**BUG PREDICTION**

**Abstract:**

Predicting software defects is one of software engineering’s holy grails. Researchers have implemented many bug prediction approaches varying in terms of accuracy, complexity and the input data they require. The motivation to have such defect prediction model is to serve as early quality indicator of the software entering system testing and assist the testing team to manage and control test execution activities. In this paper, I used linear regression model and cross-validation.

**Data Description:**

The dataset used for this quiz is in file JDT.csv. It has four variables namely,

* **Bug**: number of post-release bugs, dependent variable
* **loc**: number of lines of code, predictor
* **Bugfound**: number of bug found previously, and
* **Version**: number of versions.

**Bug Prediction Approaches:**

LINEAR REGRESSION

**Linear regression** is a method to model the relationship between a scalar dependent variable(y) and one or more independent variables, denoted X. The case of one independent variable is called simple **linear regression**.

**Simple linear regression:** We predict scores on one variable from the scores on a second variable. The variable we are predicting is called the *criterion variable* and is referred to as Y. The variable we are basing our predictions on is called the *predictor variable* and is referred to as X. When there is only one predictor variable, the prediction method is called *simple regression*. In simple linear regression, the predictions of Y when plotted as a function of X form a straight line.

**Regression line:** Linear regression consists of finding the best-fitting straight line through the points. The best-fitting line is called a *regression line*. The regression line consists of the predicted score on Y for each possible value of X. The vertical lines from the points to the regression line represent the errors of prediction.

NOTE: The closer the point is to the regression line, the smaller is the error.

lm(): Function used to do linear regression.

CROSS VALIDATION

**Cross-validation** is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent data set.

**Target** is to define a dataset to” test” the model in the training phase (i.e., the validation dataset), 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.

COMPARE PERFORMANC EOF MODELS IN RELATIVE TERMS

**Mean absolute error (MAE)** is also measured in the same units as the data, and is usually similar in magnitude to, but slightly smaller than, the root mean squared error.  It is less sensitive to the occasional very large error because it does not square the errors in the calculation. It is relatively easy to compute them in RegressIt:  just choose the option to save the residual table to the worksheet, create a column of formulas next to it to calculate errors in absolute or absolute-percentage terms, and apply the AVERAGE function.

RESULT

R-Squared values:

summary(foundbug)$r.squared

[1] 0.4560663

> summary(located)$r.squared

[1] 0.3487671

> summary(version)$r.squared

[1] 0.3806763

> summary(everything)$r.squared

[1] 0.4785589

Bug found highest

Cross Validation:

mean(foundbug1)

[1] 0.391635

> mean(located1)

[1] 0.4512669

> mean(version1)

[1] 0.4368763

> mean(everything1)

[1] 0.387281

Bug found lowest

Values of everything are highest compared to the single predictor for linear regression.

Values of everything are lowest compared to the single predictor for cross validation.

Hence, everything is better.

{\displaystyle y\_{i}=\beta \_{1}x\_{i1}+\cdots +\beta \_{p}x\_{ip}+\varepsilon \_{i}=\mathbf {x} \_{i}^{\rm {T}}{\boldsymbol {\beta }}+\varepsilon \_{i},\qquad i=1,\ldots ,n,}