EAI Endorsed Transactions
on Energy Web and Information Technologies
Research Article **EAI.EU**

Statistical Testing on Prediction of Software Defects

Satya Srinivas Maddipati [1,*] and Malladi Srinivas [2]

[1]Research Scholar, KLEF, Vaddeswaram, Guntur
[2]Professor K LEF, Vaddeswaram, Guntur

Abstract

Statistical Tests are used to make inferences from data. These tests will tell whether the observed pattern is real or just due to chance. The type of the test, to be used, depends on research design, distribution of data and type of variables. In this paper, we are addressing high dimensionality problem in software defect prediction using statistical tests. We determined the distribution of data to choose appropriate statistical test. We observed most of the variables follow gamma distribution and hence applied wilcoxon Rank Sum Test for correlation between input variables and outcome variable. We extracted the variable with high correlation. We observed the performance of the classifier was improved by addressing dimensionality problem with wilcoxon Rank Sum Test.

Keywords: Statistical testing, Software defects, Wilcoxon Rank Sum Test.

Received on 07 July 2018, accepted on 30 August 2018, published on 12 September 2018

3rd International Conference on Green, Intelligent Computing and Communication Systems - ICGICCS 2018, 18.5 - 19.5.2018, Hindusthan College of Engineering and Technology, India

doi: 10.4108/eai.12-9-2018.155748

1. Introduction

A statistical test is a mechanism to make quantitative decision about process. The objective of statistical test is to determine whether there is an evidence to reject the hypothesis about process.
Types of Statistical Tests:
There is a wide range of statistical tests. The type of the test, to be used, depends on research design, distribution of data and type of variables. If the data is normally distributed, we will choose parametric tests. If the data is non-normal, we will choose non parametric test.
Pearson Correlation test is used to test the strength of association between two continuous variables. It is determined by Covariance of two variables divided by the product of their standard deviations. It is a measure of linear correlation between two variables. It has a value between -1 and +1.

$$cov(x,y) = E[(X - \mu_x)(Y - \mu_v)]$$

Spearman correlation test is used to test the strength of association between two ordinal variables.
Chi square test is used to test the strength of association between two categorical variables.
T-Tests are used to test the mean of the data (One sample T-Test) or to compare two sets of data (Paired T-test). ANOVA test the difference between group means.
Simple regression predicts the change of predictor variable with outcome variable. Multiple regression predicts the change of two or more predictor variable with outcome variable.

2. Related Work

Tim Menzies et al inspected static code attributes for learning defective modules. He explored the merits of McCube versus Halstead versus lines of code counts for

[*] Corresponding author. Email:maddipativas@gmail.com

identifying defective prone modules. He observed 71 percent probability of detection and mean false alarm rate of 25 percent[3]. Aditi Thakur et al applied hybrid Neuro Fuzzy Approach for bug prediction using software metrics. He applied Linear Discriminant Analysis for that machine learning algorithms yields more efficieny and better performance[5].

Shivkumar Shivaji et al proposed Improving of change based bug prediction using reduced features. He Investigated various feature selection techniques to classification algorithms for bug prediction. He reduced less than 10% of features to improve the classification performance. He applied reduced dataset to Naïve bayes and support vector machine classifiers and observed 21 % improvement in F-measure[1]. Mrinal Singh Rawat et al surveyed Machine learning algorithms for Software Defect Prediction. They applied Bayesian belief networks, Genetic algorithms & Neural networks for identification of defective prone modules[2]. Vipul Vashisht et al designed a frame work for Defect prediction using Adaptive Neuro Fuzzy Inference System. He found the accuracy of validation for 10 projects during requirement analysis and construction as 93.4%[6]. Juan Murillo Morera designed a frame work for software effort prediction using genetic algorithm. The performance of learning schemes was measured using the metrics Spearman’s rank correlation, mean of magnitude relative error, median of magnitude of relative error, standardized accuracy, number of predictions within percentage of actual ones[7]. J S Pahariya et al. proposed Genetic Programming based feature selection for software cost dimensionality reduction and reduced the dataset. Then they applied hybrid neuro fuzzy approach for reduced dataset[4]. K Punitha et al compared Genetic algorithm with ant colony optimization with Hybrid Neuro Fuzzy Inference system and Naïve Bayes classiers. He proved estimation. They compared the efficiency of various classification algorithms, Multiple linear regression, polunomial regression, Support Vector Regression, Radial Basis Function neural network, Dynamic Evolving Neuro Fuzzy Inference System, has been tested on ISBSG 10 datasets[8]. Ebubeogu Amarachukwu et. al conducted an experiment to determine the correlation of each predictor variable with the number of defects[9]. Wenjie Liu attempted automatic feature selection method to improve the performance of severity prediction of bug reports. They introduced ranking based strategy to improve existing feature selection algorithms[10]. Kehan Gao et al. presented a novel form of ensemble learning based on boosting that incorporates data sampling to alleviate class imbalance and feature selection to address high dimensionality problems[11].

3. Methodology

In this paper, we are proposing Wilcoxon Rank Test to determine the correlation between predictor variable & outcome variable. Relavant features are extracted by outcome of the test. A Classifier was constructed using the extracted features. Figure 1 shows the methodology used in this paper.

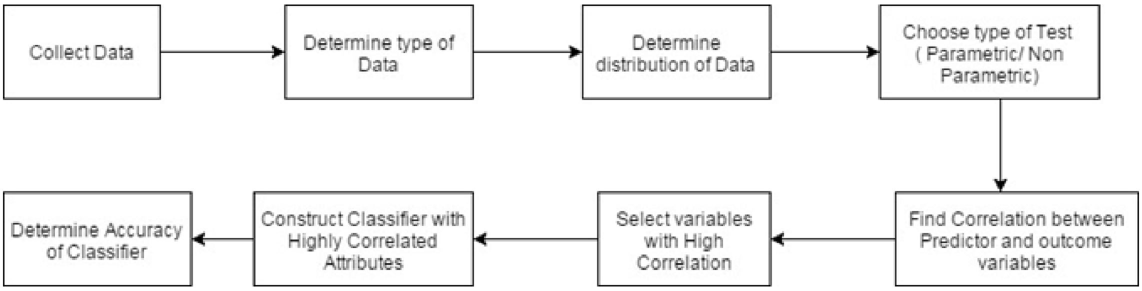
Collect Data
Determine type of Data
Determine distribution of Data
Choose type of Test (Parametric/ Non Parametric)
Find Correlation between Predictor and outcome variables
Select variables with High Correlation
Construct Classifier with Highly Correlated Attributes
Determine Accuracy of Classifier

Fig 1: Methodology for Software Defect Prediction

3.1 Data Collection

The Data for Software Defect Prediction was downloaded from NASA Dataset repository. The datasets were donated by Softlab. Softlab is the Software Research Laboratory in Bogazici University, Istanbul, Turkey. Function level static code attributes are collected using the Pretest metrics Extraction and Analysis tool. In this data, there are two types of datasets, one will keep defect information in discrete manner where as the other will keep the bug count associated with defectiveness.

Determine type of Data: The attributes in a dataset can be categorized into categorical and numeric. The categorical attributes will have limited number of values where as numeric attributes will have continuous values. Categorical attributes are categorized into ordinal and nominal. Ordinal attributes will have order among possible values of attributes where as nominal attributes will have no order among possible values of attributes.

A probability distribution is a mathematical function that provides the probabilities of occurrence of different possible outcomes in an experiment

3.2 Types of distributions

Normal Distribution

It is a most famous statistical distribution, used when the probability distribution at extreme values is low.It generally follows bell shaped curve.

Poisson distribution

A distribution is said to be Poisson, when the success of an event should not influence the outcome of another successful event with the probability of success over a short interval must equal the probability of success over a long interval.

Binomial distribution

The binominal distribution with parameters ‘n’ and ‘p’ is the discrete probability distribution of number of success in a sequence of ‘n’ independent trails.

Gamma distribution

Gamma distribution is a two parameter family of continuous probability distributions with shape parameter(k) and scale parameter (Ө)

There are number of probability distributions. The most generally used probability distributions are

3.3 Choose type of test

To fit relationship between predictor variable and outcome variable, there are various types of tests. These tests are categorized into parametric and non parametric tests.

3.3.1 Parametric test: Parametric tests are applied when the population follows probability distribution based on fixed set of parameters. Parametric tests provide more accurate results.

3.3.2 Non parametric test: Non parametric tests are applied when the assumptions about probability distributions are non known. Non parametric tests provide more robustness.

From Table 1, most of the variables in Software Defect Prediction follows gamma distribution. Figure 3 in appendix, demonstrates the probability distribution of variables.

Table 1: Data Distribution of Software Defect Prediction

Variable	**Unique Values**	**Min**	**Max**	**Mean**	**Type of Distribution**	**Skewness**	**Kurtosis**
Total-loc	32	5	477	75.8	Gamma	2.29	5.76
Blank_loc	24	0	104	18.17	Gamma	2.11	4.89
Comment_loc	18	0	89	8.86	Gamma	2.91	8.86
Code & Comment loc	6	0	8	1.36	Gamma	1.08	0.72
Executable loc	30	5	284	48.86	Gamma	2	4.35
Unique Operands	29	5	150	35.03	Gamma	1.85	3.86
Unique Opeartors	17	3	29	12.58	Gamma	0.59	-0.87
Total operands	35	5	482	93.33	Gamma	1.8	3.3
Total operators	35	9	127.47	699	Gamma	1.86	3.59
Halstead Vocabulory	31	8	47.61	179	Gamma	1.65	3
Halstead length	34	14	220.8	1181	Gamma	1.84	3.48
Halstead volume	36	29	939.56	6126	Gamma	2.2	5.32
Halstead level	22	0.01	0.67	0.14	Gamma	1.45	1.13
Halstead Difficulty	22	1.49	100	18.46	Gamma	2.19	5.49
Halstead Effort	36	43.28	306300	33011.19	Gamma	2.39	5.13
Halstead Error	26	0.01	2.04	0.31	Gamma	2.2	5.31
Halstead Time	36	2.4	17016.67	1833.95	Gamma	2.39	5.13
Branch Count	22	0	236	34.06	Gamma	1.94	4.03
Decision Count	22	0	118	17.03	Gamma	1.94	4.03
Call Pairs	13	0	20	4.08	Gamma	1.07	0.02
Condition count	21	0	116	16.31	Gamma	1.95	4.01
Multiple Condition Count	14	0	25	4.03	Gamma	1.45	1.54

Cyclometric Complexity	20	1	93	13.78	Gamma	2.11	4.84
Cyclometric Density	27	0.04	0.73	0.25	Gamma	1.11	2.14
Decision Density	13	0	2.5	0.85	Gamma	-0.16	-0.49
Decision Complexity	13	0	20	4.08	Gamma	1.07	0.02
Design Density	24	0	10	1.11	Normal distribution	2.57	5.86
Normalized Cyclometric Complexity	20	0.02	0.57	0.22	Normal distribution	1.69	4.44
Formal Parametrs	3	0	2	0.22	Gamma	1.81	1.92
Defects	5	0	4	0.47	Gamma	1.48	0.81

3.4 Find the Correlation between Predictor and Outcome variable

Wilcoxon Rank Sum Test: The two-sample non-parametric Wilcoxon rank sum test (equivalent to the Mann-Whitney test) is performed on the two specified samples. The null hypothesis is that the distributions are the same (i.e., there is no shift in the location of the two distributions) with an alternative hypothesis that they differ on location (based on median). This test does not assume that the two samples are normally distributed but does assume they have distributions of the same shape. If the p-value is less than 0.05 then we reject the null hypothesis and accept the alternative hypothesis, that the two samples have different medians, at the 95% level of confidence.

As per our finding described in Table 2 variables with Null Hypothesis accepted are considered to be relevant for mining task. In our software defect prediction dataset, Total Loc, Executable Loc, Unique Operands, Total operands, Total operators, Halstead length, Halstead volume, Halsted Effort, Halsted error, Halstead time, Call pairs, Cyclometric complexity, Cyclometric density, Decision Density, Decision Complexity, Design Density, Normalized Cyclometric Complexity and formal parameters are considered to be relevant.

Table 2: Wilcoxon Rank Test for Software Defect Prediction

Variable	**W**	**P value**	**Accept**	**Variable**	**W**	**P value**	**Accept**
Total-loc	13	0.05	Null Hypothessis	Decision Count	12	0.04	Alternative Hypothesis
Blank_loc	9.5	0.03	Alternative Hypothesis	Call Pairs	50.5	0.58	Null Hypothessis
Comment_loc	11.5	0.03	Alternative Hypothesis	Condition count	11.5	0.04	Alternative Hypothesis
Code & Comment loc	13.5	0.02	Alternative Hypothesis	Multiple Condition Count	11.5	0.03	Alternative Hypothesis
Executable loc	18	0.11	Null Hypothessis	Cyclometric Complexity	14	0.062	Null Hypothessis
Unique Operands	14	0.06	Null Hypothessis	Cyclometric Density	16.5	0.062	Null Hypothessis
Unique Opeartors	6.5	0.01	Alternative Hypothesis	Decision Density	35.5	0.67	Null Hypothessis
Total operands	15	0.07	Null Hypothessis	Decision Complexity	50.5	0.58	Null Hypothessis
Total operators	15	0.07	Null Hypothessis	Design Density	65	0.13	Null Hypothessis
Halstead	11	0.04	Alternative	Normalized	27.5	0.34	Null

Vocabulory			Hypothesis	Cyclometric Complexity			Hypothessis
Halstead length	15	0.07	Null Hypothessis	Formal Parametrs	51	0.409	Null Hypothessis
Halstead volume	13	0.054	Null Hypothessis	Halstead Error	13.5	0.06	Null Hypothessis
Halstead level	73	0.04	Alternative Hypothesis	Halstead Time	13	0.054	Null Hypothessis
Halstead Difficulty	11	0.04	Alternative Hypothesis	Branch Count	12	0.04	Alternative Hypothesis
Halstead Effort	13	0.05	Null Hypothessis				

3.5 Linear and Generalized Linear Models

A linear regression model is the traditional method for fitting a statistical model to data. It is appropriate when the target variable is numeric and continuous.

The family of generalized linear models extends traditional linear regression to targets with non-normal (non-Gaussian) distributions. Linear regression models are iteratively fit to the data after transforming the target variable to a continuous numeric.

Generalized linear regression, applied to a dataset with a numeric, continuous target variable, will build the same model, using a different algorithm.

The generalized algorithm is parameterized by the distribution of the target variable and a link function relating the mean of the target to the inputs. These two parameters describe what we often refer to as a family, such as Poisson, Logistic, etc.

If the target has just two possible outcomes it is transformed using a logistic or probit function. A probit regression gives similar results to the logistic regression, but often with smaller coefficients.

4. Results & Discussion

A logistic regression model was applied on Software Defect Prediction. The performance of classifier was measured using accuracy. By applying statistical testing to construct the classifier the accuracy was improved to 94%. Table 3 shows the Accuracy values of C4.8, SVM Neural Network and ANFIS applied on SDP datasets.

Table 3: Accuracy Values on Various SDP Datasets using Various ANFIS algorithms

Name of the Dataset	C 4.8	SVM	Neural Networks	ANFIS
Cm1	79.6	95.9	93.9	94.3
Kc1	86.7	86.7	87	92.6
Kc2	79.5	77.9	74.4	93.6
Pc1	90.4	89.8	89.8	92.1

5. Conclusion

Predicting Software Defects in advance greatly reduces the development cost of variable. We can improve the performance of the classifier by applying statistical testing on the variables to extract relevant features to address high dimensionality problem. Wilcoxon test was applied for the variables that follow Gamma distribution to find the reliance of variables on defects. We constructed logistic regression on reduced dataset. The performance of Classier was improved to 94%.

References

[1] Shivkumar Shivaji,E James Whitehead, Ram Akella, Sunghun Kim(2013), Reducing Features to Improve code Change Based Bug Prediction, IEEE Transactions on Software Engineering,39(4),.pp:552-569.

[2] Mrinal Singh Rawat, Sanjay Kumar Dubey(2013), Software Defect Prediction for Quality Improvement: A Literature Survey,International Journal of Computer Science Issues,9950.

[3]Tim Menzies, Jeremy Greenwald, Art Frank(2007), Datamining Static code Attributes to learn Defect Predictors,, IEEE Transactions on Software Engineering,33(1),Jan-2007.pp:2-13.

[4] Aditi Thakur, Dr. Ajay Goel(2013), A Hybrid Neuro Fuzzy Approach for Bug Prediction using Software Metrics, International Journal of Engineering Trends and Technology,38(2).

[5] K Punitha, B Latha(2016), Sampling imbalance dataset for Software defect prediction using hybrid neuro fuzzy systems with Naïve Bayes Classifier, Technical gazette,23(6).

[6] Vipul Vashisht, Manohar Lal and G S Sureshchandar(2016), Defect Prediction Framework using Adaptive Neuro Fuzzy Inference System(ANFIS) for Software Enhancement Projects, British Journal of Mathematics & Computer Science,19(2),pp:1-12.

[7] Juan Murillo-Morera, Christian Quesasa-Lopez,Carlos Castro-Herrera and Marcelo Jenkins(2017), A genetic algorithm based framework for software effort prediction, Journal of Software Engineering Research and Development,5(4).

[8]. J S Pahariya, V Ravi, M Carr(2009), Software cost estimation using computational intelligence techniques, Nature & Biologically Inspired Computing,2009. NaBIC 2009. World Congress on,2009.

[9] Ebubeogu Amarachukwu Felix, Sai Peck Lee(2017), Integrated Approach to Software Defect Prediction, IEEE Translations and content Mining.

[10] Wenjie Liu, Shanshan Wang, Xin Chen, He Jiang(2018), Predicting the severity of Bug reports based on feature selection, International Journal of Software Engineering and Knowledge Engineering,28(04),pp:537-558.

[11] Kehan Gao, Taghi M.Khoshgoftaar, Amri Napolitano(2018),The use of ensemble Based Data Preprocessing Techniques for Software Defect Prediction,International Journal of Software Engineering and Knowledge Engineering,24(09),pp:1229-1253.