

Predicting Bug Resolution on Eclipse Browser

Al-sadh Imadh, Hope Mullins, Gualberto Oliveira

1 August 2024

Summary

We explored a dataset on bug tracking for the Eclipse Browser.

Summary

We explored a dataset on bug tracking for the Eclipse Browser.

Our goal was to analyze the factors that impacted the likelihood of a bug being fixed.

Summary

We explored a dataset on bug tracking for the Eclipse Browser.

Our goal was to analyze the factors that impacted the likelihood of a bug being fixed.

We trained 5 types of models and found that Random Forest gave the best AUC (Area Under the Curve) score.

ROADMAP OF TALK

① Problem

ROADMAP OF TALK

① Problem

② Theory

ROADMAP OF TALK

- ① Problem
- ② Theory
- ③ Data

ROADMAP OF TALK

- ① Problem
- ② Theory
- ③ Data
- ④ Training, Validation, and Testing

ROADMAP OF TALK

- ① Problem
- ② Theory
- ③ Data
- ④ Training, Validation, and Testing
- ⑤ Conclusions

Bug Reports

1

LOGIT Regression

- 1 This model takes linear combination of the factors and transforms it to return a value $p \in [0, 1]$.

- Regression equation:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x + \dots + \beta_k x_k$$

LOGIT Regression

- 1 This model takes linear combination of the factors and transforms it to return a value $p \in [0, 1]$.
- 2 0 represents "Won't Fix", 1 represents "Fixed"

- Regression equation:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x + \dots + \beta_k x_k$$

LOGIT Regression

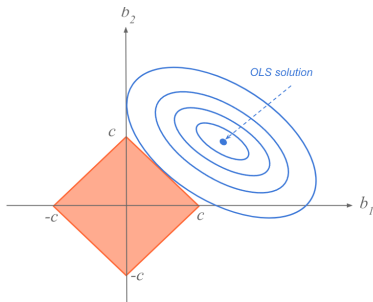
- 1 This model takes linear combination of the factors and transforms it to return a value $p \in [0, 1]$.
 - 2 0 represents "Won't Fix", 1 represents "Fixed"
 - 3 Our optimal threshold to label output as 1 was 0.56
- Regression equation:

$$\ln\left(\frac{p}{1-p}\right) = \beta_0 + \beta_1 x + \dots + \beta_k x_k$$

Lasso Regularization

- ① This improves the previous LOGIT model by adding to the loss function L

$$L + \lambda \sum_i |\beta_i|$$



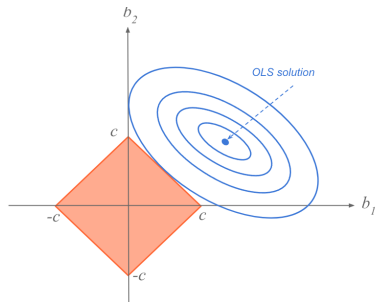
<https://allmodelsarewrong.github.io/lasso.html>

Lasso Regularization

- 1 This improves the previous LOGIT model by adding to the loss function L

$$L + \lambda \sum_i |\beta_i|$$

- 2 Model is encouraged to push (and even set) β_i 's to 0.

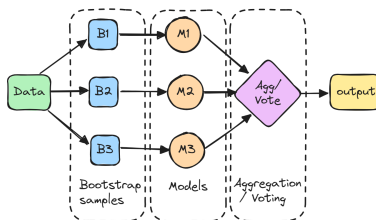


<https://allmodelsarewrong.github.io/lasso.html>

Bagging

Also known as Bootstrap Aggregation.

- 1 Reduces variance in a noisy dataset

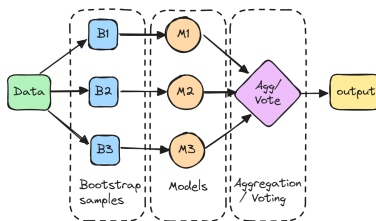


<https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples>

Bagging

Also known as Bootstrap Aggregation.

- 1 Reduces variance in a noisy dataset
- 2 Initial dataset is bootstrapped and each sample is used to make a decision tree

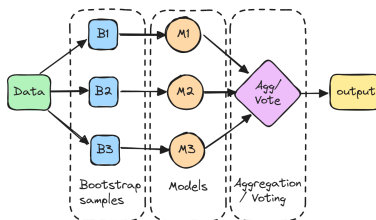


<https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples>

Bagging

Also known as Bootstrap Aggregation.

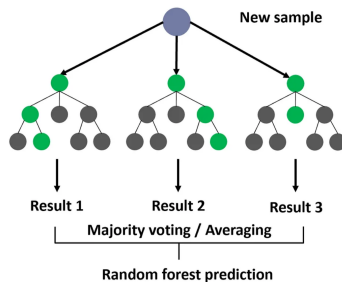
- 1 Reduces variance in a noisy dataset
- 2 Initial dataset is bootstrapped and each sample is used to make a decision tree
- 3 Smaller models are averaged/aggregated to make overall model.



<https://www.datacamp.com/tutorial/what-bagging-in-machine-learning-a-guide-with-examples>

Random Forest

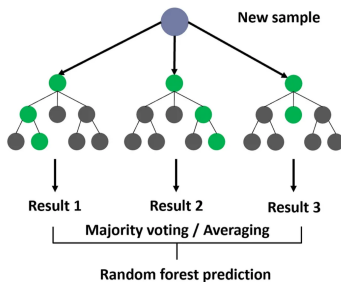
- 1 Closely related to Bagging.



<https://medium.com/@roiyehe/random-forests-98892261dc49>

Random Forest

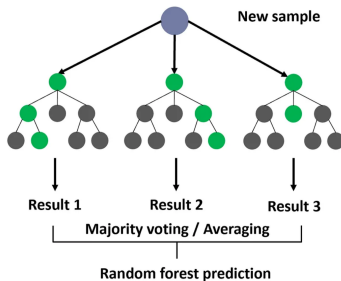
- 1 Closely related to Bagging.
- 2 Unlike bagging, only a handful of the variables are considered during a node split



<https://medium.com/@roiyehe/random-forests-98892261dc49>

Random Forest

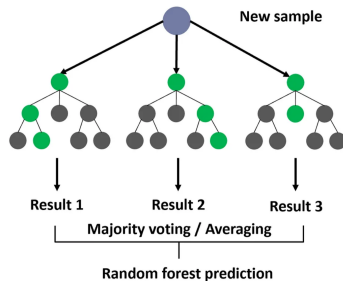
- 1 Closely related to Bagging.
- 2 Unlike bagging, only a handful of the variables are considered during a node split
- 3 The variables are chosen randomly during training



<https://medium.com/@roiyehe/random-forests-98892261dc49>

Random Forest

- 1 Closely related to Bagging.
- 2 Unlike bagging, only a handful of the variables are considered during a node split
- 3 The variables are chosen randomly during training
- 4 The random selection leads to more independent trees which can help prediction



Boosting

- 1 Weak decision trees are built sequentially.



<https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/>

Boosting

- 1 Weak decision trees are built sequentially.
- 2 Models (Learners) build on top of the previous one, each model weighted on performance.



<https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/>

Boosting

- 1 Weak decision trees are built sequentially.
- 2 Models (Learners) build on top of the previous one, each model weighted on performance.
- 3 We used Gradient Boosting, which tries to build learners that are more efficient than the previous.



<https://www.geeksforgeeks.org/boosting-in-machine-learning-boosting-and-adaboost/>

Eclipse Bug Tracking Dataset

- The Eclipse and Mozilla Defect Tracking Dataset

Eclipse Bug Tracking Dataset

- The Eclipse and Mozilla Defect Tracking Dataset
- 12 Excel files containing attributes about a bug. Attributes are updated over time

Eclipse Bug Tracking Dataset

- The Eclipse and Mozilla Defect Tracking Dataset
- 12 Excel files containing attributes about a bug. Attributes are updated over time
- Only bugs labeled "Fixed" or "Won't Fix" are kept. Duplicate bugs removed. Final Dataset has size of 90,789

Eclipse Bug Tracking Dataset

- The Eclipse and Mozilla Defect Tracking Dataset
- 12 Excel files containing attributes about a bug. Attributes are updated over time
- Only bugs labeled "Fixed" or "Won't Fix" are kept. Duplicate bugs removed. Final Dataset has size of 90,789
- New variables created using information in dataset

Eclipse Bug Tracking Dataset

- The Eclipse and Mozilla Defect Tracking Dataset
- 12 Excel files containing attributes about a bug. Attributes are updated over time
- Only bugs labeled "Fixed" or "Won't Fix" are kept. Duplicate bugs removed. Final Dataset has size of 90,789
- New variables created using information in dataset
- Excel and SQLite3 was used to cleanup the data and create new variables

Data Structure

id	curr_res	reporter	stat_upd	num_intrst	op_sys	component	prod	severity
287149	FIXED	17941	2	6	Linux-GTK	SWT	Platform	normal
89374	FIXED	57	5	1	Windows 2000	UI	Platform	normal
89378	FIXED	8126	3	5	Windows XP	SWT	Platform	normal

Sample of dataset. Some new variables include

Data Structure

id	curr_res	reporter	stat_upd	num_intrst	op_sys	component	prod	severity
287149	FIXED	17941	2	6	Linux-GTK	SWT	Platform	normal
89374	FIXED	57	5	1	Windows 2000	UI	Platform	normal
89378	FIXED	8126	3	5	Windows XP	SWT	Platform	normal

Sample of dataset. Some new variables include

- Total time bug report is open
- Max priority/severity
- Number of reassignments/status changes
- Success rate of initial assignee

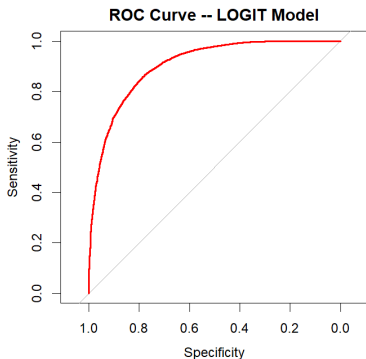
Variable Descriptions (1)

- id: Unique identifier for bug
- curr_res: Whether a bug is fixed or won't be fixed
- reporter: Unique identifier for the bug reporter
- stat_upd: Number of times the bug status was updated
- num_inrst: Number of emails interested in the bug
- op_sys: Operating system that bug affects. Most recent update value used
- prod: Software/product that the bug pertains to
- component: Subsystem of product that the bug affects
- severity: Highest severity given to the bug

Variable Description (2)

- version: The version of the product that the bug affects
- times_assigned: Number of times the bug was reassigned
- succ_rate: The success rate of the initial assignee
- res_upd: Number of times the resolution of a bug was changed
- res_time: Time until the bug was resolved
- reporter_report_cnt: How many bugs the reporter has reported in the dataset
- desc_length: Sum of lengths of the descriptions on the bug
- prio: Highest priority value assigned to the bug

Model: LOGIT Regression (with poly)

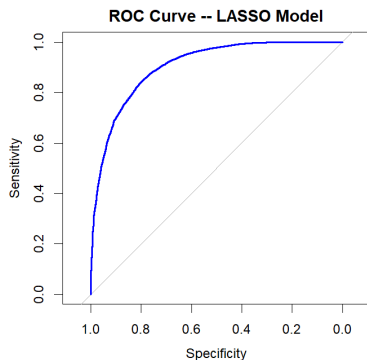


Confusion Matrix

class	Wont Fix	Fixed
Wont Fix	2848	777
Fixed	3070	29624

Figure: AUC: 0.9048

Model: LASSO Regularization (with poly)

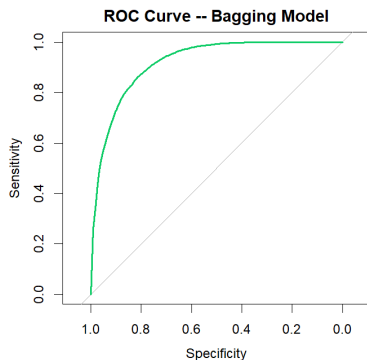


Confusion Matrix

class	Wont Fix	Fixed
Wont Fix	3161	826
Fixed	2757	29575

Figure: AUC: 0.9286

Model: Bagging

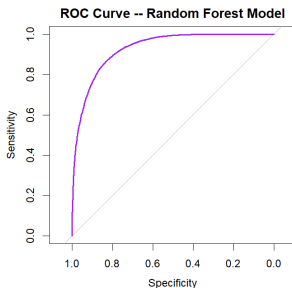


Confusion Matrix

class	Wont Fix	Fixed
Wont Fix	3425	521
Fixed	2493	29880

Figure: AUC: 0.9186

Model: Random Forest



Confusion Matrix

class	Wont Fix	Fixed
Wont Fix	3431	438
Fixed	2487	29963

Figure: AUC: 0.9286

Model: Boosting

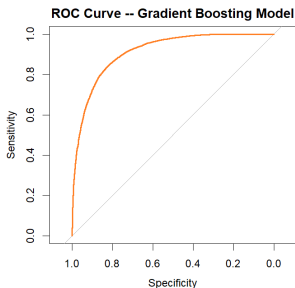


Figure: AUC: 0.9122

Confusion Matrix

class	Wont Fix	Fixed
Wont Fix	3349	916
Fixed	2569	29485

Important Variables

- 1 Based on the results of the LOGIT regression and the subsequent LASSO regularization

Important Variables

- 1 Based on the results of the LOGIT regression and the subsequent LASSO regularization
- 2 Almost every variable we examined was significant

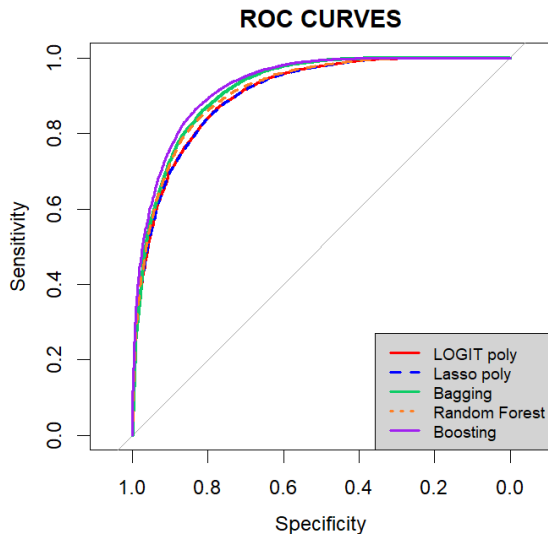
Important Variables

- 1 Based on the results of the LOGIT regression and the subsequent LASSO regularization
- 2 Almost every variable we examined was significant
- 3 The only variable that was found to not be significant was "Component"

Summary of LOGIT variables (no poly)

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-2.4051	0.1819	-13.22	0.0000
reporter	-0.0000	0.0000	-19.40	0.0000
stat_upd	0.5816	0.0208	27.90	0.0000
num_inrst	0.1420	0.0095	14.99	0.0000
op_sys	0.0039	0.0012	3.25	0.0012
component	-0.0007	0.0016	-0.40	0.6888
prod	-0.1186	0.0179	-6.62	0.0000
severity	0.2028	0.0113	17.90	0.0000
version	0.0382	0.0025	15.53	0.0000
times_assigned	0.2017	0.0179	11.28	0.0000
succ_rate	5.1959	0.1233	42.15	0.0000
res_upd	-0.7482	0.0322	-23.26	0.0000
res_time	-0.0000	0.0000	-46.34	0.0000
reporter_report_cnt	-0.0002	0.0000	-15.26	0.0000
desc_length	-0.0022	0.0003	-6.46	0.0000
prio	-0.1457	0.0151	-9.65	0.0000

ROC Curve Comparison



Selection of Best Model

- 1 All models were also tested using `poly()` to give 135 polynomial combinations, with LOGIT and LASSO seeing the only notable improvements

Selection of Best Model

- 1 All models were also tested using `poly()` to give 135 polynomial combinations, with LOGIT and LASSO seeing the only notable improvements
- 2 We used the AUC score to determine the best model. Boosting, Random Forest, and Bagging all had similarly high AUC scores.

Selection of Best Model

- 1 All models were also tested using `poly()` to give 135 polynomial combinations, with LOGIT and LASSO seeing the only notable improvements
- 2 We used the AUC score to determine the best model. Boosting, Random Forest, and Bagging all had similarly high AUC scores.
- 3 Using our training/testing set, Random Forest gave the highest AUC score, i.e. the best testing performance

Final Notes

- 1 Each model was a good predictor of how a bug would be resolved

Final Notes

- 1 Each model was a good predictor of how a bug would be resolved
- 2 The confusion matrix shows that every model struggles to identify when a bug will not be fixed correctly

Final Notes

- 1 Each model was a good predictor of how a bug would be resolved
- 2 The confusion matrix shows that every model struggles to identify when a bug will not be fixed correctly
- 3 In line with the AUC scores, Random Forest identifies both categories the best with Bagging at a close second

Final Notes

- 1 Each model was a good predictor of how a bug would be resolved
- 2 The confusion matrix shows that every model struggles to identify when a bug will not be fixed correctly
- 3 In line with the AUC scores, Random Forest identifies both categories the best with Bagging at a close second
- 4 Boosting was expected to outperform every other model but ranked below both bagging and random forest

Final Notes

- 1 Each model was a good predictor of how a bug would be resolved
- 2 The confusion matrix shows that every model struggles to identify when a bug will not be fixed correctly
- 3 In line with the AUC scores, Random Forest identifies both categories the best with Bagging at a close second
- 4 Boosting was expected to outperform every other model but ranked below both bagging and random forest
- 5 Rankings could change with different val/train splits