Revisiting Unsupervised Learning for Defect Prediction

Wei Fu  
wfu@ncsu.edu  
http://weifu.us

Tim Menzies  
tim.menzies@gmail.com  
http://menzies.us

Find these slides at: http://tiny.cc/unsup

Sep, 2017
On the market

(expected graduation: May, 2018)

Research Areas:
- Machine learning
- SBSE
- Evolutionary algorithms
- Hyper-parameter tuning

Tim Menzies
(advisor)

Publications:
- FSE: 2
- ASE: 1
- TSE: 1
- IST: 1
- Under Review: 2

weifu.us
Yang et al. reported: unsupervised predictors outperformed supervised predictors for effort-aware Just-In-Time defect prediction.

Implied: dramatic simplification of a seemingly complex task.

We Repeated and reflect Yang et al. results.
General Lessons

Open Science
Yang, Yibiao, et al. “Effort-aware just-in-time defect prediction: simple unsupervised models could be better than supervised models.” FSE’16

Yang’s Unsup Methods in Precision

Red is bad!
arXiv.org Effect

Nanjing: Mar 11, 2016
Raleigh: Feb 28, 2017
Singapore: Apr 6, 2017
Hangzhou: Apr 7, 2017
Nanjing: Apr 7, 2017
Raleigh: Oct, 2017

[Yang et al. FSE’16] [Fu et al. FSE’17] [Huang et al. ICSME’17] [Liu et al. ESEM’17] [Fu et al.arXiv’17]

Dr. Zhou
Dr. Menzies
Dr. Xia & Dr. Lo
Dr. Zhou
Dr. Menzies
<table>
<thead>
<tr>
<th>Publication</th>
<th>h5-index</th>
<th>h5-median</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. International Conference on Software Engineering</td>
<td>68</td>
<td>91</td>
</tr>
<tr>
<td>2. IEEE Transactions on Software Engineering</td>
<td>52</td>
<td>80</td>
</tr>
<tr>
<td>3. Journal of Systems and Software</td>
<td>51</td>
<td>73</td>
</tr>
<tr>
<td>4. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)</td>
<td>50</td>
<td>67</td>
</tr>
<tr>
<td>5. ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)</td>
<td>46</td>
<td>68</td>
</tr>
<tr>
<td>6. Information and Software Technology</td>
<td>45</td>
<td>68</td>
</tr>
<tr>
<td>7. ACM SIGSOFT International Symposium</td>
<td>43</td>
<td>64</td>
</tr>
<tr>
<td>8. Mining Software Repositories</td>
<td>39</td>
<td>57</td>
</tr>
<tr>
<td>9. IEEE Software</td>
<td>38</td>
<td>54</td>
</tr>
<tr>
<td>10. ACM SIGPLAN Symposium on Principles &amp; Practice of Parallel Programming (PPOPP)</td>
<td>37</td>
<td>55</td>
</tr>
<tr>
<td>11. ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)</td>
<td>37</td>
<td>51</td>
</tr>
<tr>
<td>12. Empirical Software Engineering</td>
<td>37</td>
<td>48</td>
</tr>
<tr>
<td>15. International Symposium on Software Testing and Analysis</td>
<td>31</td>
<td>50</td>
</tr>
<tr>
<td>16. IEEE/ACM International Conference on Automated Software Engineering (ASE)</td>
<td>31</td>
<td>44</td>
</tr>
<tr>
<td>17. Software &amp; Systems Modeling</td>
<td>31</td>
<td>43</td>
</tr>
<tr>
<td>18. IEEE International Conference on Software Maintenance</td>
<td>29</td>
<td>41</td>
</tr>
<tr>
<td>19. IEEE International Conference on Software Testing, Verification and Validation (ICST)</td>
<td>29</td>
<td>41</td>
</tr>
<tr>
<td>20. arXiv Programming Languages (cs.PL)</td>
<td>29</td>
<td>40</td>
</tr>
<tr>
<td>X. ACM Transactions on Software Engineering and Methodology( TOSEM)</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
We Promote Open Science

A Large-Scale Empirical Study of Just-in-Time Quality Assurance

Yasutaka Kamei, Member, IEEE, Emad Shihab, Bram Adams, Member, IEEE,
Ahmed E. Hassan, Member, IEEE, Audris Mockus, Member, IEEE, Anand Sinha, and
Naoyasu Ubayashi, Member, IEEE

Abstract—Defect prediction models are a well-known technique for identifying defect-prone files or packages such that practitioners can allocate their quality assurance efforts (e.g., testing and code reviews). However, once the critical files or packages have been identified, developers still need to spend considerable time drilling down to the functions or even code snippets that should be reviewed or tested. This makes the approach too time consuming and impractical for large software systems. Instead, we consider defect prediction models that focus on identifying defect-prone (“risky”) software changes instead of files or packages. We refer to this type of quality assurance activity as “Just-In-Time Quality Assurance,” because developers can review and test these risky changes while they are still fresh in their minds (i.e., at check-in time) characteristics of a software change, such as the source and five commercial projects from multiple defect with an average accuracy of 68 percent and review changes, we find that using only 50 percent defect in the first three days of the defect prediction, and these results are shown in the following.

Index Terms—Maintenance, software metrics, n

Effort-Aware Just-in-Time Defect Prediction: Simple Unsupervised Models Could Be Better Than Supervised Models

Yibiao Yang, Yuming Zhou,' Jinping Liu,' Yangyang Zhao,' Hongmin Lu,' Lei Xu,' Baswen Xu,' and Hareton Leung

1Department of Computer Science and Technology, Nanjing University, China
2Department of Computing, Hong Kong Polytechnic University, Hong Kong, China

ABSTRACT

Unsupervised models do not require the defect data to build the prediction models and hence incur a low building cost and gain a wide application range. Consequently, it would be more desirable for practitioners to apply unsupervised models in effort-aware just-in-time (JIT) defect prediction if they can predict defect-inducing changes well. However, little is currently known on their prediction effectiveness in this context. We aim to investigate the predictive power of simple unsupervised models in effort-aware JIT defect prediction, especially compared with the state-of-the-art supervised models in the recent literature. We first use the most commonly used change metrics to build simple unsupervised models. Then, we compare these unsupervised models with the state-of-the-art supervised models under cross-validation, time-wise cross-validation, and across-project prediction setting. We present the average results of participants. The results show that simple unsupervised models perform better than the state-of-the-art unsupervised models in effort-aware JIT defect prediction.

Our code & data at:

https://github.com/WeiFoo/RevisitUnsupervised

Thanks!
Methods
Yang et al’s Method: Core Idea

Koru et al [Koru 2010] suggest that

“Smaller modules are proportionally more defect-prone and hence should be inspected first!”

Area under the curve of Recall/LOC

- **Bad** = modules predicted defective
- **Other** = all other modules
- Sort each increasing by LOC
- Track the recall
Yang et al’s Unsupervised Method

Build 12 unsupervised models, on testing data:

<table>
<thead>
<tr>
<th>NF</th>
<th>NS</th>
<th>LT</th>
<th>FIX</th>
<th>ND</th>
<th>NDEV</th>
<th>EXP</th>
<th>REXP</th>
<th>SEXP</th>
<th>NUC</th>
<th>AGE</th>
<th>Entropy</th>
<th>LOC</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>11</td>
<td>1</td>
<td>1</td>
<td>23</td>
<td>2</td>
<td>12</td>
<td>4</td>
<td>2</td>
<td>8</td>
<td>0.3</td>
<td>32</td>
<td>?</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>24</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>13</td>
<td>3</td>
<td>1</td>
<td>6</td>
<td>0.4</td>
<td>42</td>
<td>?</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>89</td>
<td>0</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>0.6</td>
<td>18</td>
<td>?</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>34</td>
<td>0</td>
<td>3</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>103</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>537</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>22</td>
<td>9</td>
<td>7</td>
<td>12</td>
<td>0.3</td>
<td>20</td>
<td>?</td>
</tr>
</tbody>
</table>

10% effort

20% effort

Predicted as “Defective”

Yang et al report:

*Aggregating performance over all projects, many simple unsupervised models perform better than supervised models.*
OneWay
OneWay is not “the Way”

OneWay:

“The alternative way, maybe not the best way!”

--Wei
OneWay

Supervised data

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>LT</th>
<th>...</th>
<th>Xₙ</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>11</td>
<td>...</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>X₁</td>
<td>X₂</td>
<td>NS</td>
<td>...</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>1</td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

Testing data

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>NS</th>
<th>...</th>
<th>Xₙ</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>1</td>
<td>...</td>
<td>33</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>2</td>
<td>...</td>
<td>22</td>
<td>?</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>5</td>
<td>...</td>
<td>18</td>
<td>?</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>10</td>
<td>...</td>
<td>7</td>
<td>?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>12</td>
<td>...</td>
<td>22</td>
<td>?</td>
</tr>
</tbody>
</table>

Build 12 learners  Select the best one (e.g.: NS)  Test with NS
Results
Our Evaluation Metrics

- Recall
- Popt
- F1
- Precision

$$P_{opt}(m) = 1 - \frac{S(\text{optimal}) - S(m)}{S(\text{optimal}) - S(\text{worst})}$$
Our Result Format

Report results on a project-by-project basis.
Research Questions

• All unsupervised predictors better than supervised?
• Is it beneficial to use supervised data?
• OneWay better than standard supervised predictors?
Research Questions

- All unsupervised predictors better than supervised?
- Is it beneficial to use supervised data?
- OneWay better than standard supervised predictors?
RQ1: All unsupervised predictors better?

Recall: LT and AGE are better; Others are not.

Popt: the similar pattern as Recall.
RQ1: All unsupervised predictors better?

F1: Only two cases better, LT on Bugzilla; AGE on PostgreSQL;

Precision: All are worse than the best supervised learner!
RQ1: All unsupervised predictors better?

Not all unsupervised predictors perform better than supervised predictors for each project and for different evaluation measures.
Research Questions

• All unsupervised predictors better than supervised?

• Is it beneficial to use supervised data?

• OneWay better than standard supervised predictors?
RQ2: Is it beneficial to use supervised data?

Recall: OneWay was only outperformed by LT in 4/6 data sets.

Popt: The similar pattern as Recall.
RQ2: Is it beneficial to use supervised data?

F1: EXP/REXP/SEXP performs better than OneWay only on Mozilla.

Precision: Similar as F1 but more data sets.
RQ2: Is it beneficial to use supervised data?

As a simple supervised predictor, OneWay performs better than most unsupervised predictors.
Research Questions

• All unsupervised predictors better than supervised?

• Is it beneficial to use supervised data?

• *OneWay* better than standard supervised predictors?
RQ3: \textit{OneWay} better than supervised ones?

- Better than supervised learners in terms of Recall & Popt;
- It performs just as well as other learners for F1.
- As for Precision, worse!
Conclusion
Lessons Learned

• Don’t aggregate results over multiple data sets.
  – Different conclusions: Yang et al.
Lessons Learned

• Don’t aggregate results over multiple data sets.
  – Different conclusions: Yang et al.

• Do use supervised data.
  – Unsupervised learners’ are unstable.
Lessons Learned

• Don’t aggregate results over multiple data sets.
  – Different conclusions: Yang et al.

• Do use supervised data.
  – Unsupervised learners’ are unstable.

• Do use multiple evaluation criteria.
  – Results vary for different evaluation criteria.
Lessons Learned

• Don’t aggregate results over multiple data sets.
  – Different conclusions: Yang et al.

• Do use supervised data.
  – Unsupervised learners’ are unstable.

• Do use multiple evaluation criteria.
  – Results vary for different evaluation criteria.

• Do share code and data.
  – Easy reproduction.

• Do post pre-prints to arXiv.org.
  – Saved two years.
Why Seacraft?

- Successor of PROMISE repo, which contains a lot of SE artifacts.
- No data limits;
- Provides DOI for every submission (aka, easy citation);
- Automatic updates if linked to github project.

*tiny.cc/seacraft*
On the market
(expected graduation: May, 2018)

weifu.us

(Machine learning, SBSE, Evolutionary algorithms, Hyper-parameter tuning)

Publications:

OPEN science is FASTER science

FSE: 2
ASE: 1
TSE: 1
IST: 1
Under Review: 2