清華大學

On the Value of Learning From Defect Dense Components for Software Defect Prediction

Hongyu Zhang, Adam Nelson, Tim Menzies

PROMISE 2010

Defect Prediction /1

- Software quality assurance (QA) is a resource and time-consuming activity to detect defects (bugs).
  - Software testing
  - Code inspection
  - Model checking
  - Review meetings
  - …

- One contributing factor to low software quality is the limitation of resources for QA
  - Limited by time and cost

How to allocate the limited QA resources more efficiently?

Defect Prediction /2

- Software defect prediction: we want to be able to predict the defect-proneness of a software
  - to better estimate the quality of the software
  - to better allocate QA resources.
- In recent years, research on software defect prediction has drawn the attention of many researchers in the software engineering filed.

Defect Prediction /3

- A few recent works on NASA datasets:
  - Menzies et al. performed defect predictions for five NASA projects using static code metrics.
    - The average results of recall= 71% and pf = 25%, using a Naive Bayes classifier, but the precisions are low.
  - Khoshgoftaar and Seliya performed an extensive study on NASA datasets using 25 classifiers.
    - They observed low prediction performance, and they did not see much improvement by using different classifiers.
  - Lessmann et al. reported a study on the statistical differences between 19 data miners used for defect prediction.
    - The learner's performance was remarkably similar.
  - many more …

The Ceiling Effect

- The high water mark in this field has been static for several years.
- For example, for four years we have been unable to improve on Menzies' 2006 results.
- Other studies report the same ceiling effect:

> *...the importance of the classifier model is less than generally assumed ... practitioners are free to choose from a broad set of models* [Lessmann et al., 2008]

- Perhaps it is time to explore other approach - we can look more at the data rather than the miner.

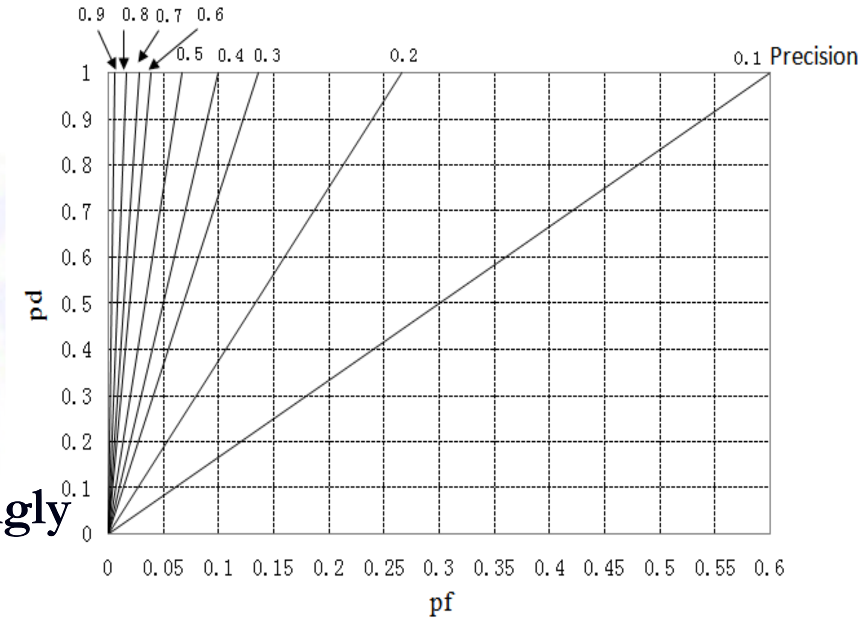
Maths about Defect Prediction

- Let {A, B, C, D} denote the true negatives, false negatives, false positives, and true positives (respectively) found by a binary detector.
- Certain standard measures can be computed as:
  - pd = recall = D/(B + D)
  - pf = C/(A + C)
  - prec = precision = D/(D + C)
  - acc = accuracy = (A + D)/(A + B + C + D)
  - F -measure = 2 *recall *prec/(recall + precision)
  - neg/pos = (A + C)/(B + D)
- The last measure (neg/pos) is important for understanding the data and the results.

The Impact of Neg/Pos ratio

- The Zhangs' equation about Precision and Recall

$$pf = \frac{pos}{neg} \cdot \frac{(1-prec)}{prec} \cdot recall$$



The change of precision with *pf* and *pd* when neg/pos = 15

- As neg/pos increases, high *recall*&*precision* is only possible when *pf* becomes vanishingly small.
- For example, in the ranges $0.65 \leq prec$, recall ≤ 0.8, *pf* falls into the following ranges:

✓ $0.023 \leq \text{pf} \leq 0.062$ for neg/pos = 7;

✓ $0.011 \leq \text{pf} \leq 0.029$ for neg/pos = 15;

✓ $0.007 \leq \text{pf} \leq 0.002$ for neg/pos = 250;

Changing the NEG/POS ratio /1

- One obvious way to change neg/pos ratio is to over-sample or under-sample the training data.
- Both methods might be useful in data sets with highly imbalances class distributions.
- At PROMISE'08, Menzies et al. demonstrated in the paper that:
  - "No treatment" performed as well as under-sampling
  - Over-sampling did not improve classifier performance.

Changing the NEG/POS ratio /1

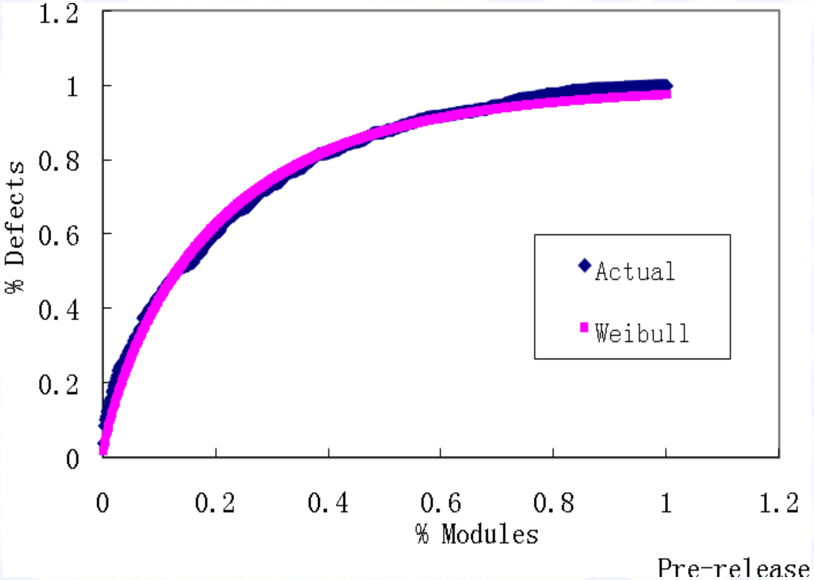
• We replicated the experiment described by Menzies et al. The results were evaluated using f-measure

• In summary, under-sampling/over-sampling approaches do not appear to be promising.

Rank	Treatment	f-measure percentiles 25%	50%	75%	2nd quartile median, 3rd quartile
1	j48 / none	27	86	94	
2	j48 / over	33	80	92	
3	nb / none	33	69	81	
4	nb / under	33	67	79	
4	nb / over	33	67	79	
5	j48 / under	30	53	79	
					0 50 100

Breaking Through the Ceiling

- In this paper, we exploit the naturally occurring distributions inside defect data sets.
  - Some interesting observations:
    - In Eclipse 3.0, 20% of the largest packages are responsible for 60.34% of the pre-release defects and 63.49% of post-release defects
    - At the file level, 20% of the largest Eclipse 3.0 files are responsible for 62.29% pre-release defects and 60.62% postrelease defects.
    - These results are consistent with those reported by other researchers.

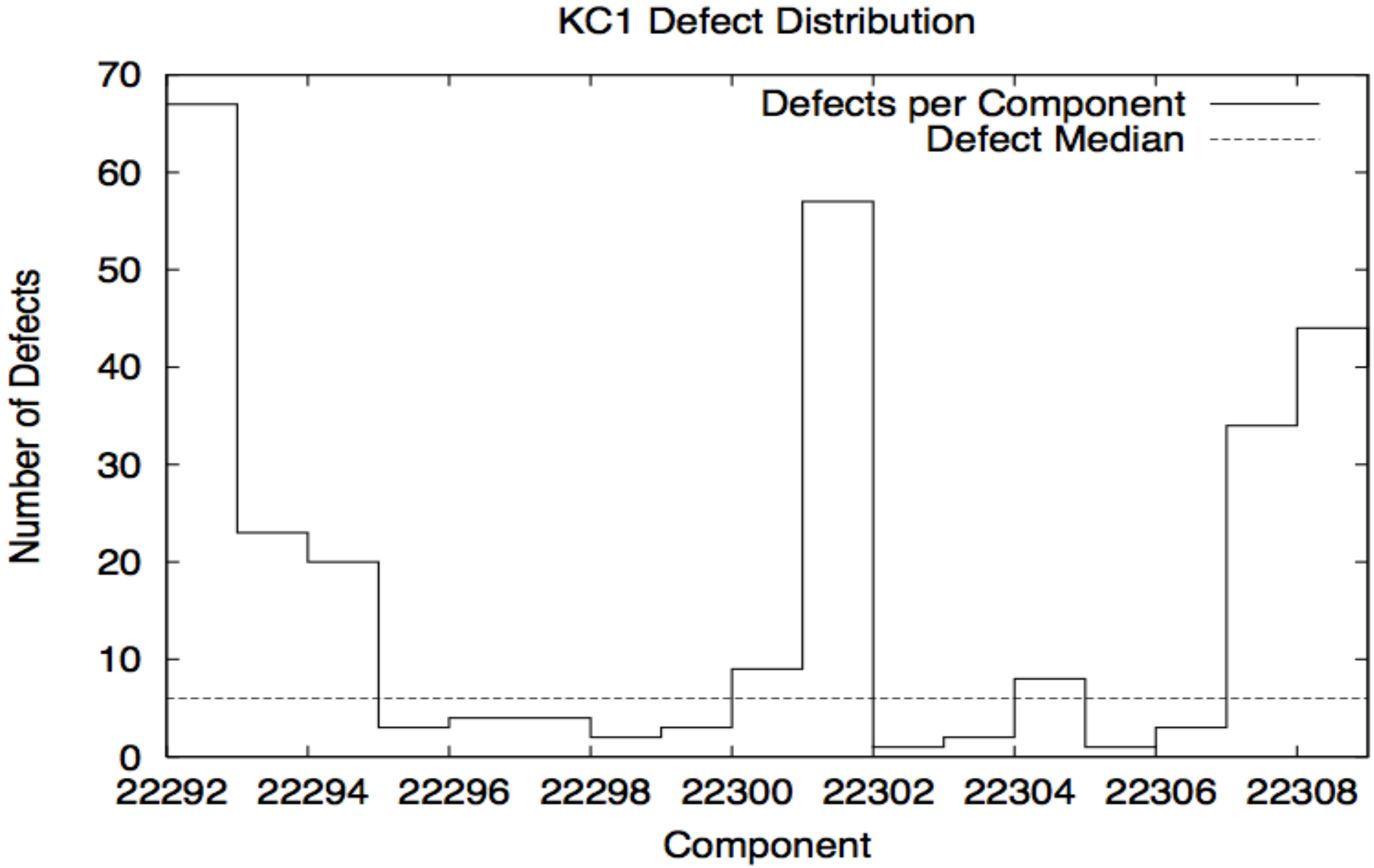


The Nature of Defect Data

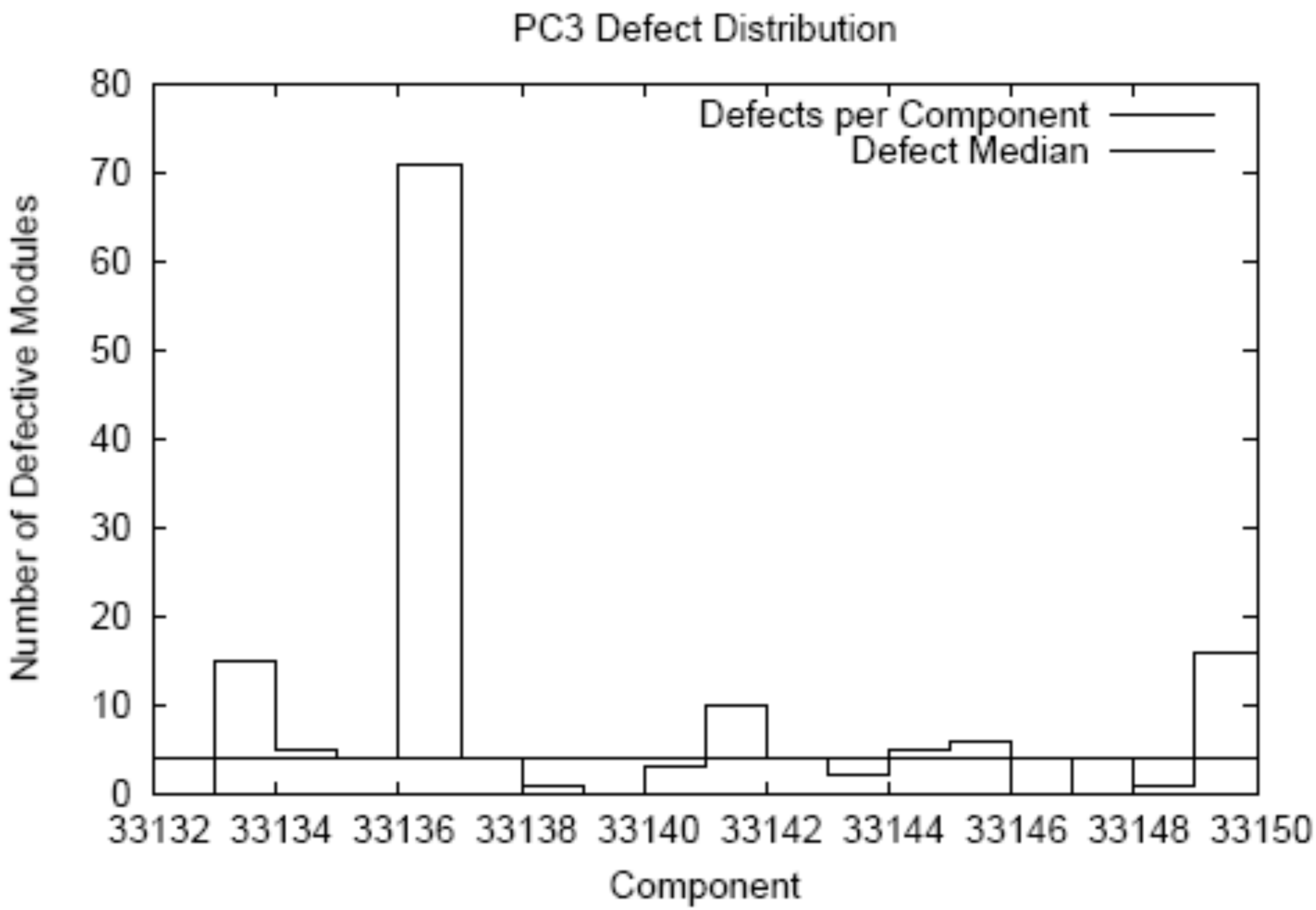
- Similar phenomenon is observed in NASA datasets
- For example, for the PC3 dataset (module level):
  - The top 5% "most defective" PC3 modules contain 68.34% of the defects
  - The top 10% "most defective" PC3 modules contain 98.84% of the defects.
- For example for the PC3 dataset (component level):
  - 6 out of 29 (20.69%) "most-defective" components contains 77.61% defects and 70% defective modules.
  - The NEG/POS ratios in these components range from 0.93 to 2.70, while for all modules in PC3, the overall NEG/POS ratio is 8.77.

in a large software system, the distribution of defects are skewed – most of the defects are found in the minority of components, which have lower NEG/POS ratios.

Defect Distributions of components



Defect Distributions of components



清華大學

Breaking Through the Ceiling

- **Identifying the defect dense components:**

If the number of defective modules per component exceeds the median number of defective modules across all components in that data set, it is labeled as a defect-dense component.

- **The idea:**

  Build defect prediction model using the training data from high defect dense components, instead of using all project data.

Datasets

Project	language	#Defects	#Modules	#Defective Modules	#Components	#Defective Components	#Dense Components
CM1	C++	70	505	48	20	9	9
KC1	C++	525	2107	325	18	18	9
MC1	C++	79	9466	68	57	26	26
PC1	C++	139	1107	76	29	17	14
PC3	C++	259	1563	160	29	17	14

- **Five NASA defect datasets from the PROMISE repository were used.**
- **For each data set, components are extracted (using a unique identifier) containing both defective and non-defective modules.**

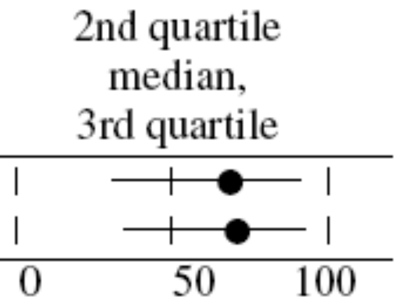
The experiments

```
For run = 1 to 10
    For each dense component C in data set D
        Train = C
        Test = D - C
        For bin = 1 to 10
            Test' = 10% of Test (picked at random)
            Train' = 90% of Train (picked at random)
            Naive Bayes (Train', Test')
        end bin
    end component
end run
```

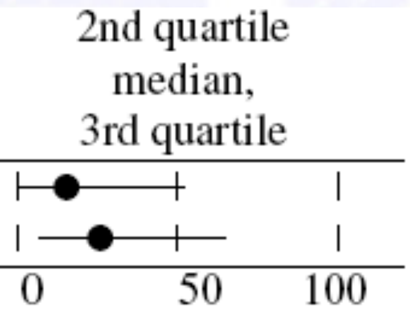
Results /1

Using data from all components, rank computed by Mann-Whitney test:

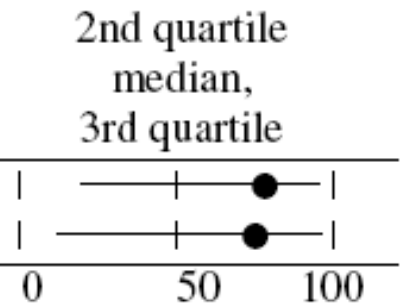
Rank	Treatment	recall percentiles 25%	50%	75%
1	Train on Dense Components	31	69	91
1	Train on All Components	35	71	93



Rank	Treatment	pf percentiles 25%	50%	75%
1	Train on Dense Components	0	15	52
2	Train on All Components	0	26	65



Rank	Treatment	precision percentiles 25%	50%	75%
1	Train on All Components	20	78	95
1	Train on Dense Components	12	75	96



Results /2

- **Over all the datasets studied here, training on dense components (those containing a higher number of defective modules) yields:**
  - **Similar medians (for *recall* and *precision*).**
  - **however, dramatic gains are seen with *pf* : the median error rates decrease by nearly *half***

Results /3

Project	Recall		0% 50% 100%	Prob. False Alarm (Pf)		0% 50% 100%	Precision		0% 50% 100%
CM1	1	All		1	Dense		1	All	
	2	Over		2	Over		1	Over	
	2	Under		2	Under		1	Under	
	3	Dense		2	All		1	Dense	
KC1	1	Dense		1	All		1	Dense	
	1	Under		1	Over		1	All	
	1	All		1	Under		1	Under	
	1	Over		1	Dense		1	Over	
MC1	1	Over		1	Over		1	Dense	
	1	Under		1	Under		1	All	
	2	All		2	All		1	Under	
	3	Dense		3	Dense		1	Over	
PC1	1	Dense		1	Dense		1	Dense	
	2	Over		2	Over		2	All	
	2	All		2	All		2	Under	
	3	Under		3	Under		3	Over	
PC3	1	Under		1	Dense		1	Dense	
	1	Over		2	Over		1	Under	
	2	All		2	Under		1	All	
	2	Dense		3	All		1	Over	

Results /4

data set	performance measure	all components	dense components	over sampling	under sampling
CM1	precision	0	0	0	0
	recall	+	-	-	-
	pf	-	+	-	-
KC1	precision	0	0	0	0
	recall	0	0	0	0
	pf	0	0	0	0
MC1	precision	0	0	0	0
	recall	-	-	0	0
	pf	-	-	0	0
PC1	precision	-	+	-	-
	recall	-	+	-	-
	pf	-	+	-	-
PC3	precision	0	0	0	0
	recall	-	-	0	0
	pf	-	+	-	-
summary	+	1	5	0	0
	0	6	6	9	9
	-	8	4	6	6

Each treatment is assigned one of {+, 0, –} depending on if it won, tied, lost in the statistical rankings

Note that dense won most often, and lost the least amount of times compared to all other treatments

Results /5

- In the majority cases, training on dense components yielded:
  - statistically significantly better defect prediction models than training on all components.
  - these detectors performs better than those learned from over- and under- samplings.

Conclusions

- A previously unexplored feature of the PROMISE defect data sets are the small number of components containing most of the defects.
- This skewed defect distribution has implications on our data analysis of the PROMISE defect datasets.
- To test this possibility, we restricted training to just the components with higher than the median number of defective modules.
- We found that training via dense sampling is useful for generating better defect prediction.
- we recommended the proposed method whenever it becomes apparent that components have different defect densities.

Thank you!

Hongyu Zhang
Associate Professor
School of Software, Tsinghua University
Beijing 100084, China
Email: hongyu@tsinghua.edu.cn
Web: http://www.thss.tsinghua.edu.cn/hongyu