Key words:
accuracy evaluation, Bayesian nets, defect prediction, model building, expert knowledge, structure learning, parameter learning

Łukasz RADLINSKI*

BUILDING BAYESIAN NETS FOR SOFTWARE DEFECT PREDICTION – A COMPARISON OF MANUAL, SEMI- AND FULLY-AUTOMATED SCHEMES

Various techniques have been used in software defect prediction. Bayesian nets (BNs) is one of such techniques. BNs have various advantages, especially important in an environment where no large dataset is available, as is typical for software defect prediction. BNs can be built using various schemes, depending on availability of domain knowledge and/or data: (1) automatically from the data, (2) manually by an expert or (3) using a mixture of expert knowledge and empirical data. This chapter discusses an experiment, which aim is to analyze and assess the accuracy of BNs for software defect prediction developed using these three schemes. An important assumption is to use a small set with data on past projects completed in a single company, with most of predictors describing development process quality. The results show that a model built by experts performs the most accurately to predict number of defects and defect rate; models generated using semi- and fully-automated schemes achieve significantly lower accuracy. Two main reasons of such result are: (1) low volume of data used to automatically generate the model structure and parameters and (2) the need to discretize numeric variables using wide intervals.

1. INTRODUCTION

Software quality prediction has been investigated for about 40 years. Various techniques have been used – from early parametric models (e.g. multiple regression), through more advanced statistical (e.g. decision trees, case-based reasoning, Bayesian analysis) to more sophisticated within artificial intelligence (e.g. neural networks, fuzzy

* University of Szczecin, Institute of Information Technology in Management, ul. Mickiewicza 64, 71-101 Szczecin, Poland

sets, rough sets, evolutionary algorithms, support vector machines, Bayesian nets). None of these techniques can be regarded as absolutely the best – i.e. ensuring the most accurate predictions in every environment. Moreover, various researchers reported that specific technique provided the most accurate predictions in a very specific experiment.

In this study the predictive accuracy of Bayesian nets (BNs) is under investigation. The main aim of this work is to analyze the applicability of various schemes of generating BNs based on small local data. The main research question is then: **is it possible to built a BN using any of three schemes using small local qualitative data to predict software quality?**

There is no single measure of software quality. Often is is regarded as a set of various measures. In this study, due to data availability, two of such measures are used: **number of defects** and **defect rate**.

The main motivation for selecting BNs in this study is its flexibility about the model generation scheme. Three types of generation schemes have been analyzed:

– **manual** – the whole model is built by an expert or a group of experts,
– **semi-automated** – model structure is created by expert(s), parameters are learnt automatically from the dataset provided,
– **fully-automated** – the whole model (both structure and parameters) are generated automatically from the dataset provided.

Another important feature of BNs is its ease of use – there are basically no assumptions which have to be considered before model generation. Other techniques often have such assumptions, like normal distribution and linear relationships for some statistical techniques. In industrial use for software project management such constraints may not be known and ignored, causing various problems.

A useful feature of BNs is that such model contains relationships between various variables. There is no need to nominate in advance which variable is a dependent variable. This is done depending on model usage scenario. When a variable receives an observation (evidence) from a user, it becomes a predictor; otherwise it is a dependent variable. As a single BN can predict the values for multiple variables at once. Other advantages of BNs have been listed in Section 4.1.

To bring the experiment closer to the industrial setting, the dataset used in this study is very small but contains local data on software projects from a single company. Such dataset is cheap – there is no need for extensive metric collection and the project manager may provide the values for process variables using a questionnaire. Thus, there is no attempt to build general-purpose predictive models.

2. DATASET USED

The dataset used in this study has been published in [11], and is also publicly available in [4]. It contains data on past projects for embedded software developed by a large company producing media equipment. Table 1 contains a list of variables in the dataset. Most variables reflect the process quality and are expressed on a 5-point ranked scale from "very low" to "very high". One additional predictor, missing in the dataset, has been used in this study – project type. There are two dependent variables:

– Defects – number of defects found after software release;

– Defect Rate – calculated as: Defects / Project Size.

The dataset contains 31 cases (projects). For two projects no data on project size is available, thus 29 cases have been used in the analysis.

Table 1. Summary of variables and their groups

Quantitative data	effort
defects	project size
New functionality	number of inputs and outputs
complexity of new functionality	scale of new functionality
Specification and documentation process	requirements stability
experience of spec&doc staff	review process effectiveness
quality of documentation	specification defects discovered in review
regularity of spec&doc reviews	standard procedures followed
Design and development process	
development staff experience	defined process followed
programmer capability	development staff motivation
Testing and rework	
testing process well defined	staff experience - unit test
staff experience - independent test	quality of documented test cases
Project management	stakeholder involvement
development staff training quality	customer involvement
configuration management	vendor management
project planning	internal communication/interaction
scale of distributed communication	process maturity
Additional	project type

3. RESEARCH PROCEDURE

To compare the predictive accuracy of different model generation schemes the research procedure illustrated in Fig. 1 has been followed.

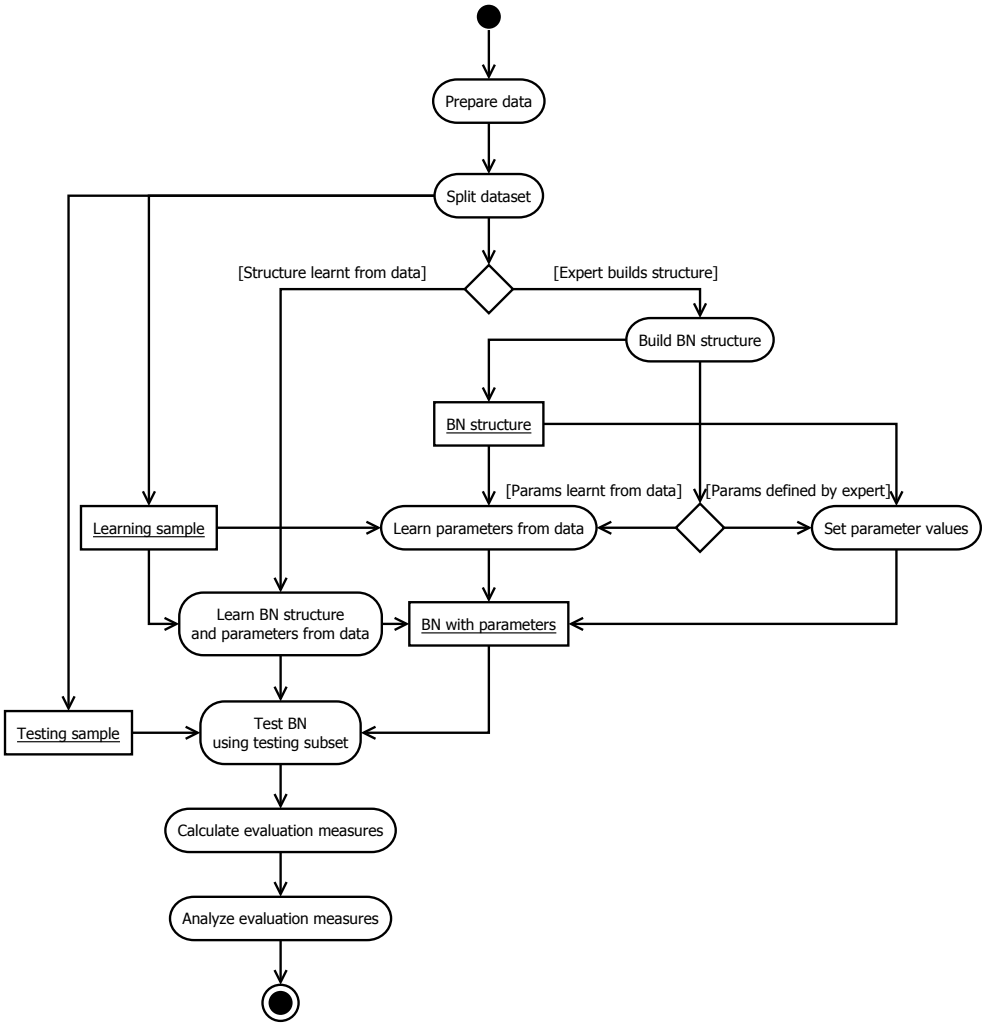

Fig. 1. Research procedure involving various generation schemes

This procedure involves the following steps:

– Data preparation – adjusting categories for "project type"; discretizing numeric variables to four or five intervals, as required by a learning algorithm.
– Random data split with proportions: 22 cases (75%) for model generation, 7 cases (25%) for model validation.
– Generating the model depending on the learning scheme – (1) purely from the data using greedy thick thinning algorithm implemented in Genie tool [13]; (2) purely by an expert – this step has been performed as an earlier study [11]; or (3) as a combination: structure by an expert (adopted from [11]), parameters from the data using expectation maximization algorithm.
– Obtaining predictions from generated models using samples for model validation – with junction-tree model calculation algorithm implemented in Agenarisk tool [1].
– Calculate and analyze the evaluation measures – Bayesian nets provide predictions not as a point value but in the form of probability distribution. In this study the calculated median of such distribution was treated as "predicted value". Earlier studies [9, 11, 21, 23] confirmed that such approach ensures higher accuracy than using the mean value of predicted distribution.

To ensure more reliable results, i.e. to avoid coincidence, generating and testing the model have been repeated ten times – for different random splits of data. The model built by an expert remained unchanged but it has been validated using the same ten testing datasets as for semi- and fully-automated schemes.

To assess the predictive accuracy of developed models the following measures have been used:

– mean magnitude of relative error (MMRE):

$$MMRE = \frac{1}{n}\sum_{i=1}^{n} MRE_i \tag{1}$$

$$MRE_i = \left| \frac{y_i - \hat{y}_i}{y_i} \right| \tag{2}$$

– median magnitude of relative error (MdMRE):

$$MdMRE = \text{Median}(MRE_i) \tag{3}$$

– prediction at level l (Pred(l)) – indicating fraction of cases for which predictions are within l percent of actuals:

$$Pred(l) = \frac{1}{n}\sum_{i=1}^{n} a_i \tag{4}$$

$$a_i = \begin{cases} 1 \text{ if } MRE_i \le \dfrac{l}{100} \\ 0 \text{ if } MRE_i > \dfrac{l}{100} \end{cases} \tag{5}$$

4. BAYESIAN NETS

4.1. BACKGROUND

Bayesian net (BN) is a modeling technique proposed in a pioneering work by Pearl in 1980's [18, 19]. However, their roots reach back the XVIII century when Bayes [3] formulated the basics of contemporary probability theory, known as a Bayes' rule.

BN model has two perspectives: graphical and numeric. Graphically, it is a set of nodes (variables) and directed links connecting pairs of variables. Numerically, each node is defined as a conditional probability distribution given the states of its parents (direct predecessors). Thus, the joint probability distribution for a set of variables $X_1, \ldots, X_n$ is defined as:

$$\mathrm{P}(X_i,...,X_n) = \prod_{i=1}^{n} \mathrm{P}(X_i \mid \mathrm{Parents}(X_i)) . \tag{6}$$

Calculating a BN is not a trivial task. Hence, several exact and approximate algorithms have been proposed. It is beyond the scope of this work to discuss them in details. More on them can be found in [7, 14, 15, 19, 24, 27].

BNs have a set of unique practical advantages (other popular modeling techniques have only some of these advantages): ability to reflect conditional independence be-

tween variables; ability to build a model purely from the data, purely by an expert or as a mixture of expert knowledge and data; ability to reflect causal relationships, explicit incorporation of uncertainty; ability to run the model with incomplete data (observations); forward and backward inference; ability to combine numeric, ranked and nominal types of variables.

4.2. RELATED MODELS

BNs have been successfully applied in diverse areas, such as: medicine, biology, chemistry, physics, law, management, and, naturally, computer science. Various researchers have built BN models for software engineering. Table 2 summarizes the most recent and relevant to this study – for software quality-related issues.

Table 2. Summary of related BNs for software quality prediction

Source	Main problem analyzed	Structure source	Parameter source	Validated using empirical data
[2]	failures	expert	data	yes
[5]	effectiveness of inspections	expert	expert	yes
[6]	defect rate	expert	data / expert	yes
[8]	need for requirements review	expert	expert	no
[9], [21]	defects	expert	expert	no
[10]	trade-off between: scope, effort, quality	expert	expert / data	no
[11], [12]	defects, partly: effort	expert	expert / data	yes
[16]	maturity of requirements	expert	expert / data	no
[17]	fault content, fault proneness	expert	data	yes
[20], [21]	trade-off between: scope, effort, quality	expert	expert / data	no
[22]	types of defects	expert	data / expert	no
[23]	defects, defect rate, effort, productivity	data	data	yes
[25]	rework effort	expert	expert / data	yes
[26]	rework effort	expert	expert / data	yes
[28]	various aspects of software quality	expert	expert / data	no
[29]	testing process	expert	expert / data	yes
[30]	change coupling	data	data	yes

Detailed analysis of these models led to the following two observations:

1. Some models have been validated with data from one or a couple of projects or using hypothetical scenarios (yet with data from real projects). Many models, especially those largely based on expert knowledge, have not been validated using empirical data at all. However, most of them have been validated against the commonly known laws of the discipline, but it is still uncertain how such models behave in industrial environment.
2. Most of these models are focused on a single issue – variable to be predicted. Although it is generally a valid approach to reflect only a part of the real world in the model, many of these models use only one group of predictors, for example source code data or process data, but not both.

One of earlier studies [11] is particularly relevant to the current one. The authors of that study proposed a BN for software defect prediction. They built their model purely based on expert knowledge. They calibrated it for a particular company. However, model parameters were not automatically learnt from the dataset but rather entered as expressions (transformed to the conditional probability distributions). Thus, these expressions already were generalizations of the empirical data. The authors used the dataset, the same as in the current work, only to validate the model by assessing the accuracy of predictions.

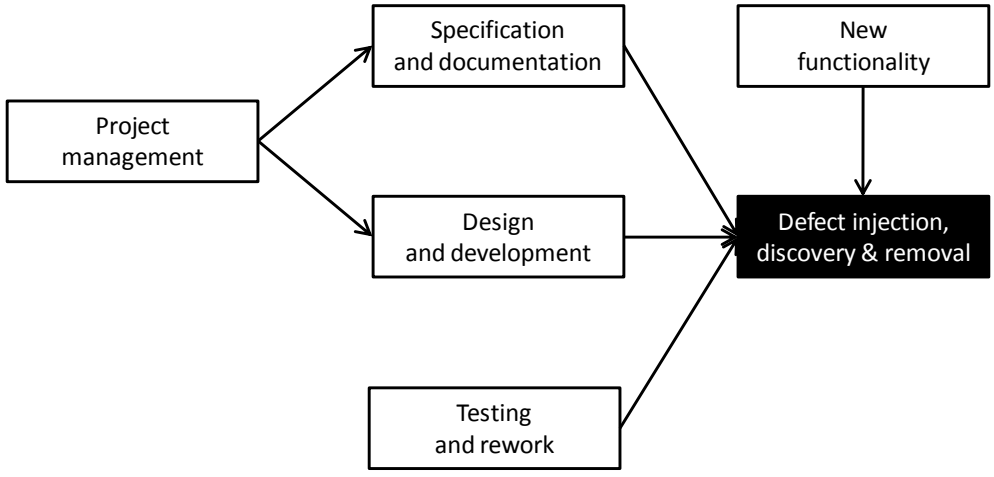

Fig. 2. Schematic of BN structure built by expert; based on [11]

The majority of variables from this dataset match the variables in that model. Thus, in the current study, where there is a need for BN structure built by an expert, the structure from that model is used. Fig. 2 illustrates the schematic of such model. All the rectangles represent subnets containing variables listed in Table 1. The subnet "Defect insertion, discovery and removal" contains the dependent variables (defects and defect rate) and a set of hidden variables for which no data is available. The details on this structure can be found in [11].

5. ACCURACY OF PREDICTIONS

5.1. BY SET ACCURACY

The dataset used in this study has been divided randomly ten times into learning and testing samples. Thus, ten testing sets have been created. Fig. 3 illustrates the measures of accuracy of predictions individually for these ten sets for predicted number of defects and defect rate.

The analysis of these results leads to a set of observations. For almost all evaluation measures and all three generation schemes there is a high variability of the values of evaluation measures depending on the dataset. This may suggest that the testing samples contained the projects considerably different than in the learning samples (for structure or parameter learning) or in the background knowledge (when building by an expert). This may also suggest a more general problem of generating predictive model from low number of cases.

For predicting the number of defects model developed by an expert have outperformed other schemes for almost all sets. Pred(25) was the highest for this scheme in all ten sets. MdMRE was the lowest in almost all sets (except set 8 for structure learning). MMRE was the lowest in seven sets (except sets 7, 9 and 10). MMRE for models built using semi-automated scheme was very poor in the half of the sets (value close to 10 or higher).

For predicting the defect rate it is more difficult to point the best performing scheme because the values of accuracy measures fluctuated from one set to another. However, for parameter learning scheme Pred(25) was lower than for model built by expert in all ten sets and lower than structure learning scheme in all but two sets. For any set no scheme achieved the value of Pred(25) higher than the model built by expert; structure learning scheme achieved the same value of Pred(25) in five sets.

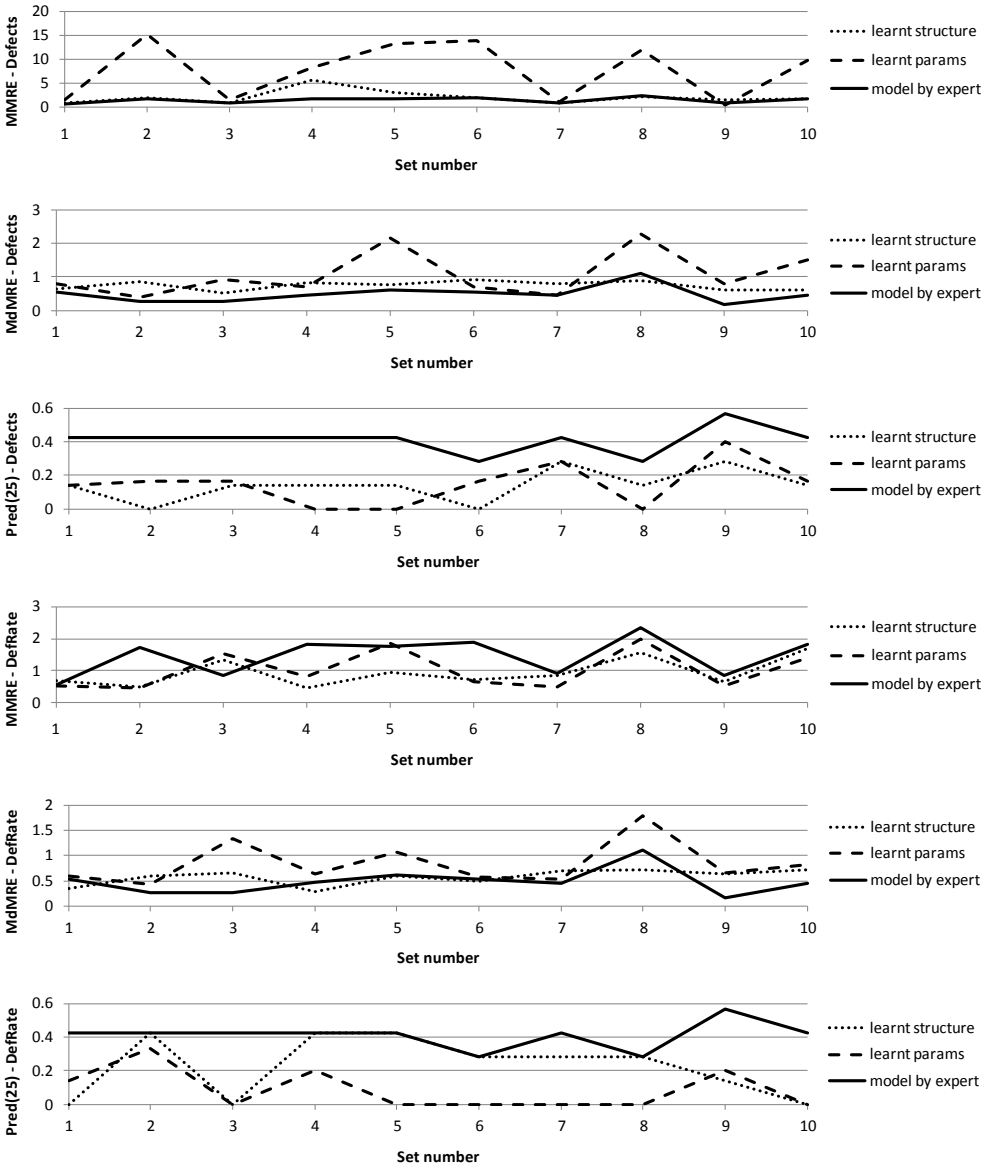

Fig. 3. Accuracy of predictions for individual sets

5.2. OVERALL ACCURACY

The analysis of overall results have been performed by analyzing all predictions obtained from every model generated using each scheme. The predicted values have been plotted against the actual in Fig. 4.

It can be observed that for predicting number of defects only the model built by expert provided reasonably accurate predictions – at least for number of defects higher than 500 (for lower values the graph is unclear due to its resolution). Both fully- and semi-automated schemes significantly underestimated the projects with this high number of defects

Predictions for defect rate appear to be highly inaccurate (under- or overestimated) for all generating schemes. However, extremely inaccurate predictions are least often from the model built by expert.

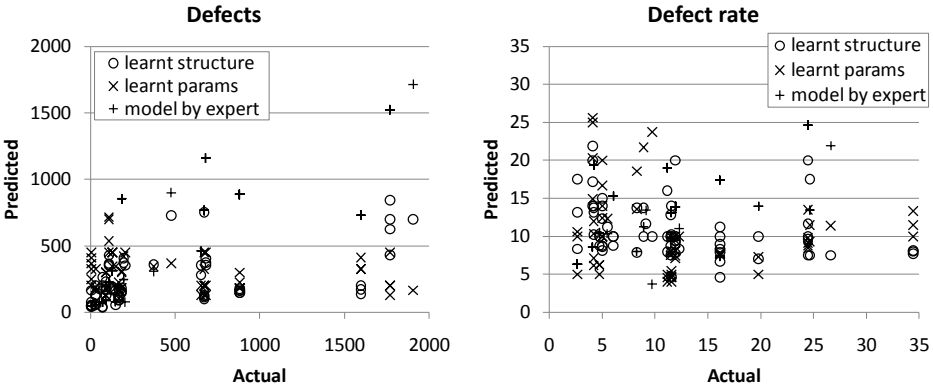

Fig. 4. Predicted vs actual values

Fig. 5 illustrates the values of accuracy measures aggregated for the whole experiment (all ten sets). These results confirm the highest overall accuracy for model built by expert when predicting the number of defects. However, when analyzing predictions for defect rate the MMRE was the highest for the model built by expert (suggesting its low accuracy). Still, this model achieved the best values of MdMRE and Pred(25) among all three schemes.

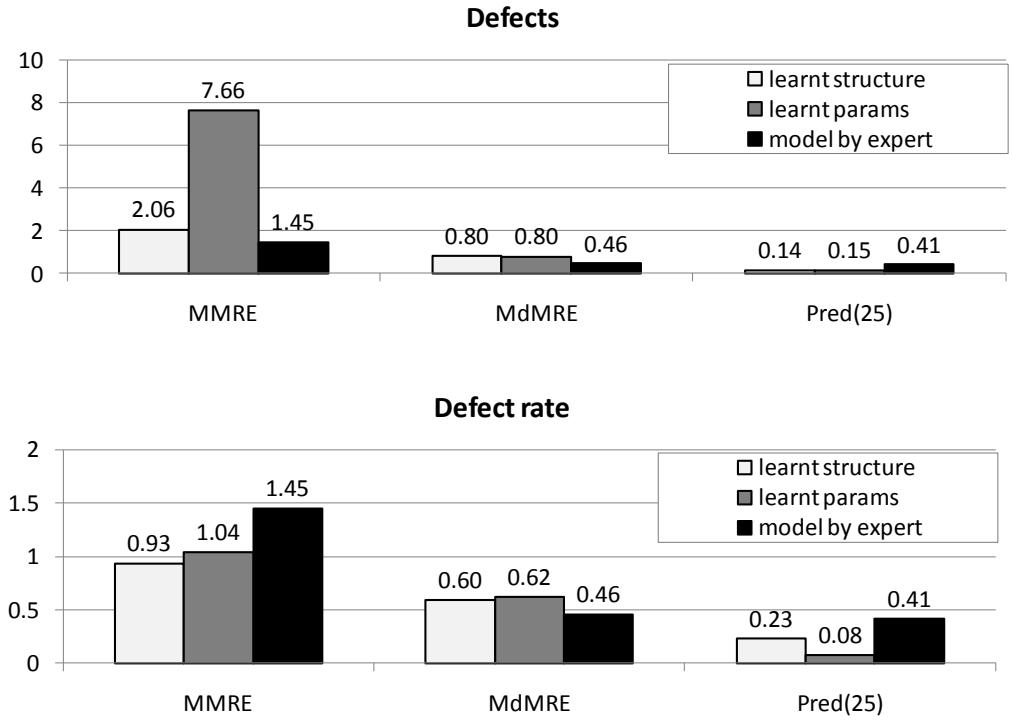

Fig. 5. Overall accuracy of predictions

After analyzing all of achieved results it can be observed that the model built by expert significantly outperformed two other schemes in predicting the number of defects and in most cases performed better for predicting the defect rate. However, the application of these schemes was slightly different and might have caused the differences in model accuracy.

The most important difference is related to the discretization of numeric variables. Due to the low number of cases in the dataset, the values of each numeric variable have been discretized into four or five intervals when using semi- and fully-automated schemes. Such step was necessary to enable learning some general rules from the data rather than just fit to exact values. The model built by expert also contains numeric variables discretized, but the number of defined intervals is much higher (for example 46 intervals for number of defects). As a result, in models generated with automated

schemes all numeric values are represented with significantly lower precision, which might have a negative effect on lower predictive accuracy.

The dataset contains the numeric value of effort for each project. It has been used, after discretization, in models generated using automated schemes. However, this variable has not been used to validate the model built by expert. It was caused by the fact that this model does not contain effort variable. Still, this model performed best even without use of effort data.

In the original study where this model have been proposed [11] the authors reported the following values of accuracy measures for predicted number of defects: MMRE=0.96, MdMRE=0.27, and Pred(30)=0.58 (defect rate was not investigated in that study). Such results indicate the higher accuracy of their model than calculated in the current work for the same testing dataset. The fact that the authors used Pred(30) instead of Pred(25) should not make any significant difference since both MMRE and MdMRE also indicate the higher accuracy in their study. There are three explanations of the differences:

– In the original study the authors used the effort data for each development stage: specification, coding, testing and rework – not numerically though, but expressed on a ranked scale. However, such data is not available in the dataset so these results cannot be replicated.
– In the original study the model has been validated using additional data on code reused in particular project from the past projects – on the assumption that existing code might have a different number of defects than the new code. Again, such data is not available in the dataset, so the default distributions for these variables have been used in the current study.
– In the original study for one project from the dataset one prediction have been obtained. In the current study, due to ten times random data split, some projects have been used several times in testing samples while some other have never been selected to any testing sample.

6. SUMMARY AND CONCLUSIONS

Building a predictive model based purely on expert knowledge is a difficult and time-consuming task. Furthermore, there is no guarantee that such model will be able to predict accurately. From the industrial point of view there is a need to generate predictive models fast. Thus, any kind of automation of this process is desired. Specifically, to

generate a predictive model purely or partially from the data. Statistical and artificial intelligence techniques satisfy this automation need but many are difficult in use because some checks need to be performed before actually building a model.

Bayesian nets can be built in various ways: by an expert, purely from the data, or by combining expert knowledge with empirical data. In the experiment discussed in this chapter several BNs have been created using the three generation schemes.

The analysis of predictive accuracy of generated models have shown that the model built by expert(s), proposed in [11], outperformed the models generated using fully- and semi-automated schemes both for prediction of number of defects and defect rate. This low accuracy of models generated with automated schemes is mainly caused by two factors:

– Low volume of data used to learn the BN structure and/or its parameters – Conditional probability distributions for many variables have not been consistent. It is difficult, or even impossible to observe any kind of more general rule describing the relationship between variables being identified and encoded in the model.
– The need to discretize the numeric variables into four or five intervals as required by the implementations of structure and parameter learning algorithms. Such discretization caused immediate lack of precision and next, low predictive accuracy of models built using automated schemes.

It appears that there is no easy way to improve the accuracy of predictions. One of these ways is to significantly increase the volume and diversity of the data used to build the models. However, software companies may be unable to do so , for example if they are focused on developing lower number but larger projects. Additionally, having a larger dataset also opens the door for other modeling techniques and thus limits the justification for using BNs.

In the future I plan to investigate the accuracy of other techniques, including with this small dataset. In another study I plan to analyze how integration of different types of data, i.e. from different types of software repositories, affects the accuracy of software quality prediction.

ACKNOWLEDGEMENT

I would like to thank Professor Marek Druzdzel (Univ. of Pittsburgh) and Professor Norman Fenton (Queen Mary, University of London) for providing tools for developing and managing BNs: Genie and Agenarisk, respectively.

This work has been supported by research funds from the Ministry of Science and Higher Education in Poland as a research grant for years 2010-2012.

REFERENCES

[1] AGENARISK, Agena, Ltd., http://www.agenarisk.com, 2009.

[2] BAI C.G., HU Q.P., XIE M., NG S.H., *Software failure prediction based on a Markov Bayesian network model*, Journal of Systems and Software, Vol. 74, No. 3, 2005, 275–282.

[3] BAYES T., *An essay towards solving a Problem in the Doctrine of Chances. By the late Rev. Mr. Bayes, F.R.S. communicated by Mr. Price, in a letter to John Canton, A.M.F.R.S.*, Philosophical Transactions of the Royal Society of London, Vol. 53, 1763, 370–418.

[4] BOETTICHER G., MENZIES T., OSTRAND T., PROMISE *Repository of Empirical Software Engineering Data*, West Virginia University, Department of Computer Science, http://promisedata.org/repository, 2010.

[5] COCKRAM T., *Gaining Confidence in Software Inspection Using a Bayesian Belief Model*, Software Quality Journal, Vol. 9, No. 1, 2001, 31–42.

[6] DABNEY J.B., BARBER G., OHI D., Predicting Software Defect Function Point Ratios Using a Bayesian Belief Network, Proc. 2nd Int. Workshop on Predictor Models in Software Engineering, Philadelphia, PA, 2006.

[7] DARWICHE A., *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[8] DEL SALGADO MARTINEZ J., DEL AGUINA CANO I.M., *A Bayesian Network for Predicting the Need for a Requirements Review*, in: Meziane F., Vadera S. (eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects, Information Science Reference, New York, 2008, 106–128.

[9] FENTON N., HEARTY P., NEIL M., RADLIŃSKI Ł., *Software Project and Quality Modelling Using Bayesian Networks*, in: Meziane F., Vadera S. (eds.), Artificial Intelligence Applications for Improved Software Engineering Development: New Prospects, Information Science Reference, New York, 2008, 1–25.

[10] FENTON N., MARSH W., NEIL M., CATES P., FOREY S., TAILOR M., *Making Resource Decisions for Software Projects*, Proc. 26th Int. Conference on Software Engineering, Washington, DC, IEEE Computer Society, 2004, 397–406.

[11] FENTON N., NEIL M., MARSH W., HEARTY P., RADLIŃSKI Ł., KRAUSE P., *On the effectiveness of early life cycle defect prediction with Bayesian Nets*, Empirical Software Engineering, Vol. 13, 2008, 499–537.

[12] FENTON N.E., NEIL M.., MARSH W., KRAUSE P., MISHRA R., *Predicting Software Defects in Varying Development Lifecycles using Bayesian Nets*, Information and Software Technology, Vol. 43, No. 1, 2007, 32–43.

[13] GENIE, Decision Systems Laboratory, University of Pittsburgh, http://genie.sis.pitt.edu/, 2009.

[14]JENSEN F.V., *An Introduction to Bayesian Networks*, UCL Press, London, 1996.

[15]NEAPOLITAN R.E., *Learning Bayesian Networks*, Pearson Prentice Hall, Upper Saddle River, 2004.

[16] PAI G., BECHTA-DUGAN J., LATEEF K., *Bayesian Networks applied to Software IV&V*, Proc. 29th Annual IEEE/NASA Software Engineering Workshop, IEEE Computer Society, Washington, DC, 2005, 293–304.

[17] PAI G.I., DUGAN J.B., *Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods*, IEEE Transactions on Software Engineering, Vol. 33, No. 10, 2007, 675–686.

[18] PEARL J., Bayesian Networks: *A Model of Self-Activated Memory for Evidential Reasoning*. UCLA Technical Report CSD-850017, Proc. 7th Conf. of the Cognitive Science Society, University of California, Irvine, 1985, 329–334.

[19] PEARL J., *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*, Morgan Kaufmann, San Francisco, 1988.

[20] RADLIŃSKI Ł., FENTON, N., NEIL, M., MARQUEZ D., *Improved Decision-Making for Software Managers Using Bayesian Networks*, Proc. 11th IASTED Int. Conference Software Engineering and Applications, Cambridge, MA, 2007, 13–19.

[21] RADLINSKI L., *Improved Software Project Risk Assessment Using Bayesian Nets*, Ph.D. Thesis, Queen Mary, University of London, London, 2008.

[22] RADLIŃSKI Ł., *Predicting Defect Types in Software Projects*, Polish Journal of Environmental Studies, Vol. 18, No. 3B, 2009, 311–315.

[23] RADLIŃSKI Ł., *Software Development Effort and Quality Prediction Using Bayesian Nets and Small Local Qualitative Data*, Proc. 22nd International Conference on Software Engineering and Knowledge Engineering, Redwood City, CA, 2010, 113–116.

[24] RUSSELL S., NORVIG P., *Artificial Intelligence. A Modern Approach, Second Edition*, Pearson Education, Inc., Upper Saddle River, 2003.

[25] SCHULZ T., RADLIŃSKI Ł., GORGES T., ROSENSTIEL W., *Defect Cost Flow Model – A Bayesian Network for Predicting Defect Correction Effort*, Proc. 6th International Conference on Predictive Models in Software Engineering, Timisoara, 2010 (in press).

[26] SCHULZ T., RADLIŃSKI Ł., GORGES T., ROSENSTIEL W., *Software Process Model Using Dynamic Bayesian Network*, in: Ramachandran M., (ed.), Knowledge Engineering for Software Development Life Cycles: Support Technologies and Applications, IGI Global, 2010 (in press).

[27]SPIRTES P., GLYMOUR C., SCHEINES R., Causation, Prediction, and Search, 2nd Edition, MIT Press, 2001.

[28]WAGNER S., *A Bayesian network approach to assess and predict software quality using activity-based quality models*, Proc. 5th Int. Conf. on Predictor Models in Software Engineering, New York, ACM Press, 2009.

[29] WOOFF D.A., GOLDSTEIN M., COOLEN F.P.A., *Bayesian Graphical Models for Software Testing*, IEEE Transactions on Software Engineering, Vol. 28, No. 5, 2002, 510–525.

[30] ZHOU Y., WÜRSCH M., GIGER E., GALL H.C., LÜ J., *A Bayesian Network Based Approach for Change Coupling Prediction*, Proc. 15th Working Conference on Reverse Engineering, Washington, DC, IEEE Computer Society, 2008, 27–36.