Think Locally, Act Globally: Improving Defect and Effort Prediction Models

Nicolas Bettenburg, Meiyappan Nagappan and 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*

***Abstract*—Much research energy in software engineering is focused on the creation of effort and defect prediction models. Such models are important means for practitioners to judge their current project situation, optimize the allocation of their resources, and make informed future decisions. However, software engineering data contains a large amount of variability. Recent research demonstrates that such variability leads to poor fits of machine learning models to the underlying data, and suggests splitting datasets into more fine-grained subsets with similar properties. In this paper, we present a comparison of three different approaches for creating statistical regression models to model and predict software defects and development effort. Global models are trained on the whole dataset. In contrast, local models are trained on subsets of the dataset. Last, we build a global model that takes into account local characteristics of the data. We evaluate the performance of these three approaches in a case study on two defect and two effort datasets. We find that for both types of data, local models show a significantly increased fit to the data compared to global models. The substantial improvements in both relative and absolute prediction errors demonstrate that this increased goodness of fit is valuable in practice. Finally, our experiments suggest that trends obtained from global models are too general for practical recommendations. At the same time, local models provide a multitude of trends which are only valid for specific subsets of the data. Instead, we advocate the use of trends obtained from global models that take into account local characteristics, as they combine the best of both worlds.**

***Keywords*-software metrics, models, techniques.**

I. Introduction

Data modelling is an important technique that is frequently employed in empirical software engineering research, for studying observations from software projects. Common modelling approaches use a variety of tools from machine learning methods, such as decision trees or support vector machines, to statistical descriptions in the form of regression models [2], [6], [7], [15]. In the past, the foremost application of modelling was to create prediction models. These models describe how a set of software 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. For example, empirical methods have demonstrated that the amount of source code changes carried out on a file give a reliable indication of the potential future defects one is likely to observe in that file [18]. In recent research, modelling has also gained use as a means for understanding. Instead of using prediction models solely as a blackbox, which is fed a set of measures and outputs a prediction, closer inspection of a prediction model can provide insights into the underlying processes [5], [6], [15], [16].

Regardless of the final goal (modelling or prediction), models are derived from data about the artifact under study. This data is commonly extracted by mining software repositories. Past research has shown that most of such software engineering data contains a great amount of variability [14]. Since day one, we have been using software engineering datasets for model building as is, without further considering such variability.

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 [14]. The study showed that building models using such subsets (i.e. local) models lead to better fits when using specialized machine learning algorithms. However, it is not known:

1) Whether this finding holds for statistical techniques such as linear regression.
2) Whether such models would lead to better predictions since better fits might be due to overfitting of the data. Such overfit might in turn lead to lower prediction performance.
3) Whether others (practitioners and researchers) can replicate such findings using off-the-shelf techniques implemented in today's data analysis tools such as R.
4) And finally, whether one can create models that bridge the benefits of both, local and global worlds.

Therefore, we set out to investigate the following research questions:

RQ1 *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 regression 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 predicted variable. **We find that a local approach produces significantly better fits**

of statistical prediction models to the underlying data.

RQ2 *Is there an advantage of using local models over global models, with respect to prediction performance?* Better performing prediction models are of great value to practitioners since they allow to take better informed decisions and actions. **We find that the better fits do not overfit the prediction models. The local approach significantly increases the predictive power of statistical models, up to three times lower prediction error.**

RQ3 *What are 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 can distinguish the significant variables for each local region of the data, the recommendations between different regions can be conflicting.**

Our findings impact both practitioners and researchers alike. Researchers in empirical software engineering might want to further investigate modelling of local regions to use regression models for understanding the differences of local regions in the data. Furthermore, we see potential of local modelling for building improved prediction models.

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 preprocessing, as well as a discussion of how we derived the global and local prediction models used in the case study. Section IV, discusses the findings of our case study, and Section V, outlines our conclusions, as well as recommendations for future research efforts.

II. Background and Related Work

Modelling Goals. Models are used in the area of empirical software engineering research 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 failure probability for future new modules [19]. 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 models. Similarly, Mockus et al. studied the use of linear regression models to predict the risk of source code changes [15].

At the same time, models can be used to gain an understanding of why certain effects happen. For instance, Mockus et al. use regression models to understand the relationships between observations on the software process and how customers perceive the quality of the product [17]. A similar approach was used in a study by Mockus et al. to investigate how changes impact the overall software development effort [16].

Modelling Techniques. The majority of modelling 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 [2]. Elish et al. use a machine learning technique called support vector machines (SVM), to predict defect-prone modules at NASA [7]. They find that machine learning techniques perform as well as statistical approaches for defect prediction.

One of the most popular statistical techniques is regression modelling. For instance, Nagappan et al. use regression models to predict defect density based on a measure of changeability, named code churn [18]. Within the same vein of research, Zimmermann et al. use regression models to study which metrics are the best performing predictors for software defects in the Eclipse project [30], as well as for cross-project defect prediction [29].

Modelling Approaches. Since day one, researchers in empirical software engineering have built global models, to predict development effort and software defects with high accuracy [15], as well as for understanding of software engineering phenomena [5], [16]. *In this work, we call models that are learned on a complete dataset, a "global" model, as they take the complete data with all its inherent variation into account.*

However, recent research suggests that empirical SE should focus more on context specific principles, in particular Menzies et al. advise that future work in empirical software engineering explore lessons learned from individual subsets in contrast to lessons learned across whole data [14]. *In this work, we call models that are learned by subdividing larger datasets into smaller subsets with similar data properties, and learning individual prediction models on each subset of data, "local" models.*

A similar issue was also recently raised in the works of Kamei et al. and Nguyen et al. [13], [20], 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. Posnett et al. [22] recently confirmed the existence of this bias in a study within the same vein of research as Kamei et al. [13].

Where this paper fits in. The research most closely related to this paper, is the work by Menzies et al., who demonstrate that for their WHICH treatment 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 [14]. 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. In the following we identify key differences between the work by Menzies et al. and our study.

(1) We follow the idea of data partitioning into local regions within the context of prediction models based on linear regression. Our study investigates, whether off-the-shelf technology that is readily available for practitioners and researchers can experience the same benefits of data partitioning.

(2) In addition to evaluating goodness of fit, we 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.

(3) 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 multivariate adaptive regression splines [11] (MARS). MARS models have found extensive and successful use in modelling problems in fields outside of empirical software engineering such as economics [21], genetics [28], and civil engineering [3].

To the best of our knowledge, our work is the first to explore the benefits of partitioning software engineering data along the axis of highest variability within the context of prediction models based on linear regression functions.

III. Case Study Design

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

A. Data Collection and Preparation

All datasets used in our case study have been obtained from the PROMISE [1] repository, and have been reported to be as diverse of datasets as can be found in this database [14]. 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). A summary of the obtained data is shown in Table I. For a complete set of descriptions of the variables in each of the four datasets, we refer the interested reader to [14].

[1] http://promisedata.org

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

Dataset	Predictions for	Metrics	Datapoints
Xalan 2.6	Defects	23	885
Lucene 2.4	Defects	23	340
CHINA	Development Effort	19	499
NasaCoc	Development Effort	27	154

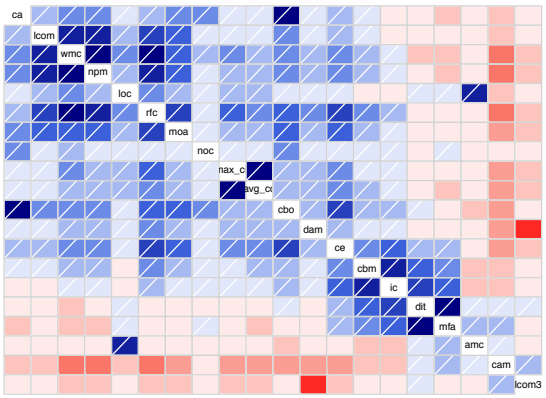


Figure 1: Visualization of the correlation analysis for the Xalan 2.6 dataset. Higher correlations between metrics result in darker coloured areas (red for positive correlation, blue for negative corellation).

Correlation Analysis. To prepare each of the four datasets for use in our experiments, we first carry out a correlation analysis. In each case, we start from a complete dataset, and carry out an analysis of potential multi-collinearity between the metrics (columns) in the datasets. Previous research [4], [25] has demonstrated that many process and source code metrics are correlated, both with each other, and with lines of code (LOC). As an example to illustrate this problem, we show a visualization of the auto-correlations of all variables in the Xalan 2.6 dataset in Figure 1. From this visualization, we observe a number of very high correlations (dark coloured squares) that we need to take care of, before using this data for defect prediction. Ignoring such correlations would lead to increased errors in the estimates of model performances, and increased standard errors of predictions [12].

VIF Analysis. Within the same vein of prior research, we handle multi-collinearity through analysis of variance inflation factors (VIF) for each dataset [8]. 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 [8].

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.

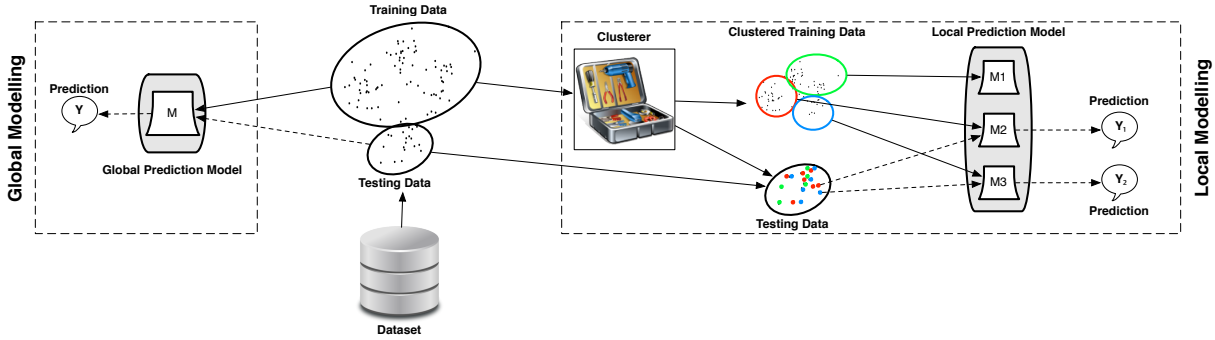


Figure 2: Overview of our approach for building global and local regression models. This process is repeated 10 times.

B. Model Building Approach

An overview of our approach to build the models used in our case study is illustrated in Figure 2. In the following we discuss each model type in more detail.

Global Models. We start the process of building a global model, 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 use a statistical modelling approach, linear regression, to build a global model. 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 that we base our predictions on.

In empirical software engineering, a common practice is to first obtain a multitude of metrics for use in prediction models. For example, the popular PROMISE2007 dataset [30] contains a total of 197 different software process and source code metrics on the Eclipse software. When building regression models, it is important to select a meaningful subset of independent variables in order to maximize the predictive power of the regression model, while at the same time keeping the overall set of variables low enough to avoid over-fitting. With over-fitting, we describe a situation, where a prediction model has been tailored too closely to the data it was learned on, and subsequently has significant problems to accurately deal with unseen data. For prediction models in particular, over-fitting is a risk that needs to be avoided if possible.

Local Models. The process for building local models starts out similarly to that of building global models: by randomly splitting the dataset into training data and testing data.

As a main difference between global and local models, the training data is then partitioned into regions with local properties by a clustering algorithm. We want to note at this point that the primary goal of this paper is using partitions of similar data for building defect and effort prediction models, rather than the discussion of obtaining the best possible partitions with one clustering technique or the other.

Since we have no prior knowledge as to the optimal number of subsets in our datasets, we decided to employ a state-of-the-art model-based clustering technique called MCLUST [10]. 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 [9].

We applied the MCLUST clustering technique, to divide each of the four prediction datasets into smaller subsets, within which observations have similar properties. Figure 3 shows an overview of the subsets derived with this clustering technique. We observe that MCLUST produces a different amount of clusters depending on the random sample of the training data. For 10 random splits of the dataset into training and testing data, we found that the number of clusters ranges between 2 and 9.

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 data (clusters produced by MCLUST). To carry out predictions on the testing data using local prediction models, we have to take the additional step of determining for each input the most similar cluster first. After the appropriate cluster has been determined for each entry in the testing data, we then subsequently use the local model that has been fitted to that particular cluster, and carry out the individual prediction.

C. Automatic Model Selection

To handle the potential threat of overfitting in global and local models, we need to select an appropriate subset of independent variables. We follow a common automated model selection approach based on the Bayesian information criterion (BIC) [24], called Bayesian model averaging (BMA) [23]. Model selection based on the BIC measure resolves the problem of over-fitting by introducing a penalty

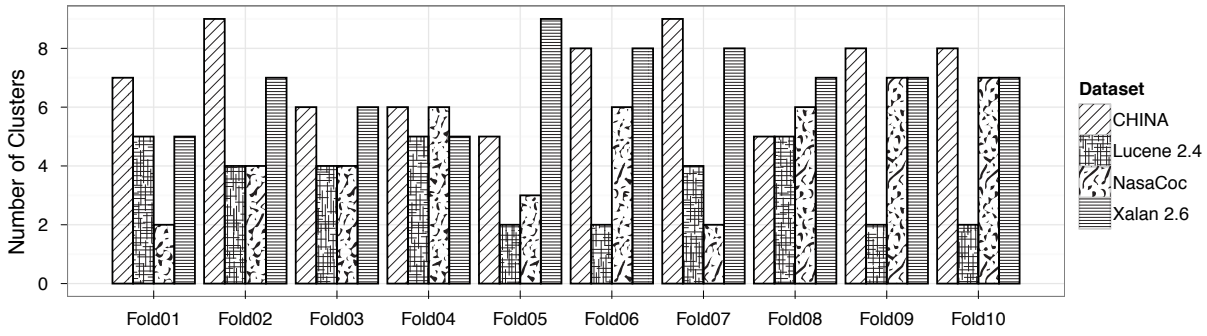


Figure 3: Number of clusters generated by MCLUST in each run of the 10-fold cross validation.

term for each additional prediction variable entering the regression model [23].

For practical purposes, we use a publicly available implementation of BIC-based model selection, contained in the `R` package: `BMA`. The input to the BMA implementation is the dataset itself, as well as a list of all dependent and independent variables that should be considered. In our case study, we always supply a list of all independent variables that were left after VIF analysis. The output of the BMA implementation is a selection of independent variables, such that the linear regression model built with these variables will produce the best fit to the data, while avoiding overfitting.

D. Multivariate Adaptive Regression Splines

We use Multivariate Adaptive Regression Splines [11], or MARS, models, as an example of a global prediction 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 [3], [21], [28]. A MARS model has the form $Y = \epsilon_o + c_1 * H(X_1) + \cdots + c_i * H(X_n)$, with Y called the dependent variable (that is to be predicted), c_i 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 partition 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.

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 model's generalization ability. The backward phase removes individual terms, deleting the least effective term at each step until it finds the best submodel. Model subsets are compared using a performance criterion specific to MARS models, and the best model is selected and returned as the final prediction model.

MARS models have the advantage that a model selection phase is already built-in by design, so we do not need to carry out a model selection 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 partitioning of the dataset with a clustering algorithm. Both advantages make this type of prediction model very easy to use in practice.

E. Cross Validation

For better generalizability of our results, and to counter random observation bias, all experiments described in our case study are repeated 10 times on random stratified subsamples of the data into training (90% of the data) and testing (10% of the data) sets. Stratification is carried out on the measure that is to be predicted. We evaluate all our findings based on the average over the 10 repetitions. This practice of evaluating results is a common approach in Machine Learning, and is often referred to as "10-fold cross validation" [27].

IV. Results

In this section, we present the result of our case study. This presentation is carried out in three steps that follow our initial three research questions. For each part, we first

describe our evaluation approach, and then 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 section is to evaluate the goodness of fit of the prediction models produced by each of the three modelling approaches. In general, a goodness of fit measure describes how well the prediction 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 is trying to use prediction models as a means of understanding underlying processes.

For example, if one was 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. A commonly used goodness of fit measure for linear regression models is the coefficient of determination, R^2. In general, the 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 measure is highly biased towards the number of independent variables in the model [12]. 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. The two measures are discussed below.

Akaike Information Criterion. To better judge the goodness of fit between global prediction models and local prediction models, we use the Akaike information criterion (AIC) [1]. AIC is a relative goodness of fit measure based on information entropy. On of the main advantages over R^2 is the robustness of this measure against a bias towards using more independent variables in a model. AIC is widely used to judge the relative goodness of fit between different regression models [23]. In general, a lower AIC measure corresponds to a better fit of the model to the data. 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.

Table II: Case study results: goodness of fit for global models. For AIC, smaller is better, for Correlation, higher is better.

	Global		Local		MARS
Dataset	**AIC**	**Cor**	**AIC**	**Cor**	**Cor**
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**

Correlation between predicted and actual values on trained data. 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 correlation between actual values and predicted values. This measure of correlation is of particular significance for defect prediction models. For defect prediction, models are often not concerned with the absolute number of post-release defects, but rather in a ranking of source code entities from "most buggy" to "least buggy" [18]. Resources and testing effort are then allocated according to that ranking.

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

The same observation holds true for the analysis of correlation fit. Our analysis of correlation suggests, that MARS models produce very good fits to the underlying data, outperforming both, global prediction models, as well as local prediction models. This further strengthens our conjecture of the beneficial effect of data localization.

To test for the statistical significance of differences between correlation fits, we performed a Fisher's Z-test. We found that all differences were statistically significant at $p < 0.01$, except one: the correlation fit of the local model and the MARS model in the CHINA dataset (in both cases correlation fit was 0.89).

Overall, the results of our case study confirm that in the context of regression models for defect and effort prediction,

Table III: Summary of experimental results on prediction model 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 $p < 0.01$.

Global Models				
Dataset	**Error Sum**	**Median Error**	**Error Var**	**Cor**
Xalan 2.6	61.07	0.64	0.37	0.36
Lucene 2.4	49.72	1.15	**2.22***	0.71
CHINA	91,592.52	765.00	14,194,155.12	0.82
NasaCoc	48.75	3.26	31.63	0.95
Local Models				
Dataset	**Error Sum**	**Median Error**	**Error Var**	**Cor**
Xalan 2.6	57.35	0.52	0.57	0.50
Lucene 2.4	55.15	1.15	217.63	0.67
CHINA	83,420.53	552.85	19,159,955.36	0.85
NasaCoc	41.49	2.14	703.19	0.95
Global Models with Local Considerations				
Dataset	**Error Sum**	**Median Error**	**Error Var**	**Cor**
Xalan 2.6	**50.90***	**0.40***	**0.36***	**0.56***
Lucene 2.4	**43.61***	**0.94***	2.51	**0.72**
CHINA	**25,106.00***	**234.43***	**1,102,256.01***	**0.99***
NasaCoc	**26.95***	**1.63***	**25.46***	**0.97***

data localization leads to significantly better fits of prediction models to the 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 prediction models when applied on unseen data. Better performing prediction models are of great value to practitioners since they allow to take better informed decisions and actions. For this purpose, we have carried out experiments on each dataset as follows.

Approach. To evaluate the prediction performance of each of the three prediction modelling approaches, we follow the same steps of model building described in the previous section. We partition 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 prediction model. For local models, we first need to determine for each datapoint in the testing set the most similar cluster, and then use the corresponding local prediction model from that cluster to carry out the prediction.

We compare these predicted values to the actual values recorded in the testing data, and evaluate prediction performance based on the four different criteria. These criteria are discussed in detail below.

Evaluation Criteria and Findings. The results of our prediction experiments are summarized in Table III. Based on previous research in the area [14], we decided on the following evaluation criteria to compare the performance of prediction models.

1. Absolute sum of prediction error. The sum of all prediction errors $\sum abs(Y_{actual} - Y_{predicted})$ tells us how good the prediction model performed overall on the testing data. The closer the sum of prediction errors is to 0, the better the performance of the prediction model. This performance criterion is of importance to practitioners as it gives an indication of how good a prediction model captures reality.

Our results for this evaluation criterion are recorded in the first column of Table III. Overall, we observe that the local model outperforms the global prediction model in three out of four cases, Lucene 2.4 being the single exception. However, the global model with local considerations outperforms both the global model and the local model in all cases.

2. Median prediction error. This performance criterion tells us, how far off predictions were from the actual value on average. The closer this measure is to 0, the better the performance of the prediction model. In general, this criterion can be seen as the prediction accuracy of the model.

Figure 5 illustrates a comparison of the distributions of median prediction errors across all 10 runs on each individual dataset. Overall, we observe that the local model outperforms the global model in three out of four cases, and in one case (Lucene 2.4) shows comparable performance. At the same time, the global model with local considerations (MARS) demonstrates a significantly lower median prediction error distribution, which in three out of four cases has a shorter quartile range than both other approaches.

We performed Mann-Whitney-U tests to confirm that the differences in distributions of median prediction error are statistically significant. The only difference that was not deemed statistically significant based on this t-test is GLOBAL vs. LOCAL in the Lucene 2.4 dataset.

3. The variance of prediction errors. This performance criterion tells us, how wide the values of prediction errors are spread. In practice, we are seeking prediction models with a low spread, i.e., a consistent range can be better planned for, than unexpected large errors.

Overall, we observe that the global prediction model has less variance than the local model. However, the global model with local considerations outperforms the local model in 3 out of 4 times. Upon further analysis of prediction errors on a case-by-case basis, we found that a) the average predictions from global models are further off from the actual value, but this interval of prediction error is narrow; and b) even though the median error in local models is improved, these models can, on occasion, be very far off the actual value. As an example of this observation, we show the differences between actual values and predicted values in the CHINA dataset for each modelling approach in Figure 4.

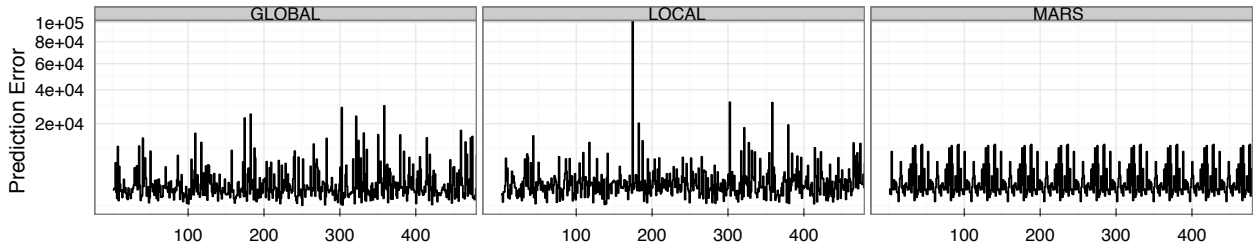


Figure 4: Differences between actual value and predicted value across 10 folds, CHINA dataset.

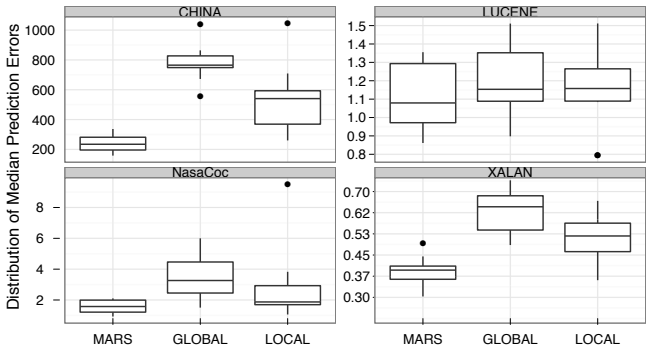


Figure 5: Distributions of median predictions errors for each dataset and modelling approach.

Our results show that both, the MARS model, and the global model perform consistent across all predictions. However, we observe 3 distinct spikes in the local model around predictions number 180, 300 and 360, whereas prediction error spikes of similar magnitude are absent from the predictions of global and MARS models. For practical purposes, this introduces a risk to practitioners who might come to rely on predictions that are close to reality, but encounter the rare, yet utterly wrong prediction.

4. The correlation between predicted and actual values. This performance criterion is of particular interest for defect prediction, as it measures the extent to which the prediction model is able to come up with a correct ranking of defective files, which in practical applications of defect prediction models is often used for resource allocation.

With respect to this evaluation criterion, our results are inconclusive. All three modelling approaches show similar performance in three out of the four datasets. In the case of the Xalan 2.6 defect prediction dataset, we observe that both the local model and the global model with local considerations produce a significantly better ranking than the global model.

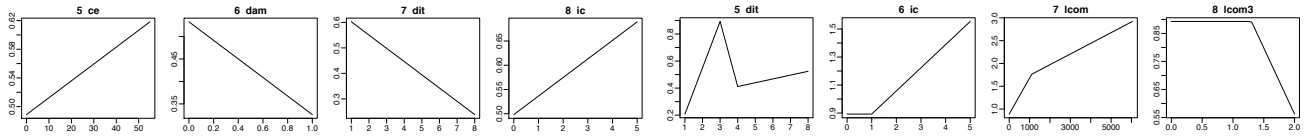
RQ3. What are the considerations in the use of local models over global models for practitioners?

One of the main applications of prediction 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 as prediction models is the use of response plots [12]. These plots describe, how the dependent variable (i.e., bugs and effort) reacts when we change the value of a single prediction variable (while at the same time keeping all other variables at their median values).

As an example, we show the response plots for four prediction variables found in the global model that was learned on the Xalan 2.6 dataset for the 9th fold 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 prediction 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 prediction models, the individual prediction models that are learned from each of the clusters differs both, in the prediction variables that were deemed significant for that portion of the data, as well as the overall trends.

For example, Figure 7 shows three response plots for local models learned from Cluster 1 and Cluster 6 in the 9th fold of our experiments on the Xalan 2.6 dataset. 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 predicted to lead to a decrease in bug-proneness.

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 his problem at hand, and b) for each cluster there might be many recommendations to choose from. 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 prediction variables in the MARS model learned on the Xalan 2.6 dataset is shown in Figure 6b. By design, the hinge functions of the MARS



(a) Part of a global Model learned on the Xalan 2.6 dataset

(b) Part of a Global model with local considerations learned on the Xalan 2.6 dataset

Figure 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 prediction variables at their median values.

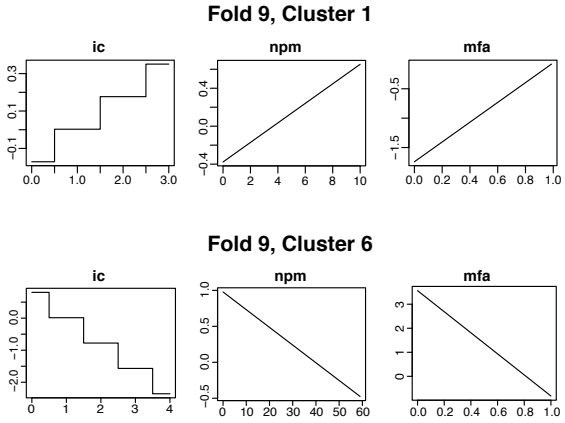


Figure 7: Example of contradicting trends in local models (Xalan 2.6, Cluster 1 and Cluster 6 in Fold 9).

model already partition 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.

V. Conclusions

In this study, we investigated the difference of three different types of prediction models. 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 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.

Think Locally

We evaluated each of the three modelling strategies in a case study on four different datasets, which have been used in prior research on the WHICH machine-learning algorithm [14]. 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 prediction 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. 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. For example, we have seen in the Xalan 2.6 defect prediction dataset that particular sets of classes are influenced differently by attributes such as inheritance, cohesion and complexity. Our findings reinforce the recommendations of Menzies et al. against the use of a "one-size-fits-all" approach, such as a global model, when trying to account for such localized effects.

Act Globally

When the goal is carrying out actual predictions, rather than understanding, local models show only small improvements over global models, with respect to prediction error and ranking. In particular, 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. [14] might run in linear time, we still have to learn a multitude of models, one for each cluster. One particular point that we have not addressed in our study is whether the choice of clustering algorithm influences the final performance of the local models. While our choice was to use a state-of-the-art model-based clustering algorithm that partitions data along dimensions of highest variability, future research may want to look deeper into the effect that different clustering approaches have on the performance of local models.

Surprisingly, we found that the relatively small increase in prediction performance of local models is offset by an increased error variance. While predictions from local models are close to the actual values most of the time, we observed the occasional very high errors. In other words, while global models are not as accurate as local models, their worst case scenarios are not as bad as we observe with local models. We want to note however that this finding

stand in conflict with the findings of Menzies et al., who observed the opposite: their clustering algorithm decreases error variance of local models both within the inter-quartile range and at the 100th percentile [14]. Future research may warrant further insight into this disparity.

Lastly, 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.

REPEATABILITY

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

REFERENCES

[1] H. Akaike, "A new look at the statistical model identification," *Automatic Control IEEE Transactions on*, vol. 19, no. 6, pp. 716–723, 1974.

[2] A. Andreou and E. Papatheocharous, "Software cost estimation using fuzzy decision trees," in *Automated Software Engineering, 2008. ASE 2008. 23rd IEEE/ACM International Conference on*, sept. 2008, pp. 371 –374.

[3] N. Attoh-okine, S. Mensah, M. Nawaiseh, and D. Hall, "Using multivariate adaptive regression splines (mars) in pavement roughness prediction," *Strategy*, 2001.

[4] H. Barkmann, R. Lincke, and W. Lowe, "Quantitative evaluation of software quality metrics in open-source projects," in *Proceedings of the 2009 International Conference on Advanced Information Networking and Applications Workshops*, ser. WAINA '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 1067–1072.

[5] N. Bettenburg and A. E. Hassan, "Studying the impact of social structures on software quality," in *Proceedings of the 2010 IEEE 18th International Conference on Program Comprehension*, ser. ICPC '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 124–133.

[6] M. Di Penta, "Nothing else matters: what predictive model should i use?" in *Proceedings of the 7th International Conference on Predictive Models in Software Engineering*, ser. Promise '11. New York, NY, USA: ACM, 2011, pp. 10:1–10:3.

[7] K. O. Elish and M. O. Elish, "Predicting defect-prone software modules using support vector machines," *J. Syst. Softw.*, vol. 81, pp. 649–660, May 2008.

[8] J. Fox, *Applied Regression analysis and generalized linear models*, 2nd ed. Los Angeles, London: Sage, 2008.

[9] C. Fraley, "Bayesian regularization for normal mixture estimation and model-based clustering," *Journal of Classification*, vol. 181, no. 2, pp. 155–181, 2007.

[10] C. Fraley and A. E. Raftery, "Mclust version 3 for r : Normal mixture modeling," *Office*, vol. Technical, no. 504, pp. 1–54, 2009.

[11] J. H. Friedman, "Multivariate Adaptive Regression Splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[12] F. E. Harrell, *Regression modeling strategies : with applications to linear models, logistic regression, and survival analysis*. Springer, 2001.

[13] Y. Kamei, S. Matsumoto, A. Monden, K.-i. Matsumoto, B. Adams, and A. E. Hassan, "Revisiting common bug prediction findings using effort-aware models," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ser. ICSM '10. IEEE Computer Society, 2010, pp. 1–10.

[14] T. Menzies, A. Butcher, A. Marcus, T. Zimmermann, and D. Cok, "Local vs global models for effort estimation and defect prediction," in *Proceedings of the 26th IEEE/ACM International Conference on Automated Software Engineering*, November 2011.

[15] A. Mockus and D. M. Weiss, "Predicting risk of software changes," *Bell Labs Technical Journal*, vol. 5, pp. 169–180, 2000.

[16] A. Mockus, D. M. Weiss, and P. Zhang, "Understanding and predicting effort in software projects," in *Proceedings of the 25th International Conference on Software Engineering*, ser. ICSE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 274–284.

[17] A. Mockus, P. Zhang, and P. L. Li, "Predictors of customer perceived software quality," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. New York, NY, USA: ACM, 2005, pp. 225–233.

[18] N. Nagappan and T. Ball, "Use of relative code churn measures to predict system defect density," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE '05. ACM, 2005, pp. 284–292.

[19] N. Nagappan, T. Ball, and A. Zeller, "Mining metrics to predict component failures," in *Proceedings of the 28th international conference on Software engineering*, ser. ICSE '06. New York, NY, USA: ACM, 2006, pp. 452–461.

[20] T. H. D. Nguyen, B. Adams, and A. E. Hassan, "Studying the impact of dependency network measures on software quality," in *Proceedings of the 2010 IEEE International Conference on Software Maintenance*. IEEE Computer Society, 2010, pp. 1–10.

[21] K.-M. Osei-Bryson and M. Ko, "Exploring the relationship between information technology investments and firm performance using regression splines analysis," *Information and Management*, vol. 42, no. 1, pp. 1 – 13, 2004.

[22] D. Posnett, V. Filkov, and P. Devanbu, "Ecological inference in empirical software engineering," *Automated Software Engineering, International Conference on*, vol. 0, pp. 362–371, 2011.

[23] A. E. Raftery and A. E. Raftery, "Bayesian model selection in social research adrian e. raftery sociological methodology, vol. 25. (1995), pp. 111-163." *Social Research*, vol. 25, no. 1995, pp. 111–163, 2007.

[24] G. Schwarz, "Estimating the dimension of a model," *Ann. Statist.*, vol. 6, no. 2, pp. 461–464, 1978.

[25] M. Shepperd, "A critique of cyclomatic complexity as a software metric," *Software Engineering Journal*, vol. 3, no. 2, pp. 30 –36, 1988.

[26] W. Thomson, *Popular Lectures*. Baron William Thomson Kelvin, 1894, vol. I, p. 73.

[27] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[28] T. P. York and L. J. Eaves, "Common disease analysis using multivariate adaptive regression splines (mars): Genetic analysis workshop 12 simulated sequence data." *Genetic Epidemiology*, vol. 21 Suppl 1, pp. S649–S654, 2001.

[29] T. Zimmermann, N. Nagappan, H. Gall, E. Giger, and B. Murphy, "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*, ser. ESEC/FSE '09. New York, NY, USA: ACM, 2009, pp. 91–100.

[30] T. Zimmermann, R. Premraj, and A. Zeller, "Predicting defects for eclipse," in *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, ser. PROMISE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 9–.