Does transformation help?

Yue Jiang,Bojan Cukic,Tim Menzies
Dept. of LCSEE
West Virginia University
Morgantown,WV,USA
cukic,yue@csee.wvu.edu;tim@menzies.us

ABSTRACT

Data preprocessing(or transformation) plays an important role in data mining and machine learning. In this study, we investigate the effect of 4 different preprocessing methods to fault-proneness prediction by using 9 datasets from NASA Metrics Data Programs (MDP) and 10 machine learners. From our experiments,we found that log transformation do not affect most learners' performance except bagging classifier. However, discretization affect the performance of many different learners. Learners such as random forest perform better in original and log transformation data. Learners such as boosting and NaiveBayes perform significantly better on discretized data. After comparing the performance of different learners in different transformation methods, we found that the one which has the best performance is random forest in original and log transformation data. Therefore,when conducting transformation on the data, in order to gain better performance for a learner, we had better based on the specific learner instead of using universal transformation for all kinds of learners.

1. INTRODUCTION

Data mining is the process of finding useful information from datasets. Dataset is generally contain noise which affect the performance of a machine learner. Hence, many preprocessing (transformation) methods are proposed to carried out before the actual learning process. Normalization [7], linear and non-linear transformation [7], feature subset selection [10], principal component analysis [11], and discretization [6] are often conducted before prediction.

In this study, we compare four different preprocessing methods in 9 NASA MDP datasets across 10 different machine learners. These four different preprocessing methods are the original data ($none$), log transformation data (log), discretized data (nom), and discretized on the log transformation data ($log\&nom$). From our experiments,we found that log transformation do not affect most learners' performance. However, discretization affect the performance of many different learners. Learners such as random forest perform better in original ($none$) and log transformation (log) data. More specifically, Random forest performs better in original data with more number of individual data values. Bagging has the best performance in log transformation data (log). Discretization data is better for boosting, NaiveBayes,logistic and IBk machine learners. Random forest in the original has the best performance across all kinds of learners in these four kinds of transformation methods.

2. EXPERIMENT SETUP

In this study, we use 10 by 10 way cross-validation ($10x10$ CV) to generate Receiver Operating Characteristic (ROC) curves through all possible threshold. The Area Under the ROC curve, referred to as AUC, is used as the performance measurement to evaluate different learners.

Cross-validation is the statistical practice of partitioning a sample of data into two subsets: training and testing subset. We use 10 by 10 way cross-validation in all experiments. 90% of data is randomly assigned to the training subset and the remaining 10% of data is used for testing. The data is randomly divided into 10 fixed bins of equal size. We leave one bin to act as test data and the other 9 bins is used to train the learners. This procedure is repeated 10 times.

ROC curves provide an intuitive way to compare the classification performance of different metrics. An ROC curve is a plot of the Probability of Detection (pd) as a function of the Probability of False alarm (pf) across all the possible experimental threshold settings. AUC is a numeric performance evaluation measure directly associated with an ROC curve. It is very common to use AUC to compare the performance of different classifiers. In this study, we use AUC as a measurement of the performance of learners.

A box and whisker diagram,referred as a boxplot diagram, graphically depicts numerical data distributions using five first order statistics: the smallest observation, lower quartile (Q1), median, upper quartile (Q3), and the largest observation. The box is constructed based on the interquartile range (IQR) from Q1 to Q3. The line inside the box depicts the median which follows the central tendency. The whiskers indicate the smallest observation and the largest observation. Boxplot diagrams are shown in this study to visually depict the learners performance too.

2.1 Used DataSet and Preprocessing Methods

There are 9 datasets shown in Table 1 from Metrics Data Programs (MDP) [2] are used. In this study, we predict whether a module is fault-prone or not instead of predicting how many faults a module has.

Four kinds of preprocessing methods are listed as follows.

1. **none:** This is the original data set without applying any preprocessing methods. All the independent variables in MDP are continuous attributes.

Table 1: Datasets used in this study

Data	mod.#	faulty mod. #	% faulty	Note	language
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++

Table 2: The number of distinct values for an attribute of 4 preprocessing methods in average.

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

2. **log:** The original continuous independent variable values are transformed by taking mathematical log operation. To avoid numerical errors with $\ln(0)$, all numbers under 0.000001 are replaced with $\ln(0.000001)$. This transformation method is reported effective transformation in [12].

3. **nom:** The original independent variable continuous values are discretized to nominal values by using Fayyad and Irani's minimum description length (MDL) method (Weka default discretization method) [6, 15]. We discuss this method in detail below.

4. **log&nom:** The log-transformed data are discretized to nominal values using Fayyad and Irani's discretization method.

Fayyad and Irani's method : We use the recursive minimal entropy discretization method proposed by Fayyad and Irani [8]. It is a supervised, non-parametric discretization method. It uses the class information entropy to partition bin boundaries. The criterion to stop is the minimum description length which do not need the user intervention. For example, assume there is a dataset with S instances, a variable A, and a partition boundary T, there are two classes (S_1, S_2) for the dataset, the class entropy is denoted as *E(A,T;S)*: $E(A,T;S) = \frac{|S_1|}{|S|}E(S_1) + \frac{|S_2|}{|S|}E(S_2)$ (1)
It creates a tree of discretization from top to down. The branch of the partition is recursively discretized and evaluated independently. Thus, in some branches, the continuous values with relatively high entropy will be partitioned very finely, while others that have relatively low entropy will be partitioned coarsely.

Table 2 shows the average number of distinct values for an attributes for the MDP datasets we study. For example, in the first cell, the number is "63.27". It means that in CM1 data, under the preprocessing method of "none" (the original data set), there are 63.27 number of distinct values per attribute in average. From Table 2, we find that:

- The average number of distinct values for $none$ and log are the same.
- The average numbers of distinct values for $none$ and log are larger than that of nom and $log\&nom$.
- The average number of distinct values for nom and $log\&nom$ are similar.

Table 3: Machine learners used.

	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

2.2 Different Learners

Table 3 shows the learners we used in this study from Weka [15].

Random Forest (rf) [3] is a decision tree-based classifier demonstrated to have good performance in software engineering studies by Guo $et\ al$ [9]. As implied from its name, it builds a "forest" of decision trees. The trees are constructed using the following strategy:

- The root node of each tree contains a bootstrap sample data of the same size as the original data. Each tree has a different bootstrap sample.
- At each node, a subset of variables are randomly selected from all the input variables to split the node and the best split is adopted.
- Each tree is grown to the largest extent possible without pruning.
- When all trees in the forest are built, new instances are fitted to all the trees and a voting process takes place. The forest selects the classification with the most votes as the prediction of new instance(s).

NaiveBayes (nb) "naively" assumes data independence. This assumption may be considered overly simplistic in real life application scenarios. However, in software engineering data sets it's performance is surprisingly good. Naive Bayes classifiers have been used extensively in fault-proneness prediction, for example in [12].

Bagging (bag) stands for bootstrap aggregating. It relies on an ensemble of different models. The training data is resampled from the original data set. According to Witten and Frank [15], bagging typically performs better than single method models and almost never significantly worse.

Boosting (bst) combines multiple models by explicitly seeking models that complement one another. First, it is similar to bagging in using voting for classification or averaging for numeric prediction. Like bagging, boosting combines the models of the same type. However, boosting is iterative. "Whereas in bagging individual models are built separately, in boosting each new model is influenced by the performance of those built previously. Boosting encourages new models to become experts for instances handled incorrectly by earlier ones." [15].

Logistic regression(lgi) is a classification scheme which uses mathematical logistic regression functions. The most popular models are generalized linear models. J48 is a Weka [15] implementation of J.R. Quinlan C4.5 [13] decision tree algorithm.

Decorate is a Weka classifier that builds ensembles of diverse classifiers by using specially generated artificial training examples [15]. According to [15], it outperforms bagging and random forest. It outperforms boosting on small training sets and rivals it on larger training sets [15]. Decorate classifier are only run on data of *none* and *log* two preprocessing methods because of the reason that it does not allow an attribute with only one distinct value, which is not a rarely case when a continuous variable is discretized.

"AODE (averaged, one-dependence estimators) is one of Bayesian method that averages over a space of alternative Bayesian models that have weaker independence assumptions than Naive Bayes. The algorithm may yield more accurate classification than Naive Bayes on datasets with nonindependent attributes." [15]. AODE is only able to classify nominal data. So, we only compare it with others on the nominal data (on *nom* and *log&nom*).

All the learners are run in their default parameter except random forest with 500 trees because in the original random forest algorithm proposed by Breidman [3], the default random forest algorithm is build with 500 trees. The first 8 learners are run on all 4 proprecessing methods of 9 datasets. Decorate only runs on *none* and *log* two preprocessing methods and AODE only run on *nom* and *log&nom* these two preprocessing methods. Thus, in a specific preprocessing method of a data, there are 9 learners run against it.

2.3 Statistical test hypotheses

The most popular method used to evaluate a classifier's performance on a data set is based on 10 by 10 ways cross-validation (10×10 CV). The 10×10 CV results in 10 individual values of AUC. These 10 values are usually similar to each other, given that they come from the same population after randomization. With only 10 values it is difficult to say whether the values follow the normal distribution (i.e., indicate that they obey the central limit theorem). Therefore, using parametric statistical methods which assume a normally distributed population to compare the performance of classifiers may not be justified. A prudent approach calls for the use of nonparametric methods. The loss of efficiency caused by using nonparametric tests is typically marginal [4, 14].

In [5], Demsar overviewed the theoretical work on statistical tests for the comparison of multiple classifiers over multiple data sets. He recommended the Wilcoxon signed rank test for the comparison of two classifiers and the Friedman test with the corresponding post-hoc tests when the comparison includes more than two classifiers. The Wilcoxon signed rank test and the Friedman's test are nonparametric counterparts for paired $t - test$ and analysis of variance (ANOVA) parametric methods, respectively. Demsar advocates these tests largely due to the fact that nonparametric procedures make less stringent demands on the data. However, two issues need attention. First, nonparametric tests do not utilize all the information available. The actual data values (in our case, for example, AUCs) are not used in the test procedure. Instead, the signs or ranks of the observations are used. Therefore, nonparametric procedures will be more powerful than their parametric counterparts, when justifiably used. The second point is that signed rank tests are constructed for the null hypothesis that the difference of the performance measure is symmetrically distributed. For non-symmetric distributions, this test can lead to a wrong conclusion.

All statistical tests conducted here utilize the routines provided in the statistical package R [1]. Based on Demsar's recommendation, we use the Friedman test to evaluate whether there is a difference in the performance amongst the models developed from the 9 different algorithms over the 9 data sets. Provided that the Friedman test indicates statistically significant difference, we need pairwise comparisons of models to determine which model performs the best for each data set.

Assume we are looking at two different learners in a dataset, A and B,in order to derive which learner has a better performance, the appropriate statistical test hypotheses are:
H_0*: There is no difference in the performance of the models which use group A or group B metrics;*
H_1*: The performance of the group A model is better than the performance of group B model;*
H_2*: The performance of the group A model is worse than the performance of group B model.*

First, using the 95% confidence interval, we test whether two learners have the same performance using H_0. The p-value greater than 0.05 in this case indicates no difference in the performance of group A and B models. In such a case, further tests of hypotheses H_1 and H_2 are not necessary since H_0 is the correct one. Otherwise, we test H_1. If the p-value of H_1 is less than 0.05, then H_1 is accepted. Otherwise, if H_1 is rejected H_2 will be tested. After conducting these three hypothesis tests, the relationship in the performance of group A and B models will be clear on the given data set: if H_0: $A = B$, if H_1: $A > B$, if H_2: $A < B$.

Recall that, in a specific preprocessing method of a data, there are 9 learners' performance needed to be compared. In order to get the overall performances of 9 learners across 9 datasets in a specific preprocessing method, we count the number of wins($>$), losses($<$), and ties ($=$) over pairwise comparisons. For a specific learner in a preprocessing of a dataset, there are $8+7+6+5+4+3+2+1 = 36$ comparisons. When a learner A is compared to a learner B, learner B is compared to learner A as well. Thus, totally, there are $36*2 = 72$ comparisons.

3. EXPERIMENTAL RESULT

Since the results of the Mann-Whitney and the Wilcoxon two statistical nonparametric tests do not vary too much, in the following we only show the number of wins,losses and ties of the Wilcoxon test.

3.1 Original Data

Table 4 shows the comparison result from the original data ($none$) listed in the decreasing order of the number of wins. We can tell that random forest (rf) is the best learner in $none$ preprocessing data set from Table 4 which wins 67 out of 72 cases, ties with other

Table 4: Comparison result in original data($none$).

Learner	prep	ties	wins	losses
rf	none	5	67	0
bag	none	12	48	12
bst	none	9	41	22
lgi	none	10	38	24
dec	none	12	35	25
nb	none	11	32	29
IBk	none	6	12	54
j48	none	6	11	55
jrip	none	5	2	65

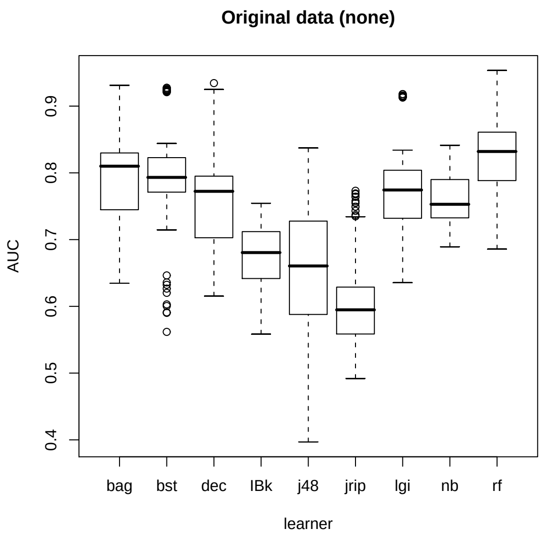

Figure 1: Boxplot diagrams in the original data.

Table 5: The trend in original ($none$) dataset

dataset	Order of log transformation
general	rf >bag>bst>lgi>dec>nb>IBk>J48>jrip
CM1	rf >bst>lgi>nb>bag>dec>IBk>J48=jrip
KC1	rf>bag>lgi>nb>bst>dec>IBk>J48>jrip
KC3	rf>nb=bag=bst>lgi=dec>IBk>J48=jrip
KC4	rf=bag =dec>J48=bst>lgi>jrip=nb>IBk
PC1	rf >bag>bst>lgi=dec>nb>IBk=J48>jrip
PC3	rf>lgi>bag=bst>nb=dec>IBk=J48>jrip
PC4	rf>bag=bst=dec>lgi>nb>J48>IBk=jrip
MW1	rf=bst>nb=bag>log=dec>IBk>jrip>J48
MC2	rf>lgi = dec>IBk=nb=bag>J48>bst>jrip

Table 6: Comparison result of log transformation.

Learner	prep	ties	wins	losses
rf	log	7	65	0
bag	log	14	46	12
bst	log	13	38	21
lgi	log	10	37	25
nb	log	11	37	24
dec	log	9	34	29
IBk	log	6	17	49
j48	log	5	9	58
jrip	log	3	2	67

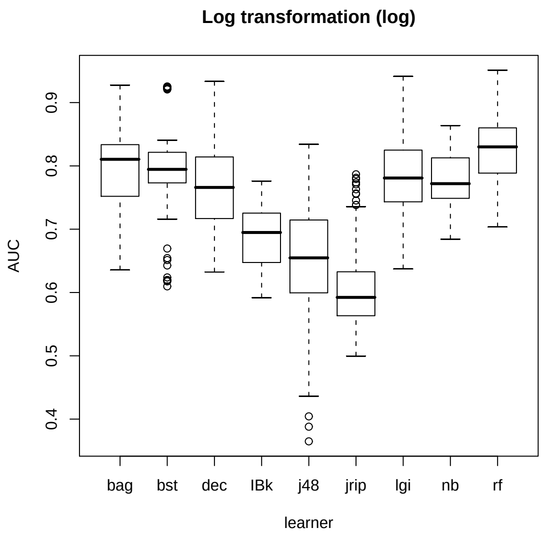

Figure 2: Boxplot diagrams of log transformation (log) data.

learners in 5 cases, and never lose in any case to other learners. Figure 1 shows the boxplot diagrams of these 9 learners over 9 datasets. Table 7 shows the trend of 9 learners in each dataset. The following observations can be drawn from the above:

1. Random Forest has the best performance among 9 learners with 67 wins, 5 ties and none loss.
2. Bagging, logistic and boosting are not bad in the original data.
3. IBk, j48, and Jrip do not have good enough performance comparing to other classifiers in original dataset.

3.2 Log transformation

From Table 6, Table 7, and Figure 2, we can see that the performance of log transformation (log) is similar to that of the original dataset ($none$) in terms of the number of wins, losses and tie, boxplot diagrams and overall trend of individual dataset. Similar observations can be made in log as that of $none$ method.

3.3 Discretized data

Table 8 shows the number of wins, losses, and ties in discretization data in the decreasing order of the number of wins. Table 9 shows the general trend of each learner's performance in individual data. Figure 3 shows the boxplot diagrams of discretization data. From these tables and figure, we have these observations:

Table 7: The trend of log transformation (log)

dataset	Order of log transformation
General	Rf>bag>bst>nb>Dec>IBk>j48=jrip
CM1	Rf>lgi>nb>bag>bst>Dec>IBk>J48=jrip
KC1	Rf>bag>lgi>bst>nb=Dec>IBk>j48=jrip
KC3	Rf>nb>bag=bst>IBk=lgi=Dec>j48>jrip
KC4	Rf=nb=bag=Dec>bst>j48>jrip=lgi>IBk
PC1	Rf>bag>lgi=bst>Dec>nb>IBk=j48>jrip
PC3	Rf>lgi=bag=bst>nb>Dec>IBk>j48>jrip
PC4	Rf>lgi>bag>bst=Dec>nb>j48>IBk=jrip
MW1	Rf=bst>nb=bag>Dec>lgi>IBk>jrip>j48
MC2	Rf=IBk=Dec>nb>lgi=bag>j48=bst>jrip

Table 8: Comparison result of discretized data (nom).

Learner	prep	ties	wins	losses
bst	nom	7	57	8
nb	nom	6	50	16
aode	nom	5	49	18
lgi	nom	10	39	23
bag	nom	15	34	23
rf	nom	13	29	30
IBk	nom	3	23	46
j48	nom	6	7	59
jrip	nom	3	2	67

Table 9: The trend of discretized data(nom).

dataset	Order of discretized data
General	bst>nb>aode>lgi>bag>rf>IBk>j48>jrip
CM1	bst >nb>aode>IBk>lgi>rf>bag>j48=jrip
KC1	nb=bag>bst=aode>lgi>j48=rf>IBk>jrip
KC3	nb>aode>bag=bst>lgi>rf>j48>IBk
KC4	nb>lgi=bst>aode>rf=bag>IBk>j48>jrip
PC1	bst>lgi=aode>IBk>nb=rf=bag>jrip>j48
PC3	bst>lgi=rf=bag>aode>nb>IBk>jrip>j48
PC4	lgi=bst>rf=bag>aode>j48=nb>IBk>jrip
MW1	bst>nb=aode>lgi=rf>IBk=bag>j48=jrip
MC2	nb>aode>IBk>rf>bag=bst>j48=lgi>jrip

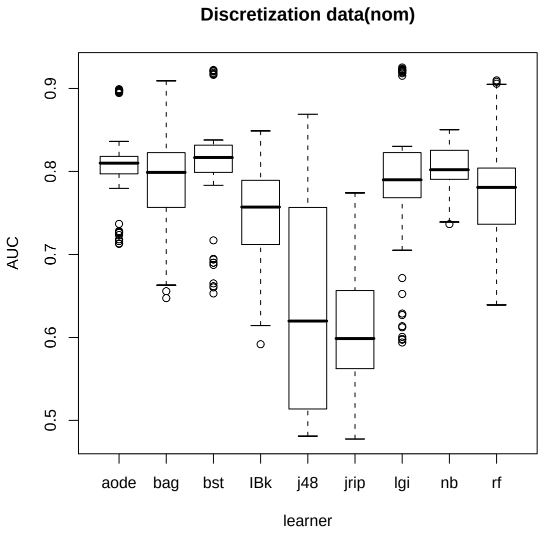

Figure 3: Boxplot in the discretized data(nom).

Table 10: Comparison result of discretization on log data (log&nom).

Learner	prep	ties	wins	losses
bst	log&nom	6	58	8
nb	log&nom	4	52	16
aode	log&nom	5	48	19
lgi	log&nom	11	39	22
bag	log&nom	13	36	23
rf	log&nom	13	31	28
IBk	log&nom	6	17	49
j48	log&nom	5	7	60
jrip	log&nom	3	3	66

1. Boosting is the best learner on discretized data.
2. NaiveBayes, AODE, logistic, and bag are not bad.
3. In a result consistent with prior work [6], we observe here that Naivebayes performs much better on discretized data than that of in $none$ and log methods.
4. The performance of Random Forest is decreased in discretized data.
5. IBk, j48, jrip still remain three worst learners in discretized data.

3.4 Discretized on log data

Table 10 shows the number of wins,losses,and ties in discretization data in the decreasing order of the number of wins. Table 11 shows the general trend of each learner's performance in individual data. Figure 4 shows the boxplot diagrams of discretization data. From these tables and figure, we see that all the observations we have in nom method are hold in this $log\&nom$ method.

3.5 Comparison between Random Forest and Boosting

We perform the Wilcoxon nonparametric statistical test to compare the performance between the Random Forest in the form of

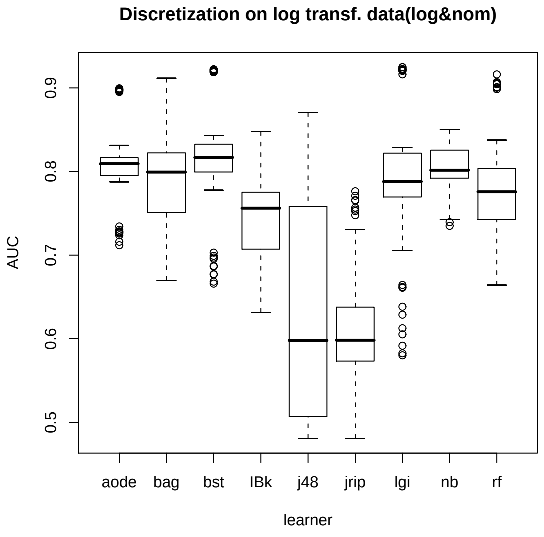

Figure 4: Boxplot in $log\&nom$.

Table 11: The trend of discretized on log transformation ($log\&nom$).

dataset	Order of discretized data
General	bst>nb>aode>lgi>bag>rf>IBk>j48>jrip
CM1	bst>nb>aode>lgi>IBk>rf>IBk>j48>jrip
KC1	nb=bag>bst=aode>lgi>j48=rf>IBk>jrip
KC3	nb>bag=aode>bst>lgi>j48=rf>IBk>jrip
KC4	nb>lgi=bst>aode>rf=bag>IBk>j48=jrip
PC1	bst>lgi=aode>nb>rf>bag>IBk>jrip>j48
PC3	bst>lgi=rf=bag>aode>nb>IBk>jrip>j48
PC4	lgi=bst>rf=bag>aode>j48>nb>IBk>jrip
MW1	bst>nb>aode>lgi=rf>IBk=bag>jrip>j48
MC2	nb>aode>IBk=rf=bag=bst>lgi>j48=jrip

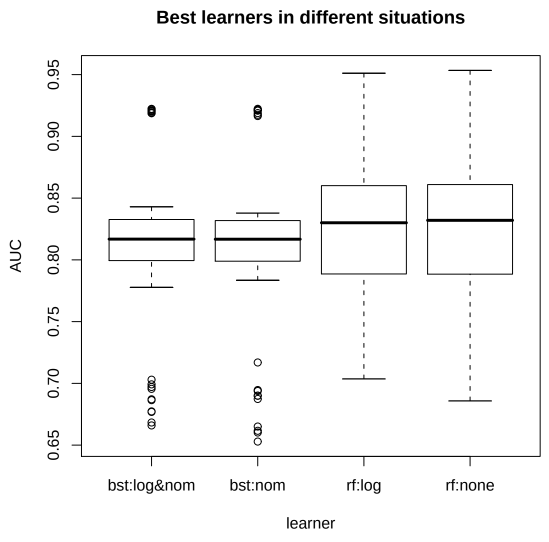

Figure 5: Comparison of best learners in 4 preprocessing methods.

Table 12: Comparison among the best learners in 4 preprocessing methods

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

$none$, log preprocessing data and that of boosting in the form of nom and $log\&nom$ data. The p value of H_0 is $0.00000001510363 (< 0.05)$ which suggests a significance difference between these two learners. The p value of H_1 is $0.000000007.551813 (< 0.05)$ which support that the performance of random forest in $none$ and log is better than that of boosting in nom and $log\&nom$. We performs the Mann-Whitney test which also gives the same result although with different p value. Figure 5 shows the boxplot diagrams of them. Thus, comparing two best performance learners in different preprocessing methods, random forest of $none$ and log beats boosting of nom and $log\&nom$ methods.

3.6 Comparison among different preprocessing methods

We perform the Wilcoxon nonparametric statistical test to compare the performance of 4 different preprocessing methods in the MDP dataset. Table 12 shows the comparison result. Different learners perform differently in different preprocessing methods. We have the following observations:

1. Boosting, NaiveBayes, IBk, and logistic have better performance in discretization data.

2. J48 and jrip has the same performance in 4 different preprocessing methods.

3. Decorate has the same performance in the original and log transformation data.

4. AODE has the same performance in two discretized data.

5. Random forest's performance is better in $none$ and log preprocessing methods than that of discretization data (nom and $log\&nom$)

6. Bagging is the only learner which has better performance in log transformation than that of other preprocessing methods.

4. CONCLUSIONS

This paper investigates the effect of four preprocessing methods on prediction of software fault-proneness quality models based on NASA MDP dataset. We have the following observations.

1. Random forest in the original has the best performance across any other kinds of learners in any kinds of transformation methods. Its performance is decreased in discretized data.

2. Boosting is the best learner among all learners in discretized data.

3. NaiveBayes' performance is greatly improved in discretized data which confirms prior results in [6].

4. IBk, J48, and jrip probably are not right learners to be considered in MDP dataset because they are all ranked in the worst three learners in all 4 preprocessing methods.

5. Log transformation do not affect predictive models performance except on learner bagging.

Therefore, in the different situation, in order to gain better predictive result, we should choose a preprocessing method based on a specific learner instead of using universal transformation for all kinds of learners.

5. REFERENCES

[1] The R Project for Statistical Computing, available `http://www.r-project.org/`.

[2] Metric data program. NASA Independent Verification and Validation facility, Available from `http://MDP.ivv.nasa.gov`.

[3] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[4] W. J. Conover. *Practical Nonparametric Statistics*. John Wiley and Sons, Inc., 1999.

[5] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 2006.

[6] J. Dougherty, R. Kohavi, and M. Sahami. Supervised and unsupervised discretization of continuous features. In *International Conference on Machine Learning*, pages 194–202, 1995.

[7] J. J. Faraway. *Practical Regression and Anova using R*. online, July 2002.

[8] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. pages 1022–1027, 1993.

[9] L. Guo, Y. Ma, B. Cukic, and H. Singh. Robust prediction of fault-proneness by random forests. In *Proc. of the 15th International Symposium on Software Relaibility Engineering ISSRE'04*, pages 417–428, 2004.

[10] I. Guyon, S. Gunn, M. Nikravesh, and L. A. Zadeh. *Feature Extraction: Foundations and Applications*. Springer, 2006.

[11] I. Jolliffe. *Principal Component Analysis*. Springer,New York, 2002.

[12] T. Menzies, J. Greenwald, and A. Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, 33(1):2–13, January 2007. Available from `http://menzies.us/pdf/06learnPredict.pdf`.

[13] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[14] S. Siegel. *Nonparametric Satistics*. New York: McGraw- Hill Book Company, Inc., 1956.

[15] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, Los Altos, US, 2005.