Ryerson University
Data Science & Analytics Graduate Studies

2018-2019 M.Sc. in Data Science and Analytics

The Use of Recommender Systems for Defect Rediscoveries

Kristie House-Senapati **Supervisor: Dr. Andriy Miranskyy**

Objective

The objective of this research is to build a predictive model for future defect rediscoveries. The aim is to find defects that a customer may discover in the future and eliminate those defects before they are actually found by the customer.

Background

Defects in software arise during the development stage. Once testing of the software is complete, if defects are not found they are shipped with the final product and are left to be found by the end user. When a customer finds a defect in the software they submit a bug report to the software company with detailed information regarding the defect. Once the report is received it will be sent to a developer to fix the problem.

Given resource constraints and the large volume of submitted bug reports it may not be possible to fix each and every defect that is found by the customer. Therefore, our approach is to find a list of defects that may be rediscovered in the future by the customer and focus on that list of defects.

Methodology

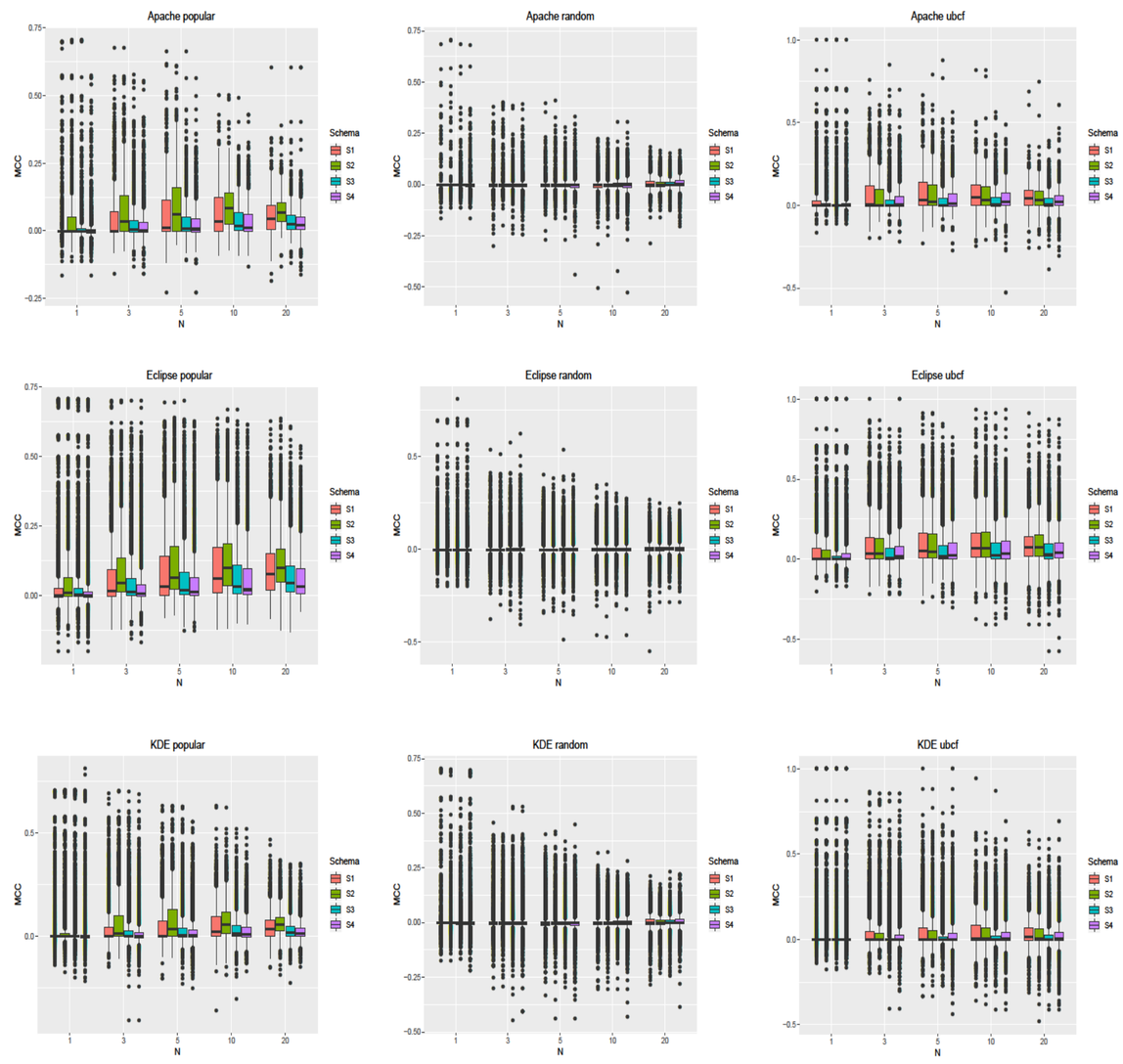
1. Extract the data from Apache, Eclipse, and KDE open source software projects.
2. Split the data into train and test sets. Four different splitting schema were used for the split.
3. For every split of the data:
   - Further split the test set into a known and unknown component.
   - Run one of the popular, random, user-based collaborative filtering, funk SVD, apriori, or eclat algorithms from the R packages recommenderlab and arules. Multiple top-N lists will be run for each algorithm.
   - Record the following performance metrics: FPR, TPR, recall, precision, and MCC (Matthew Correlation Coefficient).
4. Compare the MCC for each project, algorithm, splitting schema to determine which is the best one.

Splitting Schema:

- Schema 1: accumulate historical, train immediate future
- Schema 2: train on recent historical, test on immediate future
- Schema 3: train on recent historical, test on subsequent future
- Schema 4: accumulate historical, test on subsequent future

Results

- For the Apache project the popular algorithm performs the best
- For the Eclipse project the UBCF algorithm performs the best
- Overall, schema 2 performs the best
- The random algorithm performs poorly
- The Funk SVD, Apriori, and Eclat algorithms do not perform well on highly sparse data



Conclusions

The idea of our research was to build a recommender algorithm that can predict which defects a user may find in the future, so that a fix can be put into place before the defect is discovered.

We believe that this research may be used in order to identify a list of defects that a client may discover in the future. This research may also help researchers to gather insights about detecting defect rediscoveries in the future.