Can Data Transformation Help in the Detection of Fault-Prone Modules?

Y. Jiang, B. Cukic, T. Menzies

Lane Department of CSEE

West Virginia University

DEFECTS 2008

Background

- **Prediction of fault-prone modules is one of the most active research areas in empirical software engineering.**
  - Also the one with a significant impact to practice of verification and validation.
- **Recent results indicate that current methods reached a "ceiling effect".**
  - Differences between (most) classification algorithms not statistically significant.
  - Different metrics suites do not seem to offer a significant advantage. Feature selection indicates relatively small number of metrics perform as well as larger sets.

Motivation

- **Overcoming the "ceiling" requires experimentation with new approaches appropriate for our domain.**
  - Recent history matters the most [Weyuker et. al]
  - Inclusion of the developer's social networks [Zimmerman et. al.].
  - Incorporating expert opinions [Khoshgoftaar et. al.].
  - Utilization of early life-cycle metrics [Jiang et. al.]
  - Incorporating misclassification costs [Jiang et. al.]
  - (*your best ideas here*)
- **Transformation of metrics data suggested as a possible venue for the improvement [Menzies, TSE'07]**

Goal of study

- **Evaluate whether transformation (preprocessing) helps improving the prediction of fault-prone software modules?**

- **Four data transformation methods are used and their effects on prediction compared:**
  a) The original data, no transformation (*none*)
  b) Ln transformation (*log*)
  c) Discretization using Fayyad-Irani's Minimum Description Length algorithm (*nom*)
  d) Discretization of log transformed data (*log&nom*)

The Impact of Transformations

Table 2: The average number of distinct values for attributes in MDP.

dataset	none	Log	nom	log&nom	# attrib.
cm1	63.27	63.27	1.81	1.78	37
kc1	68.38	68.38	3.1	3.1	21
kc3	51.46	51.46	1.9	1.9	39
kc4	34.77	34.77	1.69	1.69	13
pc1	69.84	69.84	1.68	1.65	37
pc3	72.54	72.54	2.11	2.11	37
pc4	64.89	64.89	2.22	2.22	37
mw1	53.14	53.14	1.68	1.65	37
mc2	51.85	51.85	1.64	1.62	39
ave.	58.90	58.90	1.98	1.97	33

Experimental Setup

- 9 data sets from Metrics Data Program (MDP).
- 4 transformation methods.
- 9 classification algorithms for each transformation.
- Ten-way cross-validation (10x10 CV).
- Evaluation technique: Area Under the ROC curve (AUC).
- Total AUCs: 9 datasets x 4 transformation x 9 classifiers x 10CV = 3240 models
- Boxplot diagrams depict the results of each fault prediction modeling technique.
- Nonparametric statistical hypothesis test tests the difference between the classifiers over multiple data sets.

Metrics Data Program (MDP) data sets

Table 1: Datasets used in this study

data	# mod.	# faulty mod.	% faulty	notes	lang.
CM1	505	81	16.04%	Spacecraft instrument	C
KC1	2107	293	13.9%	Storage management for receiving/processing ground data	C++
KC3	458	29	6.3%	Storage management for ground data	Java
KC4	125	60	48%	A ground-based subscription server	Perl
PC1	1107	73	6.59%	Flight software from an earth orbiting satellite	C
PC3	1563	163	10.43%	Flight software for earth orbiting satellite	C
PC4	1458	178	12.21%	Flight software for earth orbiting satellite	C
MW1	403	27	6.7%	A zero gravity experiment related to combustion	C
MC2	161	52	32.30%	A video guidance system	C++

10 different classifiers used

Classification algorithms used in the study.

	learner	Abbrev.
1	Random Forest	rf
2	Bagging	bag
3	Logistic regression	lgi
4	Boosting	bst
5	Naivebayes	nb
6	Jrip	jrip
7	IBk	IBk
8	J48	j48
9	Decorate	dec
10	AODE	aode

Statistical hypothesis test

- **We use the nonparametric procedure for the comparison.**
  - 95% confidence level used in all experiments.
- **Performance comparison between more than two experiments:**
  - *Friedman* test determines whether there are statistically significant differences amongst in classification performance across ALL experiments.
  - If yes, after-the-fact Nemenyi test ranks different classifiers.
- **For the comparison of two specific experiments, we use Wilcoxon's signed rank test.**

Classification results using the original data

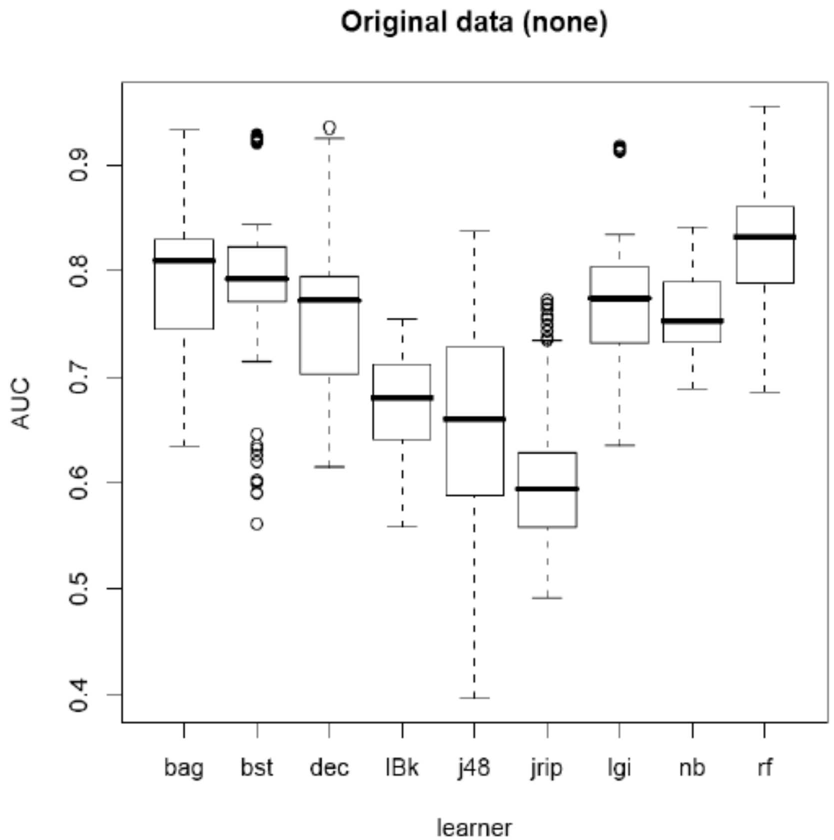

Figure 1: Boxplot diagrams with original data.

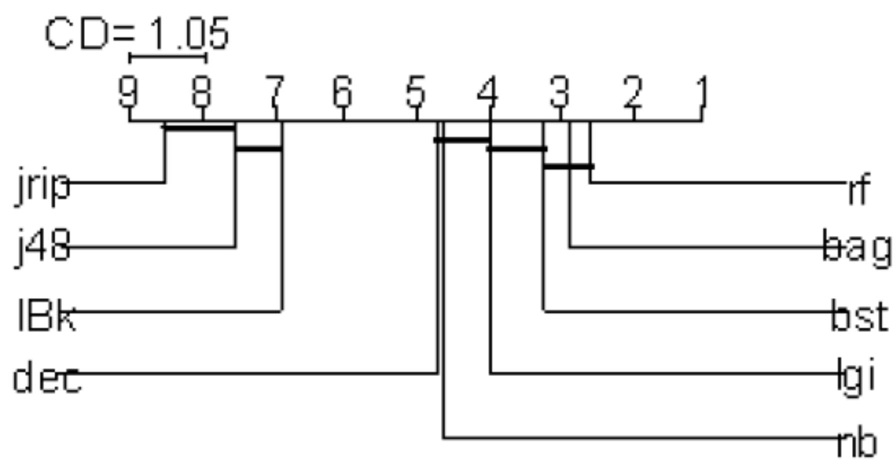

Figure 2: Comparison of all classifiers in the original data domain (*none*) with the Nemenyi test using 95% confidence interval.

Classification results using the log transformed data

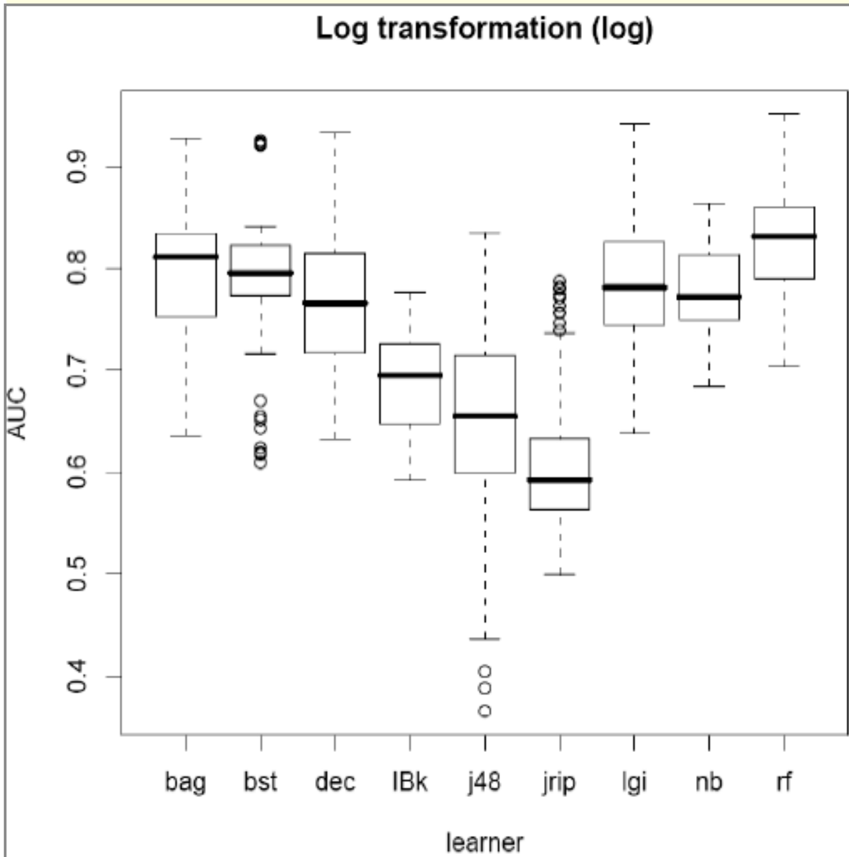

Figure 3: Boxplot diagrams with (*log*) data.

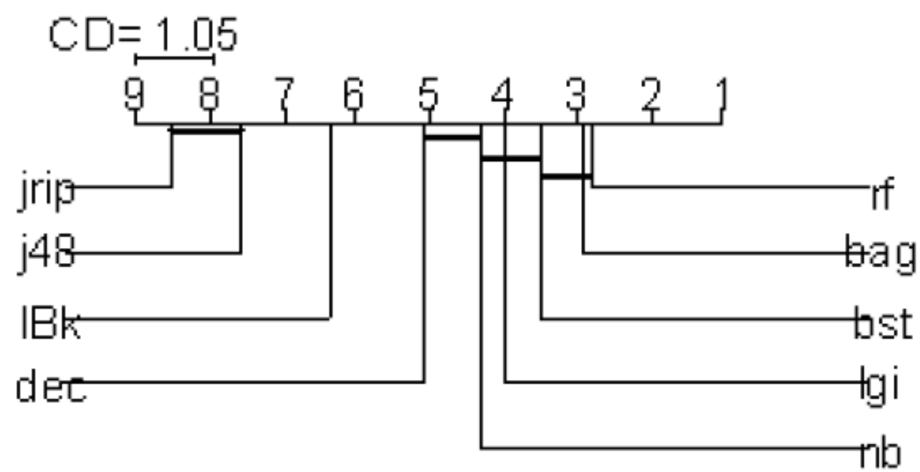

Figure 4: Comparison of all classifiers in the *log* data domain using the Nemenyi test

Classification results using the discretized data

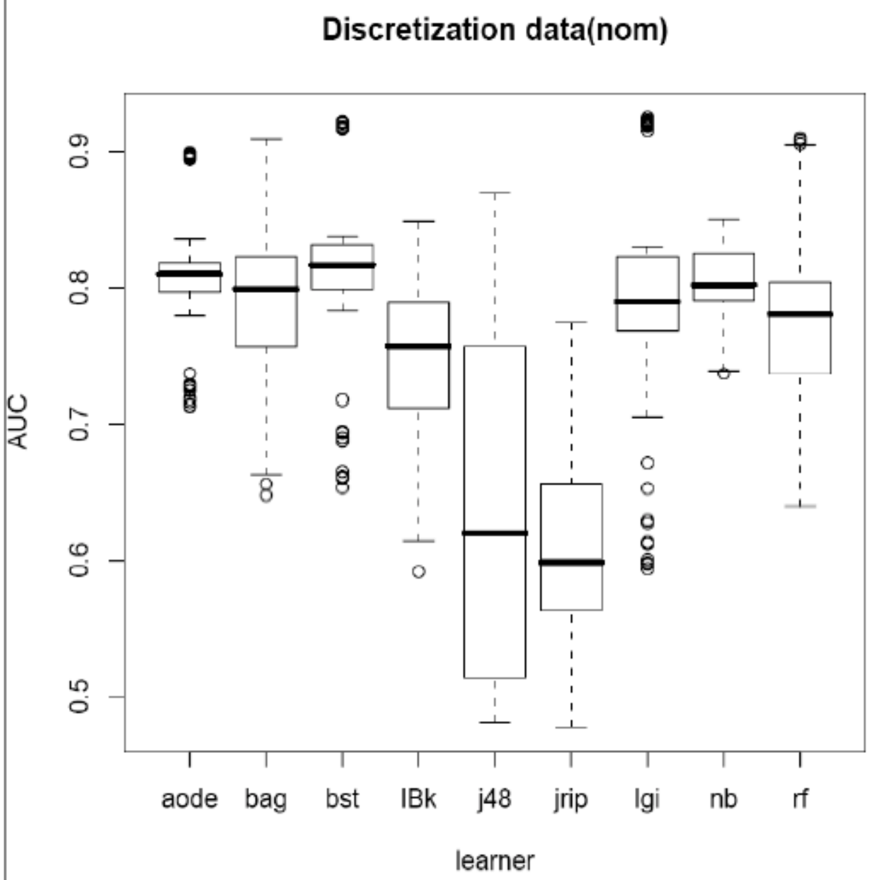

Figure 5: Boxplot with the discretized (nom) domain.

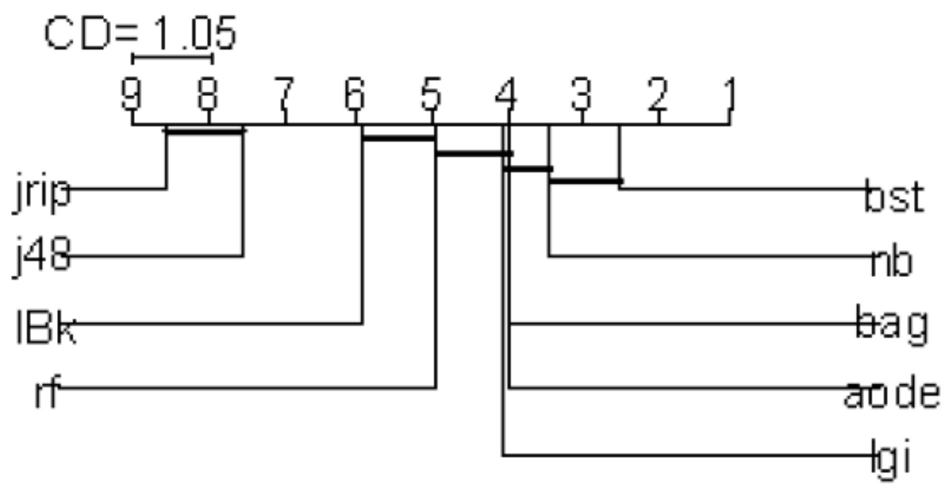

Figure 6: Comparison of all classifiers in the nom data domain with the Nemenyi test

Classification results using the discretized log transformed data

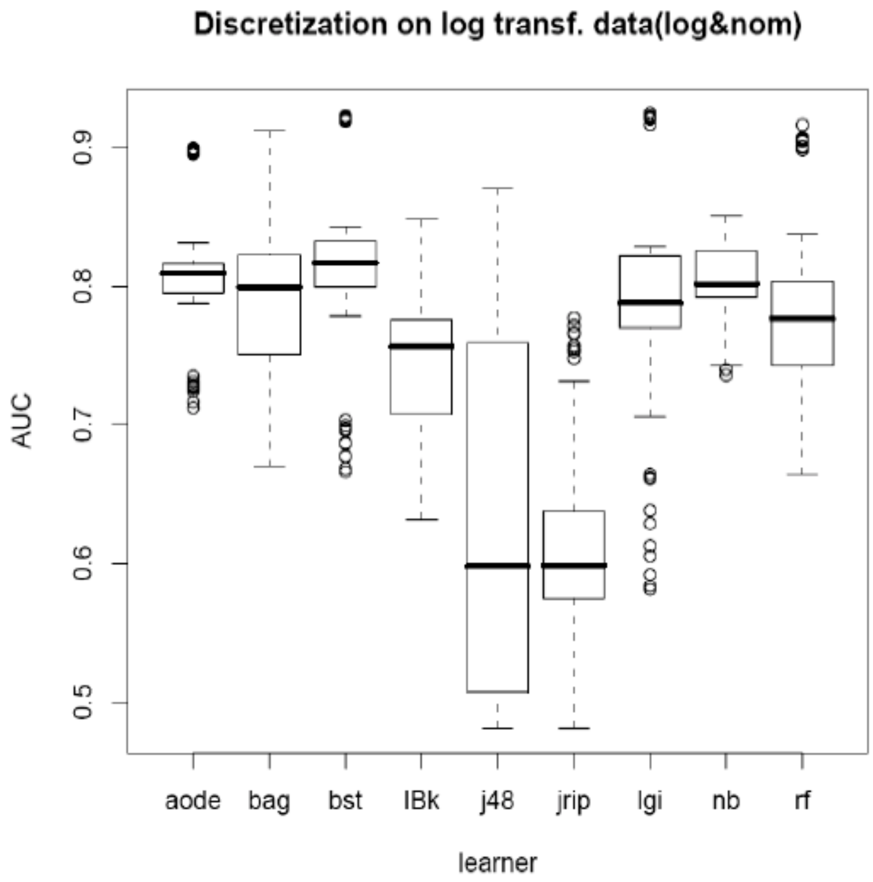

Figure 7: Boxplot in $log\&nom$**.**

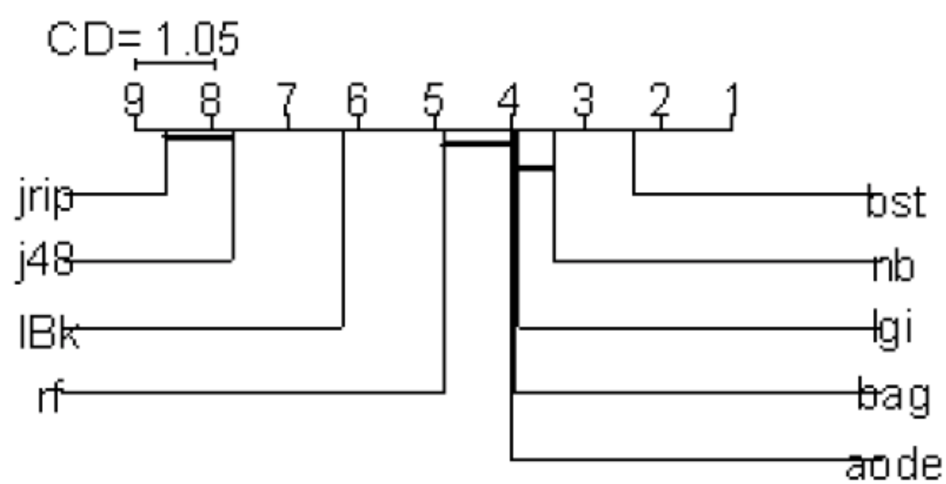

Figure 8: Comparison of all classifiers in the $log\&nom$ **data domain with the Nemenyi test**

Comparing results over different data domains

- **Random forest ranked as one of the best classifiers in *the original* and *log transformed* domains.**
- **Boosting ranked as one of the best classifiers in the *experiments* with the discretized data.**
- **The performance comparison reveals statistically significant difference.**
  - We compared random forest (*none* and *log*) *vs*. boosting (*nom* and *log&nom*) using the Wilcoxon signed ranked test, using 95% confidence interval
- ***Random forest in original and log transformed domains beats Boosting in discretized domains.***

Comparing the classifiers across the four transformation domains

Table 5: Classifier performance in the four transformation domains.

rf	none=log > nom=log&nom
bag	none=log > nom=log&nom
jrip	none=log > nom=log&nom
bst	nom=log&nom > none=log
IBk	log&nom > none; nom > log
nb	nom=log&nom > none=log
lgi	log=nom=log&nom > none
j48	none=log=log&nom=nom
dec	none=log
aode	nom=log&nom

Better for none and *log*

Better for discretized data

all the same

Conclusions

- Transformation did not improve overall classification performance, measured by AUC.
- Random forest is reliably one of the best classification algorithms in the original and log domains.
- Boosting offers the best models in the discretized data domains.
- NaiveBayes is greatly improved in the discretized domain.
- Log transformation rarely affects the performance of software quality models.

Ensuing Research

- **Data transformation unlikely to make the impact on breaking the "performance ceiling".**
- **The heuristics for the selection of the "most promising" classification algorithms.**
- **So, how to "break the ceiling"?**
  - We may have ran out of "low hanging research fruit".
  - Possible directions:
    - Fusion of measures from different development phases.
    - Human factor.
    - Correlating with operational profiles.
    - Business context.
    - ???