Predicting the bug fixing likelihood

Florian Spychiger

University of Zurich

December 9, 2018

Roadmap

1. Background
2. Problem Formulation & Goal
3. Data
4. Solution Approach
5. Results
6. Q&A

Background Information

The annual cost of software bugs is estimated at \$59.5 billion[1]. For the Eclipse project, there are thousands of bugs reported. An efficent bug-triaging can help developers to focus their resources and thus, save companies a lot of money.

[1] P Bhattacharya and I Neamtiu, "Fine-grained incremental learning and multi-feature tossing graphs to improve bug triaging", Software Maintenance (ICSM) 2010 (ieeexplore.ieee.org)

Problem Formulation & Goal

Problem

Bug-triaging is an important, but labor-intensive process if done manually.

Goal

Train a bug-triaging machine, which predicts whether a bug is likely to be fixed.

Raw Data

The Eclipse data set can be found at
`https://github.com/ansymo/msr2013-bug_dataset`.
The raw data set consists of 12 tables:

Eclipse Bug Data Set	
reports	priority
assigned_to	product
bug_status	resolution
cc[2]	severity
component	short_desc
op_sys	version

Table 1: Tables of the bug data set.

[2] The data has been newly formated with Excel VBA.

Data Preselection

After a visual exploratory analysis, four datasets were excluded:

Eclipse Bug Data Set*	
reports	priority 1
assigned_to	product
bug_status	resolution
cc	severity 2
component	short_desc 3
op_sys	version 4

Table 2: Excluded data tables.

*All duplicate bugs are excluded.

1. Priority is set by the assignee, but as we want to help them triaging the bugs, we exlude it.
2. Severity is currently set by the triaging team.
3. The descriptions are hard to encode.
4. The version dataset is quite messy and sometimes it is not clear which version is being referred to.

Data Model

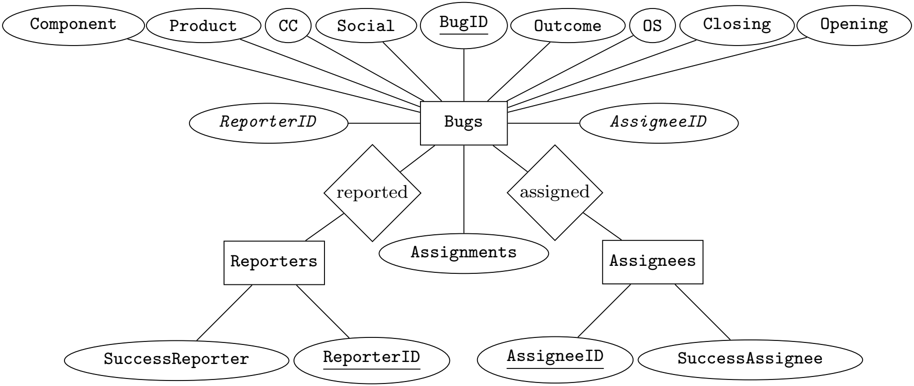

Figure 1: ER model of data used.

Feature Creation

From the data model, the feature matrix X is constructed with:

$$
\begin{aligned}
x_1 &= \text{OpenTime (Open - Close)} & [discrete]\\
x_2 &= \text{Assignments (Nr. of assignees)} & [discrete]\\
x_3 &= \text{CC (Nr. of interested parties)} & [discrete]\\
x_4 &= \text{Product (Affected product)} & [discrete]\\
x_5 &= \text{OS (Major OS)} & [discrete]\\
x_6 &= \text{SuccessAssignee (Success rate of Assignee)} & [proportion]\\
x_7 &= \text{SuccessReporter (Success rate of Reporter)} & [proportion]\\
x_8 &= \text{Component (The affected subcomponent)} & [discrete]\\
x_9 &= \text{Social (Past bug collaborations)} & [discrete]\\
x_{10} &= \text{Equal (Reporter equals Assignee)} & [binary]
\end{aligned}
$$

The labels are $y =$ Fixed with values in $\{0, 1\}$.

Univariate Analysis

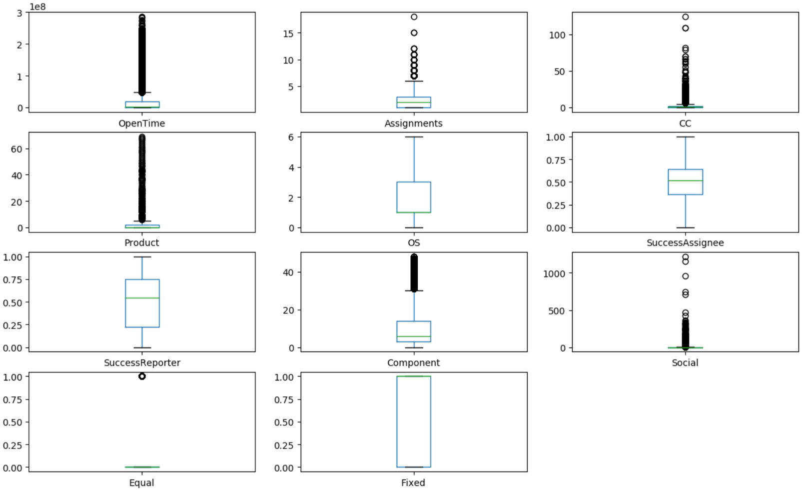

Figure 2: Boxplots of the features and the label.

Correlation Analysis

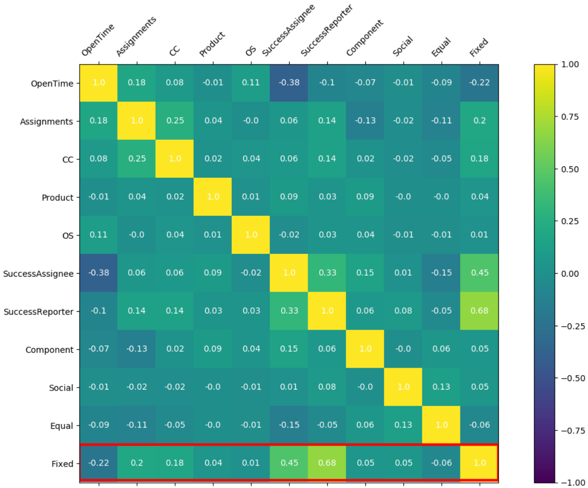

Figure 3: Correlations of the features and the labels.

Models

We consider 6 models:

1. Naive Bayes
2. Logistic Regression
3. Random Forest
4. Boosting Classifier
5. Support Vector Machine
6. Neural Network

We split the data set into a training (50%), a cross-validation (25%) and a test (25%) set. The training set is used to train the models and we calibrate the parametes on the cross-validation set. The final accuracy is caculated on the test set.

Accuracy

We achieve the following accuracies on the test set:

Naive Bayes	82.8098%
Logistic Regression	84.9409%
Random Forest	86.1529%
Boosting Classifier	85.4661%
Support Vector Machine	85.9105%
Neural Network[3]	86.1125%

Table 3: Accuracies of the models.

[3] Results are not exactly reproducible, as some randomness with GPU usage cannot be avoided.

ROC-Curves

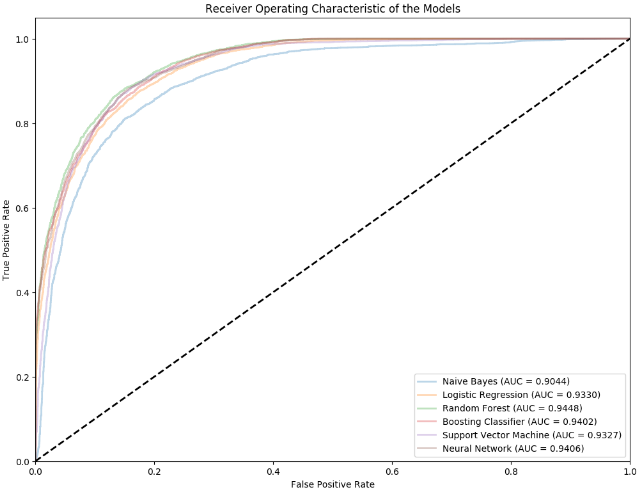

Figure 4: ROC-Curves of all models.

Q&A

The code of the project can be found at

`https://github.com/Speaker90/BusinessAnalytics_RPIcase`