Software Analytics - Data Analyses on Defect Data and Experiments with Defect Prediction Methods

Sarat Kiran Andhavarapu, *A01960723*

***Abstract*—In spite of diligent planning, good documentation and proper process control during software development, appearance of defects are inevitable. These software defects may lead to degradation of the quality and even sometimes extremely serious problems. Defect Prediction Models aim at identifying error-prone modules of a software system to guide quality assurance activities such as tests or code reviews. But checking every metrics includes a lot of cost and man power, so in this paper, I study whether the precision of existing models for defect prediction can be improved using our top correlated metrics and/or by combining the top metrics.**

In this paper, I compared three different regression models including Linear regression model, Tree regression model, Negative binomial generalized linear model to include treatment effort into the prediction process, and evaluate the predictive power of such models.

***Keywords*—Defect data analysis, Software metrics, Evaluation, Machine Learning Algorithms.**

I. Introduction

Due to increasing extent and complexity of software systems, it is common to have large teams, communities of developers working on the same project at the same time this may cause bugs. One example that burned up a 327.6 million project in minutes is in 1998, when the Mars Climate Orbiter built by NASA's Jet Propulsion Laboratory approached the Red Planet at the wrong angle. The biggest problem was that different parts of the engineering team were using different units of measurement. One group working on the thrusters measured in English units of pounds-force seconds; the others used metric Newton-seconds and this is not been checked, Even these small bugs can collapse the whole system.

Quality assurance activities, such as tests or code reviews, are an expensive, but vital part of the software development process as we can see from the above mentioned example. Any support that makes this phase more effective may thus improve software quality or reduce development costs. This lead to the development of defect prediction models, aiming at identifying error-prone parts of a system so that quality assurance activities can be focused. This task is often seen as a classification problem with the goal to label files as bug free or not. Since the prediction is most often based on data for files or modules, the evaluation is then performed at the file or module level as well. [3]

If we know the areas of future software defects will help developers to optimize the resources available for the maintenance of a certain project by focusing on the most prevailed erroneous component rather than going through the whole project to find the bugs. To achieve this we can make use of defect prediction algorithms. But not all the defect predicting algorithms preform well at all times. Each defect predicting algorithms should be analyzed to verify that it works best for the given dataset.

In this experiment we aim to answer three questions:.

Question 1: Identifying the top metrics that have the highest correlation to the post- release defects ("bugs").

Question 2: Finding the prediction performance on three regression including linear regression model, Tree regression model, Negative binomial generalized linear model, using a particular cross- validation method.

Question 3: what are the highest prediction performance when combining the metrics.

So we will look at each question, discuss the method we tried to solve the problem and finally results that we obtained.

II. Datasets

A. What I understand from the datasets ?

This project contains five datasets namely Eclipse, Equinox, lucene, mylyn, pde. In turn these 5 datasets contain multiple "csv" files that contains object-oriented Java software systems using classes as the targets. In the csv files listed in each dataset we are interested in three specific csv files.

1) single-version-ck-oo.csv - contains code metrics for all source files.
2) change-metrics.csv - contains metrics for change histories of all source files.
3) bug-metrics.csv - contains metrics for bug histories of all source files.

Each file also contains the number of post-release defects in column "bugs". [4]

III. Analysis

A. Question 1: Correlation Analysis

We use Spearman's ranked correlation coefficient to get the highest correlation to the number of post-release defects. Spearman's ranked correlation coefficient is a nonparametric measure of statistical dependence between two variables. It

TABLE I. TOP METRICS IN ALL DATASETS.

	Eclipse	**Equinox**	**lucene**	**mylyn**	**pde**
Bug Metrics	numberOfBugsFoundUntil.	numberOfBugs FoundUntil.	numberOfBugs FoundUntil.	numberOfMajor BugsFoundUntil.	numberOfNonTrivial BugsFoundUntil.
	numberOfNonTrivial BugsFoundUntil.	numberOfNonTrivial BugsFoundUntil.	numberOfNonTrivial BugsFoundUntil.	numberOfNonTrivial BugsFoundUntil.	numberOfBugs FoundUntil.
	linesAddedUntil.	numberOfVersionsUntil.	maxCodeChurnUntil.	linesAddedUntil.	linesAddedUntil.
Change Metrics	numberOfVersionsUntil.	linesAddedUntil.	linesAddedUntil.	maxLinesAddedUntil.	numberOfVersionsUntil.
	linesRemovedUntil.	linesRemovedUntil.	numberOfVersionsUntil.	avgLinesAddedUntil.	linesRemovedUntil.
	maxLinesAddedUntil.	maxLinesRemovedUntil.	linesRemovedUntil.	codeChurnUntil.	maxLinesAddedUntil.
	maxCodeChurnUntil.	maxLinesAddedUntil.	maxLinesAddedUntil.	maxCodeChurnUntil.	maxLinesRemovedUntil.
	wmc	cbo	numberOfAttributes	numberOfPrivateMethods	rfc
Single version	numberOfLinesOfCode	lcom	cbo	fanOut	numberOfLinesOfCode
Metrics	fanOut	wmc	lcom	cbo	fanOut
	rfc	numberOfMethods	numberOfMethods	numberOfLinesOfCode	wmc
	cbo	numberOfLinesOfCode	fanIn	rfc	numberOfPrivateMethods

TABLE II. TOP METRICS IN EACH CSV FILE FOR ECLIPSE DATASET

No.	**Bug Metrics**	**Change Metrics**	**Single version**
1.	numberOfBugs FoundUntil.	linesAddedUntil.	wmc
2.	numberOfNonTrivial BugsFoundUntil.	numberOf VersionsUntil.	numberOfLines OfCode
3.		linesRemovedUntil.	fanOut
4.		maxLinesAddedUntil.	rfc
5.		maxCodeChurnUntil.	cbo

assesses how well the relationship between two variables can be described using a monotonic function. The coefficient lies in between +1 to -1. If there are no repeated data values, a perfect Spearman correlation of +1 or 1 occurs when each of the variables is a perfect monotone function of the other.

In this research question, we want to find the top 5 metrics in single-version-ck-oo.csv, 5 in change-metrics.csv, and 2 in bug-metrics.csv.

we can solve this problem by taking each csv file (excluding the post release bugs such as ”majorBugs”,”criticalBugs”,”highPriorityBugs”,”nonTrivialBugs”) and finding the correlation with between all the metrics. we now only take “bugs” metrics (we do this because the “bugs“ metrics have the correlation with all other metrics in the csv file.

we now select the top metrics that have the highest correlated values. After getting th top metrics we find correlation between them.

B. Question 2: Prediction Performance

Here we use different regression models to get the defect prediction.

1) Linear regression model - linear regression is an approach to modeling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X.In linear regression, data are modeled using linear predictor functions, and unknown model parameters are estimated from the data. Such models are called linear models. Most commonly, linear regression refers to a model in which the conditional mean of y given the value of X is an affine function of X. Less commonly, linear regression could refer to a model in which the median, or some other quantile of the conditional distribution of y given X is expressed as a linear function of X. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of y given X, rather than on the joint probability distribution of y and X, which is the domain of multivariate analysis.
2) Tree regression model - Recursive partitioning is a fundamental tool in data mining. It helps us explore the structure of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. Classification and regression trees can be generated through the rpart package.
3) Negative binomial generalized linear model - In a generalized linear model (GLM), each outcome of the dependent variables, Y, is assumed to be generated from a particular distribution in the exponential family, a large range of probability distributions that includes the normal, binomial, Poisson and gamma distributions, among others. Generalized linear models are formulated as a way of unifying various other statistical models, including linear regression, logistic regression and Poisson regression. GLM proposed an iteratively reweighted least squares method for maximum likelihood estimation of the model parameters. Maximum-likelihood estimation remains popular and is the default method on many statistical computing packages.

We use the above mentioned regression models taking the top metrics we got from question 1 as the independent variables and the post release defects as the dependent variable. we execute each regression model based on all the three csv files in all the datasets that have the top metrics but the dependent variable is always the ”bugs“ that is the post release bugs.

TABLE III. PREDICTION PERFORMANCE WHILE COMBINING THE METRICS.

Top Metrics Combination	**Model**	**Eclipse**		**Equinox**		**lucene**		**mylyn**		**pde**	
Top two from each CSV file		mae	rcor	mae	rcor	mae	rcor	mae	rcor	mae	rcor
	lm	0.39	0.44	0.71	0.59	0.11	0.31	0.27	0.12	0.34	0.27
NumberOfBugsFoundUntil.+	rpart	0.42	0.48	0.64	0.58	0.18	0.29	0.29	0.22	0.33	0.29
numberOfNonTrivialBugsFoundUntil.	glm.nb	1.83	0.43	1.35	0.58	2.59	2.04	2.04	0.13	2.09	0.29
LinesAddedUntil.+	lm	0.44	0.39	0.63	0.62	0.17	0.28	0.28	0.12	0.38	0.19
NumberOfVersionsUntil./	rpart	0.45	0.46	0.66	0.57	0.18	0.28	0.28	0.24	0.35	0.23
maxCodeChumUntil.	glm.nb	1.83	0.38	1.41	0.63	2.56	2.04	2.04	0.13	2.03	0.26
NumberOfLinesOfCode + wmc/rfc/	lm	0.46	0.39	0.64	0.57	0.23	0.25	0.25	0.27	0.35	0.28
Cbo + lcom/numberOfAttributes + cbo/	rpart	0.66	0.44	0.69	0.52	0.23	0.27	0.29	0.29	0.35	0.24
NumberOfPrivateMethods + fanOut	glm.nb	1.84	0.4	1.47	0.56	2.36	2.15	0.27	0.27	1.88	0.28
	lm	0.39	0.45	0.59	0.65	0.2	0.26	0.28	0.28	0.36	0.24
A1+B1+C1	rpart	0.43	0.47	0.65	0.57	0.17	0.27	0.3	0.3	0.34	0.29
	glm.nb	1.85	0.44	1.48	0.62	2.62	2.11	0.28	0.28	2.12	0.32
	lm	0.39	0.43	0.62	0.61	0.19	0.25	0.26	0.26	0.34	0.24
A1+A2+B1+B2+C1+C2	rpart	0.43	0.46	0.65	0.57	0.19	0.27	0.28	0.28	0.34	0.29
	glm.nb	1.87	0.41	1.48	0.61	2.64	2.15	0.27	0.27	2.13	0.31

1) Using regression models: To check the working of the regression models we do cross validation. we use 80 % of the dataset (tuples or classes) to build the prediction model, and the remaining 20 % for predicting in the dataset. While selecting the random data, we set the random seed as the last 4 digits of my A number.

2) Evaluating prediction power: To evaluate the prediction power of the regression models, we use the cross validation, selecting random data, we use 80 % training set to build the three regression models and we use the outputs from the training set on prediction set (20 %). we compute mean absolute error and Spearman's correlation between the predicted number of post-release defects and the actual number. We repeat the above thing thirty times and save the mean absolute error and Spearman ranked correlation to find the average of the calculated mean absolute error and Spearman ranked correlation.

C. Question 3: Prediction Performance while combining the top metrics

We use the regression models taking the post release defects as the dependent variable but we take the combination of the top metrics from the question 1 to check if the prediction performance increases. In order to check this we take top two metrics from each csv file and run through all the three regression models, and also we make a data frame that has the top two metrics in the three csv files in each dataset. With the above formed dataset we take two combinations of the independent variables as a).Top one of all csv files in a dataset, b). Top two of all csv files in a dataset.

Finally we end up with the mean absolute error and Spearman ranked correlation for different combinations of independent variables.

IV. DISCUSSION

A. Question 1: Correlation Analysis

The top metrics that are calculated from the highest correlation to the number of post-release defects ("bugs") are listed in the table 1. This is repeated for all the datasets to find the metrics that has the highest corelation to the post-release defects.Observations that can be made by looking at the correlation between the top metrics of all datasets

1) "numberOfBugsFoundUntil." and "numberOfNonTrivialBugsFoundUntil." are always the top metrics for bug-metrics.csv for all datasets - correlation between them is always more than 0.98 in spearmans ranked correlation.
2) In change metrics csv file "linesAddedUntil." always appears in top two places of highest correlation to the number of post-release defects. Few metrics always apear in top five highest correlated in all the datasets. They all have good correlation between them (i.e. more than 0.6).
3) maxLinesRemovedUntil./maxLinesAddedUntil. is highly correlated with the LinesRemovedUntil./LinesRemovedUntil. - this makes sense because If the lines are added/removed the maxLinesAdded/removed also changes
4) The highest correlation between the top metrics is found between "lcom" and "numberOfMethods" with $R_{spearman} = 0.9975$.
5) When we make 12x12 matrix for the (top 5 from single version + top 5 from change metrics +top 2 from

TABLE IV. PREDICTION PERFORMANCE

	Model	Eclipse		Equinox		lucene		mylyn		pde	
		mae	rcor	mae	rcor	mae	rcor	mae	rcor	mae	rcor
	lm	0.39	0.46	0.71	0.60	0.20	0.31	0.28	0.21	0.35	0.30
numberOfBugsFoundUntil.	rpart	0.43	0.48	0.64	0.59	0.19	0.28	0.30	0.18	0.34	0.30
	glm.nb	1.83	0.46	1.35	0.60	2.59	0.31	2.01	0.21	2.09	0.30
	lm	0.49	0.40	0.74	0.60	0.18	0.24	0.29	0.14	0.33	0.26
linesAddedUntil.	rpart	0.44	0.46	0.74	0.56	0.20	0.31	0.29	0.25	0.36	0.24
	glm.nb	1.71	0.40	1.36	0.60	2.53	0.24	2.04	0.14	2.03	0.26
	lm	0.47	0.40	0.72	0.55	NC	NC	0.30	0.22	0.35	0.28
numberOfLinesOfCode	rpart	0.44	0.45	0.74	0.49	NC	NC	0.28	0.27	0.36	0.25
	glm.nb	1.80	0.40	1.36	0.55	NC	NC	2.09	0.22	1.88	0.28

bug metrics) we get strong correlation between them suggesting that combining these will have good effect.

B. Question : Prediction Performance

Popularity of software components do correlate with software defects. This can be observed in the below table. This table gives the values of mean absolute error and Spearmans ranked correlation values for different regression values including Linear regression model, Tree regression model, negative binomial generalized linear model for all the datasets [1] - Eclipse , Equinox, lucene, mylyn, pde.

Looking at the Table 2 we can see the prediction performances when combining the top metrics for independant variables.

We have combined A1 + B1 + C1 (top metrics from each csv file in dataset) and A1 + A2 + B1 + B2 + C1 +C2 (top 2 metrics from the csv files) I see that when combine the top metrics the prediction performance improves and this can extended if we do more combinations of the independant variables.For eclipse Linear regression model suits and for mylyn tree regression suits.The three regression models vary a lot, for mean absolute error linear regression model performs better(lower values) and tree regression also does a decent job. For spearman ranked correlation all the three regression models perform in a similar way. One thing we can get from the experiment is that each regression model varies according to dataset, not all regression models can be used for same dataset. And the independant variable selection also plays a important role.[2]

V. CONCLUSION

In this project, we computed the top metrics that has the highest correlation with the post-release defects that is 'bugs' and applied different regression models including Linear Regression model, Tree regression model, negative binomial generalized linear model to predict the post- release bugs using cross- validation method. I have observed that defect prediction performance varies significantly if we change the independent variables in the function. Even though taking the top metrics from the corresponding datasets improved the prediction performance, I observed that while combining the independent variables from other top dataste files improved the prediction performance in most of the cases. One other intesting thing to observe is regression models play a significant role while performing the defect prediction, this is observed while performing these experiments on the same datasets. Not all regression models satisfy all datasets. Few regression models are good for few datasets and bad for few.

ACKNOWLEDGMENT

I would like to thank Dr. Tung Nguyen for his excellent guidance for this project

REFERENCES

[1] Alberto Bacchelli, Marco D'Ambros, and Michele Lanza. Are popular classes more defect prone? In *Proceedings of the 13th International Conference on Fundamental Approaches to Software Engineering*, FASE'10, pages 59–73, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-12028-8, 978-3-642-12028-2. doi: 10.1007/978-3-642-12029-9_5. URL http://dx.doi.org/10.1007/978-3-642-12029-9_5.

[2] 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*, pages 31–41, May 2010. doi: 10.1109/MSR.2010.5463279.

[3] Patrick Knab, Martin Pinzger, and Abraham Bernstein. Predicting defect densities in source code files with decision tree learners. In *Proceedings of the 2006 International Workshop on Mining Software Repositories*, MSR '06, pages 119–125, New York, NY, USA, 2006. ACM. ISBN 1-59593-397-2. doi: 10.1145/1137983.1138012. URL http://doi.acm.org/10.1145/1137983.1138012.

[4] Thomas Zimmermann, Rahul Premraj, and Andreas Zeller. Predicting defects for eclipse. In *Proceedings of the Third International Workshop on Predictor Models in Software Engineering*, PROMISE '07, pages 9–, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2954-2. doi: 10.1109/PROMISE.2007.10. URL http://dx.doi.org/10.1109/PROMISE.2007.10.