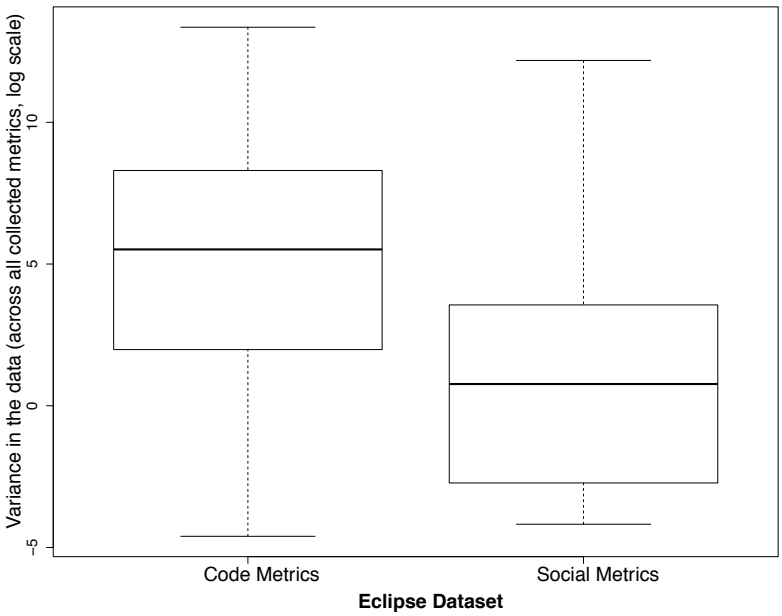


Fig. 5: The variance across all metrics collected in the Eclipse dataset is statistically significantly (Mann-Whitney U test at $p \leq 0.01$ level) for social metrics compared to code metrics, on a logarithmic scale. Our results suggest that such variance is important for clustering the dataset into local regions, which in turn are essential for building local models.

Discussion
Figure 5 might give a possible explanation as to why we fail to build local models when we are using social metrics, but succeed when using code metrics for the same data and predicted outcome. The figure summarizes the variance in the data across all metrics of one type (social or code complexity). In particular, we observe that social metrics contain statistically significantly (Mann-Whitney U at $p \leq 0.01$) less variance in each metric (column) collected for observations (rows) in the data. However, clustering algorithms need that variance to group similar observations into clusters and determine cluster boundaries [1].

Our observations in RQ5 show that if the data already has low variance to begin with, no clustering approach will find meaningful subdivisions of the data. When there are no meaningful subdivisions, local modeling converges to the trivial case and is no different from global modeling. However, the idea of local learning originally proposed by Menzies et al. [19] was exactly motivated by the observation that software engineering data contains high variance and that such high variance has a detrimental impact on modeling.

We want to note that a recent study by Rahman and Devanbu [32] found that the code metrics have higher stasis compared to process metrics. In their work they measure stasis as the change in the metric values from one version to another. Thus they find that code metrics remain the same across versions whereas process metrics changes from one version to next. However, in our work, when we refer to variance in the social metrics, we calculate variance of the metrics within the data from a single version.

Our experiments in connection with social data suggests, that social data is much more homogeneous out of the box, and thus analysts do not need to go the extra mile of carrying out local modeling.

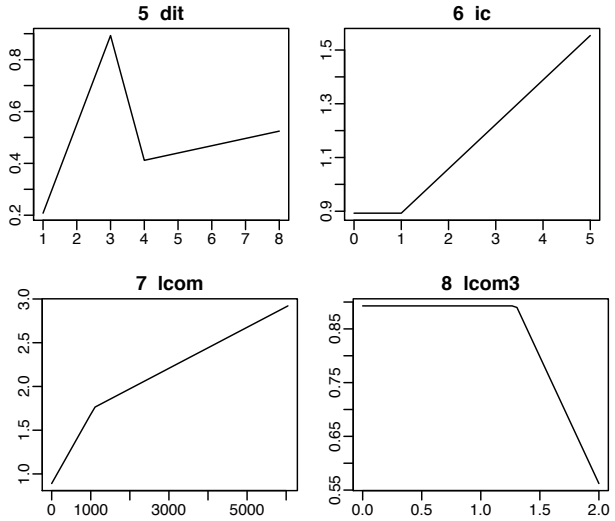
Overall, our results suggest that even for the same case study system, clustering algorithm, evaluation criteria, and outcome, different kinds of metrics are more suitable for building local models than others. Thus we recommend that researchers and practitioners not blindly build local models and assume that these models will lead to significant performance benefits over global models. The underlying data must support the local modeling process in that it needs to contain enough variability in the collected measurements to allow for good clustering into local regions, which then in turn can be used to learn high-quality localized models.

RQ6. What are the considerations in the use of local models over global models for practitioners?

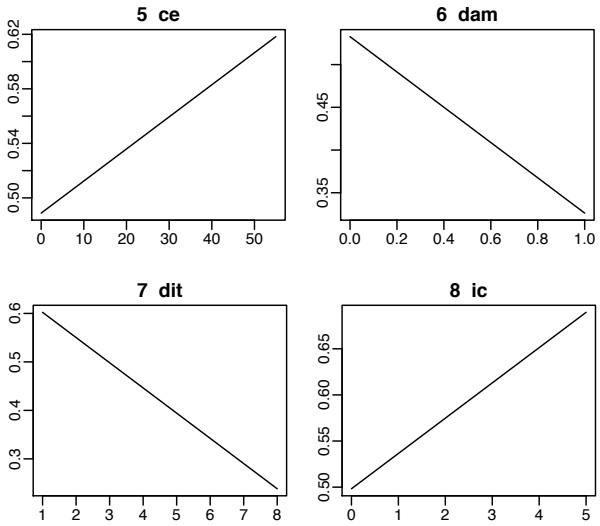
One of the main applications of models when used by practitioners is obtaining an understanding of the software project, to better guide future actions. For example, a manager might not be interested in the absolute predicted value of bugs per file, but rather would like to know what actions he or she should take in order to increase software quality. One possible way to obtain such insights when using regression models, is the use of response plots [14]. These plots describe, how the dependent variable (i.e., bugs or effort) reacts when we change the value of a single independent variable (while at the same time keeping all other variables at their median values).

As an example, we show the response plots for four independent variables found in the global model that was learned on the Xalan 2.6 dataset in Figure 6a. For example, the response plot for variable `ce`, which measures the efferent couplings (how many other classes are used by the specific class for which the model predicts bugs), shows us that as the coupling increases, so does the bug-proneness.

However, response plots obtained from global models show only general trends, as global models are fitted across the complete dataset. Throughout our case study, we have observed that when building local models, the individual models that are learned from each of the clusters differs, in the variables that



(a) Visualization of four independent variables of a global model learned on the Xalan 2.6 dataset



(b) Visualization of four independent variables of a global model with local considerations learned on the same Xalan 2.6 dataset

Fig. 6: Global models report general trends, while global models with local considerations give insights into different regions of the data. The Y-Axis describes the response (in this case bugs) while keeping all other independent variables at their median values.

were deemed significant for that portion of the data, as well as the overall trends.

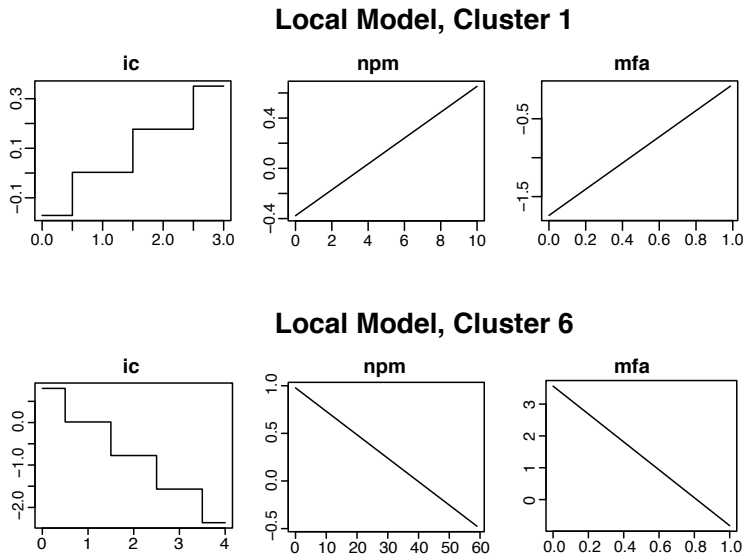


Fig. 7: Example of contradicting trends in local models (one run of local modeling of the Xalan 2.6 dataset, Cluster 1 and Cluster 6.

For example, Figure 7 shows three response plots for local models learned from Cluster 1 and Cluster 6 in one of our experiments on the Xalan 2.6 dataset. The example illustrated in this figure shows the relationship between three predictor variables and the predicted outcome (defects), according to two local regions of the data, and these relationships are completely opposite). Notably, the observed effects of all three independent variables on bugs are contradictory. For Cluster 1, an increase in `ic` (measuring the inheritance coupling through parent classes), `mfa` (the degree of functional abstraction), and `npm` (number of public methods in a class) is predicted to lead to an increase in bug-proneness. At the same time, for Cluster 6, the increase in the same variables is expected to go in hand with a decrease in bug-proneness. Thus analysts need to interpret each local region individually for each predictor variable. As the number of predictor variables, and the number of local regions in the data increases, this task becomes increasingly complex. For example, in the case of Xalan the analyst is presented with 19 x 8 = 152 different trend lines.

While local models are more precise, the trends are a) specific to particular regions of the data, so a practitioner will first have to determine the appropriate cluster for the problem at hand, and b) for each cluster there might be many recommendations to choose from (one recommendation for each metric in each cluster). As an alternative, we propose the use of response plots obtained from global models with local considerations, such as MARS. An

example of a response plot for four independent variables in the MARS model learned on the same Xalan 2.6 dataset is shown in Figure 6b.

By design, the hinge functions of the MARS model already divide the data into regions with individual properties. For example, we observe that an increase of `ic` (measuring the inheritance coupling through parent classes) is predicted to only have a negative effect on bug-proneness when it attains values larger than 1. Thus a practitioner might decide a different course of action than he or she would have done based on the trends outlined by a global model.

Local modeling offers more precise interpretations over global models. However, local modeling results in a multitude of model interpretations (one for each data cluster) that can be contradictory. We recommend global models with local considerations as a hybrid approach that strikes a balance between being able to interpret data regions and simplicity, combining the best of local and global modeling worlds.

5 Conclusions

In this study, we investigated the difference between three different approaches for building models for empirical software engineering datasets. Global models are built on software engineering datasets as-is, while for local models we first subdivide the datasets into subsets of data with similar observations, before building individual models on each subset. In addition, we also studied a third approach: multivariate adaptive regression splines (MARS) as a global model with local considerations. MARS by design takes local considerations of individual regions of the data into account, and can thus be considered a hybrid between global and local models.

A. Think Locally

We evaluated each of the three modeling approaches in a case study on five different datasets, four of which have been used in prior research on the WHICH machine-learning algorithm [20]. The results of our case study demonstrate that clustering of a dataset into regions with similar properties and using the individual regions for building of models leads to an improved fit of these models. Our findings thus confirm the results of Menzies et al., who observed a similar effect of data localization on their WHICH machine-learning algorithm [20]. These increased fits have practical implications for researchers concerned in using regression models for understanding: local models are more insightful than global models, which report only general trends across the whole dataset, whereas we have demonstrated that such general trends may not hold true for particular parts of the dataset. However, we demonstrated that goodness of fit criteria alone are not sufficient to evaluate local models and to

compare them to global models. Yet, considering evaluation criteria that measure predictive performance of models, we observed that local models trained on clustered data and global models with local considerations considerably and consistently outperform their global model counterparts.

Our findings reinforce the recommendations of Menzies et al. [20] against the use of a "one-size-fits-all" approach, such as a global model, when trying to account for such localized effects. However, as we have demonstrated, local modeling depends on:

1. the variability in the data to obtain meaningful clusters.
2. the clustering algorithm that is used to divide the data into local regions.
3. careful calibration of parameters for the used clustering approaches.

Based on our findings, we recommend strongly against blindly building local models in the hope that they will outperform global models. Our findings show that without careful a priori calibration and evaluation, analysts might end up building local models whose performance is considerably worse than their global counterparts. As such, we strongly support the notion of Menzies et al. for a dedicated "local modeling team" [19] with the expertise and knowledge to benefit from this modeling approach. On the other hand, our study also demonstrates that out-of-the box approaches that do not require analysts to provide parameters (i.e., clustering using MCLUST, learning global models with local considerations), provide performance benefits over global models for four out of five of our case study subject datasets. These approaches may be more suitable when no dedicated "local modeling team" is available, or resources for a-priori calibration and evaluation are limited.

B. Act Globally

Building local models involves a significant overhead due to clustering of the data. Even though clustering algorithms such as the one presented in the work by Menzies et al. [20] might run in linear time, researchers and practitioners still have to work with a multitude of models, one for each cluster. Paired with the need for preliminary analysis and calibration when using a parametric clustering approach, analysts may want to skip building local models based on data clustering and turn to global models with local considerations, which require less effort and are much less resource intensive.

For practical applications of guiding future decisions, we observed that global models produce general trends, which might not hold true for particular regions of the data. However, as an alternative, local models produce too much insight, that practitioners may find hard to put into practice, especially with respect to conflicting observations across different clusters. Global models that take local considerations into account, such as the MARS model, combine the best of both worlds.

Based on the findings of our study, we believe that the community would greatly benefit from insights gained in the following future research directions. For *researchers*, we want to emphasize that clustering is paramount for building high-performance local models, yet software engineering datasets

have unique properties (e.g., many metrics are not normally distributed) and current clustering approaches are not SE friendly. For *practitioners*, we want to note that blindly building local models might not lead to the desired performance advantages over global models. In fact, our results demonstrate that without careful calibration of parameters, the resulting local models can be considerably worse than their global model counterparts. Thus, we need to find approaches for building local models, which are accessible to practitioners without data modeling backgrounds. For now, we recommend using MCLUST as a non-parametric clustering approach (even though it is not specifically designed for SE data) for local modeling, or using global models with local considerations. Both modeling approaches show comparable performance to global models in the worst case, and significant performance advantages in the majority of datasets that we studied.

Repeatability

To enable repeatability of our work, and invite future research, we provide all datasets, tools, and the complete set of R code that have been used to conduct this study at:
`http://sailhome.cs.queensu.ca/replication/local-vs-global-emse/`.

References

1. Ackerman, M., Ben-david, S.: Clusterability: A theoretical study. In: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS'09), JMLR Workshop and Conference Proceedings Volume 5, pp. 1–8 (2009)
2. Akaike, H.: A new look at the statistical model identification. Automatic Control IEEE Transactions on **19**(6), 716–723 (1974)
3. Andreou, A., Papatheocharous, E.: Software cost estimation using fuzzy decision trees. In: Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on, pp. 371 –374 (2008)
4. Attoh-okine, N., Mensah, S., Nawaiseh, M., Hall, D.: Using multivariate adaptive regression splines (mars) in pavement roughness prediction. Strategy (2001)
5. Barkmann, H., Lincke, R., Lowe, W.: Quantitative evaluation of software quality metrics in open-source projects. In: Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops, WAINA '09, pp. 1067–1072. IEEE Computer Society, Washington, DC, USA (2009)
6. Bettenburg, N., Hassan, A.E.: Studying the impact of social structures on software quality. In: Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension, ICPC '10, pp. 124–133. IEEE Computer Society, Washington, DC, USA (2010)
7. Bettenburg, N., Nagappan, M., Hassan, A.: Think locally, act globally: Improving defect and effort prediction models. In: Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on, pp. 60–69 (2012)
8. Di Penta, M.: Nothing else matters: what predictive model should i use? In: Proceedings of the 7th International Conference on Predictive Models in Software Engineering, Promise '11, pp. 10:1–10:3. ACM, New York, NY, USA (2011)
9. Elish, K.O., Elish, M.O.: Predicting defect-prone software modules using support vector machines. J. Syst. Softw. **81**, 649–660 (2008)
10. Fox, J.: Applied Regression analysis and generalized linear models, 2 edn. Sage, Los Angeles, London (2008)

11. Fraley, C.: Bayesian regularization for normal mixture estimation and model-based clustering. Journal of Classification **181**(2), 155–181 (2007)
12. Fraley, C., Raftery, A.E.: Mclust version 3 for r : Normal mixture modeling. Office **Technical**(504), 1–54 (2009)
13. Friedman, J.H.: Multivariate Adaptive Regression Splines. The Annals of Statistics **19**(1), 1–67 (1991)
14. Harrell, F.E.: Regression modeling strategies : with applications to linear models, logistic regression, and survival analysis. Springer (2001)
15. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. JSTOR: Applied Statistics **28**(1), 100–108 (1979)
16. Kamei, Y., Matsumoto, S., Monden, A., Matsumoto, K.i., Adams, B., Hassan, A.E.: Revisiting common bug prediction findings using effort-aware models. In: Proceedings of the 2010 IEEE International Conference on Software Maintenance, ICSM '10, pp. 1–10. IEEE Computer Society (2010)
17. Li, M., Zhang, H., Wu, R., Zhou, Z.H.: Sample-based software defect prediction with active and semi-supervised learning. Automated Software Engg. **19**(2), 201–230 (2012). DOI 10.1007/s10515-011-0092-1. URL `http://dx.doi.org/10.1007/s10515-011-0092-1`
18. McQuitty, L.: Similarity analysis by reciprocal pairs for discrete and continuous data. Educational and Psychological Measurement pp. 825–831 (1966)
19. Menzies, T., Butcher, A., Cok, D., Layman, L., Marcus, A., Shull, F., Turhan, B., Zimmermann, T.: Local vs. global lessons from defect prediction and effort estimation. IEEE Transactions on Software Engineering, to appear (2013)
20. Menzies, T., Butcher, A., Marcus, A., Zimmermann, T., Cok, D.: Local vs global models for effort estimation and defect prediction. In: Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering (2011)
21. Menzies, T., Caglayan, B., Kocaguneli, E., Krall, J., Peters, F., Turhan, B.: The promise repository of empirical software engineering data (2012). URL `http://promisedata.googlecode.com`
22. Menzies, T., Greenwald, J., Frank, A.: Data mining static code attributes to learn defect predictors. Software Engineering, IEEE Transactions on **33**(1), 2–13 (2007)
23. Mockus, A., Weiss, D.M.: Predicting risk of software changes. Bell Labs Technical Journal **5**, 169–180 (2000)
24. Mockus, A., Weiss, D.M., Zhang, P.: Understanding and predicting effort in software projects. In: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, pp. 274–284. IEEE Computer Society, Washington, DC, USA (2003)
25. Mockus, A., Zhang, P., Li, P.L.: Predictors of customer perceived software quality. In: Proceedings of the 27th international conference on Software engineering, ICSE '05, pp. 225–233. ACM, New York, NY, USA (2005)
26. Nagappan, N., Ball, T.: Use of relative code churn measures to predict system defect density. In: Proceedings of the 27th international conference on Software engineering, ICSE '05, pp. 284–292. ACM (2005)
27. Nagappan, N., Ball, T., Zeller, A.: Mining metrics to predict component failures. In: Proceedings of the 28th international conference on Software engineering, ICSE '06, pp. 452–461. ACM, New York, NY, USA (2006)
28. Nguyen, T.H.D., Adams, B., Hassan, A.E.: Studying the impact of dependency network measures on software quality. In: Proceedings of the 2010 IEEE International Conference on Software Maintenance, pp. 1–10. IEEE Computer Society (2010)
29. Osei-Bryson, K.M., Ko, M.: Exploring the relationship between information technology investments and firm performance using regression splines analysis. Information and Management **42**(1), 1 – 13 (2004)
30. Posnett, D., Filkov, V., Devanbu, P.: Ecological inference in empirical software engineering. International Conference on Automated Software Engineering, pp. 362–371 (2011)
31. Raftery, A.E., Raftery, A.E.: Bayesian model selection in social research adrian e. raftery sociological methodology, vol. 25. (1995), pp. 111-163. Social Research **25**(1995), 111–163 (2007)

32. Rahman F., Devanbu, P.: How, and why, process metrics are better. In: Proceedings of the 2013 International Conference on Software Engineering, ICSE '13, pp. 432–441. IEEE Computer Society, Washington, DC, USA (2013)
33. Rice, J.A.: Mathematical Statistics and Data Analysis. Duxbury Press (2001)
34. Schwarz, G.: Estimating the dimension of a model. Ann. Statist. **6**(2), 461–464 (1978)
35. Shepperd, M.: A critique of cyclomatic complexity as a software metric. Software Engineering Journal **3**(2), 30 –36 (1988)
36. Shihab, E., Bird, C., Zimmermann, T., Zimmermann, T.: The effect of branching strategies on software quality. In: ESEM, pp. 301–310 (2012)
37. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2005)
38. York, T.P., Eaves, L.J.: Common disease analysis using multivariate adaptive regression splines (mars): Genetic analysis workshop 12 simulated sequence data. Genetic Epidemiology **21 Suppl 1**, S649–S654 (2001)
39. Zimmermann, T., Nagappan, N., Gall, H., Giger, E., Murphy, B.: Cross-project defect prediction: a large scale experiment on data vs. domain vs. process. In: Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering, ESEC/FSE '09, pp. 91–100. ACM, New York, NY, USA (2009)
40. Zimmermann, T., Premraj, R., Zeller, A.: Predicting defects for eclipse. In: Proceedings of the Third International Workshop on Predictor Models in Software Engineering, PROMISE '07, pp. 9–. IEEE Computer Society, Washington, DC, USA (2007)

Appendix A

Further descriptions of the individual metrics used in these datasets can be found in the work by Menzies et al. [20] and our previous work on social metrics [6].

Metrics in the Lucene 2.4 Dataset

Dependent Variables: `bug`
Independent Variables: `amc, avg_cc, ca, cam, cbm, ce, dam, dit, ic, lcom, lcom3, max_cc, mfa, moa, noc, npm`

Metrics in the Xalan 2.6 Dataset

Dependent Variables: `bug`
Independent Variables: `avg_cc, ca, cam, cbm, ce, dam, dit, ic, lcom, lcom3, loc, max_cc, mfa, moa, noc, npm`

Metrics in the CHINA Dataset

Dependent Variables: `Effort`
Independent Variables: `Input, Output, Enquiry, File, Interface, Changed, PDR_UFP, Resource, Duration`

Metrics in the NASACOC Dataset

Dependent Variables: `months`
Independent Variables: `pmat, rely, data, cplx, time, stor, pvol, pcap, apex, plex, ltex, site, sced, kloc, effort`

Metrics in the Eclipse 3.0 (Code) Dataset

Dependent Variables: `post`
Independent Variables: `pre, ImportDeclaration, VG_sum, PrefixExpression, NOM_avg, NOF_avg, TLOC, NullLiteral, NOM_max, BooleanLiteral, SwitchCase, SimpleName, SuperMethodInvocation, LabeledStatement, Block, NORM_Assignment, Initializer, NSM_avg, InfixExpression, Assignment, NumberLiteral, VariableDeclarationFragment, Javadoc, NORM_FieldDeclaration, ForStatement, Modifier, NORM_ArrayCreation, MethodInvocation, VariableDeclarationExpression, ArrayCreation, ExpressionStatement, NORM_PostfixExpression, InstanceofExpression, SwitchStatement, ArrayType, SynchronizedStatement`

Metrics in the Eclipse 3.0 (Social) Dataset

Dependent Variables: `poat`
Independent Variables: `NSCOM, PATCHS, NSOURCE, NPATCH, NTRACE, TRACES, NLINK, NPART, NDEVS, NUSERS, SNACENT, NMSG, REPLY, REPLYE, DLEN, DLENE, INT, INTE, WA, WAE`

Appendix B

Table 9: Results: Using different clustering techniques for building local models in comparison to global models and global models with local considerations (denoted with MARS). Best values are marked in bold face font and the value of k corresponds to lowest prediction error sum. Parametric clustering approaches have the potential to outperform non-parametric clustering approaches with a careful selection of parameter k.

Lucene						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	622.90	0.45	49.59	1.07	0.38
Hierarchical	2	1,255.18	0.44	46.48	0.98	0.41
K-Means	3	1,172.89	0.39	**42.83**	**0.91**	0.36
MCLUST	na	**462.43**	0.60	55.15	1.15	0.67
Global	na	1,380.35	0.32	49.72	1.15	0.71
MARS	na	na	**0.83**	43.61	0.94	**0.72**
Xalan						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	1,004.61	0.41	59.41	0.48	0.36
Hierarchical	5	2,006.17	0.40	58.28	0.46	0.38
K-Means	3	371.59	0.44	58.50	0.44	0.38
MCLUST	na	**352.98**	0.52	57.35	0.52	0.50
Global	na	2,190.30	0.33	61.07	0.64	0.36
MARS	na	na	**0.69**	**50.90**	**0.40**	**0.56**
CHINA						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	4,221.43	0.67	130,887.50	1015.36	0.67
Hierarchical	4	8,494.27	0.70	117,833.30	839.56	0.69
K-Means	8	**624.07**	0.85	51,568.75	276.00	0.82
MCLUST	na	1,805.06	**0.89**	83,420.53	552.85	0.85
Global	na	8,696.17	0.83	91,592.52	765.00	0.82
MARS	na	na	**0.89**	**25,106.00**	**234.43**	**0.99**
NASACOC						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	398.77	0.78	77.58	3.16	0.70
Hierarchical	3	826.07	0.76	72.04	2.74	0.73
K-Means	2	699.19	0.82	63.84	2.57	0.71
MCLUST	na	**158.37**	0.97	41.49	2.14	0.95
Global	na	585.95	0.93	48.75	3.26	0.95
MARS	na	na	**0.99**	**26.95**	**1.63**	**0.97**
ECLIPSE 3.0						
	K	**AIC**	**FitCor**	**ErrSum**	**MedErr**	**PredCor**
Random	2	9,458.64	0.29	338.97	**0.10**	0.28
Hierarchical	9	19,205.36	0.29	332.08	**0.10**	0.28
K-Means	3	**3,317.39**	0.31	**324.47**	**0.10**	0.30
MCLUST	na	5,701.08	0.37	389.77	0.18	0.57
Global	na	19,575.91	**0.62**	340.89	**0.10**	**0.59**
MARS	na	na	**0.67**	342.85	**0.10**	0.56

Table 10: Performance of local models built through hierarchical clustering, with different values for parameter k (number of clusters). All performance metrics, except for median prediction error, show very small differences depending on the choice of parameter k.

K	AIC	FitCor	ErrSum	MedErr	PredCor
Lucene					
2	1,255.18	0.44	**46.48**	0.98	0.41
3	1,254.89	0.44	47.02	0.96	0.41
4	1,254.55	0.44	46.80	0.92	0.41
5	1,255.03	0.44	46.93	0.89	0.41
6	**1,254.79**	0.44	46.82	0.84	**0.42**
7	1,254.37	0.44	47.20	0.81	0.41
8	1,255.39	0.44	46.74	0.77	0.41
9	1,255.47	0.44	46.51	0.72	**0.42**
10	1,255.26	0.44	46.57	**0.70**	0.41
Xalan					
2	2,004.15	0.40	58.67	0.48	0.38
3	**2,003.66**	0.40	58.86	0.47	0.38
4	2,004.45	0.40	58.70	0.47	0.38
5	2006.17	0.40	**58.28**	0.46	0.38
6	2,004.75	0.40	58.42	0.46	0.39
7	2,004.87	0.40	58.45	0.45	0.38
8	2,003.81	0.40	58.78	0.45	0.38
9	2,004.58	0.40	58.43	**0.44**	0.38
10	2,004.82	0.40	58.61	**0.44**	0.38
CHINA					
2	8,495.32	0.70	118,638.90	866.65	0.69
3	**8,492.78**	0.70	119,672.40	850.48	**0.70**
4	8,496.50	0.70	**117833.30**	839.56	0.69
5	8,495.28	0.70	118,378.90	818.78	0.69
6	8,492.83	0.70	119,665.00	791.26	**0.70**
7	8,493.73	0.70	119,275.80	780.55	0.69
8	8,495.23	0.70	118,577.40	771.16	0.69
9	8,493.63	0.70	120,123.30	748.75	0.69
10	8,495.96	0.70	118,249.60	**726.14**	**0.70**
NASACOC					
2	825.42	0.76	73.38	2.90	0.73
3	826.07	0.76	**72.04**	2.74	0.73
4	825.95	0.76	72.22	2.53	0.72
5	825.79	0.76	72.75	2.44	0.73
6	825.40	0.76	73.68	2.28	0.73
7	825.44	0.76	73.65	2.12	0.73
8	825.98	0.76	72.18	1.92	0.73
9	825.21	0.76	73.62	1.76	0.73
10	**824.96**	0.76	74.33	**1.63**	0.73
Eclipse 3.0					
2	19,202.72	0.29	332.63	0.10	0.29
3	19,210.63	0.29	332.36	0.10	0.28
4	**19,193.89**	0.29	333.38	0.10	0.29
5	19,197.26	0.29	333.60	0.10	0.28
6	19,201.34	0.29	332.51	0.10	0.29
7	19,196.12	0.29	333.68	0.10	0.29
8	19,202.15	0.29	333.19	0.10	0.28
9	19,205.36	0.29	**332.08**	0.10	0.28
10	19,203.52	0.29	333.26	0.10	0.28

Table 11: Performance of local models built through k-means clustering, with different values for parameter k (number of clusters). Depending on the modeling objective, different choices of parameter k are optimal.

K	AIC	FitCor	ErrSum	MedErr	PredCor
Lucene					
2	1,179.73	0.40	**43.23**	0.95	**0.37**
3	1,172.89	0.39	42.83	**0.91**	0.36
4	558.96	0.40	78.30	0.95	0.29
5	**92.99**	0.41	98.14	1.04	0.26
6	105.15	0.42	89.65	1.07	0.25
7	104.68	0.42	86.35	1.04	0.25
8	135.24	0.45	110.64	1.12	0.22
9	97.77	0.48	155.96	1.14	0.19
10	113.88	**0.53**	131.79	1.11	0.23
Xalan					
2	941.64	0.41	59.03	0.45	0.37
3	371.59	0.44	**58.50**	0.44	0.38
4	215.21	0.44	109.45	0.43	0.34
5	251.52	0.47	60.61	0.41	**0.39**
6	174.66	0.47	72.49	**0.40**	0.35
7	123.58	0.51	83.14	0.41	0.36
8	**102.88**	0.51	131.57	0.41	0.32
9	-225.25	0.51	122.80	0.42	0.31
10	-501.45	**0.52**	156.59	0.43	0.28
China					
2	3,964.04	0.75	101,731.46	648.05	0.74
3	934.37	0.78	79,068.97	629.40	0.76
4	872.69	0.80	66,220.52	524.68	0.78
5	651.79	0.82	58,563.48	396.06	0.80
6	652.09	0.80	63,553.69	380.82	0.78
7	658.80	0.82	60,341.00	334.70	0.79
8	625.82	0.85	**51,568.75**	276.00	0.82
9	673.66	**0.87**	52,944.17	237.54	**0.84**
10	**529.11**	**0.87**	60,944.49	**234.40**	0.82
NASACOC					
2	699.19	0.82	**63.84**	2.57	**0.71**
3	313.11	0.76	112.58	2.05	0.62
4	117.77	0.79	119.74	1.77	0.64
5	157.38	0.81	133.98	1.57	0.59
6	88.23	**0.83**	179.08	1.33	0.56
7	**15.46**	0.78	119.61	1.31	0.56
8	40.67	0.80	83.19	**1.07**	0.60
9	-2.30	0.82	129.73	1.21	0.49
10	23.41	0.73	181.06	1.29	0.28
Eclipse 3.0					
2	8,625.81	0.31	329.54	**0.10**	0.29
3	3317.39	0.31	**324.47**	**0.10**	**0.30**
4	2,833.96	0.31	359.12	0.11	0.29
5	1,533.78	0.31	483.02	**0.10**	**0.30**
6	1,730.24	0.31	470.03	0.12	**0.30**
7	1,178.33	0.31	377.62	0.11	**0.30**
8	1,284.91	**0.32**	364.45	0.11	0.29
9	1,037.70	**0.32**	428.98	0.11	0.28
10	**845.23**	**0.32**	497.34	0.11	0.27