Business Analytics Project

Luca Gaegauf, Daniel Salamanca

University of Zurich

December 15, 2015

Overview

1. Problem and dataset
2. Exploratory work
3. Creation of prediction variables
4. Logic of solution approach

Problem and dataset

- The goal of the research is to identify the factors that are likely to be predictive of a bug being fixed.
- The motivation behind this issue is to save labor time and thus increase productivity.
- Data: The Eclipse and Mozilla Defect Tracking Dataset[1]

[1] https://github.com/ansymo/msr2013-bug_dataset

Exploratory work

To familiarize ourselves with the data we:

- Read the article by Lamkanfi, Perez and Demeyer (2013)[2]
- Had a look at an example of bug report and its attributes[3].
- Compared the number of occurrences of a few bug reports in each CSV table in order to grasp the process through which the bug has to go through during its lifetime.
- Ran a logistic regression for every attribute (product, component, version, operating System, priority, severity) in order to get an idea which attributes were the most relevant and thus decided not to include the product, component and version attributes and only a smaller subset of operating systems. (cf. script ExploratoryLogit.r)

[2] A. Lamkanfi, J. Perez and S. Demeyer, "The Eclipse and Mozilla Defect Tracking Dataset: a Genuine Dataset for Mining Bug Information", MSR'13:Proceedings of the 10th Working Conference on Mining Software Repositories, May18–19, 2013.

[3] https://bugs.eclipse.org/bugs/page.cgi?id=fields.html

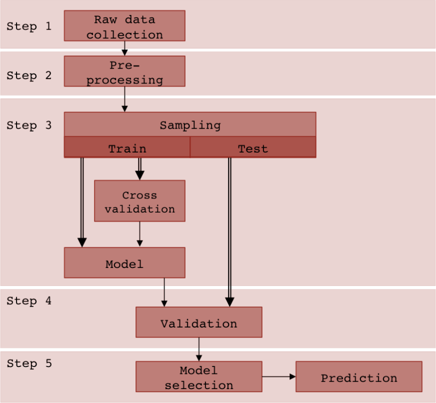
Creation of prediction variables (2)

Variable	Type	Description	Out of 165547
fixed	Dummy $\{0, 1\}$	Bug was fixed	76268
reopened	Dummy $\{0, 1\}$	Bug was historically reopened.	16172
success rate of assignee	Continuous [0,1]	Proportion of assigned bugs which the assignee fixed	mean: 0.2930
time opened	Count $\{\mathbb{Z}^+\}$	Duration of bug existence.	mean: 15949108
success rate of the reporter	Continuous [0,1]	Proportion of reported bugs which have been fixed	mean: 0.4607
number of assignment	Count $\{\mathbb{N}\}$	Amount of times assigned	mean: 1.996
number of edition	Count $\{\mathbb{N}\}$	Amount of times the bug status is edited.	mean: 2.744
linux	Dummy $\{0, 1\}$	Bug affects only Linux	20337
macOS	Dummy $\{0, 1\}$	Bug affects only Mac OS	5644
all	Dummy $\{0, 1\}$	Bug affects all OS	28158

Creation of prediction variables

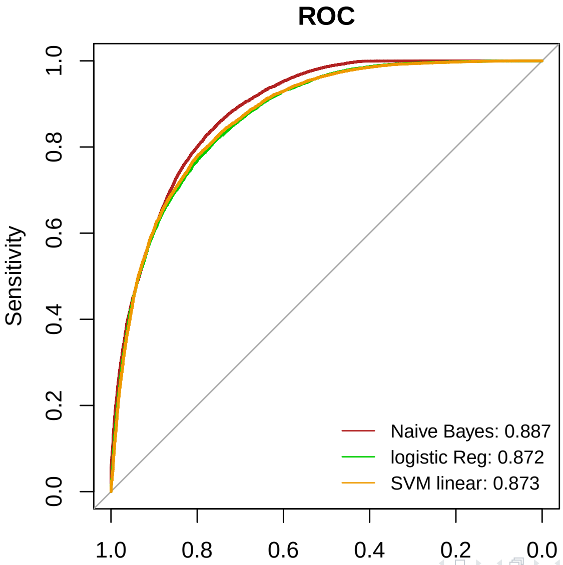
- We decided to consider CSV tables which contain relevant information. Thus we ignored the "cc" and "short description" tables.
- We also chose not to include the priority and severity tables because those attributes are the results of human appreciations and are inherent to the reports themselves.
- For the computation of the assignee's success rate, we calculated the number of bugs which a given assignee was assigned to and the number of bugs which he himself fixed. Then we linked each bug to the success rate of its latest assignee.
- To calculate the time during which the bug was opened we computed the difference between its opening time and its last modification in the resolution table.

Logic of solution approach

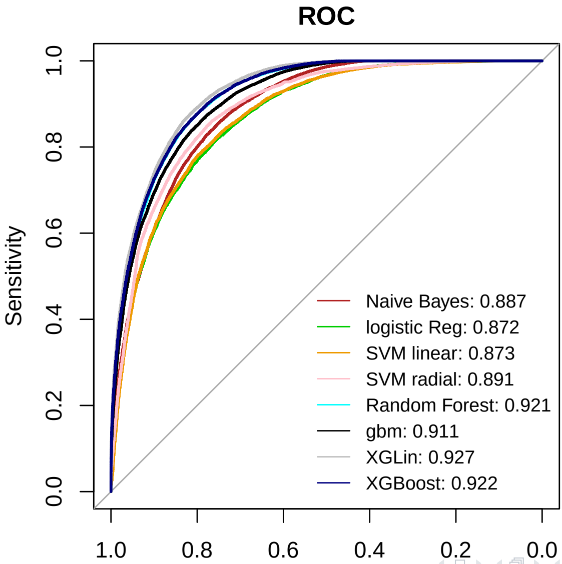


- The data was sampled into a 60/40 split.
- 3-fold cross validation was used for the machine learning algorithms.
- The models were then used to predict whether the test bugs had been fixed and evaluated based on their predictions (ROC and UAC).

ROC(1)


ROC
Sensitivity
1.0
0.8
0.6
0.4
0.2
0.0
1.0
0.8
0.6
0.4
0.2
0.0
Naive Bayes: 0.887
logistic Reg: 0.872
SVM linear: 0.873

ROC(2)


ROC
Sensitivity
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.8
0.6
0.4
0.2
0.0
Naive Bayes: 0.887
logistic Reg: 0.872
SVM linear: 0.873
SVM radial: 0.891
Random Forest: 0.921
gbm: 0.911
XGLin: 0.927
XGBoost: 0.922

Conclusion

- Based on the ROC and the AUC the naive bayes model performed the best out of the three prescribed models.
- However, out of all computed models, the linear extreme gradient boosting model performed best.