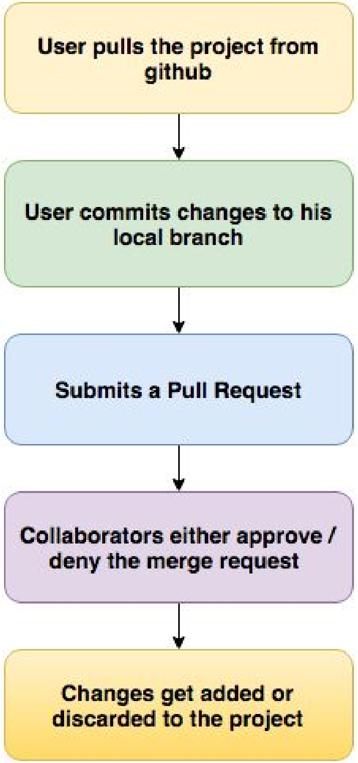
Analyzing Github Pull-Requests

ECS 260 Project, Arjun Bharadwaj and Christopher Lock

Introduction



User pulls the project from github

User commits changes to his local branch

Submits a Pull Request

Collaborators either approve / deny the merge request

Changes get added or discarded to the project

Features

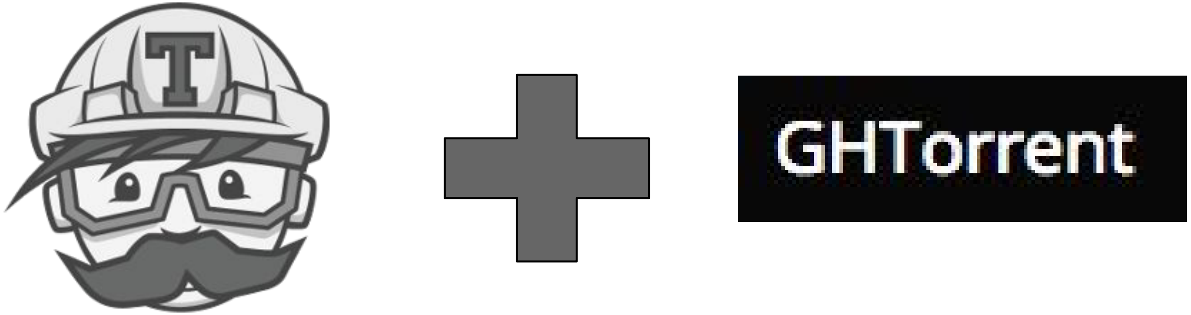
- Description Length
- Team size
- Size of project (in terms of sloc)
- Subscribers Count
- Open Issues Count
- Commit Comments
- Pull Request Comments
- Issue Comments
- Churn
- .....

Research Questions

1. What are *some features a developer can control* in order for his/her pull request to likely get accepted?

2. What is the importance of *tests* in pull- requests acceptance? (i.e are testing variables powerful indicators for the acceptance of pull requests?)

Data Gathering



- Projects which had more than 10 Pull Requests and part of *2016*
- 47 Open Source Projects including popular ones such as *IPython,Pandas,Scipy, etc.*
- ~36,000 Pull Requests
- 31 Features

Experimentation & Results

- Phase 1: Preprocessing of Data
- Phase 2: Feature Selection
- Phase 3: ML Models on Train and Test Set
- Phase 4: Evaluation
- Phase 5: Conclusion

Phase 1 - Feature Selection

- Trees Selection - Ranking by Feature Importance
- Univariate Selection - Chi Squared Test
- Recursive Feature Elimination

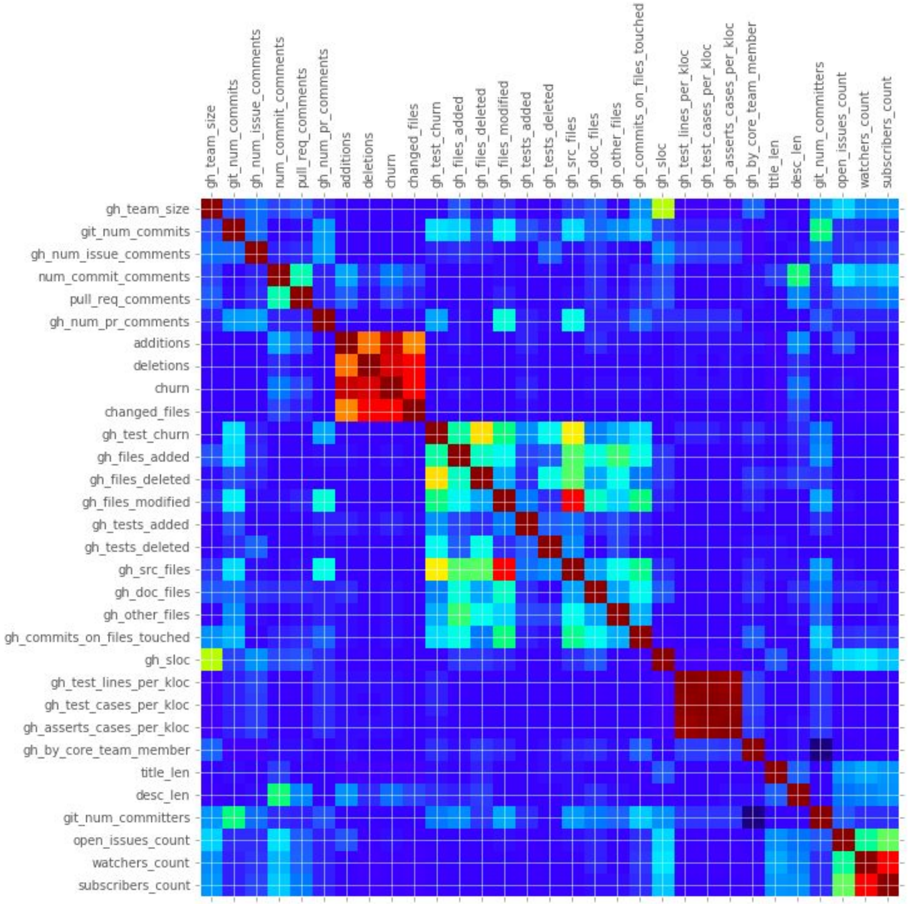
Top Features

- 'num_commit_comments',
- 'desc_len',
- 'churn',
- 'changed_files',
- 'gh_sloc',
- 'open_issues_count',
- 'subscribers_count',
- 'gh_team_size’
- 'pull_req_comments',
- 'gh_test_lines_per_kloc’

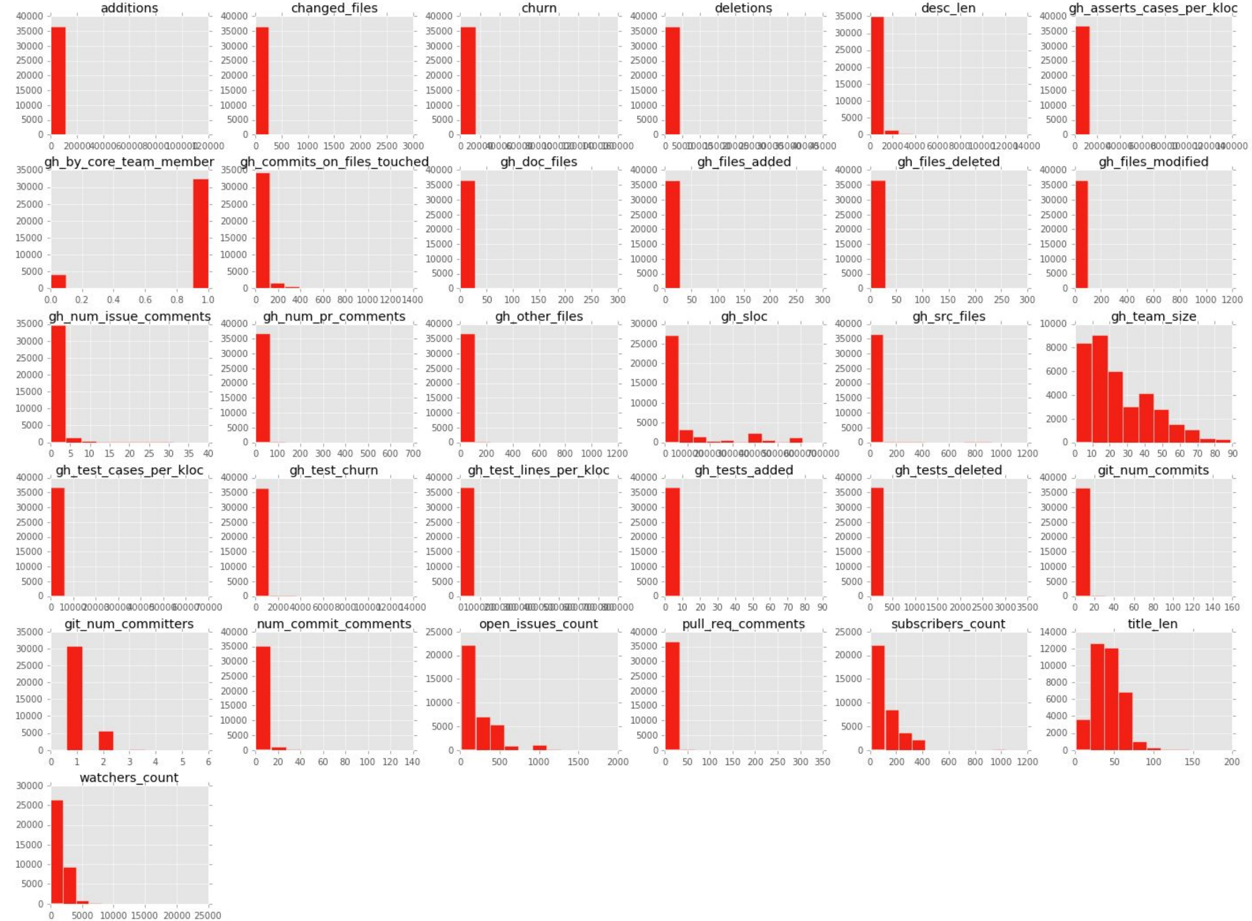
Why Diff Results?

- Correlation between Variables
- Distribution of Data

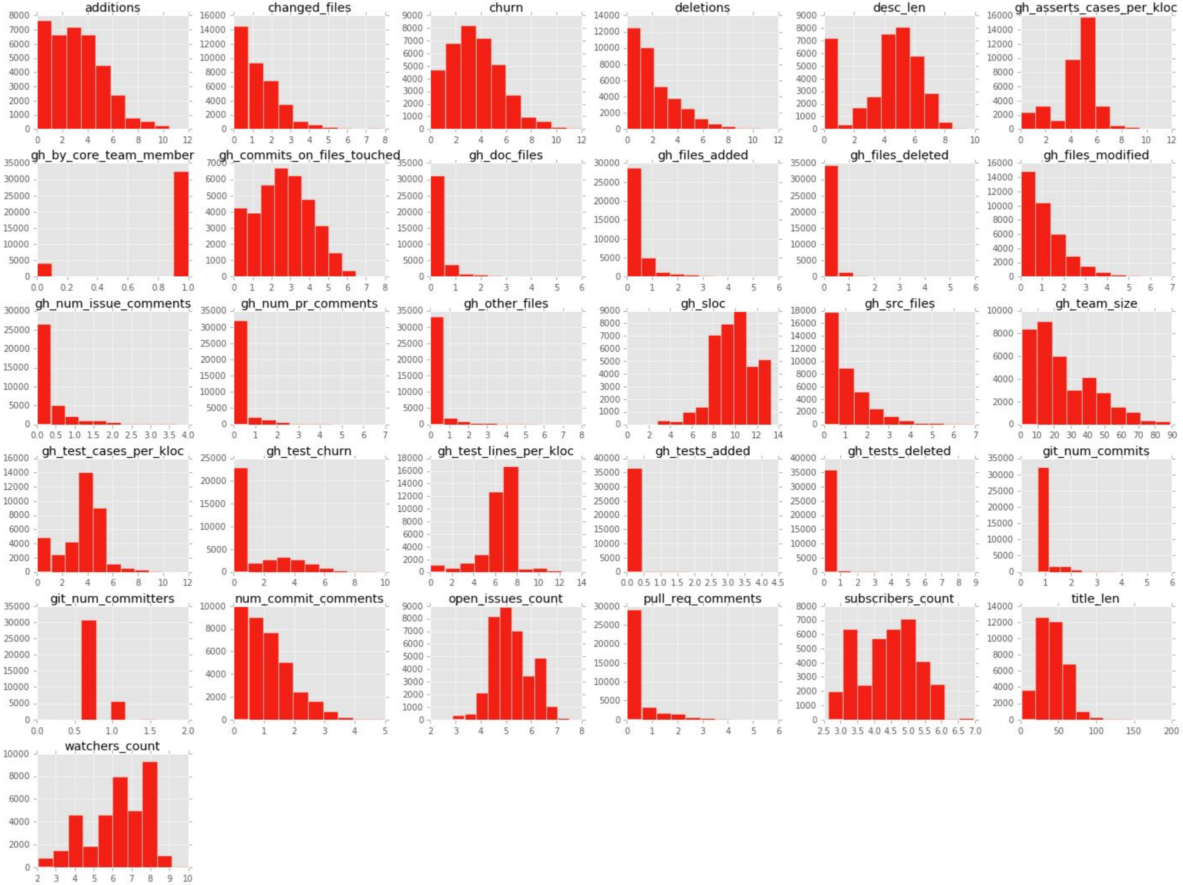
Correlation between Variables



Distribution of Data within Each Feature



Distribution of Data within Each Feature (Logged)



Modelling Phase

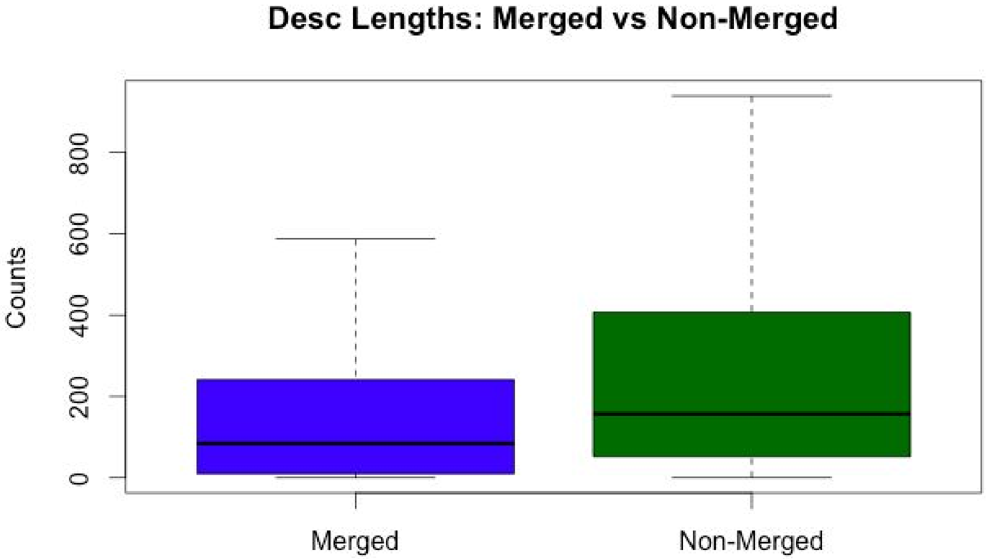
- Supervised Learning (Labels are T/F for merged or not)
- Chose 10 Features for final modelling phase
- Time-DEPENDENT split into training and test sets
- First *three months* in order to predict the rest of them gave *best* results.

Evaluation of Models

Model	**Accuracy**
Predicting at Random	69.87%
Logistic Regression	67.07%
k-NN	67.55%
Random Forests	71.81%
Adaboost	71.98%
SVM	66.36%

Boxplots of Merged vs Unmerged

One example: Description Lengths of M vs NM



Threads to Validity

- Skewed Samples
- Overfitting. (PCA helps reduce this risk)
- Models may reveal a true correlation but may not necessarily imply the correct cause of the relationship. (Example: PR’s on a Friday)

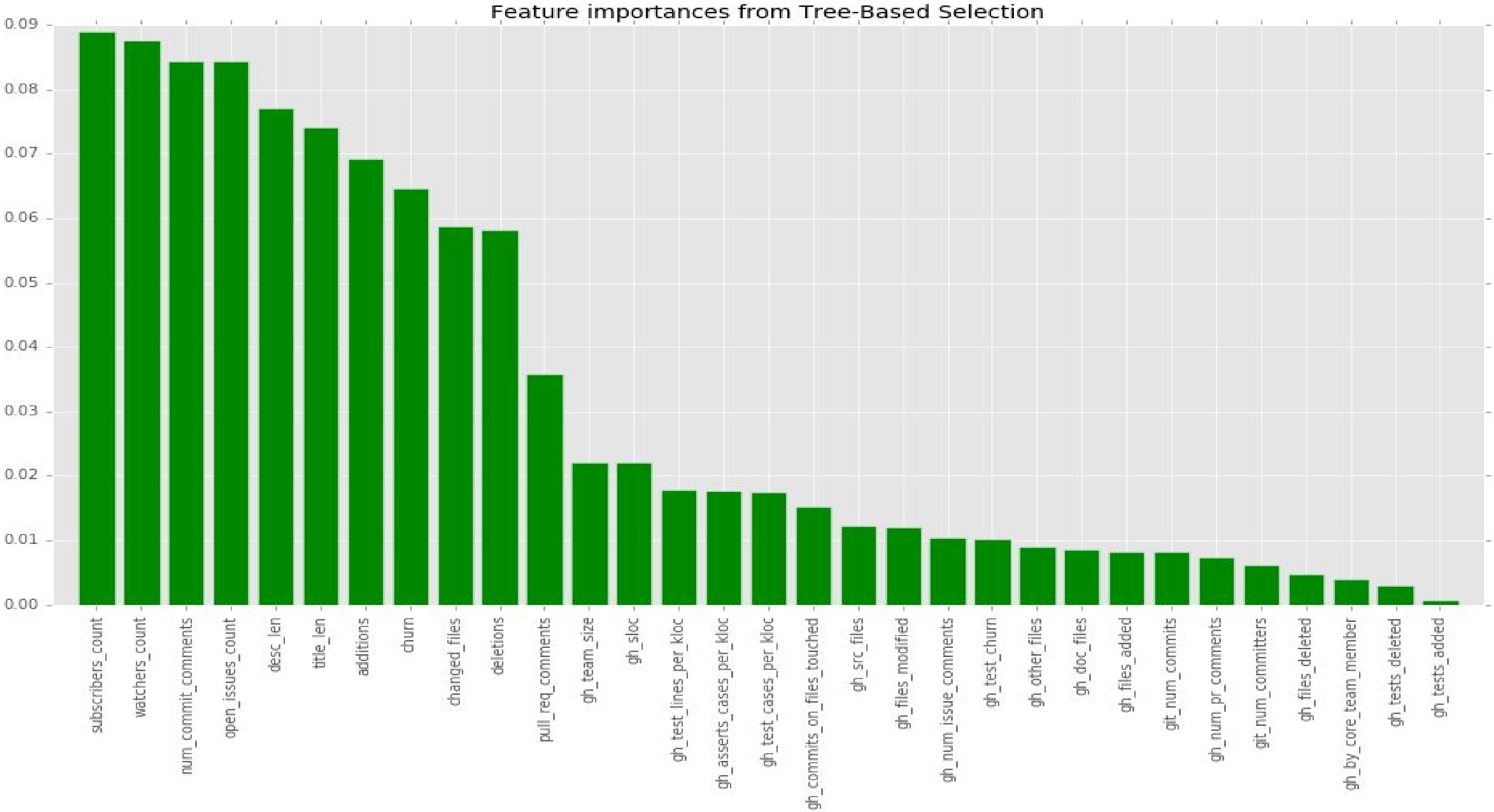
Future Work

- Larger datasets
- Using heuristics of software engineering
- Textual Analysis of Commits and Pull Request Messages

Conclusion

- Pull Requests Prediction can be really hard and unpredictable!
- Most likely, developers can control desc len, churn, etc. to make pull req likely acceptable.
- Even though tests are important, model shows churn & project characteristics are more important indicators.

Trees Selection



Univariate Selection - Chi^2 Selection

1) churn

2) additions

3) gh_sloc

4) deletions

5) watchers_count

6) desc_len

7) changed_files

8) open_issues_count

9) subscribers_count

10) gh_test_lines_per_kloc

11) num_commit_comments

12) gh_test_churn

13) gh_commits_on_files_touched

14) title_len

15) gh_doc_files

Recursive Feature Elimination

```
('gh_team_size', 1),
('git_num_commits', 1),
('gh_num_issue_comments', 1),
('num_commit_comments', 1),
('pull_req_comments', 1),
('gh_tests_added', 1),
('gh_doc_files', 1),
 ('gh_by_core_team_member', 1),
 ('title_len', 1),
 ('git_num_committers', 1),
('subscribers_count', 2),
 ('changed_files', 3),
 ('gh_files_modified', 4),
 ('gh_src_files', 5),
```