Change-Aware Build Prediction Model for Stall Avoidance in Continuous Integration

{ Foyzul Hassan and Xiaoyin Wang}

UTSA Computer Science

Abstract

Continuous Integration(CI) is a widely used development practice where developers integrate their work after submitting code changes at central repository. CI servers usually monitor central repository for code change submission and automatically build software with changed code, perform unit testing, integration testing and provide test summary report. If build or test fails developers fix those issues and submit the code changes. Continuous submission of code modification by developers and build latency time creates stalls at CI server build pipeline and hence developers have to wait long time to get build outcome. In this paper, we proposed build prediction model that uses TravisTorrent data set with build error log clustering and AST level code change modification data to predict whether a build will be successful or not without attempting actual build so that developer can get early build outcome result. With the proposed model we can predict build outcome with an average F-Measure over 87% on all three build systems (Ant, Maven, Gradle) under the cross-project prediction scenario.

Contributions

We focused on Java projects using Ant,Maven and Gradle build systems because they are supported by the TravisTorrent data set. Our evaluation results show that our model can achieve an average F-Measure of 87% on all three build system for cross-project build-outcome prediction, which is a very challenging but more realistic usage scenario. Our work makes the following main contributions.

1. A statistic study of CI build status and time on the TravisTorrent data set.
2. A build-outcome prediction model based on combined features of the build-instance meta data and code difference information of the commit.
3. A large-scale evaluation of our project with both scenarios of cross-validation and cross-project prediction on the TravisTorrent data set with more than 250,000 build instances.

Data Set

We studied TravisTorrent data set on Oct 27,2016. This data set includes 402 Java projects with data for 256,055 build instances. Among this 256,055 build instances, the Ant based build system is used for 104,417 cases, while Maven and Gradle based build systems are used for 104,876 and 44,056 instances, respectively.

Time Required for Building

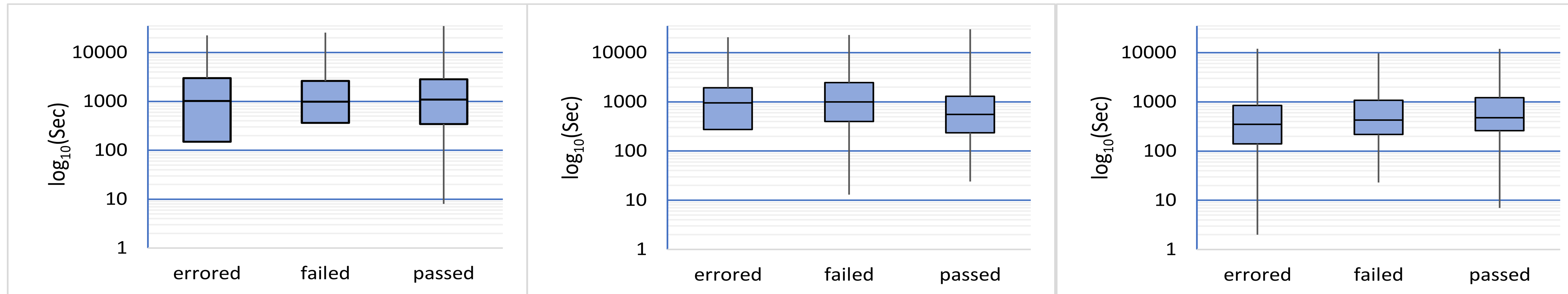

fig1: Ant Build Execution Time   fig2: Maven Build Execution Time   fig3: Gradle Build Execution Time

Figure 1: Build Time

Median build execution time for errored status with Ant, Maven and Gradle build tool are 1,019, 955 and 352 seconds respectively. For failed build status, the median execution time for Ant, Maven and Gradle are 981, 998, and 426 seconds, respectively. Passed build execution time for Ant, Maven and Gradle are 1,090, 558 and 477 second respectively.

Commit Time Interval

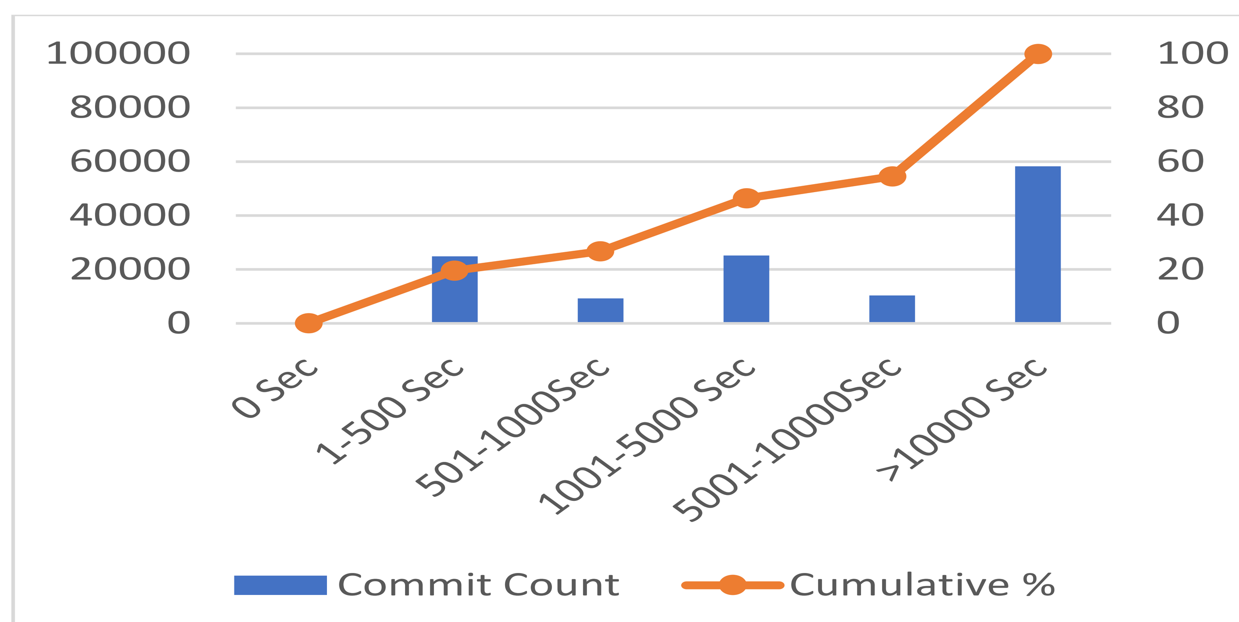

Figure 2: Commit Time Interval

19.54% code commits occurs within 500 seconds, 26.74% of commits occurs within 1,000 seconds, and 54.53% of commit occurs within 10,000 seconds.

Build Status Change Frequency

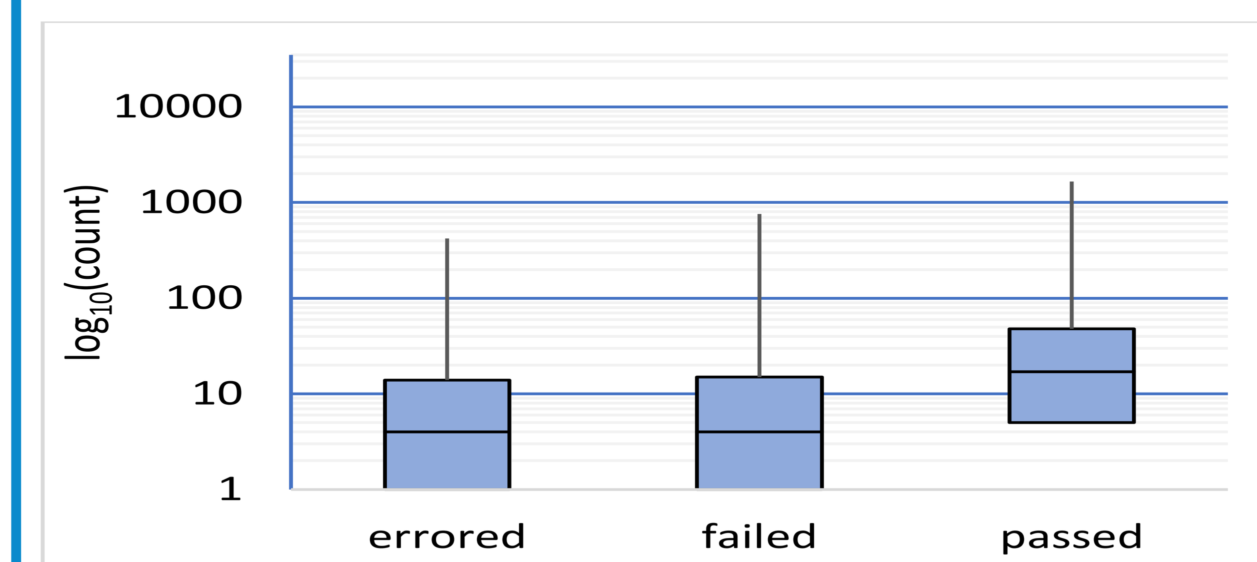

Figure 3: Build Status Statistics on Consecutive Build Sequence

For errored and failed build status, build outcome remains unchanged with median of four build instances. For maximum case, 422 consecutive build was errored, while 760 consecutive build was failed.

Overview of Build Prediction Model

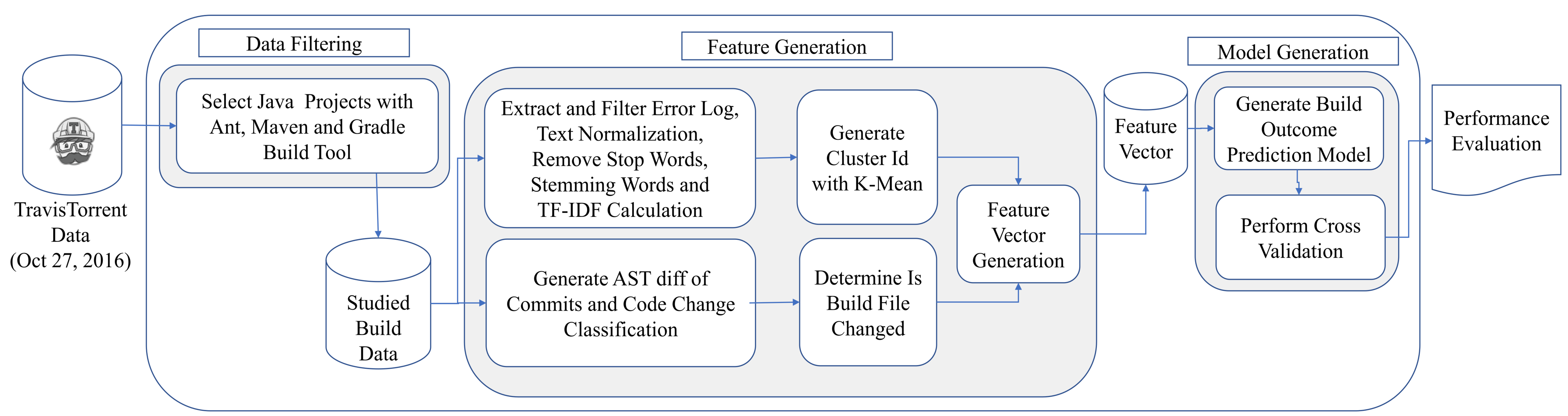

Figure 4: Overview of Proposed System

Feature Selection

Feature Name	Entropy
prev_bl_cluster	0.4812
prev_tr_status	0.4696
gh_team_size	0.0313
gh_src_churn	0.0078
prev_gh_src_churn	0.0059
cmt_buildfilechangecount	0.0058
cmt_importchangecount	0.0057
cmt_methodbodychangecount	0.0056
gh_test_churn	0.0056
prev_gh_test_churn	0.0055

Table 1: InfoGainAttributeEval Entropy for Top Ten Features

For feature selection we applied Information Gain Attribute Evaluation on Ant, Maven and Gradle data set and select those attributes having average entropy>0.005.

Performance Evaluation

Build Tool	Precision	Recall	F-Measure
Ant	0.938	0.938	0.938
Maven	0.937	0.937	0.937
Gradle	0.921	0.922	0.921

Table 2: Performance Evaluation of Build Prediction Model

Ant and Maven average Precision, Recall and F-Measure are above 0.93.While for Gradle, average precision, recall and F-Measure is above 0.92.

Build Tool	Precision	Recall	F-Measure
Ant	0.914	0.914	0.914
Maven	0.909	0.910	0.909
Gradle	0.872	0.873	0.872

Table 3: Cross Project Performance Evaluation

For Cross Project evaluation, the effectiveness of build prediction models drop a bit, but for Ant and Maven it can still predict build outcome with over 0.90 F-Measure. For Gradle, our build prediction model can predict build outcome with over 0.87 F-Measure.

Conclusion

We propose the scalable approach for predicting build outcome in CI environment with evaluation on large scale data. In future, we are planning to use build configuration change type as feature for build outcome prediction model. Apart from that, different learning algorithms can be used for better accuracy.