Universität Innsbruck

Department of Computer Science
Research Group Quality Engineering

SEMINAR PAPER

Comparison of Defect Prediction Models

Martin Durcek, Alberte Thegler

Supervisor: Ass.-Prof. Dr. Michael Felderer

Innsbruck, June 15, 2017

Abstract: Many data mining methods have already been introduced into defect predictions. In our work, we tested and compared the results obtained from Naive Bayes and J48 decision tree based defect prediction model. For prediction itself, metrics like .... were used. We also argue about the topic of general defect prediction model and usage of the very same model within several environments.

1 Introduction

Software Fault Prediction(SFP) is one of the most active research areas in software engineering.In spite of diligent planning, documentation, and proper process adherence in software development, occurrences of defects are inevitable. Finding and fixing defects costs companies around the world huge amounts of money and therefore any automated help in reliably predicting where faults are, and focusing the efforts of testers, has a significant impact on the cost of production and maintenance of software. Various regression techniques and recently also machine learning algorithms had been utilized to provide better insight into software repositories and help developers as well as testers to invest their time at work more effectively. Although some voices are critical about the importance and rentability of creating a defect prediction models as a part of software development projects, there is no doubt such model, if implemented correctly, represents additional tool to increase work effectiveness as well as software reliability.

2 Metrics, classifier and dataset selection

In this section we describe the reasons behind choosing specific code metrics, classifiers as well as datasets to work with. Since we don't possess any deep previous knowledge on this particular topic, we mostly refer to other paper and studies - mainly the article *Systematic Review of Machine Learning Techniques for Software Fault Prediction*[2] from Ruchika Malhotra.

2.1 Metrics

Typical software defect prediction model is trained by using previously collected fault data. Metrics characteristics or count influences the performance of the model.[3]. Since exaustive literature review is out of space consideration, to decide ideal number of metrics, we took a look into Wang's paper on this topic[3]. There we found, that models perform very good with already just 3 metrics. Paper also demonstrates that models generally perform better with restricted number of metrics then with many(all) of them. There are several metric groups and metrics itself used to describe software repositories and code itself. Besides traditional McCabe and Halstead static metrics, there are also Object-oriented metrics(cohesion, coupling and inheritance count), miscellaneous

metrics(change and requirement metrics, code churn) or even hybrid metrics combining various metrics from previous groups. In her extensive research, Malhotra found that in more than 50% of 64 considered papers and articles on fault prediction, traditional procedural metrics had been used. Object-oriented metrics had also been used a lot - particularly coupling between objects(CBO) and response for class(RFC) emerged as highly useful. On the other side, number of children(NOC) and depth of inheritance tree(DIT) didn't seem as good metrics for SFP purposes[2, p. 15].

2.2 Classifier

Choosing the right classifier is also not a straightforward process since there are many statistics and machine learning methods to be used and combined. During last ten years, many papers and articles have been published on the topic of classifier performance in SFP field. According to Malhotra in her paper, most widely used classifiers are decision trees and Bayesian learners followed by neural networks and support vector machines[2, p. 11].

2.3 Datasets

Several dataset options are available for SFP studies, but NASA Metrics Data Program data sets are definitely most widely used. In her research, Malhotra claims that they are used in more than 60% of 64 examined studies. KC1 is the most widely used (both module and class level) which is applied in 43% of studies and has 15.4% faulty modules and 40.6% faulty classes. Similarly, PC1 data set is used in 40% of studies and contains only 6.8% of faulty modules.[2, p. 17].

3 Experiment design

While designing our experiment, we realised there are really lot of options. The nature of experiment always stays the performance comparison/analysis, but the circumstances can significantly differ. We can for example compare groups of classifiers (decision trees vs. Bayesian learners) or particular classifiers within one group. Comparison can be also analyse performance on different datasets, number of used metrics or compare particular software metrics. Another option could also be just changing the parameters of one classifier in the stable/unchanging(?) environment. We've decided to carry out rather simple tests of several decision tree and bayesian algorithms to see how they perform on NASA MDP data sets. We carried tests with the whole datasets as well as with data sets restricted just on chosen code metrics. To deal with the lack of data but still avoid overfitting, we're mostly using K-fold cross validation approach. Here is needed to mention, that in WEKA, K-fold validation is executed internally to estimate the generalization error and the output model is the model trained on the whole dataset

(partial K-fold models are not shown to the user). We utilize this to also perform our own tests with the datasets where this is possible (due to their sufficient size).

3.1 Evaluation metrics

To compare models(classifiers) and evaluate their performance, there are several metrics which can be used. Of course, each metric carries different information and should be used in different situations. In our case (SFP) it can't be generalised which metrics are important, because it depends on the particular project and policy utilized within the project (should we rather pay attention to the false positives or the false negatives - in other words, do we rather want our test team to concentrate of defect-less but falsely classified class or is it worse when testing team doesn't spend enough time with the defect-rich class because it has been classified as "safe"). Since we can't favor precision over recall or vice versa, F-measure offers itself as good way to consider both of them. We have also again taken look into Malhotra's article[2] to find out that the most used ones are usually recall, precision, area under the curve(AUC - the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example[1]) and F-measure. WEKA outputs the whole confusion matrix plus several more metrics so we utilized it to compare trained classifiers with several of these.

4 Experiment results

4.1 Decision Trees

To compare the decision trees performance, we used the PC1 NASA dataset because of their popularity for SFP purposes.

Results of models' performance can be seen in following tables. Tables 0.1,0.2, 0.3 and 0.4 document performance of decision trees.

Algorithm	**Correct [%]**	**F-Measure**	**AUC**
Decision Stump	91.347	.872	.551
HoeffDing Tree	91.205	.871	.620
J48	91.347	.872	.491
LMT	91.489	.876	.573
Random Forest	91.489	.885	.601
Random Tree	92.198	.899	.595
REP tree	91.347	.872	.522
Mean	91.488	.878	.565

Table 0.1: Decision trees performance using 1 repository metric

Algorithm	Correct [%]	F-Measure	AUC
Decision Stump	91.347	.872	.714
HoeffDing Tree	91.205	.872	.491
J48	90.780	.874	.605
LMT	91.347	.888	816
Random Forest	87.943	.874	.748
Random Tree	84.539	.851	.568
REP tree	90.922	.875	.701
Mean	89.726	.872	.663

Table 0.2: Decision trees performance using 3 repository metrics

Algorithm	Correct [%]	F-Measure	AUC
Decision Stump	91.347	.872	.714
HoeffDing Tree	91.347	.872	.491
J48	90.354	.872	.616
LMT	91.347	.890	840
Random Forest	91.063	.898	.845
Random Tree	88.227	.882	.624
REP tree	91.347	.880	.636
Mean	90.719	.880	.681

Table 0.3: Decision trees performance using 5 repository metrics

Algorithm	Correct [%]	F-Measure	AUC
Decision Stump	91.347	.872	.729
HoeffDing Tree	91.347	.872	.491
J48	90.070	.880	.679
LMT	91.489	.889	827
Random Forest	91.773	.903	.851
Random Tree	89.503	.895	.668
REP tree	90.638	.873	.704
Mean	90.881	.883	.707

Table 0.4: Decision trees performance using 7 repository metrics

4.2 Bayesian Learners

To compare the bayesian performance, we used the PC1 NASA dataset because of their popularity for SFP purposes.

Results of models' performance can be seen in following tables. Tables 0.5,0.6, 0.7 and 0.8 document performance of bayesian learners.

Algorithm	Correct [%]	F-Measure	AUC
Bayes Net	91.347	.872	.491
Naive Bayes	88.794	.863	.524
Naive Bayes Multinomial	91.347	.872	.491
Naive Bayes Multinomial Text	91.347	.872	.491
Naive Bayes Multinomial Updatable	91.347	.872	.375
Naive Bayes Updatable	88.794	.863	.524
Mean	90.496	.869	.482

Table 0.5: Bayesian Learners performance using 1 repository metric

Algorithm	Correct [%]	F-Measure	AUC
Bayes Net	91.347	.872	.688
Naive Bayes	88.510	.878	.592
Naive Bayes Multinomial	88.227	.882	.749
Naive Bayes Multinomial Text	91.347	.872	.491
Naive Bayes Multinomial Updatable	88.227	.882	.748
Naive Bayes Updatable	88.510	.878	.592
Mean	89.361	.877	.643

Table 0.6: Bayesian Learners performance using 3 repository metrics

Algorithm	Correct [%]	F-Measure	AUC
Bayes Net	89.645	.877	.754
Naive Bayes	88.085	.878	.624
Naive Bayes Multinomial	87.375	.883	.779
Naive Bayes Multinomial Text	91.347	.872	.491
Naive Bayes Multinomial Updatable	87.375	.883	.779
Naive Bayes Updatable	88.085	.878	.624
Mean	88.652	.878	.675

Table 0.7: Bayesian Learners performance using 5 repository metrics

Algorithm	Correct [%]	F-Measure	AUC
Bayes Net	89.787	.876	.732
Naive Bayes	87.375	.872	.656
Naive Bayes Multinomial	85.815	.865	.638
Naive Bayes Multinomial Text	91.347	.872	.491
Naive Bayes Multinomial Updatable	87.375	.872	.643
Naive Bayes Updatable	87.375	.872	.656
Mean	88.179	.871	.636

Table 0.8: Bayesian Learners performance using 7 repository metrics

5 Hypothesis testing

In previous experiment, we obtained specific values for each and every algorithm in 2 groups. To compare those groups generally, we will work with mean values of each group. We computed means from all values as well as while leaving out maximal and minimal values of each group (to prevent outliers from affecting our results). In both scenarios, values have been almost the same. Because of the limited scope of our report, we're offering the hypothesis testing done for means computed from all values.

[tables of means next to each other]

At first sight, there are hints that decision trees perform better than the Bayesian learners. To confirm this, we executed paired t-tests for each metric.

For F-measure, we executed two-tailed t-test since mean values do not clearly display performance supremacy of any group. On the other side, correctness and ROC area have always showed slightly better performance of decision trees over bayesian learners so in their case we can perform one-tailed t-test. We measured performance for 4 different scenarios what results into 6 degrees of freedom (n1+n2-2). With confidence level set to standard default value 95% the critical t-value found in statistical table for t-distribution is 2.447 and 2.015 (for 2-tailed and 1-tailed t-test respectively).

Metric	**T-test tails**	**Degrees of freedom**	**Confidence level [%]**	**t-value**	**t-value threshold**	**p-value**
Correctness	1	6	95	2.8945	2.015	0.06278
F-measure	2	6	95	1.1852	2.447	0.3213
AUC area	1	6	95	2.3921	2.015	0.09656

Table 0.9: Hypothetis testing results

As can be seen in table 0.9, according to t and p-values we can reject null hypothesis in favor of alternative hypothesis in cases of both one-tailed tests (correctness, AUC area). This can be in other words interpreted that when speaking in terms of correctness and AUC area, decision trees really performed better than bayesian learners. On the other side, experiment with F-measure as evaluation metric didn't show performance advantage in any of 2 groups.

Another area we concentrated on in our experiment was the effect of number of software metrics used to train the prediction models. Results can be seen in tables...

Bibliography

[1] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[2] Ruchika Malhotra. A systematic review of machine learning techniques for software fault prediction. *Applied Soft Computing*, 27:504–518, 2015.

[3] Huanjing Wang, Taghi M Khoshgoftaar, and Naeem Seliya. How many software metrics should be selected for defect prediction? In *Twenty-Fourth International FLAIRS Conference*, 2011.