NC STATE UNIVERSITY

Enhanced Issue Lifetime Prediction Using Contextual Features

Matthew J. Martin
mjmartin@colby.edu

Goal

Create a model that can accurately predict issue lifetime from information only available at the time of issue creation.

Introduction

Why should anyone care?

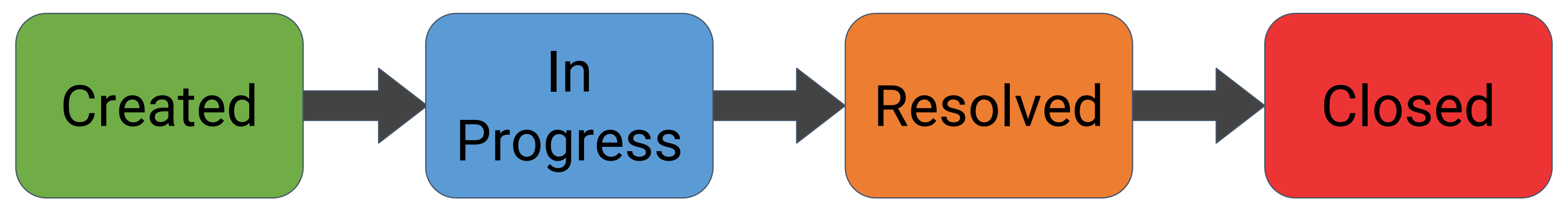
Business applications

- Useful to know if an issue can be closed within a day
- Divert more or fewer resources to issues based on their estimated completion time
- Provide more accurate information to clients and bosses

Academic interest

- Closely related to the idea of “cost to change” which is a tenet of many software development methodologies
- Highly researched in recent years
- No definitive model for issue lifetime prediction exists

Example Issue Lifetime



Background

The current state of the art in issue prediction is found in Kikas, et. al.’s paper from MSR 2016 [1]. Here are they key features of their work:

- Predictor was based heavily on project features and relied on information related to different points in an issue’s lifetime
- Used data from 4,000 GitHub projects for their models
- Used 28 different Random Forest classifiers
  - Each of the 28 different models was a binary prediction based on an Observation Point and a Horizon Point
- Obtained F1 measures ranging from 0.236 to 0.898

Methods

Feature selection

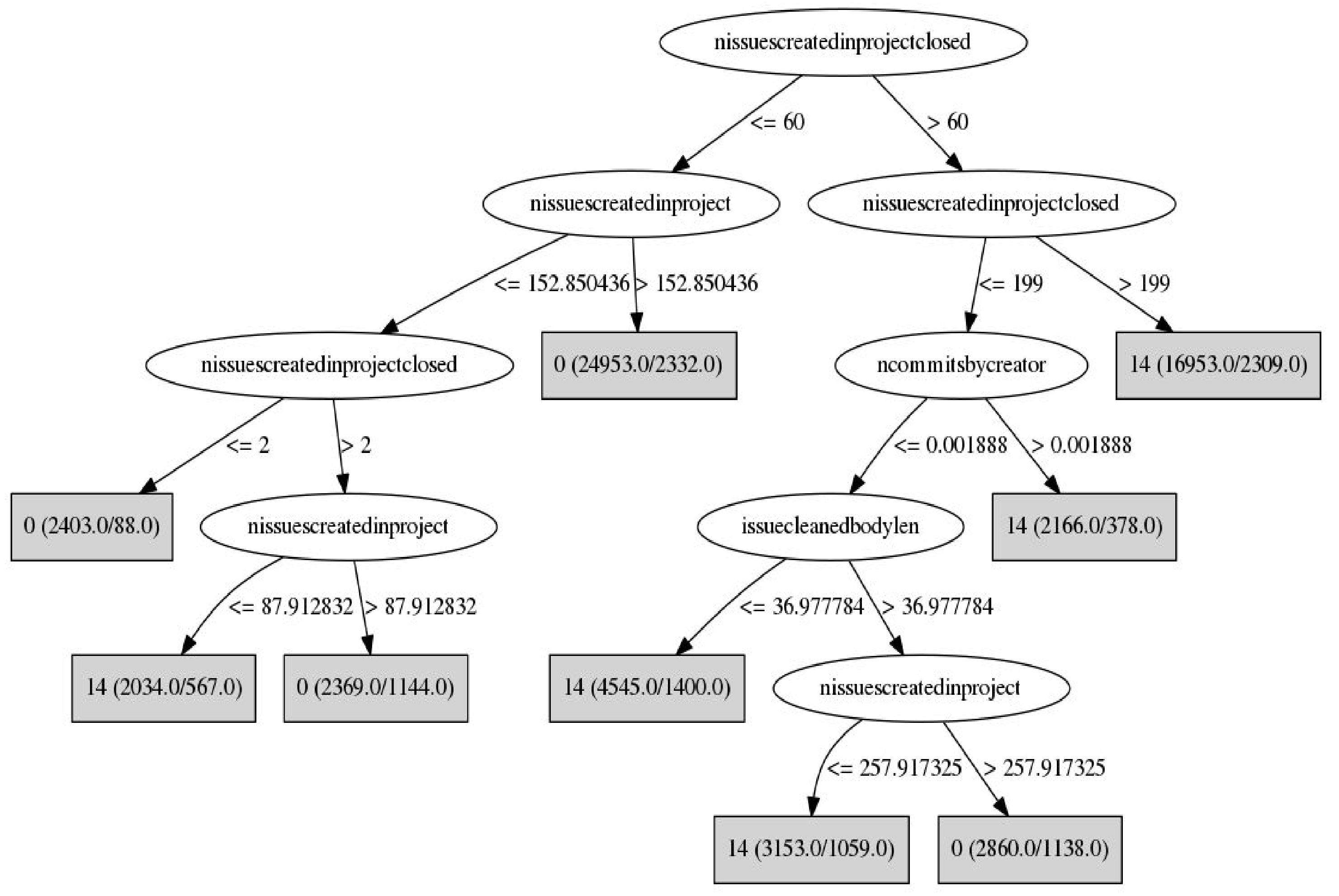
I began with the list of 21 features used in the Kikas paper, then eliminated any that relied on data from after issue creation. This brought the list to seven features:

- issueCleanedBodyLen
- nCommitsByCreator
- nCommitsInProject
- nIssuesByCreator
- nIssuesByCreatorClosed
- nIssuesCreatedInProject
- nIssuesCreatedInProjectClosed

Methods

Model creation

- Used a C4.5 decision tree
- Four models using GitHub/JIRA issue data from 10 projects
- Passed the data through some two pre-processors:
  - SpreadSubsample and SMOTE [2]
- Tuned each tree with an aggressive pruning parameter
  - Improve the readability of the tree
- Each tree predicted for whether an issue would be closed by a certain Horizon Point: 1 day, 14 days, 30 days, or 90 days



Results

Prediction Model	My F1 Measure	Kikas’ F1 Measure
1 day	0.601	0.437
14 days	0.813	0.659
30 days	0.834	0.715
90 days	0.838	0.781

- My F1 measures, on average, are 21.2% higher than those found in Kikas, et. al.’s work
- The same measure for the opposing class (predicting the issue will be closed *after* N days rather than before) renders even higher values, averaging 0.813 across my four models
- Also, from the visualization of the decision tree, business users can look to these values to see what factors indicate issues that are easier to close

Conclusions

Compare to Kikas, et. al.’s work

- I have fewer features
  - 7 instead of 21
- I have fewer models
  - 4 instead of 28
- I have simpler models
  - Random forests are hard to get actual insights out of, you can read my trees and observe cause and effect
- I have better accuracy
  - Average of 21.2% higher F1 measure

My model outperforms the Kikas model in nearly every way

Threats to Validity

- Only used 10 projects in my analysis
  - Could be the reason for my improved evaluation metrics
- Threw away open issues
  - May have skewed the importance of some features
- Difference in issue tracking styles between GitHub and JIRA

Future Work

The search for a perfect issue lifetime predictor is likely to continue for a long time. Future improvements to this work could include:

- Much larger data set
  - See if my higher performance metrics are the result of a small sample size
- Integration of open issues in the analysis
- Exploration of other learners

Notes

References

[1] Kikas, Riivo, Marlon Dumas, and Dietmar Pfahl. "Using dynamic and contextual features to predict issue lifetime in GitHub projects." InProceedings of the 13th International Workshop on Mining Software Repositories, pp. 291-302. ACM, 2016.

[2] Chawla, Nitesh V., Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. "SMOTE: synthetic minority over-sampling technique."*Journal of artificial intelligence research* 16 (2002): 321-357.

Acknowledgements

[1] Dr. Tim Menzies and Dr. Christopher Parnin, mentors

[2] This work was supported in part by funding from the NSF, and the DevOps Analytics team in the Cloud Division of IBM, Research Triangle Park, NC.