Available online at www.sciencedirect.com

ScienceDirect

IERI Procedia 6 (2014) 15 – 21

Procedia IERI

www.elsevier.com/locate/procedia

2013 International Conference on Future Software Engineering and Multimedia Engineering

Suitability of KNN Regression in the Development of Interaction Based Software Fault Prediction Models

Rinkaj Goyal[a, *], Pravin Chandra[a], Yogesh Singh[a]

[a]*University School of Information & Communication Technology, Guru Gobind Singh Indraprastha University, Delhi 110078*

Abstract

Accurate fault prediction is an indispensable step, to the extent of being a critical activity in software engineering. In fault prediction model development research, combination of metrics significantly improves the prediction capability of the model, but it also gives rise to the issue of handling an increased number of predictors and evolved nonlinearity due to complex interaction among metrics.

Ordinary least square (OLS) based parametric regression techniques cannot effectively model such nonlinearity with a large number of predictors because the global parametric function to fit the data is not known beforehand. In our previous studies[1–3], we showed the impact of interaction in the combined metrics approach of fault prediction and statistically established the simultaneous increment in the predictive accuracy of the model with interaction.

In this study we use K-Nearest Neighbor (KNN) regression as an example of nonparametric regression technique, otherwise well known for classification tasks in the data mining community. Through the results derived here, we empirically establish and validate the hypothesis that the performance of KNN regression remains ordinarily unaffected with increasing number of interacting predictors and simultaneously provides superior performance over widely used multiple linear regression (MLR).

Peer review under responsibility of Scientific Committee of of Information Engineering Research Institute

[*] Corresponding author. Tel.: +91-8826020315;
E-mail address: rinkajgoyal@gmail.com.

Peer review under responsibility of Scientific Committee of of Information Engineering Research Institute
doi:10.1016/j.ieri.2014.03.004

Keywords: Fault prediction; Interaction; K-Nearest Neighborhood; Object Oriented Metrics; Regression analysis

1. Introduction

Software metrics are statistical predictors and estimators which assigns numerical values to some attributes of a product, process, or resource in software development [4][5]. These are internal metrics, categorized as product, process and people metrics.

Fault-proneness is an external metric and a fault prediction model uses internal software metrics to predict faults in a newly developed module of the project even before it is released and tested [6].

Usage of the combination of metrics in fault prediction model development is relatively young. D'Ambros et al. [7] statistically analyzed the benefits and relative advantages of using a combination of source code metrics and other metrics derived using information theory to predict bugs. Lee et al. [8] proposed 56 interaction metrics capturing developer's behavioral pattern and empirically analyzed their effect on software quality.

In recent years, empirical software engineering has seen a huge upsurge in the use of nonparametric regression based techniques accruing to the public availability of a multitude of software repositories and progressive research shown by machine learning and data mining community [9]. Nonparametric techniques like Regression Tree, Random forest, support vector machine, Neural Network etc. have been extensively reported in literature. Nonetheless, the effect of the combination and interaction of metrics in empirical analyses has not been explored in much detail.

In our previous studies[1–3] we took the interrelatedness of these metrics into account and statistically established the extent to which such interaction improves the explanatory power of MLR based predictive models. We then conducted stepwise regression to identify influential metrics to avoid over fitting of data. Furtherance of this, prevalence of tree based approach generating piecewise linear models at leaves was also investigated. However, in this study we investigate the relevance of nonparametric regression technique i.e. KNN regression in the development of fault prediction model whilst also considering interaction between metrics. The effectiveness of method is supported and validated by established statistical measures when applied to a publicly available bug dataset. Interaction between metrics results in a large number of predictors depending upon the degree of relatedness and sometimes such interaction between metrics may not be explicitly recognized. This makes traditional parametric regression of limited use, since the functional anatomy of the dependency among metrics and the fault cannot be determined a priori.

Nonparametric regression is an alternative approach to model complex interactions by deriving the functional form of models from the data itself.

The remainder of this paper is organized as follows: Section 2 describes the experimental setup and methods employed; section 3 interprets the results of this study. Our conclusion is presented in section 4.

2. Experimental setup

2.1. The Data - Bug prediction data set

This study uses the same publicly available bug-prediction dataset which has been interrogated in our earlier studies and available at (http://bug.inf.usi.ch) [7]. We have taken single version data of four Java based software module i.e. Eclipse, Mylyn, Equinox and PDE and considered 6 Chidamber and Kemerer (CK) metrics and 11 OO (Object Oriented) metrics for analysis purposes [10]. A brief outline of the metrics used in this study is provided in Appendix A.

2.2. Modeling Methodology

In this segment first, we briefly introduce linear regression highlighting the effect of interaction between variables. Next, KNN regression along with the details of tuning parameters is discussed [11][12].
In linear regression, for an input space vector $X^T = (X_1, X_2 ... X_N)$ of N predictors, the predicted output Y is computed as follows (Refer Eq. 1).

$$y = \beta_0 + \sum_{i=1}^{N} x_i \beta_i \tag{1}$$

To fit the linear model to training dataset, least square method is generally used to minimize the residual sum of squares (RSS) [13] (Refer Eq. 2).

$$RSS = \sum \left(y_i - f(x_i)\right)^2 \tag{2}$$

However, in linear regression independent variables (predictors) may have interdependence. To incorporate the interaction effect of combined predictors, an additional level of regression ought to be included in Eq.1 (Refer Eq. 3). Eq.3 represents two-term interaction used in this study.

$$y = \beta_0 + \sum_{i=1}^{N} x_i \beta_i + \sum_{\substack{i,j \\ i>j}}^{N} x_i x_j \beta_{ij} \tag{3}$$

KNN regression is an instance based lazy learning algorithm. Being a nonparametric regression it does not make any assumption on the distribution of data thereby stimulate training phase. It learns complex target function quickly without losing information. For a given input x of training data, K observations with x_i in the proximity are taken into account and the average of the response of those K independent variables gives $\hat{y}$ (Refer Eq. 4).

$$\hat{y}(x) = \frac{1}{k} \sum_{x_i \in N_{k(x)}} y_i \tag{4}$$

Where $N_{k(x)}$ depicts K closest points in the neighborhood of x. various distance measures quantify closeness between points but Euclidean distance is commonly practiced. In high dimensional data, it is appropriate to assign different weights to the variables in the neighborhood, especially with large value of K. These weights are specified by 'Kernel Methods' using a density function. In the implementation of KNN regression used in this study, Gaussian kernel method is used with Gaussian density function (Refer Eq. 5) [11].

$$G_k(x_0, x) = \frac{1}{k} \exp\left\{ - \frac{\|x - x_0\|^2}{2k} \right\} \tag{5}$$

$G_k(x_0, x)$ assign weights to a neighborhood point that decreases smoothly with the squared Euclidean distance from the target point x.

The performance of predictive model developed by applying aforesaid regression based techniques is represented by quantifying the difference in between the output estimated by the model and the actual output. The R^2, Adjusted R^2 (Adj.R^2), root mean square (RMSE) and root relative squared error (RRSE) values are statistically established measures to quantify this difference.

The validity of the model thus developed is further corroborated by the error functions namely mean absolute error (MAE) and Absolute CNK. The MAE measures the average magnitude (absolute values) of the errors in a set of predictions. It is an unambiguous and natural error estimate for inter-comparisons of average model performance [14]. Absolute value of $C_{\text{Neighbours}}$ - K (CNK Absolute) is a density based reliability estimate proposed by Bosnic et al. [15]. It is defined as the difference between the average response of k neighbours and the predicted value. In this study we apply this criterion to compare the reliability of the developed models with 5 nearest neighbours.

3. Case study results

The efficacy of KNN regression over multiple linear regression (MLR) has been empirically investigated with four Java based software modules of a bug prediction dataset. This comparison between regression techniques makes use of two separate categories of metrics; CK type and other object oriented (OO) type metrics (Refer Section 2.1).

A combination of these two categories of metrics is considered in isolation, as well with interacting terms (Refer Eq. 3), to appropriately highlight the relevance of nonparametric techniques in managing a large number of predictors and the effect of interaction. For each software module, data for 6 CK type metrics and 11 OO type metrics, as contained in the dataset, is used for analysis. Taking a two-term interaction without self-interacting terms results in ${}^{n}C_2$ + n number of predictors.

In order to sustain the conclusion stability of prediction models thus developed, we have utilized a sample size of 70% of the dataset for training, and the remaining 30% to test the model. A stratified random sampling method is used to generate separate training and testing data. These experiments have been simulated in Orange canvas [16]. Following are the experimental results with four software modules:

Predictive modeling statistics generated for software modules are presented in Table 1 and consider metrics in isolation as well as in combination (including interaction).

Following discussion pertains to Eclipse software module;

For CK Metrics, the value of Adj.R^2 is 0.120 and after interaction, the value goes up to 0.470. The fall in the value of the corresponding RMSE and RRSE also affirms such improvement in the value of the Adj.R^2. Statistics generated for OO and CK+ OO endorse the impact of interaction as indicated above by using MLR.

In response to the consideration that this approach may lead to increased multicollinearity, statistical dynamics shows that the high collinearity between interacting terms, e.g. PQ and its independent component P is not elusive, contrary to the collinearity between P and Q [17].

When compared to MLR, KNN regression consistently gives improved model performance, as is evident by the improved values of statistical measures for the metrics in isolation, as well as in combination. For CK metrics, the value of Adj.R^2 is 0.765 whereas after combining CK metrics with OO metrics (whilst also considering interaction) the value of Adj.R^2 becomes 0.924. Though we obtain an enhanced value of Adj.R^2 this increment is insignificant in comparison to the rise in the number of predictors i.e. from 21 to 153. This hypothesizes the superiority of KNN regression over MLR to effectively manage interactions while also remains unpretentious to the effect of interaction.

To validate the predictive accuracy of these models, we use Mean absolute error (MAE) and CNK Absolute as statistical predictors. The data in Table 1 specifies a notable change in these values, when the combination and interaction between metrics is considered and compared between the models produced by applying MLR and KNN regression.

For CK Metrics with interaction in the MLR model the value of MAE is 0.850, while the corresponding value for the KNN regression based model is 0.706. A similar dip can be observed in the value of absolute CNK. Subsequently for the combination of CK and OO Metrics with interaction, the KNN regression based model not only gives a low value of (improved) MAE and corresponding absolute CNK but also shows significant differences with the corresponding values of the MLR based model. This is reflected through the significant reduction in value of MAE from 3.635 to 0.675.

The same experiments were conducted for three other software modules, i.e. Equinox, Mylyn and PDE. KNN regression based models consistently returned better values of Adj.R^2 and a correspondingly better (decreased) value of RMSE and RRSE. Additionally similar higher values of Adj.R^2 (With and without interaction) across all the software modules substantiate the non-parametric nature of KNN regression.

Table 1: Comparison of statistical measures of model performance

	MLR			KNN			MLR		KNN	
Software modules	RMSE	RRSE	Adj.R^2	RMSE	RRSE	Adj.R^2	MAE	CNK Absolute	MAE	CNK Absolute
Eclipse						Metrics without interaction				
CK	**0.820**	**0.935**	**0.120**	0.424	0.483	**0.765**	0.646	0.267	0.687	0.291
OO	0.730	0.699	0.507	0.512	0.490	0.757	0.813	0.412	0.744	0.381
CK+OO	0.853	0.846	0.272	0.315	0.313	0.901	0.623	0.352	0.636	0.296
						Metrics with two term interaction				
CK	**0.596**	**0.721**	**0.470**	0.295	0.356	0.870	**0.850**	0.525	**0.706**	0.345
OO	0.617	0.582	0.637	0.307	0.289	0.910	1.376	1.147	0.570	0.266
CK+OO	0.452	0.420	0.791	0.273	0.254	**0.924**	**3.635**	3.681	**0.675**	0.273
Equinox						Metrics without interaction				
CK	1.050	0.679	0.415	0.688	0.445	0.798	0.467	0.231	0.443	0.284
OO	0.879	0.692	0.462	0.571	0.450	0.791	0.446	0.303	0.418	0.283
CK+OO	0.888	0.505	0.731	0.841	0.478	0.759	0.426	0.280	0.360	0.261
						Metrics with two term interaction				
CK	0.845	0.740	0.530	0.561	0.491	0.742	0.426	0.306	0.427	0.317
OO	0.638	0.654	0.505	0.315	0.323	0.869	0.922	0.707	0.405	0.216
CK+OO	0.345	0.372	0.738	0.189	0.203	0.922	8.606	8.457	0.407	0.273
Mylyn						Metrics without interaction				
CK	0.575	0.943	0.109	0.358	0.586	0.655	0.283	0.167	0.301	0.203
OO	0.395	0.920	0.149	0.151	0.352	0.875	0.266	0.156	0.253	0.178
CK+OO	0.561	0.907	0.169	0.289	0.467	0.780	0.288	0.165	0.276	0.175
						Metrics with two term interaction				
CK	0.521	0.915	0.153	0.237	0.417	0.824	0.361	0.247	0.275	0.171
OO	0.529	0.729	0.449	0.176	0.242	0.939	0.295	0.213	0.246	0.139
CK+OO	0.408	0.569	0.648	0.249	0.347	0.869	0.578	0.492	0.274	0.186
PDE						Metrics without interaction				
CK	0.594	0.937	0.119	0.359	0.566	0.678	0.356	0.262	0.351	0.213
OO	0.717	0.888	0.207	0.297	0.368	0.863	0.295	0.172	0.278	0.180
CK+OO	0.604	0.898	0.286	0.259	0.385	0.850	0.336	0.209	0.314	0.196
						Metrics with two term interaction				
CK	0.579	0.888	0.200	0.351	0.538	0.706	0.358	0.201	0.368	0.222
OO	0.443	0.779	0.366	0.141	0.248	0.936	0.393	0.236	0.333	0.155
CK+OO	0.457	0.688	0.472	0.219	0.330	0.879	1.579	1.430	0.348	0.177

4. Conclusion and Future work

Through the results derived from this study it has been empirically determined that the performance of KNN regression remains mostly unaffected due to the large number of interacting metrics whilst also resulting in ameliorated models for software fault prediction. One of the aims of this study is to present ways in which software scientists can improve empirical analyses by employing this non-parametric regression technique in other areas of software measurement where compounding effects are witnessed. The significance and future scope of this study lies in the effective management of interacting attributes during the different phases of the software development life cycle. Furthermore the human aspect of software development, quantified by people metrics, may play a role in the determination of other software measurements which is yet to be explored in depth. Such a human centric approach would include a number of interacting metrics thereby making such distance based, non-parametric regression techniques more suitable

References

[1] R. Goyal, P. Chandra, and Y. Singh, “Identifying influential metrics in the combined metrics approach of fault prediction,” SpringerPlus, vol. 2, no. 1, p. 627, 2013.
[2] R. Goyal, P. Chandra, and Y. Singh “Impact of interaction in the combined metrics approach for fault prediction,” Software Quality Professional (ASQ), vol. 15, no. 3. pp. 15–23, 2013.
[3] R. Goyal, P. Chandra, and Y. Singh, “Comparison of M5’ Model Tree with MLR in the development of fault prediction models involving interaction between metrics,” in Proceedings of the International Conference on Systems, Computing Sciences and Software Engineering (SCSS 13), University of Bridgeport, USA, 12-14th December 2013.
[4] N. E. Fenton and M. Neil, “Software metrics: roadmap,” in Proceedings of the Conference on the Future of Software Engineering, 2000, pp. 357–370.
[5] L. M. Laird and M. C. Brennan, Software measurement and estimation: a practical approach, vol. 2. John Wiley \& Sons, 2006.
[6] C. Catal and B. Diri, “Software fault prediction with object-oriented metrics based artificial immune recognition system,” Product-Focused Software Process Improvement, pp. 300–314, 2007.
[7] M. D’Ambros, M. Lanza, and R. Robbes, “An extensive comparison of bug prediction approaches,” in Mining Software Repositories (MSR), 2010 7th IEEE Working Conference on, 2010, pp. 31–41.
[8] T. Lee, J. Nam, D. Han, S. Kim, and H. P. In, “Micro interaction metrics for defect prediction.,” in SIGSOFT FSE, 2011, pp. 311–321.
[9] D. Rodriguez, I. Herraiz, and R. Harrison, “On software engineering repositories and their open problems,” in Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2012 First International Workshop on, 2012, pp. 52–56.
[10] S. R. Chidamber and C. F. Kemerer, “A metrics suite for object oriented design,”, IEEE Transactions on Software Engineering , vol. 20, no. 6, pp. 476–493, 1994.
[11] T. Hastie, R. Tibshirani, and J. J. H. Friedman, The elements of statistical learning, vol. 1. Springer New York, 2001.
[12] T. A. Runkler, Data Analytics: Models and Algorithms for Intelligent Data Analysis. Vieweg+ Teubner Verlag, 2012.
[13] I. Uysal and H. A. Guvenir, “An overview of regression techniques for knowledge discovery,” Knowledge Engineering Review, vol. 14, no. 4, pp. 319–340, 1999.
[14] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” Climate Research, vol. 30, no. 1, p. 79, 2005.
[15] Z. Bosni’c and I. Kononenko, “Estimation of individual prediction reliability using the local sensitivity analysis,” Applied intelligence, vol. 29, no. 3, pp. 187–203, 2008.
[16] J. Demvsar et al., “Orange: data mining toolbox in Python,” Journal of Machine Learning Research, vol. 14, pp. 2349–2353, 2013.
[17] R. J. Friedrich, “In defense of multiplicative terms in multiple regression equations,” American Journal of Political Science, pp. 797–833, 1982.

Appendix A

CK metrics [10]	Interpretation	OO (Object oriented)	Interpretation
Weighted Methods per Class (WMC)	Weighted sum of all the methods defined in a class.	NOM NOPM	Number of methods Number of public methods.
Coupling Between Object (CBO)	Dependency of one class on other classes in the design.	NOPRM NOMI	Number of private methods Number of methods inherited.
Depth of the inheritance Tree (DIT)	Length of the longest path from a given class to the root class in the inheritance hierarchy.	Fan-in	Number of other classes that reference the class.
Number of Children (NOC)	Count of the number of immediate child classes inherited from a given class.	Fan-out	Number of other classes referenced by the class.
Response for the classes (RCF)	Number of methods in the set of all methods that can be invoked in response to a message sent.	NOAI NOA NLOC	Number of attributes inherited. Number of attributes. Number of lines of code.
Lack of Cohesion metric (LCOM)	Count of the number of method pairs with zero similarity.	NOPRA NOPA	Number of private attributes. Number of public attributes.