Key words:
productivity rate, defect rate, environmental factors,
Naïve Bayesian Classifier, ISBSG dataset

Łukasz RADLIŃSKI*[‡]
Norman FENTON[‡]
David MARQUEZ[‡]

ESTIMATING PRODUCTIVITY AND DEFECT RATES BASED ON ENVIRONMENTAL FACTORS

In previous work we have developed a causal model for software project risk assessment, called the Productivity Model. It enables trade-off analysis between key project factors: effort, size and quality. In this paper we build a model (which we call the PDR model) providing more informed prior estimates of productivity and defect rates as inputs to the Productivity Model. The PDR model has the Naïve Bayesian Classifier (NBC) structure. We built the PDR model using results from analysis of the ISBSG dataset which we adjusted by other known results and expert knowledge. The assumption for the PDR model was to use only the factors which describe the environment in which the software is developed such as: language type, development platform, methodology and CASE tool used. Therefore, we do not expect the model to be extremely accurate since it does not capture any process factors (such factors are accounted for in the Productivity Model). We still believe that, although PDR model has only limited value in isolation, it performs in a consistent way and may significantly change the estimates given by the Productivity Model.

1. INTRODUCTION

The Productivity Model developed in [5] enables us to perform trade-off analysis between key project variables: functionality, effort and software quality. One of the unique features of that model compared to previous similar models is that users can explicitly

---

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

[‡] Department of Computer Science, Queen Mary, University of London, Mile End Road, London E1 4NS, United Kingdom

calibrate the model to their own environment using productivity and defect rate estimates from their past projects. However, in some cases users will not have such data available. This can happen when no similar software has been developed in the past or when they do not collect required data, such as effort data. In these cases we need to estimate these prior rates from different sources. We decided to build a model (called the PDR model) to provide estimates for these productivity and defect rates. This model has a Naïve Bayesian Classifier (NBC) structure. It uses environmental factors as predictors – variables which describe the environment in which software is developed and the nature of the software itself.

In Section 2 we discuss the steps of statistical analysis that we performed in order to determine predictors which influence dependent variables (rates). In Section 3 we describe the structure of the PDR model and show how it can be used to estimate productivity and defect rates.

2. STATISTICAL ANALYSIS OF THE DATASET

2.1. DATASET

In this analysis we used the ISBSG R9 dataset [1] – the largest database of software projects to which we had access. It contains data for 3024 projects described by 99 variables. We filtered the dataset to include only cases with good *data quality* (classified as ‘A’ or ‘B’). In the analysis with *productivity rate* as the dependent variable we further filtered the dataset to use only those projects where *count approach* was ‘IFPUG’ (project size measured in IFPUG function points) and *resource level* was ‘1’ (only development team effort was included in effort data). As a result of such filtering we got two datasets containing respectively 2095 projects to analyze *productivity rate*, and 510 projects to analyze *defect rate*. The much smaller dataset to analyze defect rate is caused by the fact that in many projects no defect information was available. None of the dependent variables were provided in the dataset directly. We added them and defined them according to Equations 1 and 2. We used the following units of measurement: for *productivity rate* – function points per person-hour (*FP*/*PH*), for *defect rate* – defects per function point (*D*/*FP*).

$$productivity\, rate = \frac{functional\, size}{summary\, work\, effort} \qquad (1)$$

$$defect\ rate = \frac{total\ defect\ delivered}{functional\ size} \quad (2)$$

2.2. ANALYSIS PROCEDURE

We performed statistical analysis of the ISBSG dataset in order to identify variables which influence our dependent variables: *productivity rate* and *defect rate*. This analysis contained the following main steps:

1. Analyzing dependent variables and applying transformations

First, we analyzed the distributions for the dependent variables. Without any transformation the distributions of the dependent variables were extremely non-symmetrical and dispersed. We applied the Ln (natural logarithm) transformations to the data to bring their distributions closer to normal distributions as suggested in [4]. Table 1 illustrates summary statistics for original and transformed dependent variables.

Table 1. Summary of descriptive statistics for dependent variables

Statistic / Variable	Mean	Median	Std. dev.	Lower Quartile	Upper Quartile	Skewness	Kurtosis
Productivity Rate	0.254	0.114	1.114	0.057	0.226	26.639	903.37
Defect Rate	0.061	0.009	0.251	0.000	0.033	11.471	166.60
Ln Productivity Rate	-2.175	-2.170	1.112	-2.873	-1.488	0.207	1.26
Ln Defect Rate	-4.819	-4.571	2.116	-6.273	-3.424	-0.088	-0.32

2. Preparing list of potential predictors

Based on our experience we nominated a set of possible predictors which were listed in ISBSG dataset to be used later in the model.

3. Analyzing categories of potential nominal predictors

In this step we analyzed the frequencies for nominal predictors. For some categories there were very low number of observations (below 10). To improve the predictor performance we grouped such categories with existing ones that had very close meaning. We removed one predictor for which the values were not clear enough. Table 2 summarizes predictors kept and removed in this and successive steps of the statistical analysis.

Table 2. Predictors kept after each step of statistical analysis

Predictor / Dependent / Step		Average Team Size	Project Elapsed Time	Functional Size	Summary Work Effort	Value Adjustment Factor	Application Type	Architecture	Business Area Type	CASE Tool Used	Client-Server	Development Platform	Development Type	How Methodology Acquired	Intended Market	Language Type	Organisation Type	Package Customisation	Primary Programming Language	Used Methodology	User Base: Business Units	User Base: Concurrent Users	User Base: Locations
productivity rate	3	+	+	+	+	+	+	+	+	+	+	+	+	–	+	+	+	+	+	+	+	+	+
	4	–	–	–	–	+	–	+	–	–	–	+	+		+	+	–	–	–	+	–	+	+
	5					–		–				+	+		–	+				+		–	+
defect rate	3	+	+	+	+	+	+	+	+	+	+	+	+	–	+	+	+	+	+	+	+	+	+
	4	–	+	–	+	–	–	+	–	+	+	+	–		+	–	–	–	–	+	–	–	+
	5		–		+			–		+	–	+			–					+			–
‘+’ indicates that particular predictor was kept in specific step ‘–‘indicates that particular predictor was removed in specific step																							

4. Identifying relationships between predictors and dependent variables

In this step we used:

– *Spearman rank correlation coefficient* (SRCC) – for correlations between dependent variables and numeric potential predictors.

– *Kruskal-Wallis one-way analysis of variance* (KW-ANOVA) – for associations between dependent variables and nominal potential predictors.

Beside analysing the values and significance of the above measures, we also analyzed if the identified relationships make sense from a causal perspective. For each nominal variable we analysed box-plots to determine how specific categories of nominal predictors influence dependent variables.

5. Identifying correlations / associations among predictors

Our aim was to build models with a Naïve Bayesian Classifier (NBC) structure. In such types of models predictors should not be correlated or associated. This simplifies the model creation process as only pairwise relationships between dependent variables and particular predictors need to be analyzed. Although NBC models “can work surprisingly well even when the independence assumption is not true” [7 p. 482], it is still suggested to ensure independence between predictors when possible. As a result of this condition we also have to analyze possible correlations and associations between particular predictors. Therefore, we checked separately for each of the two datasets (productivity and defect rates):

– possible correlations between each pair of numeric predictors – using SRCC,
– possible associations between each numeric and each nominal predictor – using KW-ANOVA,
– possible associations between each pair of nominal predictors – using Phi, Cramer's V and contingency coefficients.

This stage resulted in removing some predictors from further analysis.

3. PDR MODEL

3.1. MODEL STRUCTURE

The Naïve Bayesian Classifier structure for our PDR model can be represented as a Bayesian net (Fig. 1). Here the arrows do not reflect causal relationships but conditional independence among the predictors given the state of the root variable. Users enter observations in the leaf nodes (white background). Then, by applying the Bayesian Net back-propagation algorithm, the model calculates the root nodes (black background) which represent logged dependent variables. The final step of the calculation is transforming the logged dependent variables to their original scale which is clearer for users.

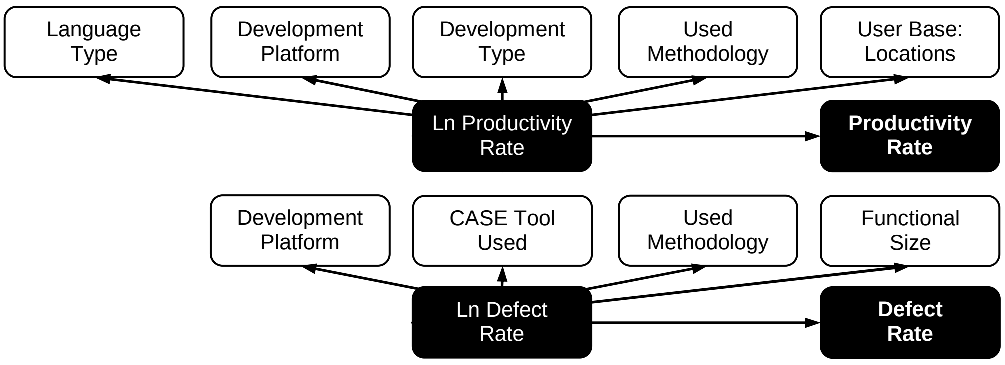


Fig. 1. Structure of PDR model

Some sources, such as [2] (cited in [8]) have reported positive correlation between *functional size* and *defect rate*. Our analysis did not confirm this but we accepted a causal relationship between these two variables and we decided to include *functional size* as a predictor in our model. Since *summary work effort* (identified earlier as a significant predictor for *defect* rate) is strongly correlated with *functional size* we could keep only one of them. We selected *functional size* because of the problems with collecting effort data by some companies [6] and because it is more naturally correlated with *defect rate*: software quality decreases as a result of the increase of project size, not as a result of increase in development effort.

The prior probability distributions for dependent variables are defined according to Equations 3 and 4:

$$\ln(\textit{productivity rate}) \sim N(-2.18, 1.232) \tag{3}$$

$$\ln(\textit{defect rate}) \sim N(-3, 4) \tag{4}$$

The prior distribution for ln(*productivity rate*) reflects the empirical distribution from the ISBSG dataset (see Table 1). The prior distribution for ln(*defect rate*) is defined to match other sources [2], [3] (both cited in [8]) which report much higher defect rates than in this dataset. Therefore, in our model we increased prior mean for *defect rate* distribution which now is a mixture of ISBSG data and other reported data.

Table 3. Impact of predictors on dependent variables reflected in the PDR model

Predictor and its type		Dependent variable	
		Productivity Rate	Defect Rate
CASE Tool Used	nominal	*none*	'Yes', 'No'
Development Platform	nominal	'Mainframe', 'Mid-Range', 'Multi', 'PC'	'Mainframe', 'PC', 'Mid-Range', 'Multi'
Development Type	nominal	'Re-development', 'Enhancement', 'New'	*none*
Functional Size	continuous	*none*	+
Language Type	nominal	'2GL', '3GL', '4GL', 'ApG'	*none*
Used Methodology	nominal	'Yes', 'No'	'Yes', 'No'
User Base: Locations	ordinal	–	*none*
'+' indicates positive impact – as predictor increases dependent also increases '–' indicates negative impact – as predictor increases dependent decreases labelled predictors – order of categories reflects increase of dependent variable: the lowest value of dependent variable is for the first category, the highest value – for the last category			

When creating NBC models, conditional probabilities (for predictors) are often estimated from the dataset by unsupervised learning. We believe that the ISBSG dataset does not allow such model learning. There are two main reasons:

1.Missing data – When analyzing crosstabs between the dependent variable and each predictor we found that in many cells there were no observations. We believe that this is not because such combinations of values of predictor and dependent variables do not exist in reality. Rather, we think that it just happened that data donors did not provide data for such combinations and that the majority of them may actually occur in reality – although some of them probably very rarely.

2.Prior probabilities for predictors – We are not convinced that prior probabilities for predictors estimated from the dataset by analyzing frequencies really reflect the whole population of the developed projects.

Therefore, we defined the conditional probabilities according to our expert knowledge partially supported by the results of earlier analysis. They reflect the impact of particular predictors as presented in Table 3.

3.2. MODEL ESTIMATES

1. Impact of factor *used methodology*

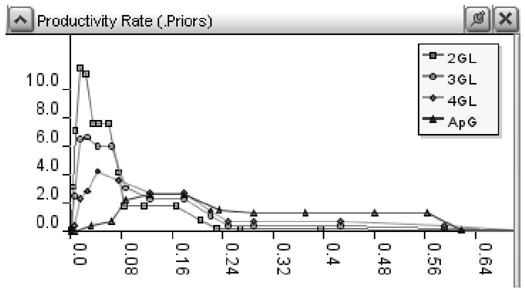
The *used methodology* factor is one of two predictors which influence both *productivity rate* and *defect rate*. The direction of its impact on both rates was leant from data. Table 4 summarizes predicted median values for dependent variables depending on value of factor *used methodology*. We would have expected that using a methodology leads to a decrease in *defect rate*. Indeed, model predictions indicate that there is a positive effect on software quality when a methodology is used. However, the impact on *productivity rate* seems to be the opposite of what we expected. The model predicts higher *productivity rate* without methodology compared to situation when development methodology is used. A possible explanation to this surprising result is captured in the PDR model, and also clearly observed in the dataset. Using a methodology involves spending additional effort on activities associated with developing documentation for product and process. It means that this effort is wasted from the productivity point of view. Instead of delivering functionality such effort is allocated to activities that do not extend functionality.

Table 4. Predicted median values for productivity and defect rates depending on factor *used methodology*

Predicted / Scenario	Productivity Rate (FP/PH)	Defect Rate (D/FP)
Used Methodology = ‘Yes’	0.109	0.050
Used Methodology = ‘No’	0.402	0.160

2. Impact of *language type*

Fig. 2 illustrates predicted probability distribution function (PDF) for *productivity rate* depending on *language type*. This relationship appears to be straightforward – the higher level of *language type* used in a project, the higher productivity achieved for this project. Thus the lowest productivity is typically achieved in projects when '2GL' language is used, and the highest with 'ApG' (application generators).



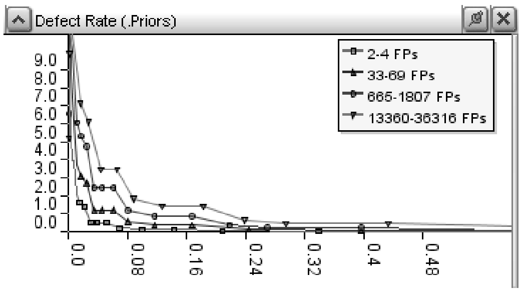
Productivity Rate

	2GL		4GL
Mean:	0.106	Mean:	0.320
Median:	0.059	Median:	0.167
SD:	0.204	SD:	0.833
	3GL		**ApG**
Mean:	0.218	Mean:	0.399
Median:	0.097	Median:	0.304
SD:	0.686	SD:	0.393

Fig. 2. Predicted PDF for productivity rate for different types of languages

3. Impact of *functional size*

In this case we analyze how the model reflects the impact of *functional size* on *defect rate*. When building this model we assumed (based on empirical data) that *defect rate* increases as *functional size* increases. It means that bigger projects have typically lower quality. Fig. 3 illustrates predicted defect rates for four sample ranges of *functional size*. The first of them reflects the lowest size supported by our model (2-4 *FPs*), the last – the highest (13360-36316 *FPs*), and two other reflect some intermediate intervals for *functional size*. Indeed, we can observe that the model predicts that *defect rate* increases with the increase of *functional size*. When *functional size* has the highest value *defect rate* is 7 times higher (comparing predicted median values) than in scenario when *functional size* is the lowest.



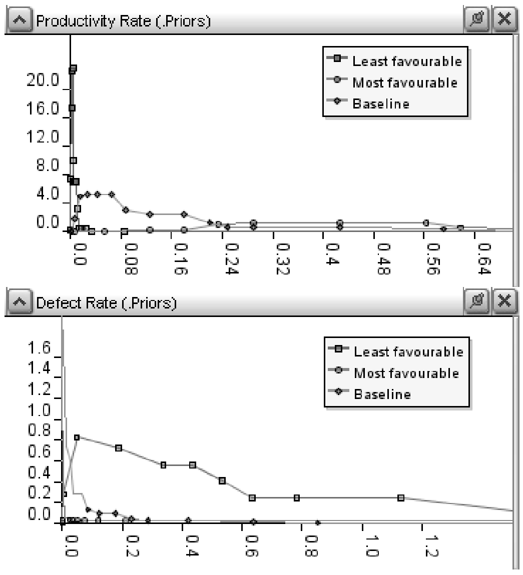
Defect Rate

	2-4 FPs		665-1807 FPs
Mean:	0.138	Mean:	0.575
Median:	0.019	Median:	0.075
SD:	0.710	SD:	2.099
	33-69 FPs		**13360-36316 FPs**
Mean:	0.295	Mean:	0.887
Median:	0.040	Median:	0.133
SD:	1.312	SD:	2.822

Fig. 3. Predicted PDF for defect rate for different project sizes

4. Least and most favourable scenarios

The next aim is to analyze the impact of the least and the most favourable combinations of predictors which determine the ranges for predicted dependent variables in the PDR model. Fig. 4 illustrates predicted productivity and defect rates in least and most favourable scenarios. In the most favourable scenario the model predicts *productivity rate* 21 times higher than in the least favourable scenario. Predicted *defect rate* is 128 times lower in the most favourable scenario comparing to the least favourable. Such wide ranges, especially for *defect rate* may seem too wide and incorrectly reflecting reality. However, we believe that some of these extremes are very unlikely to happen in reality. For example, the least favourable scenario for *defect rate* is when a project is very large and developed using multiple hardware platforms. For such projects it is almost impossible not to use any CASE tool nor any development methodology. But actually this least favourable scenario assumes that neither CASE tool nor methodology are used for such projects.



Productivity Rate

Least favourable		**Baseline**	
Mean:	0.040	Mean:	0.262
Median:	0.027	Median:	0.129
SD:	0.043	SD:	0.728
Most favourable			
Mean:	0.881		
Median:	0.675		
SD:	0.822		

Defect Rate

Least favourable		**Baseline**	
Mean:	2.772	Mean:	0.424
Median:	1.025	Median:	0.056
SD:	4.819	SD:	1.724
Most favourable			
Mean:	0.046		
Median:	0.008		
SD:	0.296		

Fig. 4. Predicted PDFs for productivity and defect rates in most and least favourable scenarios

5. Linking with Productivity Model

As suggested earlier the main purpose of the PDR model is to provide estimates for prior *productivity rate* and *defect rate* which are inputs to the Productivity Model. In this case we analyze the impact of observations entered in the PDR model on predictions provided by the Productivity Model. We analyze two scenarios for the same *functional size* (1000 FPs), the same target *defect rate* to be achieved (0.2 D/FP) and wish to get a prediction for *development effort* from the Productivity Model. In the first scenario we assume that the software project will be developed on a mainframe and with 3GL programming language. The second scenario assumes that a project will be developed on multiple platforms using an application generator. For fair comparison we further assume that in both scenarios process and people quality and other factors from the Productivity Model have 'average' values.

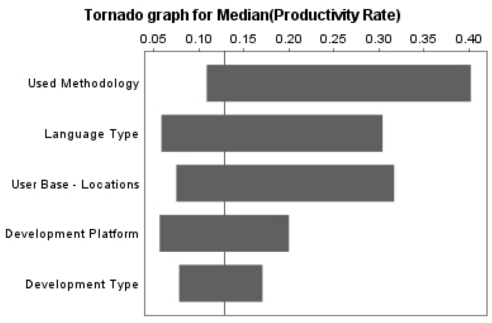
Table 5 summarizes predicted median values for various variables. We can notice that observations entered to the PDR model significantly change predictions provided by the Productivity Model. Predicted *development effort* is about 4 times higher in scenario 1 than in scenario 2. However, we note that the conclusion is not that we should only use application generators and multiple platforms while avoiding 3GL languages and mainframe platform. Predictors in PDR model should rather be treated as uncontrollable factors which describe the inherent nature of the software project. And this nature determines the best platform, language type to be used and other factors.

Table 5. Predictions from PDR model linked to Productivity Model (median values)

Predicted / Scenario	Productivity Rate PDR model	Defect Rate PDR model	Development Effort Productivity Model
Scenario 1: 'Mainframe', '3GL'	0.049	0.027	10608.0
Scenario 2: 'Multi', 'ApG'	0.258	0.145	2631.2

3.3. SENSITIVITY ANALYSIS

We perform the sensitivity analysis to determine the strength of impact of particular predictors on dependent variables. Fig. 5 illustrates the results of this sensitivity analysis.



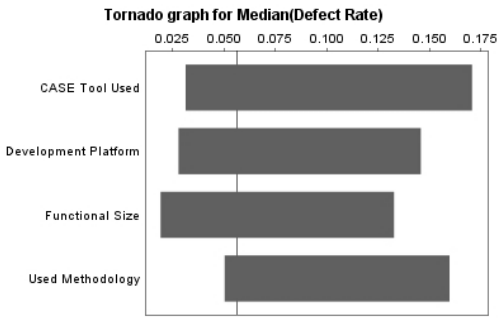


Fig. 5. Tornado graphs illustrating sensitivity of dependent variables

We can observe that the most influential predictor for *productivity rate* is *used methodology* followed by *language type* and *user base - locations*. The least influential predictors for *productivity rate* are *development platform* and *development type*. *CASE tool used* is the most influential predictor for *defect rate*. Slightly lower impact on *defect rate* have *development platform*, *functional size* and *used methodology*. We can observe that the strength of predictors' impact varies more among predictors of *productivity rate* than of *defect rate*.

4. SUMMARY

We introduced the PDR model for estimating productivity and defect rates based on environmental factors. This model contains factors which we identified as influencing productivity and defect rates in the ISBSG dataset. Due to problems with data quality we did not build the model purely from the data with unsupervised learning. Rather we adjusted many of the relationships according to our expert knowledge and incorporated results from other sources.

We discussed various cases in which we analyzed the impact of particular predictors as well as different combinations of them on productivity and defect rates. In performed sensitivity analysis we confirmed different strength of impact of particular predictors on dependent variables. Estimates obtained by this model can be successfully passed to another model developed earlier, called the Productivity Model, which enables analysing trade-offs between key software project variables.

REFERENCES

[1] ISBSG, *Estimating, Benchmarking & Research Suite Release 9*, International Software Benchmarking Standards Group, 2005, www.isbsg.org.

[2] JONES C., *Software Quality in 2002: A Survey of the State of the Art*, Software Productivity Research, 2002.

[3] JONES C., *The Impact of Poor Quality and Canceled Projects on the Software Labor Shortage*, Technical Report, Software Productivity Research, Inc., 1998.

[4] MAXWELL K.D., *Applied Statistics for Software Managers*, Prentice Hall PTR, Upper Saddle River, NJ, 2002.

[5] RADLIŃSKI Ł., FENTON N., NEIL M., MARQUEZ D., *Improved Decision-Making for Software Managers Using Bayesian Networks*, Proc. of 11th IASTED Int. Conf. Software Engineering and Applications (SEA), Cambridge, Massachusetts, Nov. 2007, pp. 13–19.

[6] RAINER A., HALL T., *Identifying the causes of poor progress in software projects*, Proc. 10th Int. Symposium on Software Metrics, Sept. 2004, pp. 184–195.

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

[8] SASSENBURG J.A., *Design of a Methodology to Support Software Release Decisions (Do the Numbers Really Matter?)*, PhD Thesis, University of Groningen, 2006.