# Data Analysis on defect data

Arukala Mounika[1]

*Abstract*— **A software bug is an error, flaw, failure or fault in a computer program or system that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. A program that contains a large number of bugs, and/or bugs that seriously interfere with its functionality, is said to be defective.**

**The goal of this paper is to perform data analyses on defect data. To achieve this goal, Spearman and Pearson correlation methods were performed on the data sets and the correlations of each metric to the number of post-release defects were computed and compared.**

**Based on the results, for each dataset, on comparison, the metric with highest correlation is identified for each of the three csv files and the cross validation followed by paired t-tests were performed at the end to identify the best metric for each csv file.**

## I. INTRODUCTION

A defect is an error or a bug, in the application which is created. A programmer while designing and building the software can make mistakes or error. These mistakes or errors mean that there are flaws in the software. These are called defects. The term "bug" to describe defects has been a part of engineering jargon for many decades and predates computers and computer software; it may have originally been used in hardware engineering to describe mechanical malfunctions. Bugs trigger errors that may have ripple effects. Bugs may have subtle effects or cause the program to crash or freeze the computer.

Software defect prediction is the process of locating defective modules in software. It helps to improve software quality and testing efficiency by constructing predictive models from code attributes to enable a timely identification of fault-prone modules, it also helps us in planning, monitoring and control and predict defect density and to better understand and control the software quality.

The goal of this paper is to perform data analyses on defect data. This paper implemented Spearman correlation method and Pearson correlation method on the given five data sets and the correlations of each metric to the number of post-release defects were computed and compared. Later, cross validation and paired t-test were performed to identify the best metric with the highest correlation.

Structure of the paper: In Section 2, data description overview was given. In Section 3, I detailed the approaches that were reproduced. We report their performance in Section 4 and concluded the paper in Section 5.

*This work was not supported by any organization

[1]H. Kwakernaak is with Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, 7500 AE Enschede, The Netherlands h.kwakernaak at papercept.net

[2]P. Misra is with the Department of Electrical Engineering, Wright State University, Dayton, OH 45435, USA p.misra at ieee.org

## II. DATA DESCRIPTION

This paper deals with five datasets. Each data set consists of several csv files of which the following three were taken into consideration while detecting bugs. 1. single-version-ck-oo.csv 2. change-metrics.csv 3. bug-metrics.csv

The first file contains code metrics for all source files while the second contains metrics for change histories of all source files. The third file contains metrics for bug histories of all source files.

One example of each file described above is what that follows next in order: "numberOfLinesOfCode" is the file size in term of lines of code, "linesAddedUntil" is the total number of lines of code added to this file, and "highPriority-Bugs" is the number of post-release defects that are marked as "high priority" ones.

Each file contains the number of post-release defects in column "bugs" and the similar bug counts for sub-categories.

## III. BUG DETECTION APPROACHES

### A. Linear Correlation

Let X and Y be two random variables. The linear correlation coefficient (or Pearson's correlation coefficient) between X and Y, denoted by Corr[X,Y] is defined as follows:

Corr[X,Y] = Cov[X,Y] / (stdev[X]stdev[Y])

where as, Cov[X,Y] is the covariance between X and Y and stdev[X] and stdev[Y] are the standard deviations of X and Y.

Of course, the linear correlation coefficient is well-defined only as long as Cov[X,Y], stdev[X] and stdev[Y] exist and are well-defined. Moreover, while the ratio is well-defined only if stdev[X] and stdev[Y] are strictly greater than zero, it is often assumed that Corr[X,Y]=0 when one of the two standard deviations is zero. This is equivalent to assuming that 0/0=0, because Cov[X,Y]=0 when one of the two standard deviations is zero.

Linear correlation is a measure of dependence (or association) between two random variables. Its interpretation is similar to the interpretation of covariance. Linear correlation has the property of being bounded between $-1$ and 1. The correlation between X and Y provides a measure of the degree to which X and Y tend to "move together":

Corr[X,Y] **greater than 0**, indicates that X and Y are said to be positively linearly correlated (or simply positively correlated); Corr[X,Y] **less than 0**, indicates that X and Y are said to be positively linearly correlated (or simply positively

correlated); when Corr[X,Y] **equal to 0**, X and Y are said to be uncorrelated.

### *B. Ranked Correlation*

The Spearman correlation between two variables is equal to the Pearson correlation between the rank values of those two variables; while Pearson's correlation assesses linear relationships, Spearman's correlation assesses monotonic relationships (whether linear or not). 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.

The Spearman correlation coefficient is defined as the Pearson correlation coefficient between the ranked variables. Spearman

$$r = 1 - \frac{6 \sum (D^2)}{N^3 - N}$$

where, D is the difference between the two ranks of each observation, and N is the number of observations.

The sign of the Spearman correlation indicates the direction of association between X (the independent variable) and Y (the dependent variable). If Y tends to increase when X increases, the Spearman correlation coefficient is **positive**. If Y tends to decrease when X increases, the Spearman correlation coefficient is **negative**. A Spearman correlation of **zero** indicates that there is no tendency for Y to either increase or decrease when X increases.

The Spearman correlation increases in magnitude as X and Y become closer to being perfect monotone functions of each other. When X and Y are perfectly monotonically related, the Spearman correlation coefficient becomes 1.

### *C. Cross Validation*

Cross-validation, sometimes called rotation estimation, is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

The goal of cross validation is to define a dataset to "test" the model in the training phase (i.e., the validation dataset), in order to limit problems like overfitting, give an insight on how the model will generalize to an independent dataset.

One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set).

To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds.

### *D. Paired t-tests*

A paired t-test is used to compare two population means where you have two samples in which observations in one sample can be paired with observations in the other sample. Suppose a sample of n students were given a diagnostic test before studying a particular module and then again after completing the module. We want to find out if, in general, our teaching leads to improvements in students knowledge/skills (i.e. test scores). We can use the results from our sample of students to draw conclusions about the impact of this module in general. Let x = test score before the module, y = test score after the module To test the null hypothesis that the true mean difference is zero, the procedure is as follows:

1. Calculate the difference (di = yi xi) between the two observations on each pair, making sure you distinguish between positive and negative differences.

2. Calculate the mean difference, d'.

3. Calculate the standard deviation of the differences, sd, and use this to calculate the standard error of the mean difference, SE(d') = sd/n

4. Calculate the t-statistic, which is given by T =d'/SE(d'). Under the null hypothesis, this statistic follows a t-distribution with n 1 degrees of freedom.

5. Use tables of the t-distribution to compare your value for T to the tn1 distribution. This will give the p-value for the paired t-test.

## IV. PERFORMANCE

### *A. jdt dataset*

**single-version-ck-oo.csv**
Pearson correlation: numberOfLinesOfCode
Spearman correlation:wmc
**change-metrics.csv**
Pearson correlation: numberOfVersionsUntil.
Spearman correlation: linesAddedUntil.
**bug-metrics.csv**
Pearsoncorrelation: numberOfNonTrivialBugsFoundUntil.
Spearman correlation: numberOfBugsFoundUntil.

### *B. equinox dataset*

**single-version-ck-oo.csv**
Pearson correlation: cbo
Spearman correlation:cbo
**change-metrics.csv**
Pearson correlation: numberOfVersionsUntil.
Spearman correlation:numberOfVersionsUntil.
**bug-metrics.csv**
Pearson correlation: numberOfBugsFoundUntil.
Spearman correlation: numberOfBugsFoundUntil.

### *C. lucene dataset*

**single-version-ck-oo.csv**
Pearson correlation: lcom
Spearman correlation: numberOfAttributes
**change-metrics.csv**
Pearson correlation: linesAddedUntil.
Spearman correlation:maxCodeChurUntil.
**bug-metrics.csv**
Pearson correlation: numberOfBugsFoundUntil.
Spearman correlation: numberOfBugsFoundUntil.

### D. mylyn dataset

**single-version-ck-oo.csv**

Pearson correlation: fanOut

Spearman correlation: rfc

**change-metrics.csv**

Pearson correlation: numberOfVersionsUntil.

Spearman correlation:linesAddedUntil.

**bug-metrics.csv**

Pearson correlation: numberOfNonTrivialBugsFoundUntil.

Spearman correlation: numberOfMajorBugsFoundUntil.

### E. pde dataset

**single-version-ck-oo.csv**

Pearson correlation: numberOfAttributes

Spearman correlation: rfc

**change-metrics.csv**

Pearson correlation: numberOfVersionsUntil.

Spearman correlation: wmc.

**bug-metrics.csv**

Pearson correlation: numberOfBugsFoundUntil.

Spearman correlation:numberOfNonTrivialBugsFoundUntil.

**AVERAGE VALUES OF CORRELATION TESTS**

**Pearson**:

CODE -rfc

CHANGE -numberOfVersionsUntil

BUG -numberOfBugsFoundUntil

**Spearman**:

CODE -numberOfLinesOfCode

CHANGE -linesAddedUntil

BUG -numberOfNonTrivialBugsFoundUntil

**FINAL VALUES OF CORRELATION TESTS**

**Pearson**:

CODE -numberOfLinesOfCode

CHANGE -numberOfVersionsUntil

BUG -numberOfBugsFoundUntil.

**Spearman**:

CODE -numberOfLinesOfCode

CHANGE -linesAddedUntil

BUG -numberOfBugsFoundUntil

**CROSS VALIDATION OF CORRELATION TESTS**

**Pearson**

Project 1. JDT Best Metric: numberOfBugsFoundUntil 2. PDE Best Metric: numberOfBugsFoundUntil 3. MYLYN Best Metric: numberOfVersionsUntil 4. Equinox Best Metric : numberOfVersionsUntil 5. Lucene Best Metric: numberOfBugsFoundUntil

**Spearman**

Project 1.JDT Best Metric: numberOfBugsFoundUntil 2. PDE Best Metric: linesAddedUntil 3. MYLYN Best Metric: numberOfBugsFoundUntil 4. Equinox Best Metric: numberOfBugsFoundUntil 5. Lucene Best Metric: linesAddedUntil

## V. CONCLUSION

Bug prediction concerns the resource allocation problem: Having an accurate estimate of the distribution of bugs across components helps project managers to optimize the available resources by focusing on the problematic system parts. Software bug causes a computer program to produce an incorrect or unexpected result. Hence, detection of bugs in a program plays a very crucial role. In this paper, five defect datasets were taken and data analysis was performed on it. Three csv files from each dataset were taken into consideration and Pearson and Spearman correlations were computed.

At the end, cross validation and paired t-tests were performed on the above resultant data and the best metric for each csv file was identified. The results were shown in Section 4.

## REFERENCES


[1] An Extensive Comparison of Bug Prediction Approaches.

[2] N. Ohlsson and H. Alberg, Predicting fault-prone software modules in telephone switches, IEEE Trans. Software Eng., vol. 22, no. 12, pp. 886894, 1996.

[3] T. Gyimothy, R. Ferenc, and I. Siket, Empirical validation of object-oriented metrics on open source software for fault prediction, IEEE Trans. Software Eng., vol. 31, no. 10, pp. 897910, 2005.

[4] ] N. Nagappan, T. Ball, and A. Zeller, Mining metrics to predict component failures, in Proceedings of ICSE 2006. ACM, 2006, pp. 452461.

[5] R. Subramanyam and M. S. Krishnan, Empirical analysis of ck metrics for object-oriented design complexity: Implications for software defects, IEEE Trans. Software Eng., vol. 29, no. 4, pp. 297310, 2003.

[6] N. Nagappan and T. Ball, Static analysis tools as early indicators of pre-release defect density, in Proceedings of ICSE 2005. ACM, 2005, pp. 580586