UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

IVAN ALEXANDRE PAIZ TIERNO

Assessment of Data-driven Bayesian Networks in Software Effort Prediction

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Prof. Dr. Daltro José Nunes
Advisor

Porto Alegre, March 2013

CIP – CATALOGING-IN-PUBLICATION

Tierno, Ivan Alexandre Paiz

Assessment of Data-driven Bayesian Networks in Software Effort Prediction / Ivan Alexandre Paiz Tierno. – Porto Alegre: PPGC da UFRGS, 2013.

63 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2013. Advisor: Daltro José Nunes.

1. Software effort prediction. 2. Bayesian networks. 3. Machine learning. 4. Data mining. I. Nunes, Daltro José. II. Título.

CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

ANN	Artificial Neural Networks
BNs	Bayesian Networks
CBR	Case Based Reasoning
CPT	Conditional Probability Table
DAG	Direct Acyclic Graph
FSS	Feature Subset Selection
MAR	Mean Absolute Residuals
MBRE	Mean Balanced Relative Error
MMRE	Mean Magnitude of Relative Error
MdMRE	Median Magnitude of Relative Error
NPT	Node Probability Table
OLS	Ordinary Least Squares

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Software prediction unveils itself as a difficult but important task which can aid the manager on decision making, possibly allowing for time and resources sparing, achieving higher software quality among other benefits. One of the approaches set forth to perform this task has been the application of machine learning techniques. One of these techniques are Bayesian Networks, which have been promoted for software projects management due to their special features. However, the pre-processing procedures related to their application remain mostly neglected in this field. In this context, this study presents an assessment of automatic Bayesian Networks (i.e., Bayesian Networks solely based on data) on three public data sets and brings forward a discussion on data pre-processing procedures and the validation approach. We carried out a comparison of automatic Bayesian Networks against mean and median baseline models and also against ordinary least squares regression with a logarithmic transformation, which has been recently deemed in a comprehensive study as a top performer with regard to accuracy. The results obtained through careful validation procedures support that automatic Bayesian Networks can be competitive against other techniques, but still need improvements in order to catch up with linear regression models accuracy-wise. Some current limitations of Bayesian Networks are highlighted and possible improvements are discussed. Furthermore, this study provides some guidelines on the exploration of data. These guidelines can be useful to any Bayesian Networks that use data for model learning. Finally, this study also confirms the potential benefits of feature selection in software effort prediction.

Keywords: Software effort prediction, bayesian networks, machine learning, data mining.

1 INTRODUCTION

In recent decades, the unremitting, relentless society's strive towards automation and the progress on information systems technologies made software virtually ubiquitous. With the ever increasing demand for software and the ensuing industry growth, software projects kept growing in size and importance. Nevertheless, the task of managing and organizing them did not become any easier or predictable. Related hereto is the difficulty of predicting their outcomes with regard to budget, quality, etc. The task of software prediction remains a far from tamed problem.

Accurate software predictions can provide significant advantages in project planning and are essential for effective project management being strongly linked to the success of software projects. Underestimating the effort can cause delays, degrade software quality and bring about increased costs and dissatisfied customers. On the other hand, overestimating the project's effort can lose a contract bid or waste resources that could be allocated elsewhere. Although the primary objective of software effort prediction is budgeting, there are also other important objectives. Boehm, Abts and Chulani (2000) mention tradeoff and risk analysis, project planning and control and software improvement investment analysis.

Software prediction models strive to perform forecasts for new projects based on knowledge, be it obtained from historical data, from experts or a hybrid approach that integrates both forms. The outset of this research field took place in the sixties and the first proposed approaches relied on expert knowledge. Soon after, the first formal models were proposed. One of the longest established approaches are parametric models which have COCOMO (BOEHM, 1981) and its updates as their most recognized representative. These models are generally based on a formula that relates the size of a task to the effort needed to perform it (SHEPPERD, 2007). A detailed review on these models and the pioneering researches on software prediction can be found in Boehm, Abts and Chulani (2000).

In case of knowledge obtained from data, the models make use of software projects data sets. These data sets contain information on projects that already finished. The prediction model takes into account the characteristics of the new project, i.e., the project we want to predict, and makes a prediction based on knowledge obtained from the projects stored in the data set. The underlying assumption is that the past patterns will hold for the new project.

Throughout this study, we use 'software prediction' as a synonym for 'software estimation'. Both forms are common in the literature. Furthermore, the terminology 'software cost estimation' is often used interchangeably with 'software effort estimation' because these two variables are very closely related to each other. Moreover, there is a further distinction between development effort estimation and maintenance effort estima-

tion. In this study we used development effort estimation data.

The variable to be predicted by the prediction model is known as dependent variable or also response variable. The attributes or variables used by the model to predict the response variable are referred to as explanatory or independent variables. The dependent variables typically found in literature can be classified as cost or quality related (FENTON; RADLINSKI, 2009).

Since the nineties, researchers began applying machine learning techniques for software prediction (FINNIE; WITTIG; DESHARNAIS, 1997) (SHEPPERD; SCHOFIELD, 1997) (FENTON; NEIL, 1999). Ever since, studies on machine learning techniques for software prediction have grown more and more common. Currently this is visibly a thriving trend with many empirical studies being published regularly and comprising a very active research field. In a systematic review, Wen et al. (2012) identified eight machine learning techniques employed in software prediction including CART (a type of decision tree) (FINNIE; WITTIG; DESHARNAIS, 1997), Case-based Reasoning (CBR) (SHEPPERD; SCHOFIELD, 1997), Artificial Neural Networks, Genetic algorithms, Support Vector Regression among others. CBR, Artificial Neural Networks and Decisions Trees were considered by Wen et al. (2012) the most popular machine learning techniques in software development effort prediction research.

One of these machine learning techniques are Bayesian Networks(henceforth BNs), which is the technique we assess in this study. BNs were initially proposed and are generally more common in software quality prediction (FENTON; NEIL, 1999). Since then there has been a steady increase of efforts towards BNs in software effort prediction (RADLINSKI, 2010) and in software projects management in general (SOUZA et al., 2011). Wen et al. (2012) ranked BNs fourth in popularity in this research field among the machine learning techniques. This technique has some distinguishing features that make it look suitable to deal with the uncertainties prevalent in this field. BNs will be discussed in more detail in the next chapter.

In spite of the increasing volume of research on machine learning models, these techniques are not yet popular in industry. Surveys have shown that expert judgment is still the predominant prediction method in industry, e.g., (JØRGENSEN, 2004) (JØRGENSEN; SHEPPERD, 2007), even though most of the research efforts focus on more formal models (JØRGENSEN; SHEPPERD, 2007). It is not clear whether expert-based or data-based predictions are most accurate. Jørgensen (2004) points out that for some situations expert estimates are likely to be more accurate, whereas in others the use of models may reduce large situational or human biases, i.e., the use of formal models can help keeping human's predictions in check. Several studies recommend the combination of both approaches to increase estimates reliability.

This research area has not been without its hindrances and difficulties. In spite of the large number of empirical studies there are conflicting results and conclusions instability (KORTE; PORT, 2008) (MAIR; SHEPPERD, 2005) (MENZIES et al., 2010). Shepperd and Macdonell (2012) state that 'empirical evaluation has not led to consistent or easy to interpret results. This matters because it is hard to know what advice to offer practitioners'. One example of this are studies comparing CBR to regression models. A systematic review (MAIR; SHEPPERD, 2005) on this subject identified seven studies favouring regression, nine studies favouring CBR and four with inconclusive results. In case of the studies on CBR and regression, a problem identified was that some studies that favoured CBR were not applying regression in line with the best statistical practices, thus providing an 'optimistic' assessment on CBR. In a very popular paper, Shepperd and Schofield

(1997) proposed a CBR method reporting that it overcomes regression, but in other studies these results were not confirmed. Shepperd and Schofield (1997) did not split the data into training and test sets and did not perform a logarithmic transformation on the data, which makes OLS regression perform poorly. This was discussed by the very author in Shepperd and Macdonell (2012) wherein the authors discuss the situation in the field and propose ways to address these problems. There are many other examples of contradictions like this in comparisons among different machine learning and statistical techniques. Errors in procedures have been widespread in this field as discussed in, e.g., Shepperd and Macdonell (2012) and Kitchenham and Mendes (2009). The latter study points out mistakes in the application of regression models. Myrtveit, Stensrud and Shepperd (2005) discuss reasons for the unreliability of conclusions in detail, chiefly focusing on validation and measuring, and concluded that more reliable research procedures are necessary. Several other researchers have made suggestions about the validation of results in comparative studies, e.g., Kitchenham et al. (2001), Mendes and Mosley (2008) and Shepperd and Macdonell (2012).

In this context, there have been efforts to find out the underlying reasons for these inconsistencies. Besides errors in procedures, part of these inconsistencies stem from differences in the experimental designs employed and the data sets used. An identified problem is that the ranking of techniques is not stable across different data sets. Thus, results obtained from experiments on a single data set cannot be generalized and categorical conclusions are not possible. To counter this problem the use of multiple data sets has been recommended. Still with regard to the data sets, some studies have investigated the effect of the type of data sets employed, i.e., investigations on whether cross-company or within-company data sets are more effective for building software prediction models. Some studies did not identify much difference between the two kinds of data sets and others did find within-company data to provide better results. Kitchenham, Mendes and Travassos (2007) present a systematic review on this subject concluding that it was not possible to bring forward a definite conclusion about which one is better due to differences in empirical procedures of the studies on the subject. They observe that studies that used small data sets and leave-one-out cross validation did find significant advantage in using within-company data. Furthermore they highlight that it is clear some companies would be ill-served by cross-company data whereas others would benefit.

An observable tendency in comparative studies of this field is publication bias. Wen et al. (2012) observed that researchers frequently tended to favour their proposed technique.

Another source of problems that was promptly investigated is the subject of accuracy metrics. Several studies, e.g., Foss et al. (2003), Myrtveit, Stensrud and Shepperd (2005), Korte and Port (2008), identified this as a reason for unreliability of comparative studies' results. Researchers have found that different accuracy metrics can bring about rank reversal problems, i.e., a change in the order of ranked techniques. Particularly, a metric that has been criticized is the once very popular MMRE due to its susceptibility to outliers and tendency to favour underestimates. A more detailed analysis of the metrics will be exposed in the next chapter.

A significant barrier for analysis of findings and replication of experiments has been the lack of publicly available data sets since the employment of proprietary data sets inhibits the replication of experiments and confirmation of results. The PROMISE repository (MENZIES et al., 2012) is an initiative that attempts to counter to some extent the lack of transparency that pervades this research field. Data sets are made available allowing for replication and scrutiny of findings with the intent of improving research efforts

and to stir up analyses and discussion.

A recurrent observation in recent times is that the vast majority of software prediction studies focusses on prediction models (SHEPPERD; MACDONELL, 2012) and little attention has been addressed to data sets, pre-processors and validation methods. Studies like Shepperd (2007) point out the importance of investigating data quality. A recent trend has been the investigation of data analysis and preparation prior to model building. This is quite an useful effort, for it clarifies and draws attention to this important but sometimes neglected topic of data preparation. In recent years, discussions in the research community about these issues have grown more common. This has been a source of many problems in software prediction research. Some studies took up specifically the investigations on data pre-processing techniques on data sets prior to performing software prediction, like Chen et al. (2005) that investigated feature selection for prediction models and Liu et al. (2008) that propose a data pre-processing framework.

In this context, this study strives to assess the employment of data-driven BNs in software effort prediction, including related analyses on data pre-processing and on how to improve their accuracy, discussing their current limitations and possibilities of improvements. Even though BNs have been promoted due to their special features, their application in this field remains relatively limited due to some practical difficulties that are rarely discussed. The investigation of data-driven BNs matters because even if this might not become the best way to apply them, the optimization of data exploration is an important direction of development for this technique. By finding ways to optimize the exploration of data there can be benefits to any BNs that use data.

This thesis is organized as follows. In chapter 2 we present background information for our experiments, including an overview of Bayesian Networks and feature subset selection, a discussion on validation methods and accuracy metrics, a numerical conversion method for BNs and also we mention some closely related studies. In chapter 3 we bring forward the experiment setup and lay out our empirical procedures and decisions. In chapter 4 we analyze the results and make comparisons with other studies and finally put forth the conclusions in the last chapter.

2 BACKGROUND

In this chapter we set forth some background on BNs and on the empirical procedures employed in software effort prediction. We make a brief overview of Bayesian Networks, the technique in focus, and feature subset selection. We discuss the data preparation and pre-processing methods necessary to carry out the experiments and the conversion method necessary to compare BNs predictions to OLS regression. It is important to bear in mind throughout that we are comparing a classifier (BNs) to a regression technique (OLS regression). The Bayesian classifiers have as an outcome a range of probability distributions for each class, whereas OLS regression predicts a number. Therefore the conversion method mentioned is necessary to make the comparison possible. We also discuss the validation and accuracy evaluation methods and the conclusions that can be drawn from them. Finally, we contextualize this work with some closely related studies.

This study focuses on data based predictions which is basically a data mining task. Fig. 2.1 illustrates the prediction process.

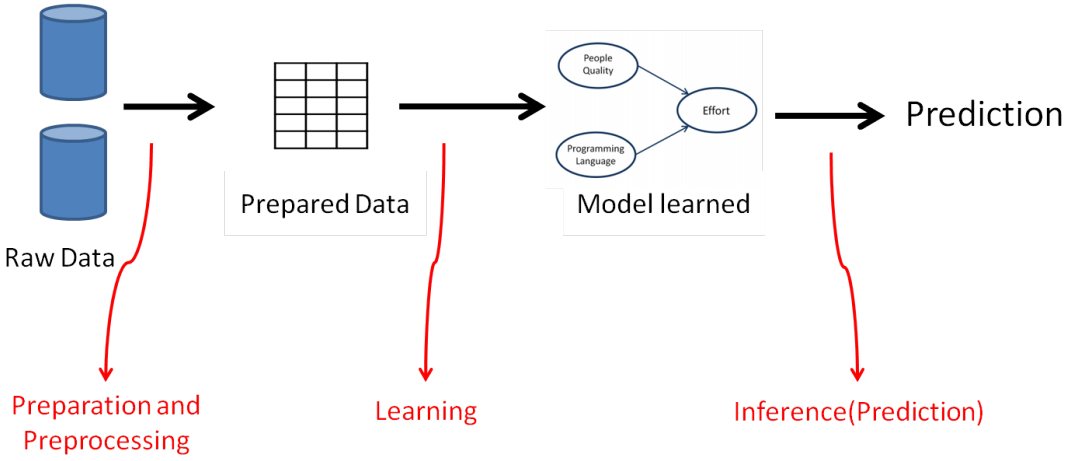
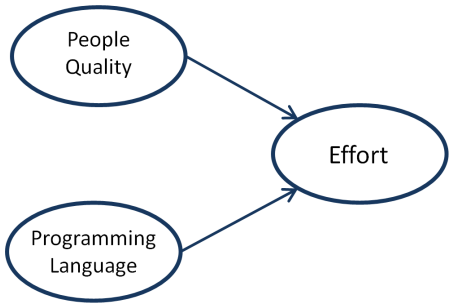


Figure 2.1: Prediction process.

2.1 Bayesian Networks

BNs (TAN; STEINBACH; KUMAR, 2005) (WITTEN; FRANK; HALL, 2011) are a modeling technique which boasts some distinguishing characteristics. A striking feature of this modeling approach is the possibility, through application of probability theory, to model uncertainty and subjectivity. The probability distributions allow for the integration of objective evaluations, learned from data, with subjective evaluations defined by experts.



Programming Language	
Language1	0,4
Language2	0,6

People Quality	
Beginner	0,2
Average	0,7
Expert	0,1

Effort				
People Quality	P.Language	Low	Medium	High
Beginner	Language1	0,1	0,4	0,5
	Language2	0,2	0,3	0,5
Average	Language1	0,3	0,4	0,3
	Language2	0,3	0,5	0,2
Expert	Language1	0,4	0,55	0,05
	Language2	0,5	0,4	0,1

Figure 2.2: A simple Bayesian Network.

Furthermore this allows the model to output several possible outcomes with varying degrees of certainty, unlike deterministic models like linear regression which simply output a single possible outcome, i.e., a numeric value.

Another highlighting characteristic of BNs is the possibility to carry out what-if analyses and tradeoff assessment. This can be done by feeding the model with different input values, referred to as evidences, and observing the effects on the outcome and on the rest of the network. Some interesting possibilities arise with this. Through this feature, BNs can stand as an experimenting platform, allowing for the investigation of several scenarios and acquainting the software manager with the software process sensitive factors and the importance of specific factors. Furthermore, the converse is also possible. Desired outcomes can be entered and the models provide the range of inputs required to achieve those outcomes.

BNs comprise a qualitative part, i.e., the graph structure that models the dependencies among a set of variables, and a quantitative part made up of node probability tables (NPT's) which contain the probability distributions for each node. The graph structure is a directed acyclic graph (DAG) encoding the dependencies among the variables. The nodes represent the relevant variables or factors in the domain being modeled, and each directed arc depicts the dependencies among these factors which can be causality relationships. The NPT's contain the prior probabilities (in case the variable has no parents) or conditional probabilities (in case the variable has one or more parents). The conditional probabilities define the state of a variable given the combination of states of its parents. With the definition of these probabilities during the training phase a test record can later be classified. These components are illustrated on a simple example in Fig. 2.2. In this figure, three variables are modelled. The dependent variable is *Effort* and the independent variables are *People Quality* and *Programming Language*. The state of the dependent variable varies according to the state of the independent variables. This is shown in the probability table of the dependent variable. The probability distributions are obtained in the training phase.

BNs are named after and founded on the Bayes theorem, which models the relationship between two variables. This theorem establishes a relationship between the prior and posterior probabilities. Before introducing the theorem we review some concepts. The **prior probability** (e.g., P(Y)) is simply the probability of a variable taking on some specified value regardless of other variables, that is, it is the default probability set before any evidence is known. The **joint probability** P(X, Y) is the probability of variable X taking on a determined value and variable Y taking on another determined value simul-

taneously. The **conditional probability** P(Y = y | X = x) is the probability of Y taking on value y given that X takes on value x (the right-hand side is usually omitted for convenience, i.e., it reads P(Y | X)). **Posterior probability** is the probability of a variable taking on some value, given some evidence, i.e., it is the updated probability after new information is inserted into the network. The Bayes theorem allows us to express the posterior probability P(Y|X) in terms of the prior probability P(Y), the class-conditional probability P(X|Y), and the evidence, P(X):

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.1}$$

After the model building takes place, the model is set with the prior probabilities. When a variable of the model is known, this information can be entered into the network. Such information is referred to as *evidence*. By means of the Bayes theorem the probabilities of the network can be updated according to new evidences inserted into the network. This updating process is known as *propagation*. This process allows for the experimenting possibilities mentioned in the first paragraphs of this section, namely performance of what-if scenarios and tradeoff analysis.

BNs can be modeled fully based on data, through a hybrid approach, i.e., integrating data modeling and experts knowledge or fully expert-based. When the BNs are learnt from data the learning algorithm strives to identify the dependencies among the variables and thus making up the network structure, i.e., the DAG. The algorithm will identify a model that best fits the relationship between the attribute set and the response variable on the input data (training data). Thereafter the probability distributions are learned for every combination of variables. This happens during the so called training or learning phase.

Besides learning from data, this modeling technique allows the experts to get involved with the construction of the model. Due to the possibility of incorporating experts knowledge, BNs can work even when data availability is limited, which is a common constraint in software companies. Expert knowledge can be coded by means of subjective or qualitative variables and also by defining the network topology. The expert can get involved with model building either in a limited way, by establishing restrictions and enforcing particular dependencies or more comprehensively by defining the graph structure and the variables and their probability distributions.

Fenton and Neil (1999) are among the pioneers on the employment of BNs in software prediction. The authors promoted BNs for software quality prediction explaining the advantages of BNs over regression based models. Other researchers, e.g., Stamelos et al. (2003). Pendharkar, Subramanian and Rodgers (2005), Mendes and Mosley (2008), employed BNs for software effort prediction. Considering the uncertainties inherent to software projects, BNs appear to be quite suitable and as shown in Radlinski (2010), a growing trend.

The BNs found in this research field most frequently consist of discrete variables. The tool used in this study currently does not support continuous variables. Although some tools offer support to continuous variables, this support has limitations, e.g., imposing restrictions in the relationships among the variables or making assumptions about the distributions of the continuous variables. Radlinski (2010) points out the difficulties in formalizing and eliciting expert knowledge and lack of usability towards this purpose in current tools. Moreover, the tools currently demand a background in artificial intelligence and statistics which is an obstacle for industry adoption. There are progresses concerning continuous variables in machine learning research and there are also constant developments in the BNs tools, so these limitations could be overcome in the future.

2.2 Feature Subset Selection

An important data pre-processing step is the selection of variables or feature selection. This technique attempts to identify a set of highly predictive attributes and discards attributes unrelated to the response variable or that do not help the model to predict more accurately. The goals are three-fold: improve prediction performance, provide faster and more cost-effective predictors and provide a better understanding of the underlying process that generated the data (GUYON; ELISSEEFF, 2003).

Often, data sets contain variables of little or no importance to the prediction task at hand. There can also be redundant variables which are basically duplicated information and also cannot provide additional knowledge to the model. These variables can only confuse the classifier algorithm and reduce the model's accuracy. In part, the negative effects on accuracy can be ascribed to the phenomenon known in the data mining field as the "curse of the dimensionality" (TAN; STEINBACH; KUMAR, 2005). This refers to the situation in which large numbers of variables make the data too sparse in the space that it occupies, making it more difficult for the algorithm to find patterns and similarities in data entries due to the increase of dissimilarities among the entries. Each new independent variable added to a model adds another dimension to the space and makes the finding of patterns more difficult and unreliable.

The removal of such variables can generate simpler and more accurate models. Some of these variables can be promptly identified by commonsense, e.g., the ID of a software project is irrelevant to predict its cost, and by domain knowledge, e.g., some sizing and duration variables are usually removed in the software prediction field due to not being known at the beginning of the project. Thus, this removal reflects the real scenario under which predictions happen. For instance, the software manager does not know how many lines of code the final product will have in the beginning of the project, so the model should not use this information during training. Other variables can be identified by the feature selection algorithm which will strive to select the variables with higher predictive value.

The feature selection process comprises four parts: a search method to deliver candidate subsets, a measure for evaluating the subset, a stopping criterion and a validation procedure (DASH; LIU, 1997). Conceptually, the search method looks for the optimal feature subset over all possible subsets. An exhaustive search is often unfeasible however, because the size of the search space grows exponentially on the number of features. So the search typically ends when the procedure meets a stopping criterion. The stopping criterion can be based on a combination of conditions like the number of iterations, the size of the subset and a threshold on the score of the subset according to the evaluation measure. The best subset according to the evaluation measure used is selected and validated with the classifier algorithm. This process is illustrated in Fig. 2.3 taken from Dash and Liu (1997).

Feature subset selection techniques can be generally classified as filters or wrappers (HALL; HOLMES, 2003) (KOHAVI; JOHN, 1997). Filters perform the selection using a scoring metric that attempts to predict how effective the candidate subset will be for the classifier. Wrappers on the other hand, evaluate the candidate subsets by running the target classifier. The classifier is invoked for every feature subset considered by the search. So, instead of attempting to predict the effectiveness of the feature subset, wrappers actually use the classification algorithm to verify its effectiveness. A natural drawback is that wrappers are more demanding computationally since the computing involved in invoking the classifier so many times is much heavier than using a simpler evaluation measure like

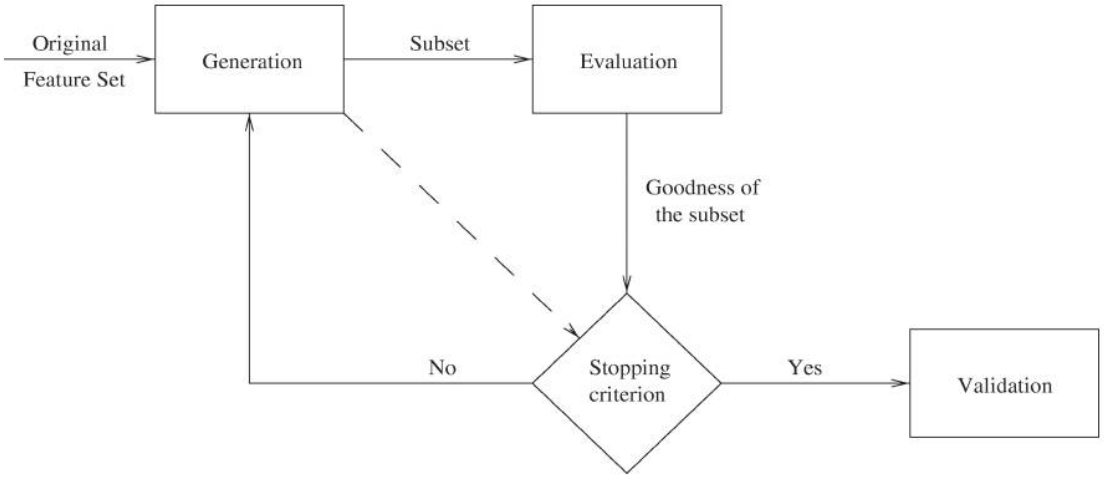


Figure 2.3: Feature selection process.

filters do.

Feature subset selection has been employed in software prediction studies with positive results. Chen et al. (2005) claim to be the first to employ feature selection in software prediction. Liu et al. (2008) incorporate feature selection in their proposed pre-processing framework and also confirm the benefits of this technique. The results in Radlinski and Hoffmann (2010) consistently show improvements across a variety of classifiers when feature selection was undertaken. Finally, Dejaeger et al. (2012) also report as one of their main findings and a confirmation of previous results, the increase in accuracy obtained by performing feature selection.

2.3 Validation

In order to assess a model's accuracy, some method of validation is used. These methods strive to provide an unbiased measure of the model's accuracy generally by separating the data used to build the model from the data used to assess accuracy, i.e., the test data set. The predictions inferred by the model are compared against the actual values of the response variable which are stored in the test data set. Common validation methods are holdout, random sub-sampling, k-fold cross validation and leave-one-out cross validation (TAN; STEINBACH; KUMAR, 2005).

The holdout validation method splits the data into two sets. One of them is the training set and the other is the test set. A common split of the data is 2/3 of the records for the training set and 1/3 of the records for the test set. In this method the model is built using the training set to feed the model. The idea underlying this split is to avoid using the training data to evaluate accuracy since the model used this very data for model building. Instead, the test set, which contains records unseen by the model, is used to measure the accuracy of the model more reliably.

The k-fold cross validation method splits the data into k partitions. The learning algorithm uses *k-1* partitions for model learning and 1 partition for model testing in each round and the accuracy is averaged out in the end of the process. The test set rotates through all partitions, so that in the end every record has been used for training and testing. This is illustrated in Fig. 2.4 taken from Liu et al. (2008). Leave-one-out cross validation is a special case of k-fold cross validation in which k is equal to the number of records. Both

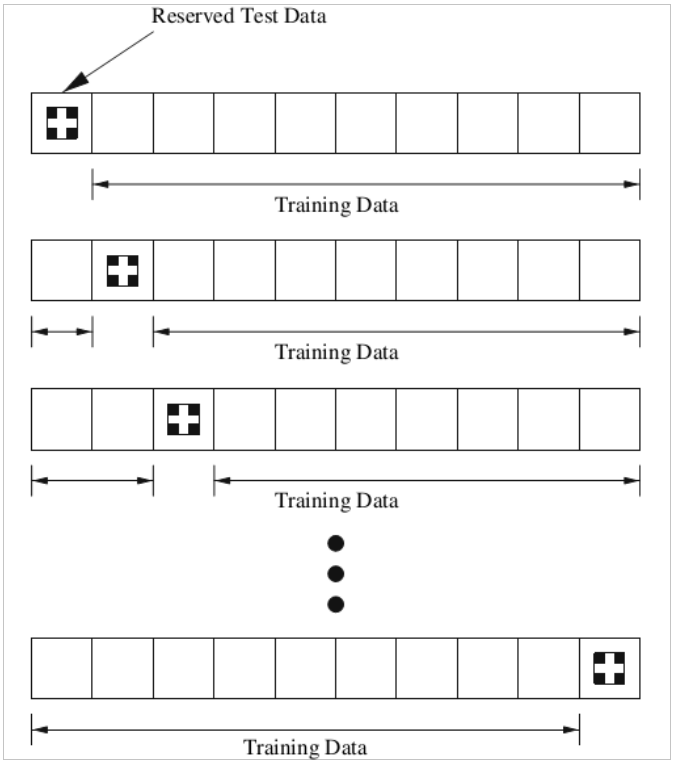


Figure 2.4: K-fold cross-validation.

these methods address many of the shortcomings of the holdout method. It is not yet a settled matter whether leave-one-out or k-fold cross validation is better.

We have observed in our experiments that the validation method can substantially affect the results in software effort prediction. Keeping all the remaining conditions the same, significantly different results can occur depending on the method chosen, sometimes bringing about the so called ranking reversal problem, i.e., changing the ranking of the techniques. We noticed in particular that the holdout method can provide different results in comparison to k-fold cross validation or leave-one-out cross validation. The holdout method is more prone to biased results and therefore considered less reliable for comparative studies. This can contribute to the conflicts and conclusions instability prevalent in the field.

One way to counter this bias is to employ random sub-sampling. This method consists in repeating the hold-out split resampling the training and test sets for a pre-specified number of times which allows constructing a confidence interval indicating the reliability of the results. Dejaeger et al. (2012) employ this validation method for the larger data sets.

Other methods frequently employed are k-fold and leave-one-out cross validation. Recently, some researchers have shown a preference for leave-one-out cross validation but this is not a consensus. Leave-one-out incurs higher variance in the models accuracy because the test sets contain only one record. We highlight that in opposition to the studies that promote this validation method, it is stated in the WEKA manual that k-fold cross validation is preferred over leave-one-out due to stratification possibilities which are implemented in this tool. The test data set in leave-one-out cross validation cannot be stratified since it contains only one record. We observed in our experiments that leave-one-out cross validation results are similar to k-fold cross validation's and since we

consider that this advantage of k-fold outweighs the claimed advantages of leave-one-out, we decided to perform 10-fold cross-validation in our experiments. Other studies in the field like Radlinski and Hoffmann (2010) have also employed this method.

We consider our validation procedures to be robust and a distinguishing feature compared to other studies assessing BNs' numeric accuracy, like Mendes and Mosley (2008) and Pendharkar, Subramanian and Rodger (2005). Studies on BNs have frequently used the holdout method to assess accuracy. We are not aware of other studies employing k-fold or leave-one-out cross-validation when converting the BNs predictions. This numeric conversion that will be discussed in section 2.5 makes the employment of more sophisticated validation methods like this somewhat cumbersome, since it is not automated in the tools.

2.4 Evaluation Metrics

This has been another controversial topic. There is no consensus on what is the most reliable metric (MYRTVEIT; STENSRUD; SHEPPERD, 2005). The *de facto* standard metric some years ago used to be MMRE (FOSS et al., 2003), but due to some flaws it lost popularity. MMRE, like other numerical metrics used in this study, is based on the magnitude of relative error. MRE is a measure of the relative error of the actual effort e_i against the predicted effort $\hat{e}_i$.

$$MRE_i = \frac{|e_i - \hat{e}_i|}{e_i} \tag{2.2}$$

MMRE measures the mean of all the predictions' MREs. This metric has not passed without criticism (FOSS et al., 2003) (KORTE; PORT, 2008), for it is highly affected by outliers and it favours models that underestimate. Miyazaki et al. (1991) were the first to observe this. MMRE is biased towards underestimates because the magnitude of the error is unbounded when overestimating and limited to at most 1 (or 100%) when underestimating. This is illustrated in a didactic example taken from Shepperd and Macdonell (2012) shown in Table 2.1. The table shows two predictions. Both have the same absolute residuals (90) and are relative to the same actual value (100). However, because MRE is a ratio measure, the magnitude of overestimates is potentially much larger. For project 1, which is an overestimate, the MRE is 900% whereas for project 2, which is an underestimate, the MRE is only 90%. This entails that models biased towards underestimates will tend to have smaller MREs overall, therefore performing better according to MRE based metrics. Even though this bias is present in all MRE based metrics it is specially so in MMRE.

Table 2.1: MMRE example

	y_i	$\hat{y}_i$	Residual $y_i - \hat{y}_i$	Absolute residual $\lvert y_i - \hat{y}_i \rvert$	MRE $\times$ 100
Project 1	10	100	-90	90	900
Project 2	100	10	90	90	90

Nevertheless, we will keep it in this study with reservations. In our opinion this metric is not worthless because it is capable of conveying information that the other two MRE based metrics do not. It can bring out which models are more prone to be occasionally

very inaccurate, since it is more sensitive to outliers, i.e., the wildly inaccurate predictions. Both MdMRE and Pred conceal this information. Furthermore, the MMRE has been widely employed in software prediction studies. Keeping this metric's flaw in mind, one can check the performance under the other metrics, and specially MAR and MBRE (explained below) which counter precisely this bias of MMRE towards underestimates.

MdMRE is the median of the MRE's. It smoothes out MMRE's bias, for it is more robust to outliers. Amply inaccurate predictions do not bear on MdMRE like on MMRE. So, on the one hand it shows which models are *generally* more accurate, but on the other hand it conceals which models can be occasionally very inaccurate. This effect is even more pronounced on Pred metric because it ignores completely the predictions with large errors. Pred measures how frequently predictions fall within a specified percentage of the actual effort, e.g., Pred_{25} tells us how often the predicted effort is within 25% of the project's actual effort (25 is a common parameter value for this metric). Therefore, this metric ignores the predictions whose errors are in excess of 25% magnitude, i.e., it does not matter for this metric if the error is 30% or 200% (assuming Pred_{25}). This is a limitation which we criticize about these metrics. Obtaining a model whose predictions rarely lie too far from the actual value is certainly advantageous. This is a desirable quality in a model and these metrics overlook this aspect.

Several studies proposed new metrics discussing their characteristics. But none of these metrics was widely adopted in the research field. MdMRE and Pred appear to be still the most popular. Foss et al. (2003) concluded that every metric studied has flaws or limitations and that it is unlikely that a single entirely reliable metric will be found. So, the use of complementary metrics is recommended.

Miyazaki et al. (1991), being the first to observe MRE's bias towards underestimates, proposed MBRE (Mean Balanced Relative Error). This metric addresses this flaw because it makes the relative error unbounded towards both underestimates and overestimates. By making the ratio relative to the lowest value (between actual and predicted values) the bias of MRE based metrics is eliminated, therefore avoiding favouring models that underestimate.

$$BRE_i = \frac{|e_i - \hat{e}_i|}{minimum(e_i, \hat{e}_i)} \tag{2.3}$$

However, it has a flaw in that it does not account for negative predictions. Linear regression models can at times predict a negative number and therefore distort a bit the results under MBRE. Kitchenham et al. (2001) propose the use of the absolute residuals as another alternative to bypass these problems of MRE based metrics. MAR (Mean Absolute Residuals) being an absolute measure also avoids this bias of ratio metrics like MRE. MAR has the disadvantage of not being comparable across different data sets.

$$MAR_i = \frac{\sum_i^n |e_i - \hat{e}_i|}{n} \tag{2.4}$$

We consider our selection of metrics to be robust with MAR and MBRE being complementary to the MRE based metrics and making the evaluation more reliable. Higher accuracy in MMRE, MdMRE, MAR and MBRE is inferred from lower values, whereas for Pred metric, the higher the value the more accurate the model. In our result tables, the results under MMRE, MdMRE, Pred and MBRE are multiplied by 100 to keep them in a percentage perspective, e.g., 0.253 turns into 25.3.

2.5 Classifiers vs Regression - Conversion of BNs predictions

Generally, comparative studies assess either classifiers or regression models. Not many researchers have endeavoured to compare classifiers to regression models. Although both these classes of models can be employed for software prediction, the representation difference of the dependent variable entails that comparisons between these models be somewhat infrequent. This is, in our opinion, an interesting undertaking, for it allows the comparison of techniques that are often in separated worlds.

Mendes and Mosley (2008) and Pendharkar, Subramanian and Rodger (2005) have done this, comparing BNs to regression techniques. In order to compare BNs' results to linear regression we used a variation of the conversion method first proposed by Pendharkar, Subramanian and Rodger (2005) in which the numerical prediction is the sum of the multiplication of each class' mean by its respective class probability after the probabilities are normalized so that their sum equals one. Instead of using the mean however, we used the median. Each class' median value *Md* is multiplied by its respective normalized class probability ρ, output in the probability distributions of the BN's predictions. See formula below.

$$Effort = \rho_{class1}Md_{class1} + \rho_{class2}Md_{class2} + ... + \rho_{classN}Md_{classN}. \quad (2.5)$$

Like the aforementioned studies, we used the mean in a preliminary study (TIERNO; NUNES, 2012). We report here accuracy improvements under MdMRE and Pred metric and significant and consistent improvements in MMRE and MAR results when using the median for the numerical conversion. This modification increased accuracy and lessened the amount of outliers, i.e., wildly inaccurate predictions. This happens because the mean of each class is more affected by outliers than the median. These data sets are positively skewed, therefore each class's mean value (and specially the highest effort class) will be closer to where the outliers are and farther from the majority of the data, pushing the numerical conversion of the output towards higher values. Therefore, when skewness is present the median is a more faithful and accurate representative of the data which makes up each class. An evidence supporting this reasoning is that the larger improvements were achieved on Maxwell data set which is the more skewed one.

Table 2.2: Numerical conversion for BNs on Desharnais data set.

Prediction System	MMRE	MdMRE	Pred	MAR
Bayesian Networks (mean)	70	35.65	38.27	2556.98
Bayesian Networks (median)	57.23	32.66	33.33	2153.52

Table 2.3: Numerical conversion for BNs with FSS on Desharnais data set.

Prediction System	MMRE	MdMRE	Pred	MAR
Bayesian Networks + FSS (mean)	68.94	35.49	39.5	2509.52
Bayesian Networks + FSS (median)	56.18	34.16	39.5	2133.84

The effectiveness of this conversion method can be seen in the tables. Table 2.2 shows the results for BNs on Desharnais data set. Table 2.3 shows the results for BNs with the employment of feature subset selection on the same data set. Tables 2.4 and 2.5 show

the results on Maxwell data set. These tables show the results comparing the conversion with the mean against the conversion with the median. BNs with and without feature selection are different prediction systems. So, the effect of the conversion method can be assessed by comparing the results *on the same prediction system*. Comparisons between the two prediction systems do not belong in this section and will be discussed in the results chapter. Here we are discussing only the improvements provided by this adaptation to the method proposed in Pendharkar, Subramanian and Rodger (2005).

Table 2.4: Numerical conversion for BNs on Maxwell data set.

Prediction System	MMRE	MdMRE	Pred	MAR
Bayesian Networks (mean)	132.69	64.44	22.58	6726.23
Bayesian Networks (median)	86.18	58.77	24.19	4655.29

Table 2.5: Numerical conversion for BNs with FSS on Maxwell data set.

Prediction System	MMRE	MdMRE	Pred	MAR
Bayesian Networks + FSS (mean)	163.53	67.74	19.35	6281.83
Bayesian Networks + FSS (median)	97.50	55.99	27.42	4854.74

The effect of using the median in the conversion is quite clear for both prediction systems and in both data sets. However on Desharnais data set under Pred metric there is no improvement. This can be ascribed to the limitation about Pred discussed in the previous section. This metric ignores predictions whose errors are larger than the parameter used, i.e., 25. All errors over this threshold are ignored. So an error that is reduced from 100% MRE to 50% MRE will not affect this metric despite being a valuable improvement. We can infer from this that the improvements happened in the predictions that lie outside the 25% error range since all the other metrics clearly show there were improvements.

We can see the impact of this adaptation is quite significant on the Maxwell data set, which is the more skewed one.

This adaptation is an interesting result on this study. It is safe to say it provides a more accurate conversion. This result can probably be more easily grasped in all detail by the reader after reading the empirical setup in next chapter and the analysis and discussion of results in the last chapter.

2.6 Related work

In this section we describe some closely related studies.

Radlinski and Hoffmann (2010) carried out a comprehensive benchmarking study comparing 23 classifiers in WEKA over four public data sets. The authors state their main research question is: “Is it possible to easily predict software development effort from local data?”. So they establish two specific constraints: easy predictions and using local data, i.e., data from a single company. This paper focused more on the practitioners viewpoint, trying to avoid complex and time-consuming procedures. So the authors do not address specific details of the techniques but provide a wide-ranging assessment of easy-to-use machine learning classifiers. By comparing so many classifiers this study illustrates very well the lack of stability of the ranking of techniques across different

data sets. They mentioned that due to the ranking instability it is difficult to recommend practitioners with a particular model even though they did conclude that K* technique with feature selection was the most accurate overall. BNs were among the most accurate predictors in two of the four data sets but did not particularly stand out. They also demonstrate that most techniques achieve higher accuracy by performing feature selection. A difficulty in drawing decisive conclusions with regard to the accuracy of the models from this study arises with their choice of accuracy metrics.

Mendes and Mosley (2008) outline thorough experiments comparing BNs, CBR, manual stepwise regression and simple mean and median based models for web effort prediction using Tukutuku, a proprietary cross-company data set. The study compares four automatic and four hybrid BN models. The results were very unfavourable to BNs, with most of the models being much more inaccurate than the median model and two of them barely matching it. The authors conclude that manual stepwise regression may be the only effective technique for web effort estimation. Furthermore, they propose and recommend that researchers benchmark proposed models against mean and median based models as they show these can be more effective than more complex models.

One of the latest investigations can be found in Dejaeger et al. (2012) wherein comprehensive and thorough experiments are laid out yielding a benchmark of some statistical and data mining techniques, not including however, BNs. This study benchmarks numeric predictors, as opposed to Radlinski and Hoffman's (2010) which assesses classifiers, i.e., discrete class predictors. This study included thirteen techniques over eight public and private data sets. Their results "indicate that ordinary least squares regression with a logarithmic transformation performs best". They also investigate feature subset selection with a wrapper approach confirming the improvements brought by this technique. They also discuss appropriate procedures and address efforts towards statistically rigorous analyses.

A survey covering bayesian networks for software development effort prediction can be found in Radlinski (2010).

3 EXPERIMENTS SETUP

In this chapter we describe details of our experiments, the data sets and the preparation steps.

We assess data-driven BNs by comparing them to ordinary least squares regression with a logarithmic transformation, which was found by Dejaeger et al. (2012) to be invariably among the most accurate predictors. Quoting the authors, they concluded that "OLS regression with a logarithmic transformation performs best". This is a comprehensive study that compared thirteen techniques over nine data sets. We remind the reader once again that we are comparing a classifier, i.e., a discrete class predictor, to a regression technique, i.e., a numerical predictor. We do this by converting the BN's class predictions to numeric ones by means of the method explained in the previous chapter.

We decided to experiment performing a logarithmic transformation on the data prior to BNs building. So this variant is included in the comparison amounting so far to three prediction systems. Furthermore, we also assess the effectiveness of feature selection as a pre-processing step. So, for each of the aforementioned models there is a variant with the application of feature selection prior to model building which multiplies by two the number of prediction systems. So there are four variants of BNs and two variants of OLS regression amounting so far to six prediction systems.

Finally, we include in the comparison mean and median based models like suggested by Mendes and Mosley (2008). These models simply use the mean and median of all projects effort as a constant prediction. These are very simple benchmark models and an effective model should be able to be more accurate than them. The comparison with such models allows us to better assess the effectiveness of the other techniques. The inclusion of such benchmark models is another recent trend proposed in several studies like Mendes and Mosley (2008) and Shepperd and Macdonell (2012), with the goal of verifying whether the models are effectively predicting and therefore bringing clarity to the results. The situation wherein some proposed models were later found not to be predicting has already happened. Shepperd and Macdonell (2012) illustrate this situation using as an example a previous study by one of the authors that compared two techniques, namely a CBR model (also known as estimation by analogies) and a regression to the mean (R2M) proposed model, concluding that the proposed model performed better. However, when replicating the experiment including a model of random predictions in the comparison it was found that those models were performing worse than random, i.e., actually they were not predicting, thus rendering it meaningless whether one was more accurate than the other. Similarly, Mendes and Mosley (2008) show that simple mean and median models can at times be more accurate than more complex models. So, with these two benchmark models we have in total eight prediction systems.

An abstract outline of the experiments we carried out is shown in Fig. 3.1. We omitted

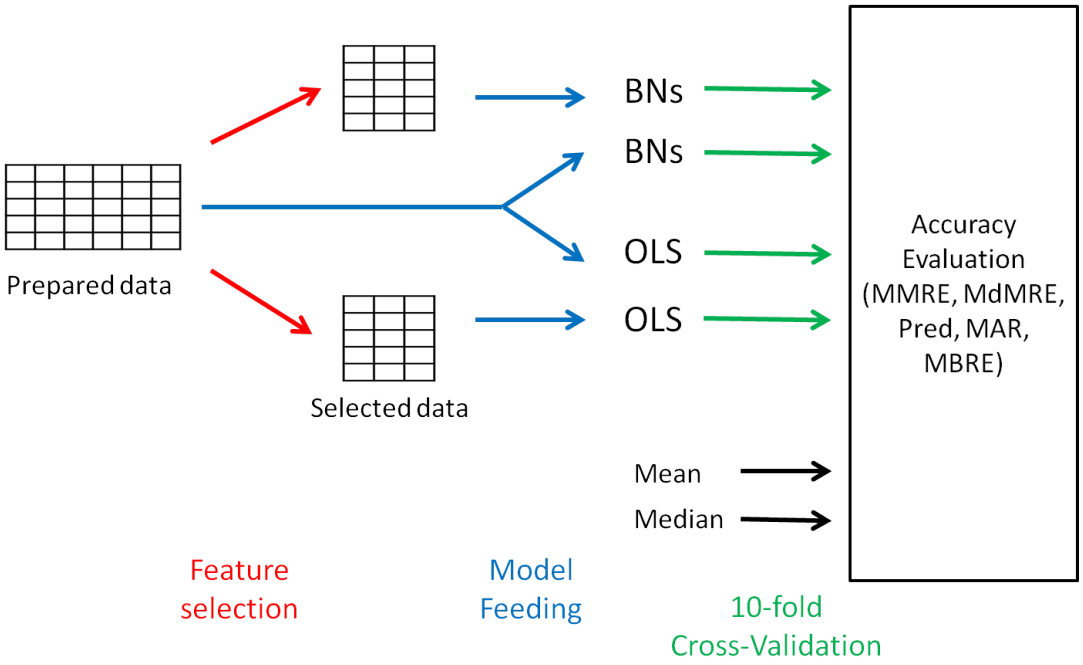


Figure 3.1: Experiments outline.

the different versions of the data set and the two models of BNs on log-transformed data to avoid cluttering up the figure, for intuitiveness' sake. Alternatively, the reader can see in the appendix the figure with the eight prediction systems, see Figure 5.1. So, *prepared data* is an abstract entity which represents any of the data sets versions (log-transformed or not, and discretized or not) and besides the six prediction systems depicted in this figure there are two BNs on log-transformed data (with and without FSS) which are not shown. We will explain these procedures in the next sections.

This comparison allows us to assess the impact of the preprocessing steps and the potential of automatic BNs and possible ways for improvements. We believe this is a sound and meaningful comparison that enables us to draw valid conclusions. As described in the previous chapter our selection of metrics is varied and robust, with the selection of complementary metrics. We highlight that although MBRE metric has a flaw in that it does not account for negative estimates, there were no negative estimates in our experiments, so this flaw had no effect in the results.

These experiments were performed in the WEKA data mining tool (HALL et al., 2009).

3.1 Data sets and Data Preparation

In this work, we used three widely studied data sets available in the PROMISE repository (MENZIES et al., 2012). These are the Desharnais, Maxwell and Cocomo81 data sets. These data sets are relatively clean in comparison to other data sets we have checked. They are local data sets, i.e., data was collected within a single company. Table 3.1 describes basic information on the data sets.

The histograms in Fig. 3.2, Fig. 3.3 and Fig. 3.4 illustrate the distribution of data over effort, the dependent variable. Effort is measured in person-hours on Desharnais and Maxwell and in person-months of 152 hours on Cocomo81. In all three cases the variables are positively skewed, i.e., variables with most records situated towards lower

Table 3.1: Basic information on data sets

Data set	Local data	Application Domain	Effort unit	Range years
Desharnais	Yes	Unknown	Person-Hours	1981-1988
Maxwell	Yes	Finnish bank	Person-Hours	1985-1993
Cocomo81	Yes	Various domains	Person-Months	1970-1981

values and a few very high outlying values. Desharnais is the least skewed of the three at *2.00*. Maxwell is significantly more skewed at *3.35* and Cocomo81 is the most skewed one at *4.48*. Skewness is a very common characteristic in software project data sets.

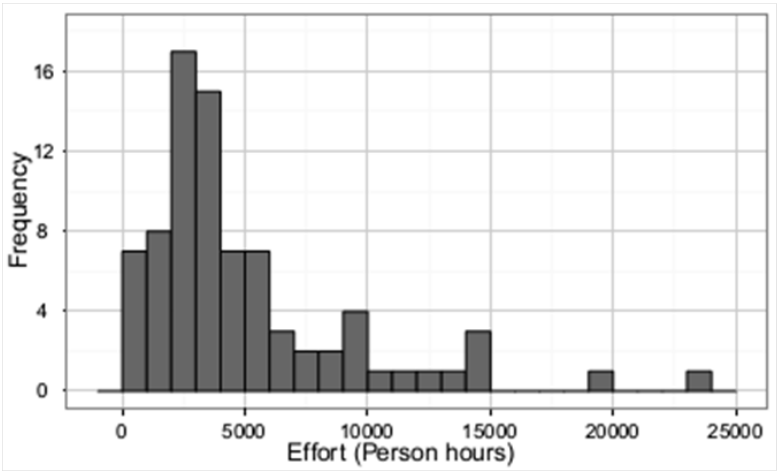
This characteristic poses some hindrances for modeling. In order to carry out linear regression these variables must be transformed as to approximate a Gaussian distribution. With regard to BNs, this is also a problem since the discretization technique could yield very uneven classes intervals. In such a scenario the equal-widths discretization technique (LIU et al., 2002) (TAN; STEINBACH; KUMAR, 2005) can produce empty classes and dispose most of the data set population within just a couple of classes, thus turning the validation highly dubious. If almost all of the data is within just a couple of classes the model can hardly predict wrong or find meaningful patterns. A very high hit-rate would not be surprising, but the predictions would be meaningless. We have observed this happen when replicating the experiments discussed in section 7 of Bibi et al. (2010). The authors use as a concept proof a BN built from a small subset of the ISBSG data set by means of an interesting methodology to improve the software development process. Although this model was a byproduct of their research and not the main focus, the authors claim to have achieved a 100% hit-rate and that was indeed achieved in our replicated experiments. However, the reasons for this uncommon performance were not discussed.

When software managers carry out effort predictions they do not know, for instance, how long a project will last, even though they can have an estimation. Therefore, variables whose values are unknown at the time the prediction is to be performed must be removed, e.g., 'Duration', 'Defects'. This is standard practice in the software prediction field. On the other hand, when a sizing variable is quantified in Function points it is usually included since it can be obtained in the specification phase, depending on the process model.

On Desharnais data set three variables were removed: the ID variable, 'YearEnd' and 'Length'. On Maxwell data set three variables were removed: 'Duration', 'Time' and 'SYear'. Finally, no variables were removed from Cocomo81 data set.

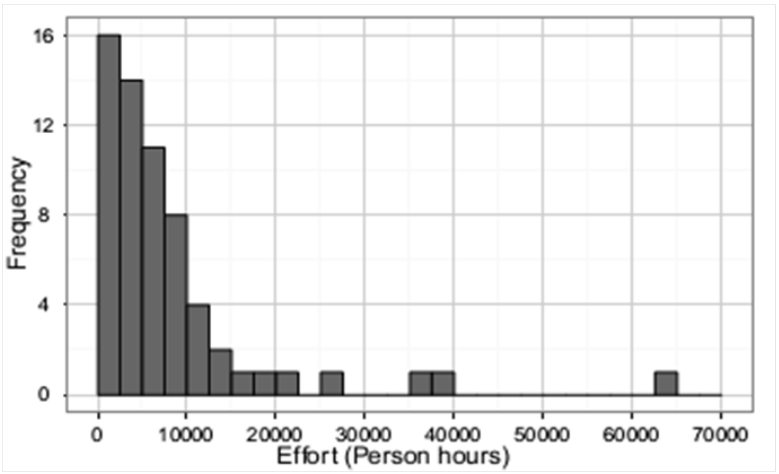
In order to carry out OLS regression we removed records with missing values. This amounts to four records on Desharnais data set and two records on Maxwell data set. There were no missing values on Cocomo81 data set. For the BNs models all the records were kept. We also experimented not removing the missing values for the OLS regression model by performing median imputation and the difference on Desharnais data set, which is the one with more missing values, was minimal. So, we decided to show the results on the data set without records with missing values because these are the same we used in our paper (TIERNO; NUNES, 2012). On Maxwell data set there were two records with missing values and on Cocomo data set there were none.

The categorical variables were coded to dummy variables for the linear regression model following good statistical practices (KITCHENHAM; MENDES, 2009). Kitchenham also suggests the removal of outliers. Although this is the standard practice for statistical procedures, we decided to keep the outliers for both models for two reasons: To keep the same conditions for both models; and chiefly because these outliers are actual



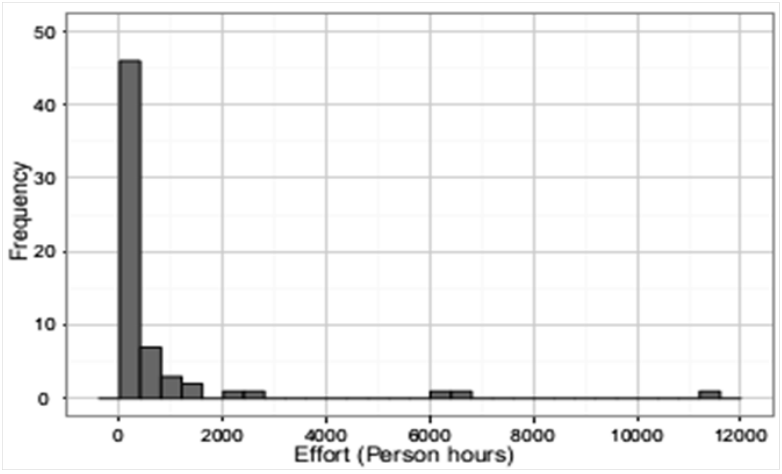
Number of projects: 81
Number of variables: 12
Skewness: 2.00
Kurtosis: 4.71
Mean Effort: 5046.30
Median Effort: 3647

Figure 3.2: Desharnais data set.



Number of projects: 62
Number of variables: 27
Skewness: 3.35
Kurtosis: 13.70
Mean Effort: 8223.2
Median Effort: 5189.5

Figure 3.3: Maxwell data set.



Number of projects: 63
Number of variables: 17
Skewness: 4.48
Kurtosis: 21.87
Mean Effort: 683.53
Median Effort: 98

Figure 3.4: Cocomo81 data set.

projects which are rare but can happen. They should not be ruled out as noisy or irrelevant entries. Other studies in software prediction also keep the outliers, e.g., Dejaeger et al. (2012), Radlinski and Hoffmann (2010).

For more detailed information on these data sets we refer the reader to the original works referenced in the PROMISE repository (MENZIES et al., 2012).

3.2 Data Pre-processing

Prior to applying OLS regression, skewed variables must undergo a logarithmic transformation (KITCHENHAM; MENDES, 2009). This transformation has the objective of making the variables approximate a Gaussian distribution in order to optimize linear regression's effectiveness.

Prior to building BNs the data underwent discretization. This is one of the most common pre-processing steps in data mining. This topic has received relatively little attention in software prediction field. In order to obtain meaningful predictions we used equal-frequencies discretization (TAN; STEINBACH; KUMAR, 2005). The continuous variables were discretized to five bins. There are no strict rules on this. Some studies use only three bins and others use seven or more bins. Five is a common number of bins in software prediction studies, e.g., Mendes and Mosley (2008), Radlinski and Hoffmann (2010). On the one hand, the larger the number of bins, the more precise the predictions can be. But on the other hand, the amount of conditional probabilities grows exponentially on the amount of bins. Thus more data is necessary for reliable model learning. Considering software project data sets are usually small, a large number of bins tends to be impractical. There is a short review study on discretization in software prediction field in Fernández-Diego and Torralba-Martínez (2012). More investigations are needed on this subject.

A recent technical development on BNs which we could not exploit is the employment of dynamic discretization (NEIL; TAILOR; MARQUEZ, 2007). This appears to be a very interesting development since the discretization bears directly on the model's accuracy. The authors reported that this discretization method improves accuracy and provides more flexibility to model builders (FENTON; NEIL; MARQUEZ, 2007). However, this development is generally not available in BNs tools yet. It was implemented on commercial BNs tool Agena Risk.

Table 3.2: Discretized effort classes of Desharnais data set

Effort Category	Range	Median value	Log transf. Median
Very Low	<= 2161.5	1211	7.0981
Low	2161.5 < x <= 3062.5	2534	7.8375
Medium	3062.5 < x <=4035.5	3636.5	8.1987
High	4035.5 < x <= 7553	5635	8.6367
Very High	> 7553	10969	9.3021

Although some BNs tools offer support to continuous variables in a limited way, e.g., making assumptions on the distribution of data or assumptions on the relationships among the variables, studies in software prediction field mostly resort to discretization when there are continuous variables. WEKA's implementation of BNs currently does not support continuous variables. Even though we used the equal-frequencies discretization method

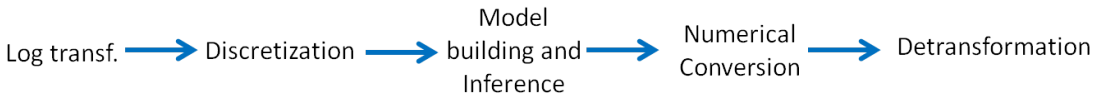


Figure 3.5: BNs on log-transformed data

(LIU et al., 2002), sometimes the classes ended up with different amount of records because of repeated values. This happened more often in the Cocomo81 data set which has many repetitions of values due to the way in which this data set was projected. In this data set the repeated values actually represent categories. The splitting up of such records would confound the model, so they are kept in the same class.

Tables 3.2, 3.3 and 3.4 show the discretized classes for each data sets' effort variables. The measurement unit for Desharnais and Maxwell is person-hours. Cocomo81's is measured in person-months (months of 152 hours). The reported median values are used by the BNs' numerical conversion method explained in the previous chapter.

Table 3.3: Discretized effort classes of Maxwell data set

Effort Category	Range	Median value	Log transf. Median
Very Low	<=1750	1080	6.9845
Low	1750 < x <= 4187	2957	7.9919
Medium	4187 < x <= 6960.5	5189.5	8.5542
High	6960.5 < x <= 10735	8710	9.0722
Very High	>10735	16776	9.7223

We can observe that the higher effort classes are much larger than the low effort classes. This pattern is directly related to the skewness of the data sets because the data gets sparser as the effort increases, therefore requiring larger classes to keep the bins with the same number of records which in turn incur precision losses to the classifier. This effect is more pronounced in the more skewed data sets.

We determined to experiment performing a logarithmic transformation on the data prior to BNs building. We have not seen this being evaluated elsewhere even though a couple of studies raised this possibility. Fernández-Diego and Torralba-Martínez (2012) apply this transformation to the data in order to compare three discretization techniques but without any assessment on its effectiveness. This transformation complicates a bit the numerical conversion. The numerical conversion must use the median of each class' log-transformed data and not the median of the original ranges. The medians of the discrete intervals obtained on log-transformed data are shown in the three tables under 'Log transf. Median'. Then the number obtained after the conversion will have to undergo the de-transformation to the original range just like done for OLS regression. This process is illustrated in Fig. 3.5. Considering the effectiveness of this transformation for OLS regression we decided to experiment performing this transformation on BNs. It addresses the problem of skewness of software project data sets and makes the classes more even.

There is a wide variety of feature subset selection algorithms in the data mining field and many of these algorithms are implemented in WEKA. We decided to use a wrapper approach with BestFirst search algorithm and using a 5-fold cross validation procedure to estimate the accuracy of the models. This algorithm generated good scoring feature subsets in our experiments and has also been well regarded elsewhere (HALL; HOLMES,

Table 3.4: Discretized effort classes of Cocomo81 data set

Effort Category	Range	Median value	Log transf. Median
Very Low	<= 34.5	9	2.1972
Low	34.5 < x <=76	47	3.8501
Medium	76 < x <=188.5	102	4.6242
High	188.5 < x <= 653.5	321	5.7714
Very High	> 653.5	1436	7.2630

2003) (KOHAVI; JOHN, 1997) (MENZIES et al., 2010). We took up this feature selection procedure for both BNs and OLS regression models.

It is interesting to observe that the application of feature selection stems from the data mining field and the combination with linear regression is not mentioned as standard statistical practice but we did achieve good results with this combination. In statistics, the selection of variables is done by means of other methods and the model is referred to as stepwise regression, e.g., see Kitchenham and Mendes (2009) and Mendes and Mosley (2008). Dejaeger et al. (2012) also used feature selection in combination with the linear regression methods. We observed that Dejaeger et al.'s results for OLS regression with and without feature selection are very similar, which shows that there is some form of variable selection taking place for OLS regression without feature selection. In our study, we made sure to not perform any selection of variables for the experiments on OLS regression without feature selection, because otherwise the models would be virtually the same.

4 RESULTS AND ANALYSIS

This chapter reports on the results of the experiments outlined in the previous chapter. The results are displayed in confusion matrices and tables. In each figure, the upper confusion matrices are for BNs without feature selection and the lower matrices for BNs with feature selection. This is the traditional way to display results for classifiers. The tables contain the results according to MRE-based metrics, MBRE and MAR, besides the basic hit-rates for the BNs. Furthermore, we included boxplots of the MREs and the residuals in the appendix.

We have observed across various experimentations with the hyperparameters available in WEKA (e.g., enforcing a limit on the number of parents for each node, enforcing a markov blanket correction) that complex networks with many nodes and relationships among them tended to perform worse than simple Naïve Bayes networks. These are simple BN models in which every independent variable relates directly to the dependent variable. The increased complexity of some models did not translate into more accurate predictions.

The confusion matrices in Figs. 4.1, 4.2 and 4.3 illustrate the BN predictions for the Desharnais, Maxwell and Cocomo81 data sets with and without feature selection on the non log-transformed data. They allow us to see how frequently the predictions hit the right class and by how much the wrong predictions missed the right class. We did not include confusion matrices of the BNs on log-transformed data.

We consider predictions to be amply wrong when they hit classes situated two classes away or more from the actual class. On Desharnais data set, without feature selection eleven times the predictions were far. Same number with feature selection. So, while there was an improvement on the hit-rates, the number of amply inaccurate predictions did not decrease. On Maxwell data set this pattern is more pronounced. Without feature selection, thirteen predictions were amply inaccurate, i.e., two classes away or more. This number increased to sixteen when feature selection was carried out. On Cocomo81 the number of very inaccurate predictions remained the same.

We can observe in the confusion matrices for all data sets that there is an improvement in the hit-rates with the employment of feature selection. However it is also observable that the magnitude of the wrong predictions also increased, i.e., the errors are situated farther away from the right class. It appears to be a tradeoff wherein the larger errors offset the increased hit-rate. So, the feature selection process appears to be geared towards improving the hit-rates.

The most relevant variables identified by the feature selection algorithm for OLS regression on Desharnais data set were the dummy variables for ‘Language’ (programming language used) and ‘PointsAdjust’ which is a size measure in functions points. For BNs the variables selected were ‘ManagerExp’ (manager experience), ‘PointsNon-

Predicted Class Effort — no feature subset selection

Very Low	Low	Medium	High	Very High	
9	4	0	3	0	Very Low
4	**8**	2	1	1	Low
2	4	**3**	5	2	Medium
1	0	4	**8**	4	High
0	0	1	5	**10**	Very High

Actual Class Effort

Predicted Class Effort — feature subset selection

Very Low	Low	Medium	High	Very High	
10	2	1	3	0	Very Low
3	**10**	1	1	1	Low
1	4	**4**	6	1	Medium
0	0	4	**9**	4	High
0	0	3	2	**11**	Very High

Actual Class Effort

Figure 4.1: BNs confusion matrices with and without feature subset selection over Desharnais data set

Predicted Class Effort — no feature subset selection

Very Low	Low	Medium	High	Very High	
7	3	2	0	0	Very Low
3	**2**	4	3	1	Low
1	4	**6**	1	0	Medium
0	3	2	**4**	4	High
0	2	1	3	**6**	Very High

Actual Class Effort

Predicted Class Effort — feature subset selection

Very Low	Low	Medium	High	Very High	
9	2	1	0	0	Very Low
1	**5**	3	2	2	Low
1	3	**6**	1	1	Medium
1	4	1	**4**	3	High
0	3	1	0	**8**	Very High

Actual Class Effort

Figure 4.2: BNs confusion matrices with and without feature subset selection over Maxwell data set

Predicted Class Effort — no feature subset selection

Very Low	Low	Medium	High	Very High	
9	3	1	0	0	Very Low
4	**3**	3	3	0	Low
2	2	**4**	3	1	Medium
0	2	1	**7**	3	High
0	0	0	3	**9**	Very High

Actual Class Effort

Predicted Class Effort — feature subset selection

Very Low	Low	Medium	High	Very High	
10	3	0	0	0	Very Low
4	**3**	4	2	0	Low
1	2	**7**	0	2	Medium
1	0	2	**7**	3	High
0	0	2	2	**8**	Very High

Actual Class Effort

Figure 4.3: BNs confusion matrices with and without feature subset selection over Cocomo81 data set

Adjust', 'Adjustment', 'PointsAdjust' (these last three variables are related to Function points measure) and 'Language' variable (programming language used). For BNs on long-transformed data the variables selected were 'ManagerExp', 'Entities' (also related to Function points measure) and 'PointsNonAdjust'.

On Maxwell data set, feature selection for OLS regression identified six variables: 'Ifc' (User interface), 'T03' (Staff availability) , 'T08' (Requirements volatility), 'T09' (Quality requirements), 'Time' (a measure related to the starting year of the project) and 'Size'. For both BNs models (i.e., with and without log transformation on the data) the algorithm identified nine variables: 'Har' (Hardware platform), 'Dba' (Database), 'Source' (whether the project was outsourced or not), 'Nlan' (Number of different development languages), 'T03' (Staff availability), 'T12' (Staff analysis skills), 'T13' (Staff application knowledge), 'T15' (Staff team skills) and 'Size'.

The feature selection procedure on Cocomo81 data set selected ten variables for OLS regression: 'rely' (required software reliability), 'time' (time constraint for cpu), 'virt' (machine volatility), 'acap' (analysts capability), 'pcap' (programmers capability), 'vexp' (virtual machine experience), 'modp' (modern programing practices), 'tool' (use of software tools), 'sced' (schedule constraint) and 'loc' (a size variable that can be computed from a function points analysis). For BNs, feature selection identified only a few variables. On the non log-transformed data it selected three variables: 'cplx' (process complexity), 'stor' (main memory constraint) and 'loc'. On the log-transformed data it selected four variables: 'data' (data base size), 'stor' (main memory constraint), 'lexp' (language experience) and 'loc'.

Table 4.1 reports the results for the Desharnais data set according to the continuous

metrics previously exposed. On the Desharnais data set there is an obvious improvement in the BNs' hit-rates when applying feature selection. However, when we consider the continuous metrics there were generally no improvements except under Pred. Pred metric resembles the hit-rates in its characteristic of only considering the accurate predictions and ignoring predictions lying far from the actual value. This shows there were more accurate predictions but that there were also more wrong predictions since the other metrics do not show improvements. This illustrates the limitation of Pred metric that we highlighted in chapter 2. For OLS regression, there is a small improvement under MMRE, MdMRE, MAR and MBRE and a marginal degradation under Pred metric. The improvements were relatively small because the number of variables in this data set is also small. The likelihood of a data set containing irrelevant or redundant variables dwindles in such cases and the feature selection technique cannot find much improvements by further decreasing the number of variables.

The accuracy of the BNs on log-transformed data was about the same as on the non transformed data. The log transformation did not bring improvements to BNs' predictions. BNs' performance was very constant regardless of data pre-processing. So, on this dataset, BNs performed relatively well but were more prone to large inaccuracies. This can be confirmed graphically in the boxplots of figures Fig. 5.2 and Fig. 5.3 shown in the appendix.

Finally, BNs clearly overcame the baseline models.

Table 4.1: Models performance on Desharnais data set

Prediction System	Hit-rate	MMRE	MdMRE	Pred	MAR	MBRE
BNs	46.91%	57.23	32.66	33.33	2153.52	65.83
BNs + FSS	54.32%	56.18	34.16	39.5	2133.84	64.52
BNs + log	44.44%	56.37	33.61	34.57	2128.65	67.38
BNs + log + FSS	48.15%	57.64	36.42	38.27	2165.47	72.19
OLS + log	-	37.62	29.19	46.75	1731.53	48.04
OLS + log + FSS	-	34.24	27.66	45.45	1567.93	42.54
Mean model	-	121.66	59.49	18.51	3161.52	140.04
Median model	-	78.46	42.03	29.62	2861.53	120.42

Table 4.2 reports on the results for the Maxwell data set. For being the data set with the largest amount of variables in this study, it is likely to contain irrelevant variables and benefit the most by undergoing feature selection. This expectation is fulfilled for OLS regression. Feature selection reduced by half the mean of residuals and all the other metrics show large improvements as well. The results obtained here with the application of feature selection were significantly more accurate than the results achieved by Dejaeger et al. (2012) on the same data set.

But again, like on Desharnais data set, BNs' performance did not improve convincingly with the application of feature selection. There is a clear improvement on the hit-rates and an improvement under Pred metric, but the other metrics show that the increase of good predictions (i.e., predictions close to the actual value) was offset by larger errors.

It is interesting to observe that when the data did not undergo feature selection, the performance of BNs is comparable to the performance of OLS regression. But with the application of feature selection OLS regression has a large improvement in accuracy as opposed to BNs which do not collect any improvement. This highlights that the BNs

models are missing very significant improvements in accuracy which are expected with the application of feature selection.

With regard to the logarithmic transformation, the results show small improvements for BNs under all metrics but Pred as opposed to the Desharnais data set in which there was no effect.

Like on Desharnais dataset, BNs clearly overcame the baseline models. In our view, an important observation on this data set is the improvement with feature selection that is being missed by BNs. We will discuss the reasons for this after exposing all results.

All these results can also be verified graphically in the boxplots of figures Fig. 5.4 and Fig. 5.5.

Table 4.2: Models performance on Maxwell data set

Prediction System	Hit-rate	MMRE	MdMRE	Pred	MAR	MBRE
BNs	40.32%	86.18	58.77	24.19	4655.29	110.48
BNs + FSS	51.61%	97.5	55.99	27.41	4854.74	122.19
BNs + log	40.32%	73.41	52.72	19.35	4550.90	104.54
BNs + log + FSS	51.61%	70.67	53.88	25.81	4576.05	106.44
OLS + log	-	76.86	43.78	30	4932.6	101.19
OLS + log + FSS	-	42.57	28.62	40	2500.04	52.28
Mean model	-	119.67	52.96	19.35	5616.54	225.64
Median model	-	108.95	66.28	20.97	5654.11	180.91

Table 4.3 reports on the results on Cocomo81. On this data set, the logarithmic transformation did yield an observable improvement on the BNs' predictions, specially under MMRE. This suggests a decrease of large overestimates. We can observe the difference in performance compared to OLS regression grew in comparison to the previous data sets, even though this effect can be slightly reduced by the application of the logarithmic transformation.

Feature selection brought an improvement for OLS regression though not as pronounced as on Maxwell. For BNs, the same pattern of improved hit-rates and no improvements under other metrics which was observed in the other data sets stands on this data set. This appears to be related to the skewness of the data sets and the loss of precision brought about by the discretization process. Skewness increases this imprecision because it makes the classes more uneven. The logarithmic transformation is only to some extent able to reduce this effect.

Nevertheless, even in this very skewed data set they were able to overcome both baseline models which is an improvement over Mendes and Mosley's (2008) results.

Table 4.4 shows the frequency of underestimates and overestimates for each model. OLS models have a tendency to underestimate, which is considered worse than a tendency to overestimate.

The variables most frequently identified by the feature selection algorithm were related to 'Size'. In all data sets studied here, a size variable was selected. This variable appears to be frequently the one with highest predictive value for estimating effort.

We can observe in all of these results that feature selection improved clearly and consistently the hit-rates of BNs and the accuracy of linear regression over all data sets. This effect is very pronounced on the Maxwell data set which is the one with the highest number of variables. Such improvements are expected because the larger the amount of

Table 4.3: Models performance on Cocomo81 data set

Prediction System	Hit-rate	MMRE	MdMRE	Pred	MAR	MBRE
BNs	50.79%	134.85	58.64	25.81	551.95	197.82
BNs + FSS	55.56%	270.64	130.37	9.68	606.22	336.39
BNs + log	52.38%	91.19	53.64	19.35	536.54	233.15
BNs + log + FSS	55.56%	76.94	64.93	25.81	530.61	212.73
OLS + log	-	46.6	30.49	44.44	278	61.83
OLS + log + FSS	-	44.28	22.98	53.96	297.47	55.97
Mean model	-	1775.35	571.16	4.76	891.64	1905.81
Median model	-	235.42	86.25	15.87	642.63	842.24

variables in a data set, the more likely it is for the data set to contain irrelevant or redundant variables. This emphasizes the importance of applying feature selection especially on data sets with many variables. It also highlights the fact that many variables in software projects data sets have a small predictive value and can actually make the models less accurate. Therefore, collecting a smaller amount of variables focusing on high data quality may be a wiser approach for data-based predictions. This finding is a confirmation of the findings of previous studies, e.g., Chen et al. (2005), Radlinski and Hoffman (2010) and Dejaeger et al. (2012).

In spite of these clear improvements however, we can see that the improvements of BNs predictions when measured by the continuous metrics was small or at times the accuracy even worsened. Specially on the Maxwell and Cocomo81 data sets, on which the predictions were significantly less accurate than without feature selection as opposed to what one would expect. According to data mining literature, wrapper approaches like the one applied here use the algorithm's own accuracy measure to assess the feature subset (HALL; HOLMES, 2003) (TAN; STEINBACH; KUMAR, 2005). And it is obvious the BNs algorithm is not using this numerical conversion to measure accuracy. The model selection is clearly favouring hit-rates. This brings into question the validity of hit-rates as an accuracy measure or at least highlights its limitation. Improved hit-rates were offset by larger magnitude errors, i.e., less wrong predictions but when the predictions were wrong they were wrong by a larger margin. So, does the improved hit-rate really reflect a more accurate model? In all these experiments BNs ended up missing the improvements expected from feature selection. This could make a significant difference in Maxwell and Cocomo81 data sets which are the ones with larger amounts of variables.

It follows from this observation that an interesting development for BNs would be to investigate the feasibility of incorporating this numerical conversion into the BNs algorithms and tools, using it as a measure of accuracy instead of the hit-rates or error-rates. This modification could bring in some improvements in the predictions and also in the effects of the feature selection technique. The application of feature selection would find improvements in overall accuracy even if with lower hit-rates. As it is, the potential improvements expected from feature selection are being wasted in the strive for higher hit-rates. Alternatively, a suggestion for future research is to experiment with other BNs search algorithms, score types and CPT estimators and check out whether these bypass this focus on hit-rates. In this study we restricted ourselves to the K2 search algorithm (COOPER; HERSKOVITS, 1992) with Bayes method for scoring of the networks and Simple estimator to estimate the NPTs.

Table 4.4: Frequency of Underestimates and Overestimates

Prediction System	Overestimates (count)	Underestimates (count)
BNs	110	96
BNs + FSS	127	79
BNs + log	99	107
BNs + log + FSS	104	102
OLS + log	86	114
OLS + log + FSS	91	109

We can observe a trend in these results. BNs accuracy degrades according to the data sets' skewness. With increases in skewness BNs struggle to predict accurately. BNs best performance in these experiments was achieved in the least skewed data set, i.e., Desharnais. When the data is too skewed the discretized classes become too uneven and there is an increased loss of precision with the largest discretized intervals. The highest effort classes tend to be very sparse. An example is the highest effort class defined for the Maxwell data set which spans a wider interval than all others put together (ranges from 10000 to 64000 person-hours), thus being very imprecise. Besides the effect on the discretization, there is also an effect on the numerical conversion because even a small probability of the highest effort class (Very High) affects the conversion quite significantly.

Much of the imprecision of the BNs can be ascribed to the discretization process. This subject has been neglected to some extent in this research field and the establishment of guidelines on this could benefit research inititiatives. The imprecision brought about by the discretization process is directly related to the skewness of the datasets. In this scenario of highly skewed data sets, the equal-frequencies discretization generates classes intervals of too different widths and the numerical conversion will show larger error margins. The alternative of equal-widths discretization causes meaningless results, for there will be empty or near empty classes and the model learning will simply state the obvious, predicting nearly always the same class which is the lowest effort class since it contains most of the records. High hit-rates are not only unsurprising but very likely when using equal-widths in very skewed data sets. Unless a log transformation is applied to the data, predictions based on skewed data discretized with the equal-widths method bring in deceitful results. Related to these findings is the study of Fernández-Diego and Torralba-Martínez (2012) which compared equal-widths, equal-frequencies and k-means discretization on one data set and concluded that equal-frequencies with a log-transformation can improve the accuracy results according to most evaluation criteria. Further investigations on discretization methods are necessary.

An interesting undertaking was to investigate the effect of the log transformation on the Bayesian classifier. Even though a couple of studies used this transformation, we are not aware of studies assessing its effects. The log transformation was able to provide only slight improvements of accuracy. The results show that in very skewed data sets, transforming the data can be beneficial. As another suggestion for future research, we observe that it would be interesting to try out this data transformation with BNs that support continuous variables since in these experiments much of the benefit of performing this transformation appears to have been lost with the discretization.

These experiments on data-driven BNs are relevant because the way data is explored

can have a significant impact on the model's performance. Much of the excitement over BNs revolves around their capability to integrate objective and subjective knowledge. Therefore, learning how to optimize the use of data (i.e., the objective part) can improve the performance of not only data-driven BNs, but also hybrid BNs which appear to be the most promising for this research field. Even though BNs solely based on data may not become the most accurate approach in software effort prediction, improvements on the use of data for BNs benefit this technique as a whole and given its relevancy in software engineering, these investigations are necessary. Optimizing the performance of the data mining capabilities of BNs is an essential part in the development of this modelling technique.

Our results on these data sets are more optimistic for BNs than the ones reported in Mendes and Mosley (2008), which were obtained on another data set. Our experiments show the BNs models struggle in very skewed data sets but are still capable of achieving a minimum standard of accuracy. In Mendes and Mosley (2008), most BNs, including hybrid BNs, performed worse than the baseline models.

From our studies on the literature and our own experiments, we observe that it appears to be hard to overcome OLS regression when it is properly applied. Our results on OLS regression confirm the conclusion of Dejaeger et al. (2012) and partially the results of Mendes and Mosley (2008). While OLS regression does perform better with regard to accuracy, one must observe that OLS regression as a well established statistical technique is optimized to its best. On the other hand, we have shown in this study that techniques like BNs have room for improvements and are under constant development. As BNs theory evolve and the tools catch up with the developments, more accurate predictions will be possible. Ideally, if data-driven BNs catch up with OLS regression and even overcome them, they will be very advantageous due to their flexibility and powerful features. When such a standard is achieved BNs users will be able to trust this technique is exploring data as well as the most accurate data-based models.

Specifically, we have observed room for improvements for BNs with regard to discretization techniques and experimenting with different model selection methods which could provide improvements in accuracy under other metrics than the hit-rates and also optimizing the effects of feature selection. This appears to be a fundamental problem. Furthermore, there are developments in data mining research concerning support for ordinal and continuous variables. These could also bring further improvements in accuracy. And besides these improvements on BNs' data mining capabilities, there are also improvements concerning support for experts' model building.

The BNs tools are currently a limitation (RADLINSKI, 2010). The latest developments are not available for most of the tools. In these experiments we did not have the opportunity to experiment with continuous variables nor with dynamic discretization. It would be interesting to verify the improvements techniques like dynamic discretization proposed in Neil, Taylor and Marquez (2007) could bring in. Although WEKA offers validation advantages over other tools, it does not have other developments from BNs theory. As we already mentioned, an interesting development would be the incorporation of the numerical conversion method. This conversion is not automated in the tools and it can be somewhat cumbersome to perform which may hinder its employment. Having this conversion automated into the tools could be an interesting development.

Some studies on BNs indicate that BNs' main strength for the software prediction area lies in their possibility to incorporate domain knowledge and qualitative factors, therefore favouring hybrid or expert-driven approaches. Currently, an advantage of data driven

models like this, as pointed out in Radlinski and Hoffmann (2010), is that by owning a projects data set it is possible to obtain quick predictions as supporting evidence for the expert's prediction, as opposed to expert based networks which take much more effort to build and to have the NPT's elicited. The employment of data-based models to support expert estimates has been indicated to industry practitioners as a means to increase safety and reliability on experts' estimates, since the situation with expert-based estimations has not been easier than the situation seen in this research field.

Finally, an observation obtained with this study and the difficulties in the field is that it is important to show faithful and realistic results even if they are not positive towards a particular technique. This research field has suffered in the last twenty years due to over-optimism towards some techniques. In recent years, efforts towards correcting mistaken studies and addressing reasons for conflicting results are on the rise even if these show a less than flattering state of affairs in the field. To move forward it is important to recognize the actual situation paving the way for improvements and solutions.

5 CONCLUSIONS

This study provided a sound and realistic assessment of automatic BNs by means of a comparison with a long established statistical technique and benchmark models, thereby illustrating its current limitations and possibilities of improvements. BNs' limitations are discussed and some guidelines on its employment are provided. Specifically, the skewness of data sets prevalent in this research field and the discretization are shown to bring about inaccuracies that limit BNs' effectiveness.

One suggestion arising from these observations and set forth to the research community is to investigate the feasibility of incorporating the numerical conversion into BNs model building as we consider it portrays accuracy more faithfully than the basic hit-rates. This could make BNs models generally more accurate even if achieving lower hit-rates. Also, the inclusion of this conversion in the tools would be interesting for research undertakings.

We believe this study discusses important matters that are scarcely discussed in software prediction studies and that can be a source of confusion. Most studies have not addressed much attention to data set properties and implications on model's functioning. Shedding light on these somewhat neglected topics is an important step to address some of the current difficulties in the field.

An interesting finding is the accuracy improvements obtained by using the median for the numerical conversion of the BNs' probability distributions instead of using each class' mean like it was originally proposed in Pendharkar, Subramanian and Rodger (2005). The improvements obtained with this modification are larger than the improvements obtained with any of the other pre-processing methods investigated.

Another result is the confirmation of the potential of feature selection to improve software prediction models' accuracy. Its effect is very clear on the linear regression technique. For BNs, the improvement is very obvious when we evaluate the hit-rate, but when converting the predictions there was little or no improvement. This lack of improvement can be ascribed to the model selection algorithm because it strived to choose models that achieve higher hit-rates. Since the feature selection approach uses the learning algorithm's accuracy measure to select the feature subset, a better feature subset could be chosen if the learning algorithm accounted for the numerical conversion or at least favoured the continuous metrics. A modification in this direction could yield significant improvements in overall accuracy. Furthermore, the effectiveness of feature selection tells us that focusing on collecting high quality data for a small number of highly predictive attributes may be more effective than collecting data on a large number of variables. The results obtained so far in researches with feature selection make it a must in software prediction endeavours.

Moreover, we assessed the possibility of performing a logarithmic transformation on

the data prior to model learning. Even though a couple of studies employed this transformation in combination with BNs, its effectiveness had not been verified. In our experiments we have shown that the logarithmic transformation for BNs is capable of improving predictions only slightly on the most skewed data sets. We believe larger improvements were lost due to the discretization. It would be interesting to assess this transformation in combination with continuous variables.

This study showed some of the problems arising from the data sets in the field and the constraints they impose specially on classifiers. Much of this is related to the discretization process and the uneven classes that it generates. We brought forward some points concerning the exploration of data which we believe to be important for the development of BNs.

There is a limit on how accurate data-driven prediction techniques can be depending on the data used. Therefore, more efforts should be addressed in studying software prediction data sets properties and data pre-processing in order to increase prediction accuracy. The performance of these models is highly dependent on data quality, which is a subject that has not received sufficient attention. Significant improvements could come from investigations on this.

Our observations indicate that BNs have a potential for data-based predictions but still need improvements to catch up with the most accurate data based models. In spite of the apparent advantage of linear models in this scenario, i.e.,data-driven modeling, it must be observed that this is only part of the potentiality of BNs. BNs offer experimenting possibilities beyond that of linear regression. The linear regression method can only provide a point estimate, whereas BNs meet other requirements expected from a prediction model.

Our results present a more optimistic view on BNs when compared to the results presented in Mendes and Mosley (2008), showing that although BNs are less accurate than linear regression, they were able to achieve a minimum standard of accuracy and that there is a potential for improvements which could lessen this gap to linear regression models. Even if BNs solely based on data might not become the most accurate predictor in software effort prediction, optimizing the exploration of data is an important step of development and necessary given the relevancy of BNs in software management research.

Purely data-based techniques can be useful to support software managers, specially as a quick, less demanding approach to aid the software manager or to keep the software manager in check in case of possible bias, e.g., bias towards optimism. Notwithstanding, due to the human factors and inherent uncertainties in software projects, the capability to incorporate expert's subjective knowledge can provide an advantage over models solely based on data. Bayesian Networks appear to be one of the most suitable techniques for future progresses in this aspect. BNs theory and tools are under constant development and some technical breakthroughs regarding discretization and NPT's elicitation appear to herald progresses for BNs in software prediction and software projects management in general.

5.1 Future Work

A topic that could provide some improvements for the software prediction field and that warrants investigations is data pre-processing. Carrying out this work we observed the impact discretization, data transformations and feature selection can have on the models' performance. Moreover, we observed the implications of and hindrances posed by the characteristics of software projects data sets. In our view, discretization is a topic that

needs thorough investigations as there are currently no guidelines on this.

In this work we applied a specific feature subset selection technique (a Wrapper approach with BestFirst algorithm (HALL; HOLMES, 2003)). It would be interesting to assess whether other feature selection techniques can bypass this focus on hit-rates that this wrapper approach demonstrated. Good improvements could be obtained if BNs could better extract the accuracy improvements expected from feature selection.

Another suggestion is to experiment with other learning and selection algorithms, as in this work we restricted ourselves to the K2 search algorithm with Bayes method for scoring of the networks and Simple estimator to estimate the NPTs. We have the expectancy that other algorithms could assess accuracy in a different way, as in this study the algorithms were clearly favouring the hit-rates, which we questioned as an accuracy measure.

Furthermore, investigating BNs with continuous variables and the related pre-processing procedures could yield interesting results.

Also, statistical significance tests could be performed to enhance the validation of the results.

APPENDIX A

Figure 5.1 outlines the experiments including the eight prediction systems.

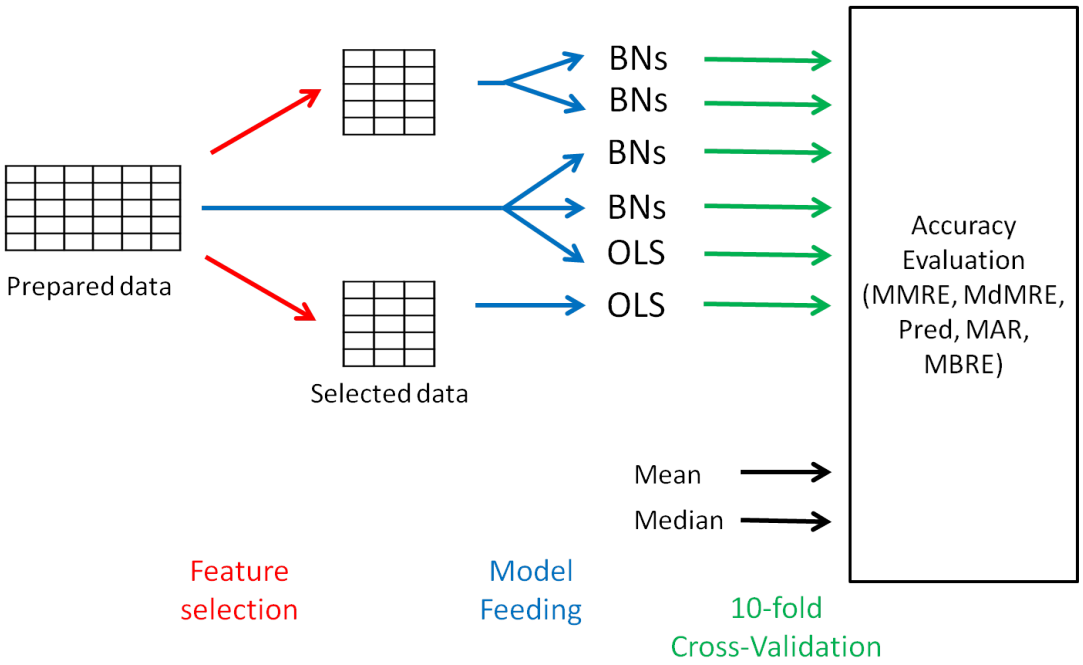


Figure 5.1: Experiments outline.

In the following pages, boxplots of the residuals and the MREs for the three data sets are shown. For all boxplots the numbers represent the following prediction systems:

(i)BNs;
(ii)BNs+FSS;
(iii)BNs+log;
(iv)BNs+log+FSS;
(v)Mean model;
(vi)Median model;
(vii)OLS+log;
(viii)OLS+log+FSS;

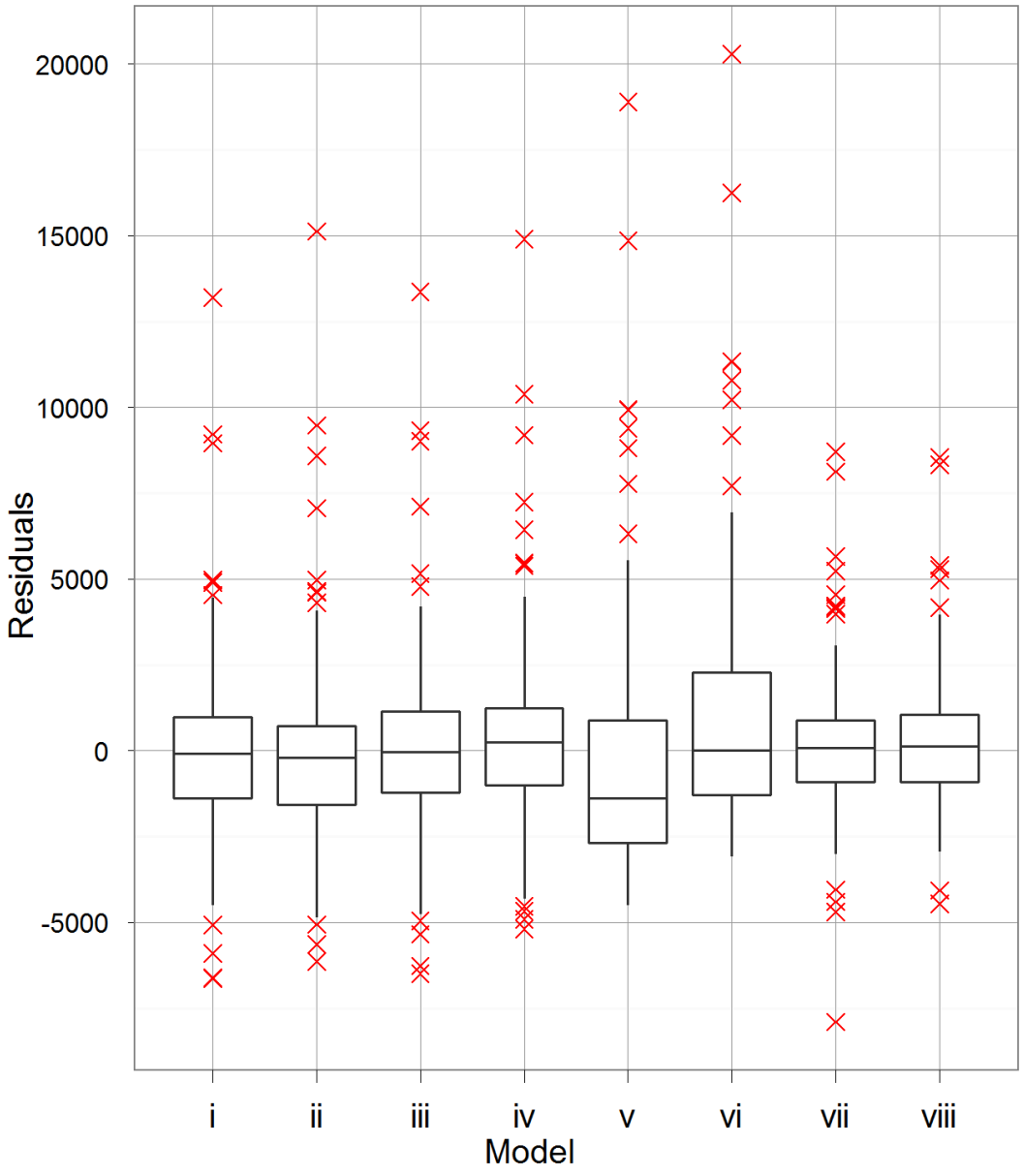


Figure 5.2: Boxplot Residuals Desharnais

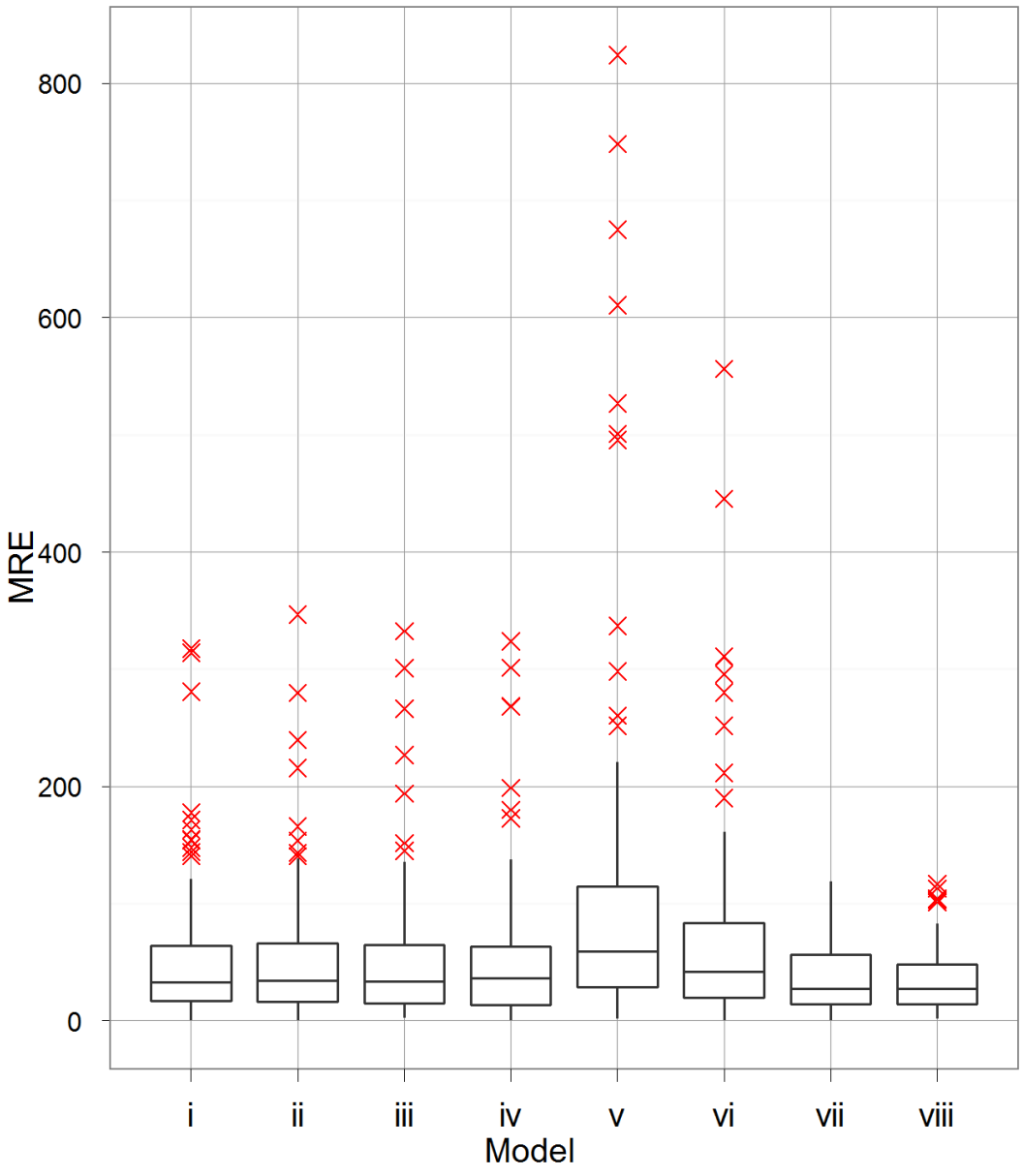


Figure 5.3: Boxplot MREs Desharnais

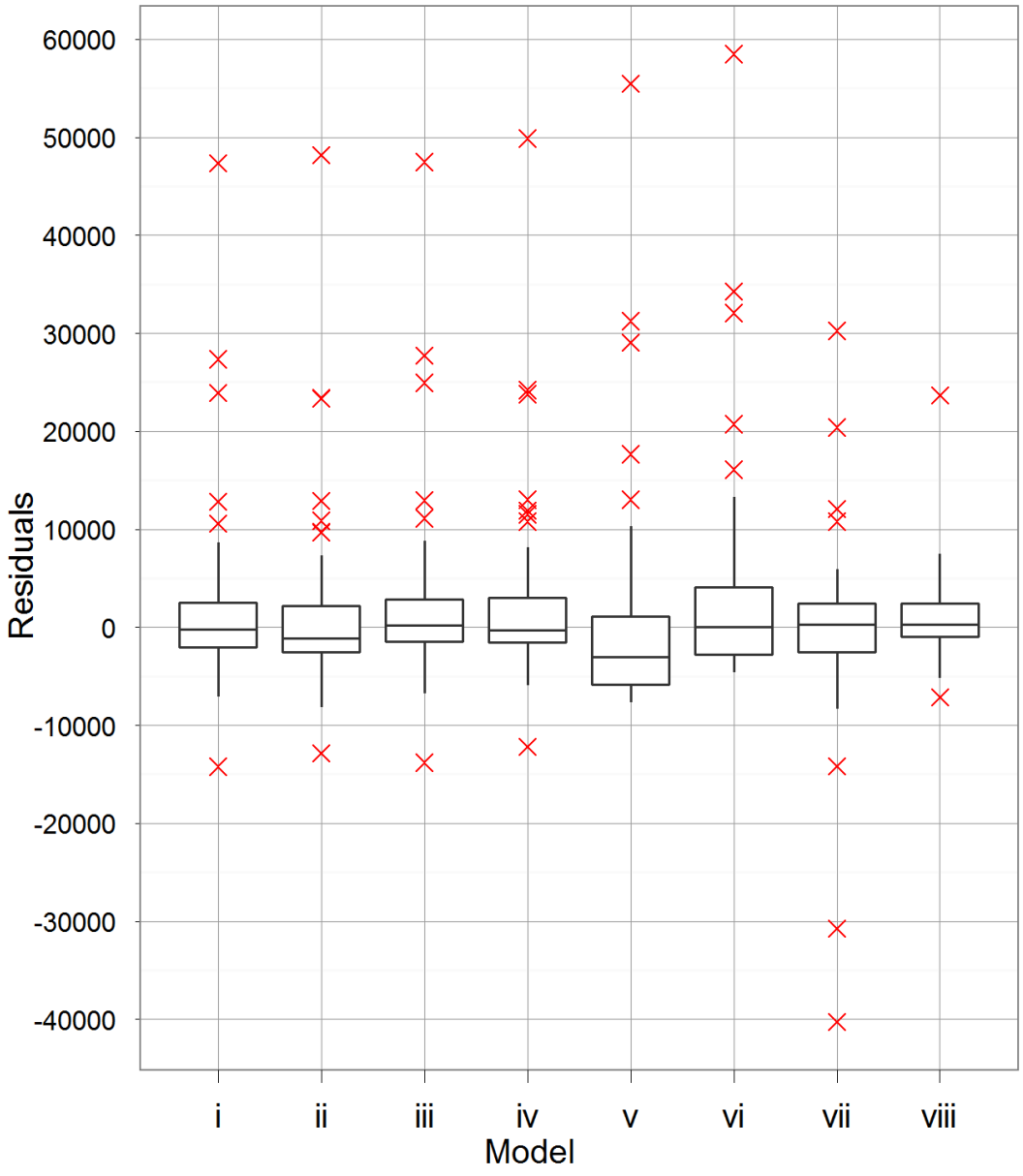


Figure 5.4: Boxplot Residuals Maxwell

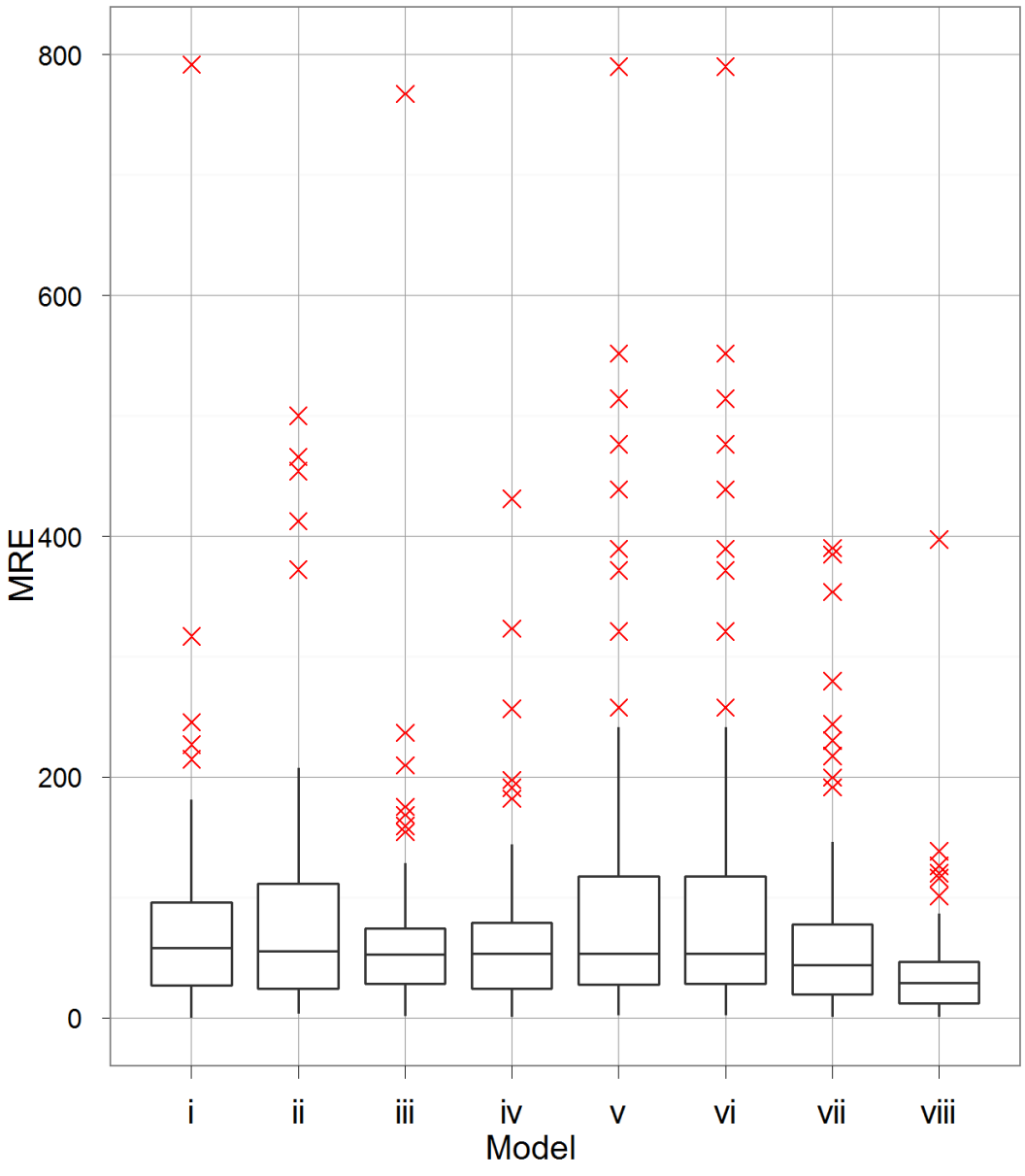


Figure 5.5: Boxplot MREs Maxwell

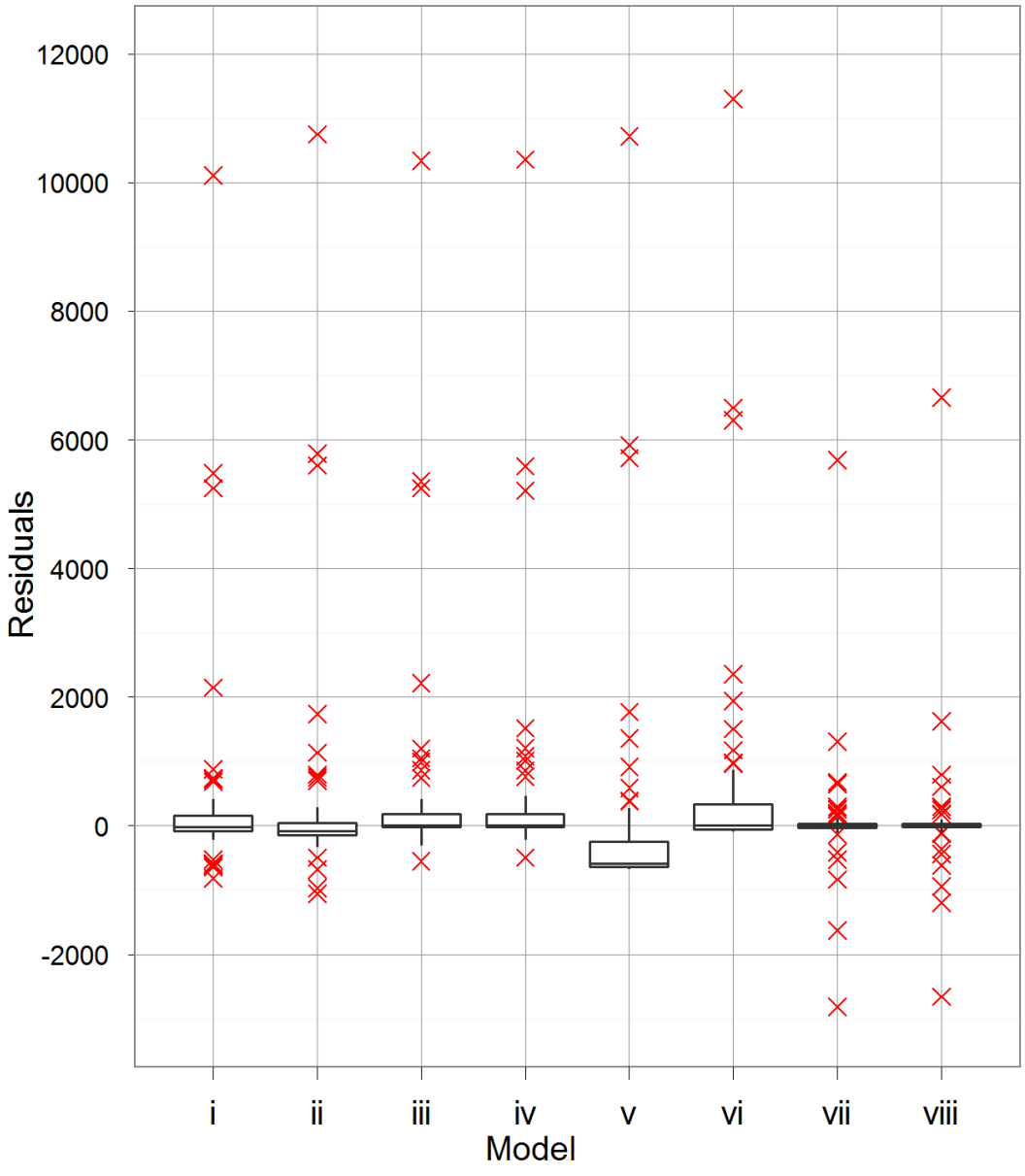


Figure 5.6: Boxplot Residuals Cocomo81

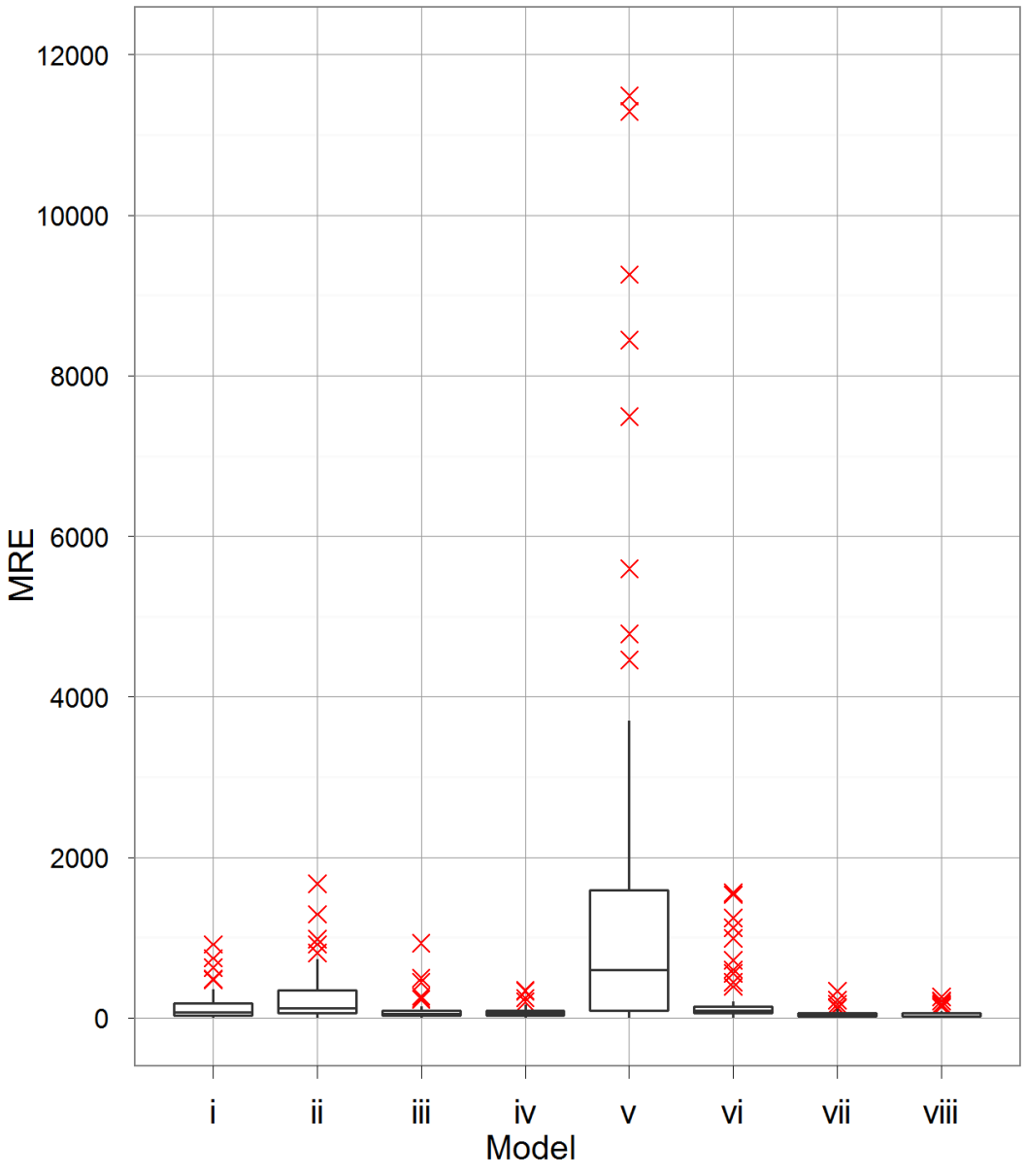


Figure 5.7: Boxplot MREs Cocomo81

APPENDIX B - RESUMO (PORTUGUÊS)

Introdução

Modelos de predição de software buscam realizar estimativas para novos projetos baseando-se em conhecimento que pode ser obtido a partir de especialistas, de dados ou de uma abordagem híbrida integrando ambas as formas. A área da predição de *software* teve seu início na década de sessenta e as primeiras abordagens consistiam em conhecimento de especialistas. Logo a seguir, foram propostos os primeiros modelos formais. O mais conhecido deles é o COCOMO (BOEHM, 1981). Estes modelos baseam-se em uma fórmula que relaciona o tamanho do projeto com o esforço necessário para realizá-lo. Uma revisão detalhada sobre esses modelos e sobre os trabalhos pioneiros da estimativa de software pode ser encontrada em Boehm, Abts e Chulani (2000).

No caso de conhecimento baseado em dados, os modelos fazem uso de bases de dados de projetos de software. Estas bases de dados contêm informação sobre projetos já concluídos. O modelo leva em conta as características do novo projeto, isto é, o projeto o qual se quer estimar, e realiza uma predição baseando-se nos projetos armazenados na base de dados. A expectativa subjacente é a de que os padrões presentes nos projetos concluídos se repetirão no novo projeto.

Na literatura, a terminologia 'predição de software' é encontrada assim como 'estimativa de software'. Além disso, 'estimativa de custo' é usada como sinônimo de 'estimativa de esforço' pois estas variáveis são bastante proximamente relacionadas. Por fim, existe uma distinção entre 'estimativa de esforço de desenvolvimento' e 'estimativa de esforço de manutenção'. Neste estudo, os dados usados são de desenvolvimento de software.

A variável a ser estimada pelo modelo de predição é conhecida como variável dependente ou variável resposta. Os atributos, isto é, as variáveis usadas pelo modelo para estimar a variável resposta são chamadas variáveis explanatórias ou variáveis independentes. As variáveis tipicamente encontradas na literatura são relacionadas ao custo ou à qualidade (FENTON; RADLINSKI, 2009).

A partir da década de noventa, técnicas de aprendizagem de máquina começaram a ser propostas. Uma revisão sistemática sobre estas técnicas na predição de esforço software pode ser encontrada em Wen et al. (2012). Uma destas técnicas de aprendizagem de máquina são as Redes Bayesianas, à qual este estudo se dedica. As Redes Bayesianas foram inicialmente propostas para a predição de qualidade por Fenton e Neil (1999). Este estudo aponta as vantagens desta técnica de modelagem sobre as técnicas estabelecidas na área até então, mencionando a possibilidade de incorporar relações de causalidade e o conhecimento subjetivo de especialistas como grandes atrativos para a área, dadas as incertezas dos projetos de *software*. Desde então, diversos estudos foram publicados empregando Redes Bayesianas e frequentemente utilizando a possibilidade de incorporar

variáveis subjetivas. Alguns estudos empregam Redes Bayesianas baseadas unicamente em dados como é feito neste estudo. Uma revisão sobre a aplicação de Redes Bayesianas na predição de esforço de software pode ser encontrada em Radlinski (2010).

Esta área de pesquisa tem sofrido com inconsistências e contradições nos resultados de estudos. São diversos os estudos que abordam este assunto, identificando causas e provendo orientações para novas pesquisas, e.g., Korte e Port (2008), Mair e Shepperd (2005), Shepperd e Macdonell (2012). Parte destas inconsistências provém de diferenças nos procedimentos empíricos e bases de dados usadas. Uma situação comum nos resultados é a instabilidades do *ranking* das técnicas dependendo da base de dados, isto é, ocorre a inversão do *ranking* dependendo de qual base de dados é utilizada. Com esta observação, foi recomendado o uso de mais de duas bases de dados. Ainda sobre a questão dos dados, vários estudos comparam o uso de bases de dados locais (bases de dados contendo projetos de uma única empresa) com bases de dados multi-empresas (*cross-company*, i.e., bases de dados com projetos de várias empresas diferentes). Em Kitchenham, Mendes e Travassos (2007) há uma revisão sistemática sobre o assunto. Os resultados foram inconclusivos devido a diferenças nos estudos comparativos, mas os autores observam que nos estudos em que foram utilizados bases de dados pequenas e validação cruzada *leave-one-out* o uso de dados locais foi significativamente mais vantajoso. Os autores também concluem que ficou claro que para algumas empresas seria benéfico o uso de bases de dados multi-empresas enquanto para outras o uso de dados locais é necessário.

Uma outra fonte de incosistências que foi investigada é o assunto das métricas de avaliação dos modelos. Diversos estudos, e.g., Foss et al. (2003), Myrtveit, Stensrud e Shepperd (2005), Korte e Port (2008), identificaram esta como uma razão para as incosistências em resultados de estudos comparativos. Foi identificado que cada métrica tem alguma limitação, tendendo a favorecer certos modelos, e causando a inversão de *ranking* dependendo da métrica usada. Particularmente, a métrica MMRE foi criticada devido a sua suscetibilidade a desvios e uma tendência a favorecer modelos que subestimam a variável de saída, i.e., o esforço de desenvolvimento.

Uma barreira para a transparência dos estudos tem sido a falta de bases de dados públicas, já que o emprego de bases de dados proprietárias inibe a replicação de experimentos e confirmação de resultados. O repositório PROMISE (MENZIES et al., 2012) foi proposto com o intuito de amenizar esta barreira. Bases de dados são disponibilizadas promovendo a replicação de experimentos e análises de resultados.

Embora as Redes Bayesianas tenham sido promovidas nesta área de pesquisa, seu uso ainda é relativamente limitado possivelmente devido a algumas dificuldades práticas. Neste contexto, este estudo procurou avaliar o emprego de Redes Bayesianas baseadas em dados na predição de esforço de software, considerando a preparação de dados necessária, discutindo as limitações atuais e possibilidades de melhoras. Mesmo que Redes Bayesianas puramente baseadas em dados, como as que são investigadas neste estudo, não venham a ser a melhor forma de aplicá-las, a otimização da exploração dos dados é um passo importante para o desenvolvimento desta técnica.

Neste resumo, é exposta uma breve descrição dos experimentos na próxima seção, para a seguir analisar-se os resultados.

Procedimentos Empíricos

As Redes Bayesianas são avaliadas através da comparação com a regressão linear (usando o método dos mínimos quadrados) com uma transformação logarítmica sobre os

dados. Esta técnica estatística foi recentemente identificada por Dejaeger et al. (2012), em um estudo abrangente, como a técnica mais precisa. Este estudo comparou treze técnicas sobre nove bases de dados. Ressalta-se que se está comparando um estimador discreto, i.e., um classificador, a um estimador contínuo. Isto é feito através da transformação das saídas das Redes Bayesianas para valores contínuos utilizando uma variante do método proposto por Pendharkar, Subramanian e Rodger (2005). Esta variante proporcionou uma melhora significativa na precisão obtida com a conversão numérica das predições das Redes Bayesianas. A saída das Redes Bayesianas é uma distribuição de probabilidades para cada uma das classes. O método de conversão numérica consiste em multiplicar a probabilidade ρ de cada classe pelo valor mediano *Md* de cada uma destas classes, como mostrado na fórmula abaixo. Os valores medianos de cada uma das classes de esforço são mostrados nas tabelas 5.1, 5.2 e 5.3. O método original de Pendharkar utiliza a média de cada classe. As tabelas na seção 2.5 mostram as melhoras obtidas com o uso da mediana comparando com os resultados de um estudo preliminar em que usamos a média (TIERNO; NUNES, 2012).

$$Esforco = \rho_{Muitobaixo}\boldsymbol{Md}_{Muitobaixo} + \rho_{Baixo}\boldsymbol{Md}_{Baixo} + ... + \rho_{Muitoalto}\boldsymbol{Md}_{Muitoalto}. \quad (5.1)$$

Neste estudo, avalia-se também o efeito da execução de uma transformação logarítmica antes da construção da Rede Bayesiana. Esta idéia surgiu com a constatação das melhoras obtidas através desta transformação para a regressão linear. Constatou-se, após a execução destes experimentos, que esta combinação foi utilizada por Fernández-Diego e Torralba-Martínez (2012), no entanto os efeitos e a validade desta combinação não foram discutidos. Até então, três sistemas de predição foram mencionados: Redes Bayesianas, Redes Bayesianas com transformação logarítmica e regressão linear.

Os experimentos incluem variantes dos sistemas de predição supracitados com a aplicação de uma técnica de seleção de variáveis, aumentando assim o número de sistemas de predição para seis.

Finalmente, são incluídos na comparação modelos base baseados na média e na mediana. A inclusão de modelos base para a comparação como estes é sugerida em diversos estudos com a finalidade de melhor avaliar a eficácia das outras técnicas, estabelecendo assim um padrão mínimo de precisão.

Uma abstração dos experimentos é mostrada em Fig. 3.1. As diferentes versões das bases de dados e os modelos de Redes Bayesianas com transformação logarítmica foram omitidos da figura para mantê-la intuitiva. Alternativamente o leitor pode conferir uma versão da figura que inclui estes dois sistemas no apêndice. A transformação logarítmica, quando aplicada, ocorre antes da seleção de variáveis e no caso das Redes Bayesianas, antes da discretização.

Destaca-se que a validação destes experimentos é considerada confiável. Foi utilizada a validação cruzada *10-fold* e uma combinação robusta de métricas, sendo que as métricas MAR e MBRE são complementares as métricas baseadas em MRE.

Bases de dados e preparação dos dados

Neste trabalho, foram utilizadas bases de dados disponibilizadas no repositório PROMISE (MENZIES et al., 2012). Estas são as bases de dados 'Desharnais', 'Maxwell' e 'Cocomo81'.

Os histogramas das figuras Fig. 3.2, Fig. 3.3 e Fig. 3.4 ilustram a distribuição dos dados na variável de saída (esforço). Esforço é medido em horas de trabalho (Person-

Hours) nas bases de dados Desharnais e Maxwell e em meses de trabalho (Person-months) na base de dados Cocomo81. Nos três casos as variáveis tem obliquidade positiva, isto é, variáveis com a maioria dos registros concentrados nos valores mais baixos e uma quantidade pequena de registros nos valores mais altos. *Obliquidade* ou *assimetria* é uma característica comum em bases de dados de projetos de software.

Esta característica proporciona algumas dificuldades para a modelagem. Para a execução da regressão linear, boas práticas estatísticas ditam que a distribuição das variáveis deve aproximar-se a uma distribuição Gaussiana, razão pela qual é realizada uma transformação logarítmica. Com relação às Redes Bayesianas, isto também é um problema porque dificulta a discretização. Os intervalos discretizados tendem a ficar bastante assimétricos. Neste cenário, a discretização das larguras iguais (*equal-widths*) pode gerar classes vazias e distribuir a grande maioria dos registro em uma ou duas classes. Neste caso, a validação seria bastante duvidosa já que se quase todos os registros estão distribuídos em uma ou duas classes o algoritmo de aprendizagem não poderá identificar padrões significativos e inclusive, dificilmente errará as predições. Uma taxa de acertos bastante alta não seria surpreendente, no entanto seria de fato, fútil. Isto ocorreu na replicação dos experimentos com a Rede Bayesiana exposta como prova de conceito no trabalho de Bibi et al. (2010).

Quando gerentes de projeto executam as predições, eles não sabem por exemplo, o quanto o projeto vai durar, muito embora possam ter uma estimativa. Logo, variáveis como esta são geralmente excluídas dos modelos. Isto é prática padrão na área de predição de software, embora raramente seja mencionada.

Pré-processamento

Antes de aplicar regressão linear as variáveis assimétricas passaram por uma transformação logarítmica com o objetivo de aproximá-las a uma distribuição normal.

Antes da aprendizagem da Rede Bayesiana os dados tiveram que ser discretizados. Esta é uma das etapas de pré-processamento mais comuns na mineração de dados. Para obter classes relevantes, optamos pela discretização das frequências iguais (*equal- frequencies*) (TAN; STEINBACH; KUMAR, 2005). As variáveis contínuas foram discretizadas em cinco intervalos. Não existem regras estritas sobre isto. Cinco é um número comum de classes encontrado em estudos de predição de software, e.g., Mendes e Mosley (2008), Radlinski e Hoffmann (2010). Por um lado, quanto maior o número de classes mais precisas podem ser as predições. Por outro lado, um número grande de classes exige uma quantidade de dados maior. Considerando que as bases de dados de projetos de software são geralmente pequenas (a maioria costuma ter menos de cem registros), um número grande de classes tende a ser inviável. Em Fernández-Diego e Torralba-Martínez (2012) há um artigo curto sobre o assunto que aponta que a discretização das frequências iguais produziu resultados melhores. Este tópico foi muito pouco explorado na área e mais investigações são necessárias.

Um desenvolvimento técnico recente, que não pôde ser avaliado neste estudo, é a discretização dinâmica (NEIL; TAILOR; MARQUEZ, 2007). Este parece ser um desenvolvimento bastante interessante já que a discretização tem efeito direto sobre a precisão do modelo. Os autores relataram significativas melhoras na precisão e também que a técnica proposta flexibiliza a construção de Redes Bayesianas com variáveis subjetivas. Esta inovação foi implementada na ferramenta comercial Agena Risk. Entretanto, este recurso ainda não está disponível em outras ferramentas.

As tabelas 5.1, e mostram os intervalos e os valores usados para a conversão numérica.

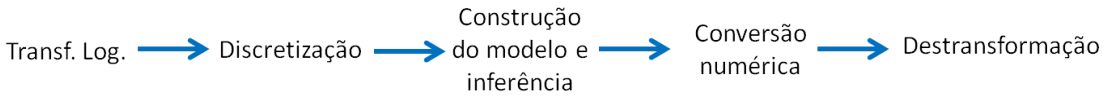


Figure 5.8: Predição numérica com Redes Bayesianas e transformação logarítmica.

A unidade de medida para as bases de dados Desharnais e Maxwell é horas de trabalho (Person-Hours). Para a base de dados Cocomo81 é meses de trabalho (Person-Months) que equivalem a 152 horas.

Table 5.1: Classes de esforço discretizadas da base de dados Desharnais.

Categoria de Esforço	Intervalo	Mediana	Mediana transf. Log.
Muito Baixo	<= 2161,5	1211	7,0981
Baixo	2161,5 < x <= 3062,5	2534	7,8375
Médio	3062,5 < x <=4035,5	3636,5	8,1987
Alto	4035,5 < x <= 7553	5635	8,6367
Muito alto	> 7553	10969	9,3021

Pode-se observar o efeito da assimetria dos dados sobre o tamanho dos intervalos. As classes de maior esforço têm intervalos muito mais largos que as classes de menor esforço. Este padrão está diretamente relacionado.

Table 5.2: Classes de esforço discretizadas da base de dados Maxwell.

Categoria de esforço	Intervalo	Mediana	Mediana transf. Log.
Muito Baixo	<=1750	1080	6,9845
Baixo	1750 < x <= 4187	2957	7,9919
Médio	4187 < x <= 6960,5	5189,5	8,5542
Alto	6960,5 < x <= 10735	8710	9,0722
Muito Alto	>10735	16776	9,7223

A partir desta observação, resolveu-se experimentar realizar uma transformação logarítmica sobre os dados para tornar as classes mais equilibradas. O efeito da aplicação desta técnica em conjunto com as Redes Bayesianas não foi avaliada anteriormente, embora Fernández-Diego e Torralba-Martínez (2012) recentemente a tenham aplicado. A inclusão desta transformação logarítmica torna um pouco mais complexa a conversão numérica. Para realização da conversão numérica devem ser usadas as medianas das classes obtidas com os dados já processados com a transformação logarítmica. Ao final do processo, obtida a predição numérica com os dados transformados, pode-se então realizar a destransformação da predição obtida. Este processo é ilustrado na figura 5.8 .

Outra etapa de pré-processamento empregada nestes experimentos é a seleção de variáveis. A ferramenta WEKA contém uma grande variedade de algoritmos para esta tarefa. Foi decidido aplicar uma abordagem *Wrapper* com algoritmo de busca *BestFirst* e utilizando um procedimento de validação cruzada 5-*fold* para estimar a precisão dos modelos. Este algoritmo produziu bons resultados nos experimentos e também foi bem avaliado em outros estudos (HALL; HOLMES, 2003) (KOHAVI; JOHN, 1997) (MENZIES et al., 2010).

Table 5.3: Classes de esforço discretizadas da base de dados Cocomo81

Categoria de esforço	Intervalo	Mediana	Mediana transf. Log.
Muito baixo	<= 34,5	9	2,1972
Baixo	34,5 < x <=76	47	3,8501
Médio	76 < x <=188,5	102	4,6242
Alto	188,5 < x <= 653,5	321	5,7714
Muito alto	> 653,5	1436	7,2630

É interessante observar que a aplicação da seleção de variáveis da forma que é apresentada na mineração de dados não é mencionada como prática estatística padrão. No entanto bons resultados foram obtidos com esta combinação. Dejaeger et al. (2012) também usaram a seleção de variáveis em conjunto com as técnicas de regressão linear. Foi observado que os resultados obtidos no estudo citado são muito similares para o modelo de regressão OLS (mínimos quadrados) independente da aplicação de seleção de variáveis. Isto evidencia que está ocorrendo alguma forma de seleção de variáveis no modelo sem seleção de variáveis (os hiperparâmetros da regressão linear do WEKA oferecem esta opção). Em contrapartida, neste estudo decidiu-se não empregar nenhuma forma de seleção de variáveis para esse modelo, já que de outra maneira, os modelos seriam praticamente os mesmos.

Resultados

Os resultados são mostrados em matrizes de confusão e tabelas. Nas figuras 4.1, 4.2 e 4.3 mostram as predições para as bases de dados Desharnais, Maxwell e Cocomo81. Em cada figura, as matrizes de confusão de cima mostram as predições para Redes Bayesianas sem a aplicação de seleção de variáveis e as matrizes de baixo mostram as predições para Redes Bayesianas com a aplicação de seleção de variáveis. Esta é a maneira tradicional de mostrar resultados para classificadores. As tabelas contêm os resultados de acordo com as três métricas baseadas em MRE, a métrica MBRE e a métrica MAR. As taxas de acerto das Redes Bayesianas também foram incluídas. Além disso, para a visualização foram incluídos no apêndice os diagramas de caixa para MRE e para os residuais.

Pode-se observar de maneira geral nas matrizes de confusão que há um aumento de predições corretas com a aplicação de seleção de variáveis. No entanto, o número de predições amplamente incorretas (i.e., predições distando duas classes ou mais da classes correta) não diminuiu. Na base de dados Maxwell houve um aumento de treze para dezesseis predições amplamente incorretas.

As variáveis identificadas pela seleção de variáveis na base Desharnais foram as variáveis mudas para ‘Language’ e a variável ‘PointsAdjust’ que é uma medida de tamanho em pontos de função. Para as Redes Bayesianas as variáveis selecionadas foram ‘ManagerExp’ (experiência do gerente), ‘Entities’ e ‘PointsNonAdjust’.

Na base de dados Maxwell, a seleção de variáveis para regressão linear identificou seis variáveis: ‘Ifc’ (interface de usuário), ‘T03’ (disponibilidade de pessoal), ‘T08’ (volatilidade dos requerimentos), ‘T09’ (requerimentos de qualidade), ‘Time’ (uma variável relacionada ao ano de início do projeto) e ‘Size’ (tamanho em pontos de função). Para as Redes Bayesianas o algoritmo identificou nove variáveis: ‘Har’ (plataforma de hardware), ‘Dba’ (base de dados), ‘Source’ (variável binária apontando emprego de *outsourcing*),

'Nlan' (número de linguagens de desenvolvimento diferentes), 'T03' (disponibilidade de pessoal), 'T12' (habilidade de análise da equipe), 'T13' (conhecimento de aplicação da equipe), 'T15' (habilidade colaborativa da equipe) e 'Size' (tamanho em pontos de função).

Na base de dados Cocomo81 foram selecionadas dez variáveis para a regressão linear: 'rely' (confiabilidade requerida do software), 'time' (restrição de tempo para CPU), 'virt' (volatilidade de máquina), 'acap' (capacidade dos analistas), 'pcap' (capacidade dos programadores), 'vexp' (experiência em máquina virtual), 'modp' (práticas modernas de programação), 'tool' (uso de ferramentas de *software*), 'sced' (limitação de cronograma) e 'loc' (uma variável de tamanho que segundo o autor pode ser computada a partir de uma análise de pontos de função).

A tabela 5.4 mostra os resultados de acordo com as métricas numéricas. Na base de dados Desharnais há uma melhora clara nas taxas de acerto com a aplicação da seleção de variáveis. Entretanto, considerando as métricas numéricas esta melhora não é confirmada, à exceção da métrica Pred. Esta assemelha-se à taxa de acertos com a característica de apenas considerar as predições precisas e ignorar as predições imprecisas. Isto mostra que houve um maior número de predições precisas mas que a imprecisão das predições imprecisas aumentou. Para a regressão linear constata-se que houve uma ligeira melhora nas predições, embora a métrica Pred não mostre. As melhoras foram pequenas porque esta base de dados contém apenas nove variáveis. Em bases de dados com poucas variáveis as melhoras esperadas com a aplicação da seleção variáveis é pequena.

A precisão das Redes Bayesianas com transformação logarítmica dos dados foi praticamente a mesma que quando sem a transformação. O pré-processamento de dados nesta base de dados surtiu apenas efeitos minúsculos. Nesta base de dados o desempenho das Redes Bayesianas foi razoável mas com uma propensão um pouco maior para grandes imprecisões. Claramente os modelos foram mais precisos que os modelos base da média e da mediana. Estes resultados podem também ser visualizados nos diagramas de caixa mostrados no apêndice.

Table 5.4: Desempenho dos modelos na base de dados Desharnais.

Sistema	Taxa de acertos	MMRE	MdMRE	Pred	MAR	MBRE
BNs	46,91%	57,23	32,66	33,33	2153,52	65,83
BNs + FSS	54,32%	56,18	34,16	39,5	2133,84	64,52
BNs + log	44,44%	56,37	33,61	34,57	2128,65	67,38
BNs + log + FSS	48,15%	57,64	36,42	38,27	2165,47	72,19
OLS + log	-	37,62	29,19	46,75	1731,53	48,04
OLS + log + FSS	-	34,24	27,66	45,45	1567,93	42,54
Média	-	121,66	59,49	18,51	3161,52	140,04
Mediana	-	78,46	42,03	29,62	2861,53	120,42

A tabela 5.5 mostra os resultados para a base de dados Maxwell. Sendo esta a base de dados com maior número de variáveis deste estudo, é provável que tenha um maior número de variáveis redundantes ou irrelevantes, e logo que seja a base de dados em que a aplicação de seleção de variáveis tenha o maior impacto. Esta expectativa é correspondida para a técnica de regressão linear. Os residuais foram reduzidos praticamente pela metade e todas as demais métricas mostram grandes melhoras também. Os resultados obtidos aqui com a aplicação de seleção de variáveis foram significativamente mais precisos do

que os obtidos por Dejaeger et al. (2012) na mesma base de dados.

Entretanto, como ocorreu na base de dados Desharnais, a precisão das Redes Bayesianas não melhorou convincentemente com a aplicação da seleção de variáveis, registrando apenas pequenas melhoras nas métricas numéricas. Há um claro aumento na taxa de acertos e na métrica Pred, mas a melhora apenas pequena nas demais métricas indica que os erros também foram maiores. É interessante observar que nos dados sem seleção de variáveis as Redes Bayesianas tiveram um desempenho próximo ao da regressão linear, o que enfatiza as melhoras em precisão que as Redes Bayesianas estão deixando de alcançar.

Com relação a transformação logarítmica, os resultados mostram pequenas melhoras para as predições das Redes Bayesianas, diferentemente da base de dados Desharnais, em que não houve melhora perceptível.

Todos os modelos foram mais precisos que os modelos base, embora apenas a regressão linear com transformação logarítmica os tenha superado por margem bastante grande.

Table 5.5: Desempenho dos modelos na base de dados Maxwell.

Sistema	Taxa de acertos	MMRE	MdMRE	Pred	MAR	MBRE
BNs	40,32%	86,18	58,77	24,19	4655,29	110,48
BNs + FSS	51,61%	97,5	55,99	27,41	4854,74	122,19
BNs + log	40,32%	73,41	52,72	19,35	4550,90	104,54
BNs + log + FSS	51,61%	70,67	53,88	25,81	4576,05	106,44
OLS + log	-	76,86	43,78	30	4932,6	101,19
OLS + log + FSS	-	42,57	28,62	40	2500,04	52,28
Média	-	119,67	52,96	19,35	5616,54	225,64
Mediana	-	108,95	66,28	20,97	5654,11	180,91

A tabela 5.6 mostra os resultados para a base de dados Cocomo81. Nesta base de dados a transformação logarítmica surtiu efeitos mais significativos. A grande melhora sob a métrica MMRE sugere uma diminuição de estimativas exageradas.

A aplicação da seleção de variáveis trouxe uma melhora na precisão da regressão linear, embora não tão significativa como na base de dados Maxwell. No caso das Redes Bayesianas repetiu-se o padrão de melhoras na taxa de acertos que não se confirmam com as métricas numéricas. Ocorre inclusive uma piora significativa na precisão das Redes Bayesianas sem transformação logarítmica.

Pode-se observar que a precisão das Redes Bayesianas, em comparação com a regressão linear, piorou progressivamente com o aumento da obliquidade dos dados, embora esta piora possa ser um pouco amenizada com a aplicação da transformação logarítmica.

Também pode-se observar nestes resultados que a seleção de variáveis melhorou clara e consistentemente a taxa de acerto das Redes Bayesianas e a precisão da regressão linear. Este efeito é mais pronunciado na base de dados Maxwell que é a base de dados com o maior número de variáveis. Isto enfatiza a importância de aplicar a seleção de variáveis, especialmente em bases de dados que contêm muitas variáveis. Estes resultados também dão destaque ao fato de que muitas variáveis nas bases de dados de projeto de software podem ter pouco valor preditivo inclusive prejudicando a obtenção a precisão do modelo. Portanto, coletar uma quantidade menor de variáveis enfocando na qualidade dos dados talvez seja uma abordagem mais adequada para predições baseadas em dados. Este resultado é uma confirmação de resultados de outros estudos, e.g., Chen et al. (2005),

Table 5.6: Desempenho dos modelos na base de dados Cocomo81.

Sistema	Taxa de acertos	MMRE	MdMRE	Pred	MAR	MBRE
BNs	50,79%	134,85	58,64	25,81	551,95	197,82
BNs + FSS	55,56%	270,64	130,37	9,68	606,22	336,39
BNs + log	52,38%	91,19	53,64	19,35	536,54	233,15
BNs + log + FSS	55,56%	76,94	64,93	25,81	530,61	212,73
OLS + log	-	46,6	30,49	44,44	278	61,83
OLS + log + FSS	-	44,28	22,98	53,96	297,47	55,97
Média	-	1775,35	571,16	4,76	891,64	1905,81
Mediana	-	235,42	86,25	15,87	642,63	842,24

Radlinski e Hoffman (2010) e Dejaeger et al. (2012).

Entretanto, apesar destas evidentes melhoras, pode-se constatar que o efeito da seleção de variáveis nas predições das Redes Bayesianas quando medidas pelas métricas numéricas foi pequeno ou até mesmo negativo, como ocorreu nas bases de dados Maxwell e Cocomo81. De acordo com a literatura de mineração de dados, a abordagem *wrapper* utiliza a medida de precisão do próprio algoritmo de aprendizagem para avaliar o conjunto de variáveis selecionadas (HALL; HOLMES, 2003) (TAN; STEINBACH; KUMAR, 2005). É evidente que as Redes Bayesianas estão favorecendo as taxas de acerto. Isto levanta questões sobre a validade das taxas de acerto como uma medida de precisão, ou pelo menos mostra a sua limitação. Taxas de acerto mais altas são alcançadas às custas de margens de erro maiores, isto é, apesar do maior número de acertos, as margens de erro das predições erradas são também maiores. Então questiona-se, uma taxa de acertos maior realmente reflete um modelo mais preciso? Nas três bases de dados as Redes Bayesianas deixaram de obter as melhoras através da seleção de variáveis. A diferença nas bases de dados Maxwell e Cocomo81 poderia ser bastante significativa se a seleção de modelos levasse em conta esta conversão numérica.

A partir desta observação, uma sugestão interessante de desenvolvimento para as Redes Bayesianas seria que os algoritmos e ferramentas incorporem esta conversão numérica e a usem para medir a precisão, ao invés de usar taxas de acerto ou taxas de erro. Então, mesmo que obtendo um menor aumento nas taxas de acerto, a seleção de variáveis tornaria os modelos mais precisos no geral. Da maneira que estão, a potencial melhora esperada com a seleção de variáveis parece estar sendo disperdiçada na busca de taxas de acerto maiores. Alternativamente, uma outra sugestão para pesquisas futuras é experimentar com outros algoritmos de busca, métodos de avaliação das redes e estimadores das tabelas de probabilidade já que este estudo restringiu-se ao algoritmo de busca K2 (COOPER; HERSKOVITS, 1992) com o método Bayes para avaliação de redes e o estimador simples para estimar as tabelas de probabilidades.

Além das observações sobre a seleção de variáveis, pode-se constatar uma tendência nestes resultados. A precisão das Redes Bayesianas se degrada com o aumento da obliquidade dos dados. O melhor desempenho das Redes Bayesianas foi alcançado na base de dados menos oblíqua, isto é, a base de dados Desharnais. A obliquidade dos dados afeta as Redes Bayesianas de duas maneiras: a) Quanto maior a obliquidade mais desequilibradas serão as classes gerada pela discretização e as classes de esforço maiores se tornam muito imprecisas. Pode-se constatar nas tabelas de discretização mostradas na seção anterior que as classes de maior esforço abrangem um intervalo muito maior do que as classes

menores. Um exemplo disto é a maior classe de esforço da base de dados Maxwell em que compreende valores entre 10000 e 64000 horas de trabalho sendo maior que todas as outras classes juntas. b) Além deste efeito na discretização, há também um efeito na conversão numérica já que mesmo uma pequena probabilidade para a classe maior afeta a conversão numérica bastante significativamente.

Então, observa-se que boa parte da imprecisão das Redes Bayesianas pode ser atribuída ao processo de discretização. Neste cenário em que os dados são muito oblíquos, a discretização das frequências iguais gera classes com larguras muio diferentes. A alternativa da discretização das larguras iguais gera resultados irrelevantes porque várias classes ficam vazias e o modelo prediz o óbvio, isto é, quase sempre as classes de menor esforço já que esta contém a grande maioria dos registros. Neste caso, taxas de acerto muito altas não só não surpreendem, como são prováveis. Mas a menos que uma transformação logarítmica seja aplicada nos dados, predições baseadas em dados oblíquos e a discretização das larguras iguais são fúteis e trazem resultados duvidosos. As técnicas de discretização são um assunto que carece de investigações na área da predição de software.

Apesar deste desempenho parecer desencorajador, especialmente nas bases mais assimétricas, mesmo na muito assimétrica base de dados Cocomo81, as Redes Bayesianas foram mais precisas que os modelos base. Esta é uma perspectiva mais positiva do que a publicada em Mendes e Mosley (2008), já que neste estudo as diversas Redes Bayesianas propostas dificilmente superavam os mesmos modelos base. Alerta-se ao leitor de que o desempenho das Redes Bayesianas não é pior do que o de muitas outras técnicas de aprendizagem de máquina propostas. Isto pode ser verificado em estudos como Radlinski e Hoffman (2010), onde em duas das quatro bases de dados as Redes Bayesianas estavam entre os classificadores com maiores taxas de acerto.

A partir dos estudos da literatura e destes experimentos, observa-se que parece ser difícil superar a regressão linear em termos de precisão quando esta é apropriadamente aplicada. Estes resultados da regressão linear confirmam a conclusão de Dejaeger et al. (2012) e parcialmente os resultados de Mendes e Mosley (2008). Mas observa-se que embora a regressão linear seja mais precisa, deve-se observar que esta técnica estatística já está bastante otimizada, enquanto outras técnicas como as Redes Bayesianas ainda têm possibilidades de melhoras e estão em constante desenvolvimento. Com evoluções na teoria das Redes Bayesianas e a atualização das ferramentas com estes desenvolvimentos, predições mais precisas serão possíveis. Se as Redes Bayesianas puramente baseadas em dados, como as investigadas neste estudo, alcançarem a regressão linear em termos de precisão, isto será bastante benéfico para esta técnica já que esta deixaria de ser mais imprecisa quando puramente baseada em dados e ainda estaria contando com as vantagens de experimentação e de integração de conhecimento de especialistas.

Especificamente, observou-se possibilidades de melhora especialmente no que se refere a discretização e otimização da seleção de variáveis. Além disso, constata-se nas pesquisas que existem progressos em relação a suporte a variáveis ordinais e numéricas que poderão vir a possibilitar predições mais precisas.

As ferramentas de Redes Bayesians são atualmente uma limitação (RADLINSKI, 2010). Seria interessante analisar que melhoras pode-se obter com técnicas como a discretização dinâmica proposta em Neil, Tailor e Marquez (2007) e desenvolvida na ferramenta comercial AgenaRisk. A vantagem da ferramenta WEKA consiste nas possibilidades extensas de validação, mas em contrapartida, esta ferramenta ainda não tem desenvolvimentos como o recém citado. Um desenvolvimento interessante para as ferramentas seria a incorporação da conversão numérica discutida neste estudo. Esta conversão não

está automatizada nas ferramentas e sua execução pode ser trabalhosa. Além distom, existem desenvolvimentos relacionados ao suporte de variáveis ordinais e contínuas. Estes desenvolvimentos podem abrir possibilidades de melhoras na precisão.

É importante expor resultados realistas mesmo que estes não sejam positivos para uma técnica em particular. Esta área de pesquisa sofreu nos últimos vinte anos devido a avaliações otimistas. Nos anos recentes, esforços dirigidos à correção de conclusões equivocadas e identificação das razões para os resultados conflitantes estão em ascenção, mesmo que isto exponha uma situação complicada desta área de pesquisa. É importante reconhecer a situação como tal para que seja possível encontrar saídas.

REFERENCES

BIBI, S. et al. BBN based approach for improving the software development process of an SME a case study. **J. Softw. Maint. Evol.**, New York, NY, USA, v.22, n.2, p..1–.1, Mar. 2010.

BIBI, S.; STAMELOS, I.; ANGELIS, L. Bayesian Belief Networks as a Software Productivity Estimation Tool. In: BALKAN CONFERENCE IN INFORMATICS, THESSALONIKI, 1. **Proceedings...** [S.l.: s.n.], 2003.

BOEHM, B.; ABTS, C.; CHULANI, S. **Software development cost estimation approaches - A survey**. [S.l.]: Annals of Software Engineering, 2000.

BOEHM, B. W. **Software Engineering Economics**. Englewood Cliffs, NJ: Prentice Hall, 1981.

CHEN, Z. et al. Finding the Right Data for Software Cost Modeling. **IEEE Softw.**, Los Alamitos, CA, USA, v.22, n.6, p.38–46, 2005.

COOPER, G. F.; HERSKOVITS, E. A Bayesian Method for the Induction of Probabilistic Networks from Data. **Mach. Learn.**, Hingham, MA, USA, v.9, p.309–347, Oct. 1992.

DASH, M.; LIU, H. Feature Selection for Classification. **Intelligent Data Analysis**, [S.l.], v.1, p.131–156, 1997.

DEJAEGER, K. et al. Data Mining Techniques for Software Effort Estimation: a comparative study. **IEEE Trans. Software Eng.**, [S.l.], v.38, n.2, p.375–397, 2012.

FENTON, M. N. N.; RADLINSKI, L. Software Project and Quality Modelling Using Bayesian Networks. In: **Artificial Intelligence Applications for Improved Software Engineering Development**: new prospects. (part of the advances in intelligent information technologies (aiit) book series). [S.l.]: Information Science Reference. ISBN: 978-1-60566-758-4, 2009. p.223–231. Edited by: F. Meziane and S. Vadera.

FENTON, N. E.; NEIL, M. A Critique of Software Defect Prediction Models. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.25, n.5, p.675–689, 1999.

FENTON, N.; NEIL, M.; MARQUEZ, D. **REVIEW PAPER 701 Using Bayesian networks to predict software defects and reliability**. 2007.

FERNÁNDEZ-DIEGO, M.; TORRALBA-MARTÍNEZ, J.-M. Discretization methods for NBC in effort estimation: an empirical comparison based on isbsg projects. In: ACM-IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, New York, NY, USA. **Proceedings...** ACM, 2012. p.103–106. (ESEM '12).

FINNIE, G. R.; WITTIG, G. E.; DESHARNAIS, J.-M. A comparison of software effort estimation techniques: using function points with neural networks, case-based reasoning and regression models. **J. Syst. Softw.**, New York, NY, USA, v.39, n.3, p.281–289, Dec. 1997.

FLORES, M. J. et al. Analyzing the impact of the discretization method when comparing Bayesian classifiers. In: INDUSTRIAL ENGINEERING AND OTHER APPLICATIONS OF APPLIED INTELLIGENT SYSTEMS - VOLUME PART I, 23., Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2010. p.570–579. (IEA/AIE'10).

FOSS, T. et al. A Simulation Study of the Model Evaluation Criterion MMRE. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.29, p.985–995, Nov. 2003.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **J. Mach. Learn. Res.**, [S.l.], v.3, p.1157–1182, Mar. 2003.

HALL, M. A.; HOLMES, G. Benchmarking Attribute Selection Techniques for Discrete Class Data Mining. **IEEE Trans. on Knowl. and Data Eng.**, Piscataway, NJ, USA, v.15, n.6, p.1437–1447, 2003.

HALL, M. et al. The WEKA data mining software: an update. **SIGKDD Explor. Newsl.**, New York, NY, USA, v.11, n.1, p.10–18, 2009.

JØRGENSEN, M. A review of studies on expert estimation of software development effort. **J. Syst. Softw.**, New York, NY, USA, v.70, n.1-2, p.37–60, Feb. 2004.

JØRGENSEN, M.; SHEPPERD, M. A Systematic Review of Software Development Cost Estimation Studies. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.33, n.1, p.33–53, 2007.

KITCHENHAM, B. A.; MENDES, E.; TRAVASSOS, G. H. Cross versus Within-Company Cost Estimation Studies: a systematic review. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.33, n.5, p.316–329, 2007.

KITCHENHAM, B. et al. What accuracy statistics really measure. **IEE Proceedings - Software**, [S.l.], v.148, n.3, p.81–85, 2001.

KITCHENHAM, B.; MENDES, E. Why comparative effort prediction studies may be invalid. In: INTERNATIONAL CONFERENCE ON PREDICTOR MODELS IN SOFTWARE ENGINEERING, PROMISE '09, 5., New York, NY, USA. **Proceedings...** ACM, 2009. p.1–5.

KOHAVI, R.; JOHN, G. H. Wrappers for Feature Subset Selection. **Artificial Intelligence**, [S.l.], v.97, n.1-2, p.273–324, 1997.

KORTE, M.; PORT, D. Confidence in software cost estimation results based on MMRE and PRED. In: PREDICTOR MODELS IN SOFTWARE ENGINEERING, 4., New York, NY, USA. **Proceedings...** ACM, 2008. p.63–70. (PROMISE '08).

LIU, H. et al. Discretization: an enabling technique. **Data Min. Knowl. Discov.**, Hingham, MA, USA, v.6, p.393–423, Oct. 2002.

LIU, Q. et al. Evaluation of preliminary data analysis framework in software cost estimation based on ISBSG R9 Data. **Software Quality Control**, Hingham, MA, USA, v.16, n.3, p.411–458, 2008.

MAIR, C.; SHEPPERD, M. J. The consistency of empirical comparisons of regression and analogy-based software project cost prediction. In: ISESE'05. **Proceedings...** [S.l.: s.n.], 2005. p.509–518.

MENDES, E.; MOSLEY, N. Bayesian Network Models for Web Effort Prediction: a comparative study. **Software Engineering IEEE Transactions on**, [S.l.], v.34, n.6, p.723–737, 2008.

MENZIES, T. et al. Stable rankings for different effort models. **Automated Software Engg.**, Hingham, MA, USA, v.17, p.409–437, Dec. 2010.

MENZIES, T. et al. **The PROMISE Repository of empirical software engineering data. Available at**: <http://promisedata.googlecode.com>. viewed in apr. 16th, 2013. 2012.

MIYAZAKI, Y. et al. Method to estimate parameter values in software prediction models. **Inf. Softw. Technol.**, Newton, MA, USA, v.33, n.3, p.239–243, Apr. 1991.

MYRTVEIT, I.; STENSRUD, E.; SHEPPERD, M. Reliability and Validity in Comparative Studies of Software Prediction Models. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.31, n.5, p.380–391, May 2005.

NEIL, M.; TAILOR, M.; MARQUEZ, D. Inference in hybrid Bayesian networks using dynamic discretization. **Statistics and Computing**, Hingham, MA, USA, v.17, p.219–233, 2007.

PENDHARKAR, P. C.; SUBRAMANIAN, G. H.; RODGER, J. A. A Probabilistic Model for Predicting Software Development Effort. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.31, n.7, p.615–624, 2005.

RADLINSKI, L. A Survey of Bayesian Net Models for Software Development Effort Prediction. **International Journal of Software Engineering and Computing**, [S.l.], v.2, n.2, p.95–109, 2010.

RADLINSKI, L.; HOFFMANN, W. On Predicting Software Development Effort using Machine Learning Techniques and Local Data. **International Journal of Software Engineering and Computing**, [S.l.], v.2, n.2, p.123–136, 2010.

SHEPPERD, M. Software project economics: a roadmap. In: FUTURE OF SOFTWARE ENGINEERING, 2007., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2007. p.304–315. (FOSE '07).

SHEPPERD, M.; MACDONELL, S. Evaluating prediction systems in software project estimation. **Inf. Softw. Technol.**, Newton, MA, USA, v.54, n.8, p.820–827, Aug. 2012.

SHEPPERD, M.; SCHOFIELD, C. Estimating Software Project Effort Using Analogies. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.23, p.736–743, Nov. 1997.

SOUZA, A. L. R. et al. A Systematic Review of Bayesian Belief Network in Management of Software Development Process. In: CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA, Quito, Peru. **Anais...** [S.l.: s.n.], 2011. v.37.

STAMELOS, I. et al. On the use of Bayesian belief networks for the prediction of software productivity. **Information & Software Technology**, [S.l.], v.45, n.1, p.51–60, 2003.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining, (First Edition)**. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005.

TIERNO, I. A.; NUNES, D. J. Assessment of Automatically Built Bayesian Networks in Software Effort Prediction. **Ibero-American Conference on Software Engineering, Buenos Aires - Argentina**, [S.l.], p.196–209, Apr. 2012.

TRENDOWICZ, A.; MüNCH, J.; JEFFERY, R. State of the practice in software effort estimation: a survey and literature review. In: THIRD IFIP TC 2 CENTRAL AND EAST EUROPEAN CONFERENCE ON SOFTWARE ENGINEERING TECHNIQUES, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2011. p.232–245. (CEE-SET'08).

WEN, J. et al. Systematic literature review of machine learning based software development effort estimation models. **Inf. Softw. Technol.**, Newton, MA, USA, v.54, n.1, p.41–59, Jan. 2012.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining**: practical machine learning tools and techniques. 3rd.ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2011.