POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

TP 3: Bug prediction models

LOG6305 Course

Prof. Giuliano Antoniol — Lab Date: 27/02/19

Lab Assistant: Houssem Ben Braiek — Due Date : 17/03/19

Evaluation criteria

1. A report in PDF format and renamed as *firstname_lastname_studentID.pdf*.

2. The completed code of the second part should be zipped and renamed as *firstname_lastname_studentID.zip*;

This work should be submitted to the mail address: houssem.ben-braiek@polymtl.ca no later than 17/03/19 at 23:59h; Otherwise, penalties will be applied **5 deduction points** for the delay of every **24 hours**.

Required Work

Goal: The application of the logistic model to the problem of bug prediction within multiple versions of one single software project.

Study Case: In our case, we intend to learn a model based on data extracted from QT version 5.0 (considered as training data) and predict the buggy classes in the next version 5.1 (considered as testing set). Obviously, the dependent variable (i.e. predict value) is the occurrence of bugs(last column). The rest of the columns are the independent variables (i.e. features), which consist of well-known software metrics as described in the table 1.

Procedure:

1. Compute the correlation between the provided features using Spearman correlation, in order to check if there are highly correlated features;

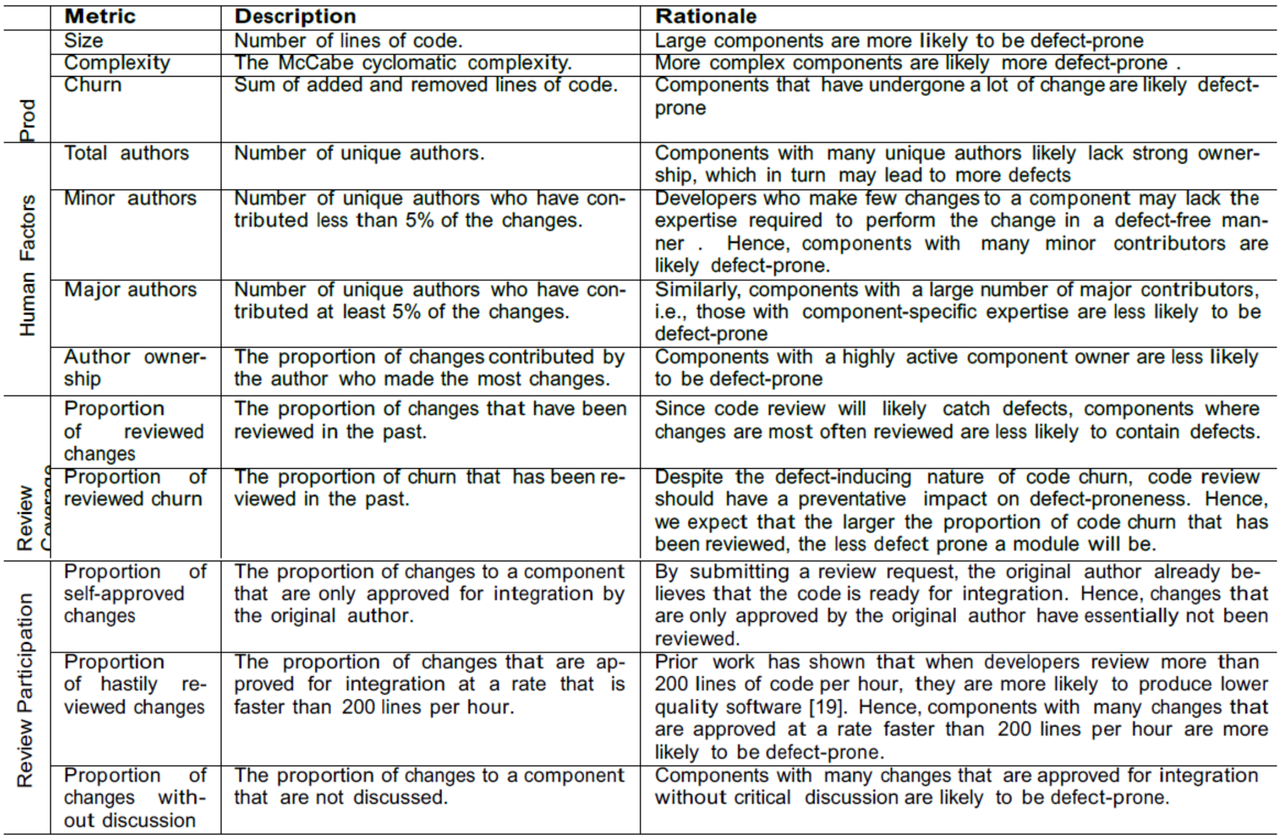
2. Build models using different combination of chosen features and check the statistical tests results in order to assess the contribution of each feature to the obtained model performance. The following techniques can help you doing the features selection.

   Coefficient test The R function *summary* displays the statistical test for the probability that the coefficient related to the feature is different from 0.

   ANOVA analysis The R built-in function *anova* performs the ANOVA analysis to determine the improvement and its significance in deviance by adding each feature.

   Drop one tests The R function *drop1* performs drop one tests that assess the impact of each feature on the model by measuring the AIC of the models consisting of: (1) all the features included (the complete model), and (2) all the features included except the one under test (the abandoned model). A χ^2 test is applied to the resulting values to detect whether the dropped feature improves significantly the AIC of the model.

   As a result, the features with *, **, and *** are the ones that contribute significantly to predict the target outcome.



	Metric	Description	Rationale
Prod	Size	Number of lines of code.	Large components are more likely to be defect-prone
	Complexity	The McCabe cyclomatic complexity.	More complex components are likely more defect-prone .
	Churn	Sum of added and removed lines of code.	Components that have undergone a lot of change are likely defect-prone
Human Factors	Total authors	Number of unique authors.	Components with many unique authors likely lack strong ownership, which in turn may lead to more defects
	Minor authors	Number of unique authors who have contributed less than 5% of the changes.	Developers who make few changes to a component may lack the expertise required to perform the change in a defect-free manner . Hence, components with many minor contributors are likely defect-prone.
	Major authors	Number of unique authors who have contributed at least 5% of the changes.	Similarly, components with a large number of major contributors, i.e., those with component-specific expertise are less likely to be defect-prone
	Author ownership	The proportion of changes contributed by the author who made the most changes.	Components with a highly active component owner are less likely to be defect-prone
Review Coverag	Proportion of reviewed changes	The proportion of changes that have been reviewed in the past.	Since code review will likely catch defects, components where changes are most often reviewed are less likely to contain defects.
	Proportion of reviewed churn	The proportion of churn that has been reviewed in the past.	Despite the defect-inducing nature of code churn, code review should have a preventative impact on defect-proneness. Hence, we expect that the larger the proportion of code churn that has been reviewed, the less defect prone a module will be.
Review Participation	Proportion of self-approved changes	The proportion of changes to a component that are only approved for integration by the original author.	By submitting a review request, the original author already believes that the code is ready for integration. Hence, changes that are only approved by the original author have essentially not been reviewed.
	Proportion of hastily reviewed changes	The proportion of changes that are approved for integration at a rate that is faster than 200 lines per hour.	Prior work has shown that when developers review more than 200 lines of code per hour, they are more likely to produce lower quality software [19]. Hence, components with many changes that are approved at a rate faster than 200 lines per hour are more likely to be defect-prone.
	Proportion of changes without discussion	The proportion of changes to a component that are not discussed.	Components with many changes that are approved for integration without critical discussion are likely to be defect-prone.

Figure 1: A brief description of the features (i.e. independent variables)

3. Adding to the feature selection, you can also create more complex features from existing ones by applying, for example, non-linearity functions (i.e. log, square, square root, etc...) to see if it improves the resulting model performance.

4. The process of searching the best combination of features is an iterative process guided by the model performance metrics. Following, you find some concepts to understand how the model is trained and how it can be evaluated.

   Maximum likelihood The maximum likelihood estimate of a parameter is the value of the parameter for which the probability of obtaining the observed data is the highest.

   Deviance The deviance is negative two times the maximum log likelihood up to an additive constant. The parameter estimation consists of finding parameter values that minimize the deviance.

   AIC It combines a measure of model fit with a measure of model complexity: The smaller, the better.This allows to strike a good balance between model fit and model simplicity. $AIC = Deviance + 2 \times p$, where $p =$ total number of coefficients.

   K-folds cross-validation It is a model validation technique for assessing how accurately a predictive model will perform in practice (test data). In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k1$ subsamples are used as training data. The cross-validation process is then repeated k times (the folds), with each of the k subsamples used exactly once as the validation data. The k results from the folds can then be averaged to produce a single estimation of the model accuracy on random unseen data.

Important Note: Beware that including simple variables with negligible effect and complex variables that are meaningless could overfit your model, so it would have high performance on training data, but low prediction ability on the unseen testing data.

Part 1 - Report

Objective 1. Model Engineering Steps: Enumerate and describe all the steps you followed to obtain your final model.
Objective 2. Comprehension Questions:

1. How many variables you found highly-correlated?

2. How do you choose between the highly correlated variables?

3. Are code review metrics useful to predict bugs in future versions?

4. How many iterations do you require to find the best model?

5. Why is important to compare different models using statistically tests e.g., ANOVA test?

Part 2 - Code

Objective 1. Rscript: Write a new Rscript *firstname_lastname_QT_PredictionModel* that loads the training data from the csv file named *qt50.csv*(using load command), builds and compares different logistic models(using glm command), and evaluates the prediction performance of the chosen optimal model on the testing data loaded from *qt51.csv*. You find a starter snippet code 1 to help you write a good R script and to find the optimal model.

Objective 2. Documentation: Do not forget to thoroughly comment every single line of code you write.

```r
#To install and import the package containing the cross-validation method (cv_binary)
install.packages("DAAG")
library(DAAG)

#To define the directory of data
data_path <- "./data/"
#To change the directory as our working directory
setwd(data_path)
#To define your training data file
train_file <-"train_data.csv"
#To load the training data
train_data <- read.csv(train_file, header=T, sep = ",")

#To check correlation between two features
f1_f2_corr <- cor(train_data[,"feature_1"],train_data[,"feature_2"], method="spearman")
print(f1_f2_corr)
#To choose the columns names of chosen features
features_cols<-c("feature_1","feature_2","feature_3")

#To create a logistic model
m1_formula <- as.formula(paste("outcome ~ ", paste(features_cols, collapse= "+")))
m1_model <- glm(m1_formula, data=train_data, family=binomial("logit"))
#To display the model built-in summary to display the statistical test
summary(m1_model)
#To analyze the contribution of each variable using ANOVA
anova(m1_model, test="Chisq")
#To perform drop one tests to measure the impact of each explanatory variable on AIC value
drop1(m1_model, test="Chisq")
#To display the AIC
AIC(m1_model)
#To apply cross-validation with choosing 10 folds
cv.binary(m1_model, rand=NULL, nfolds=10, print.details=TRUE)

#Assuming that you intend to do some optimizations in order to create another model
m2_model <- ...
#To use the ANOVA analysis in order to determine the significance of the improvement in variance explained
anova(m1_model,m2_model, test="Chisq")

#Assuming that you choose the second model as your final model
#To define the testing data file
test_file <-"test_data.csv"
#To load the test data
test_data <- read.csv(test_file, header=T, sep=",")
#To use the model to predict on test data
outcome_pred <- predict(m2_model, type="response", newdata=test_data)
#To fix a threshold probability equal to 0.5
confusion_matrix<-table(outcome_pred > 0.5, test_data$outcome)
#To calculate the accuracy of your model on testing data
accuracy <- (confusion_matrix["FALSE","FALSE"]+confusion_matrix["TRUE","TRUE"])/sum(confusion_matrix)
print(accuracy)
```

R Code 1: A starter snippet code to do logistic regression with R