BELLWETHER 2.0

By:
Aayushi Agrawal
Akshay Ravichandran
Yudong RAO

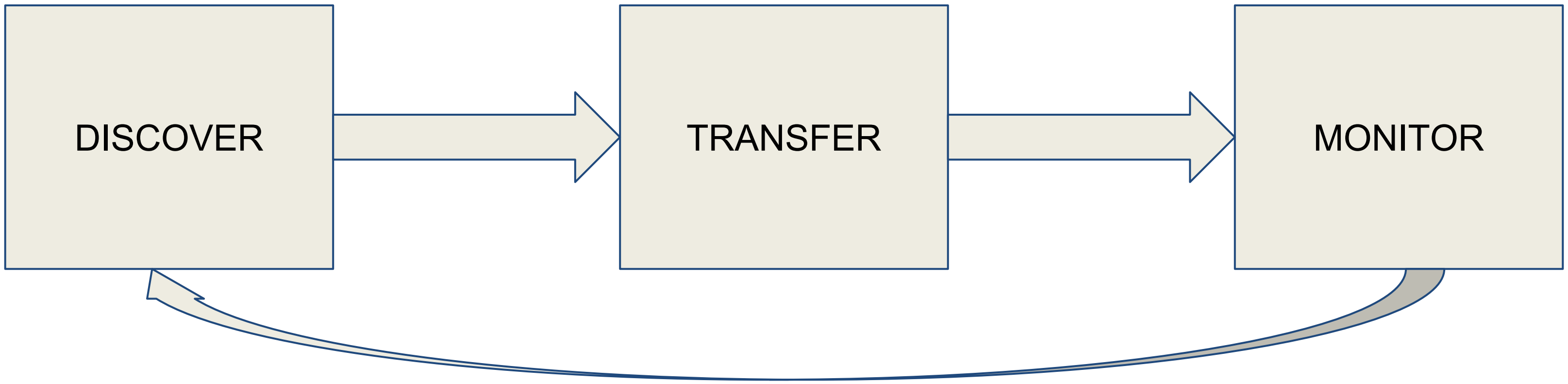
INTRODUCTION

- What is a ‘*bellwether*’?

  A: Given N projects from a community of projects, the ‘*bellwether*’ is that project whose data yields the best predictions on all others

ORIGINAL STUDY

- Conducted by Rahul Krishna and Prof. Menzies[5]
- Introduced the ***bellwether method:***



MOTIVATION

- Bellwethers appear prevalent in SE data sets - *defect prediction, code smell, effort estimation*
- Useful baseline method for transfer learning
- Bellwethers mitigate conclusion instability

ISSUE ADDRESSED

- The *Discover* phase is slow

```
def discover(datasets):
"Identify Bellwether Datasets"
  for data_1, data_2 in datasets:
    def train(data_1):
      "Construct quality predictor"
      return predictor
    def predict(data_1):
      "Predict for quality"
      return predictions
    def score(data_1, data_2):
      "Return accuracy of Prediction"
      return accuracy(train(data_1),\
      test(data_2))

  "Return data with best prediction score"
```

RESEARCH QUESTION

RQ: Can the time taken to find the bellwether dataset be reduced?

INSPIRATION - RFE[6]

Algorithm 1: Recursive feature elimination

```
Tune/train the model on the training set using all predictors
Calculate model performance
Calculate variable importance or rankings
for Each subset size S_i, i = 1 ... S do
    Keep the S_i most important variables
    [Optional] Pre–process the data
    Tune/train the model on the training set using S_i predictors
    Calculate model performance
    [Optional] Recalculate the rankings for each predictor
end
Calculate the performance profile over the S_i
Determine the appropriate number of predictors
Use the model corresponding to the optimal S_i
```

INSPIRATION - BEETLE PAPER[4]

```
def FindBellwether(sources, step_size, budget, thres, ↩
        lives):
    while lives or cost > budget:
      "Sample configurations"
      sampled = list()
      for source in sources:
        "Sample step_size number of configurations"
        sampled += source.sample(step_size)
      "Get cost"
      cost = get_cost(sampled)
      "Evaluate pair−wise performances"
      perf = get_perf(sampled)
      "Remove non−bellwether environments"
      sources=remove_non_bellwethers(sources, perf, ↩
            thres)
      "Loose life if no sources are removed"
      if prev == len(sources): lives −= 1
      "Return a bellwether"
    return sources[argmin(perf)]
```

OUR APPROACH

```
discoverBellwether(datasets, n_rep):
    repeat n_rep times:
        Initialize lives, x, step_size
        while lives > 0:
            for dataset_1, dataset_2 in datasets:
                data1 = sample x% of dataset_1
                data2 = sample x% of dataset_2
                predictor = train(data1)
                G_score[dataset_1, dataset_2] = score(test(predictor, data2))
            rank and eliminate datasets based on median g-score
            if no dataset to eliminate:
                lives  = lives - 1
            x = x + step_size
        for each dataset not eliminated:
            increment its wins
    return dataset with most wins
```

TWO VERSIONS FOR ELIMINATION

- Throw away bottom ⅓ of datasets sorted with median G-Score in each iteration
- Throw away datasets with median G-Score lower than threshold in each iteration

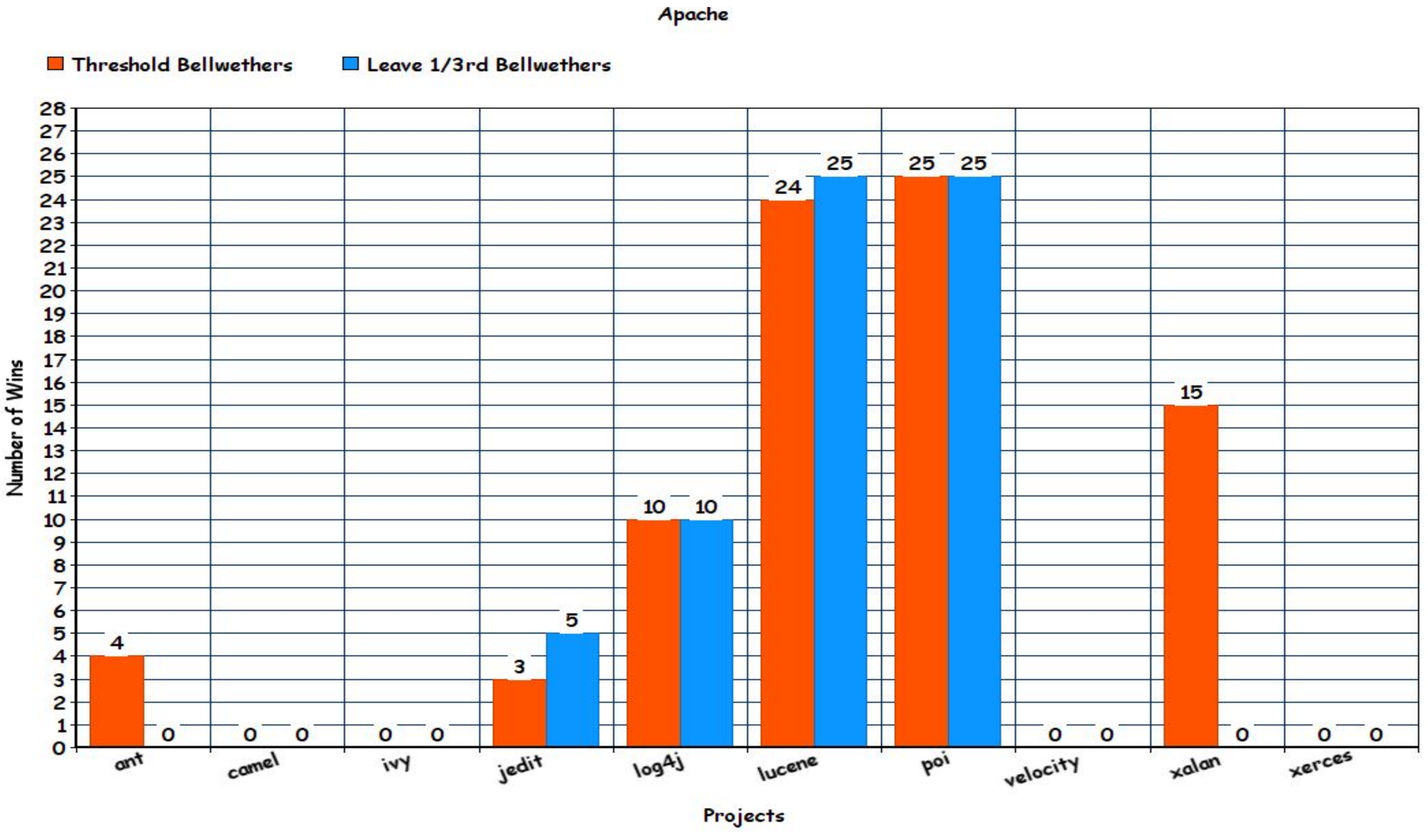
DATA SET DESCRIPTION

- The AEEEM dataset was gathered by D'Amborse et al. [1], it contains 61 metrics, 5371 instances, 893 defective instances.
- The RELINK community data was obtained from work by Wu et al. [2] , it contains 26 metrics, 649 total instances, 238 defective instances
- The Apache community data was gathered by Jureczko et al. [3]. This dataset contains records of the number of known defects for each class using a post-release bug tracking system. The classes are described in terms of 20 OO metrics, including CK metrics and McCabes complexity metrics.

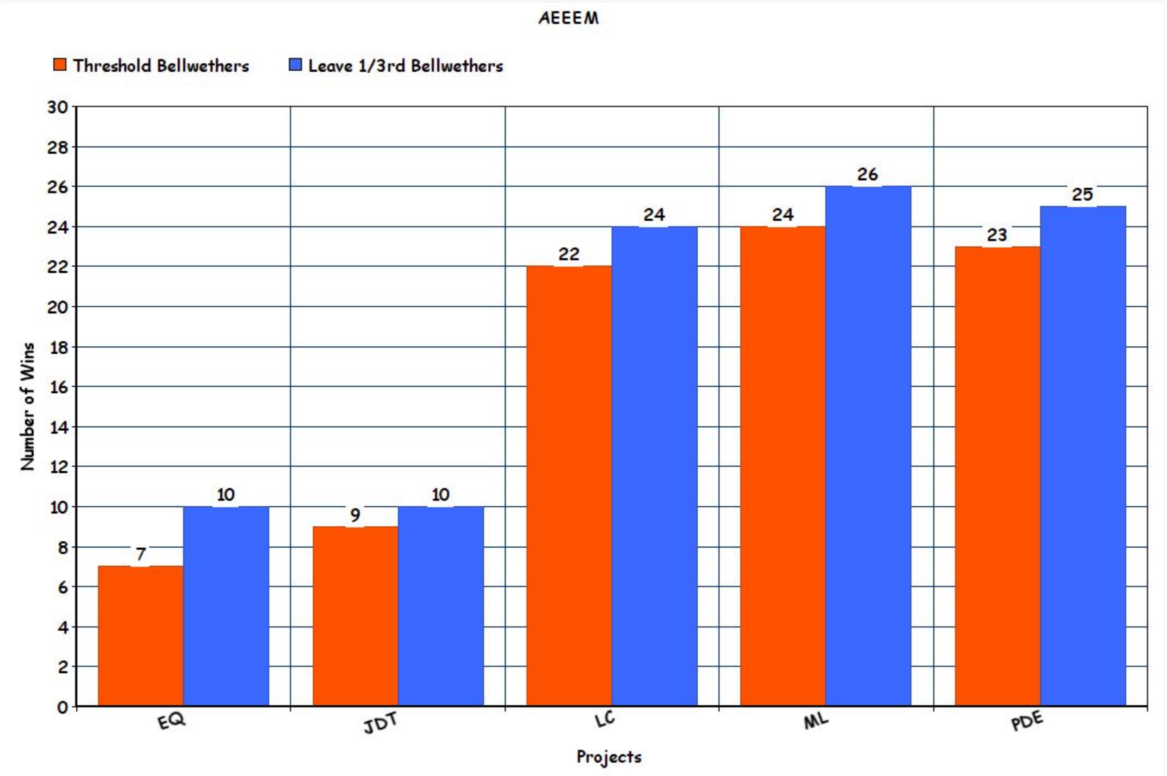
EXPERIMENTAL SETUP

- Threshold value = 52
- sample size = 0.25, incremented by 0.05
- Lives = 10

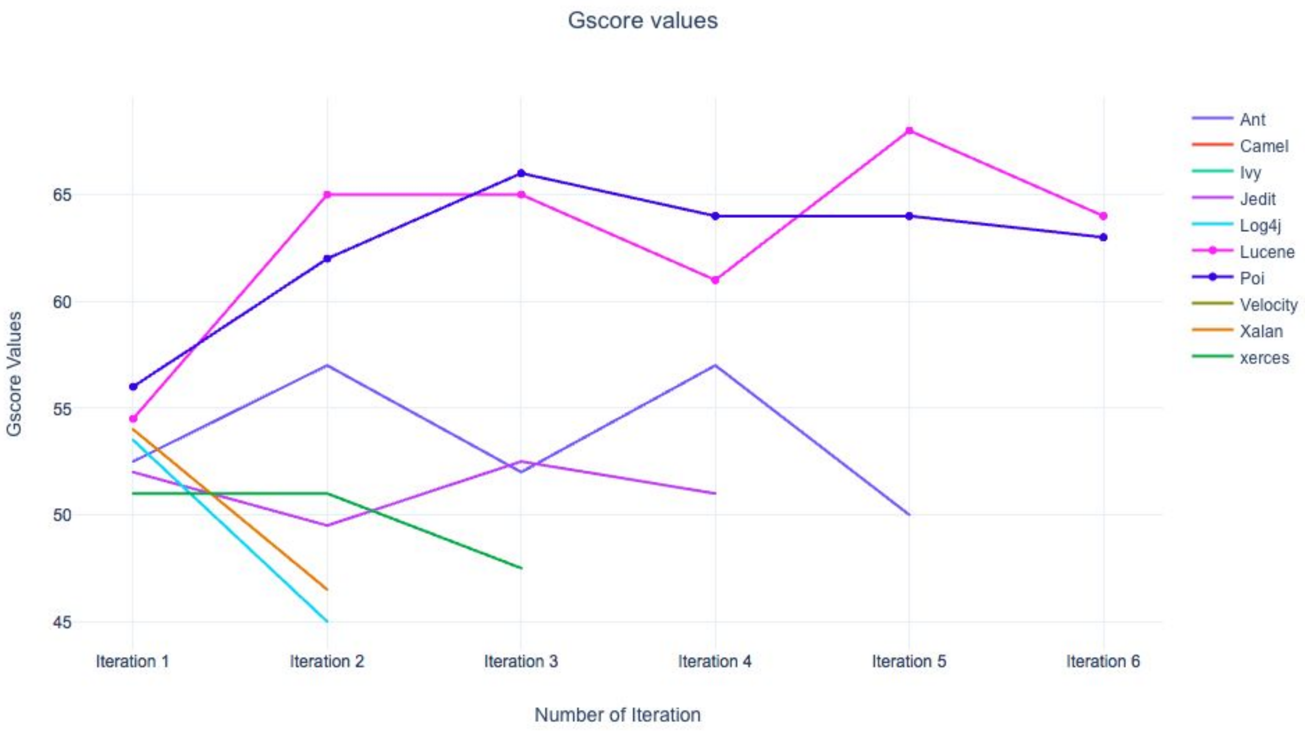
RESULTS


Apache
Threshold Bellwethers
Leave 1/3rd Bellwethers
Number of Wins
Projects
ant
camel
ivy
jedit
log4j
lucene
poi
velocity
xalan
xerces
4
0
0
0
0
0
3
5
10
10
24
25
25
25
0
0
15
0
0
0

RESULTS


AEEEM
Threshold Bellwethers
Leave 1/3rd Bellwethers
Number of Wins
Projects
EQ
JDT
LC
ML
PDE
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
7
10
9
10
22
24
24
26
23
25

G-SCORES


Gscore values
Gscore Values
65
60
55
50
45
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Number of Iteration
Ant
Camel
Ivy
Jedit
Log4j
Lucene
Poi
Velocity
Xalan
xerces

FUTURE WORK

- Predicting the threshold values for optimal bellwethers
- Analysis of trade-off between different parameters
- Combining our method with Hoeffding Racing

References

[1] D'Ambros, Marco, Michele Lanza, and Romain Robbes. "Evaluating defect prediction approaches: a benchmark and an extensive comparison." *Empirical Software Engineering 17*. 4-5 (2012): 531-577.

[2] Bavota, Gabriele. "Mining unstructured data in software repositories: current and future trends." *Software Analysis, Evolution, and Reengineering (SANER), 2016 IEEE 23rd International Conference on*. Vol. 5. IEEE, 2016.

[3] Jureczko, Marian, and Lech Madeyski. "Towards identifying software project clusters with regard to defect prediction." *Proceedings of the 6th International Conference on Predictive Models in Software Engineering*. ACM, 2010.

[4] Nair, Vivek, et al. "Transfer Learning with Bellwethers to find Good Configurations." *arXiv preprint arXiv:1803.03900* (2018).

[5] Krishna, Rahul, and Tim Menzies. "Bellwethers: A Baseline Method For Transfer Learning." *IEEE Transactions on Software Engineering* (2018).

[6] Max, "Recursive Feature Elimation" https://topepo.github.io/caret/recursive-feature-elimination.html#search

THANK YOU!