Selecting Quality Data

Tim Menzies (Ph.D.)
Assoc Prof; LCSEE, WVU
tim@menzies.us
304-376-2859

Open issues in data quality (for software engineering)

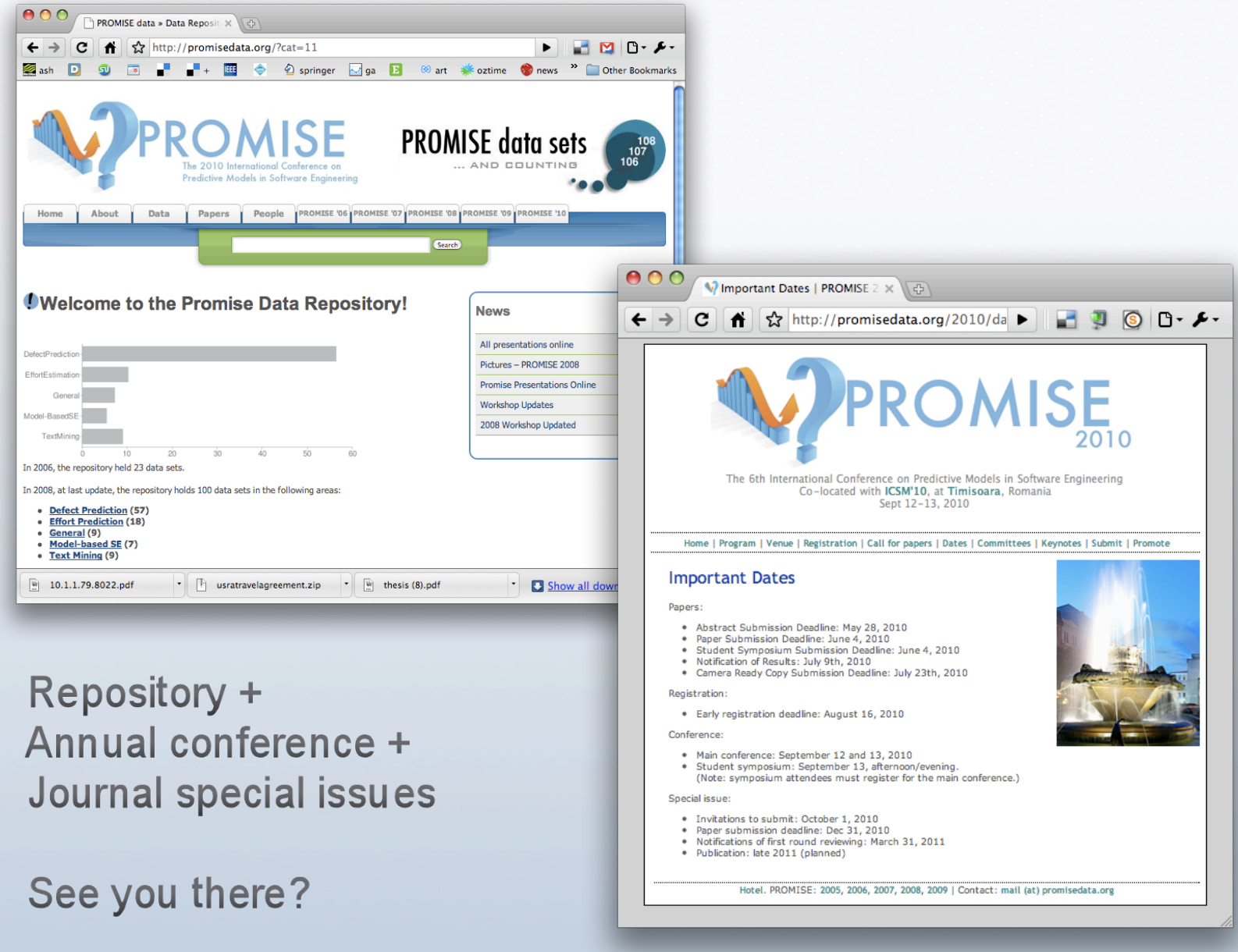
- Conclusion Instability
- Locality
- Best way to find significant data
  - How to import data from other sites
  - Relevancy filtering
  - Feature selection

Background

The PROMISE Experiment

- 2005, … 2011
  - Repeatable, improvable, maybe even refutable experiments in software engineering
  - Put up or shut up
  - [Menzies07]
- If you publish a paper, it is strong encouraged that you also offer the data on which the conclusion was made
- 130 data sets:
  - Defect prediction
  - Effort estimation
  - Model-based reasoning
  - Text mining
  - etc

http://promisedata.org/data



Repository +
Annual conference +
Journal special issues

See you there?

Relevance to quality

Other repos

- Extensive software support for the data storage
  - E.g. [Gentleman07, Leisch02]
- Elaborate design of research questions:
  - E.g. CeBASE [Basili02]
- None of those SE repos are still on-line

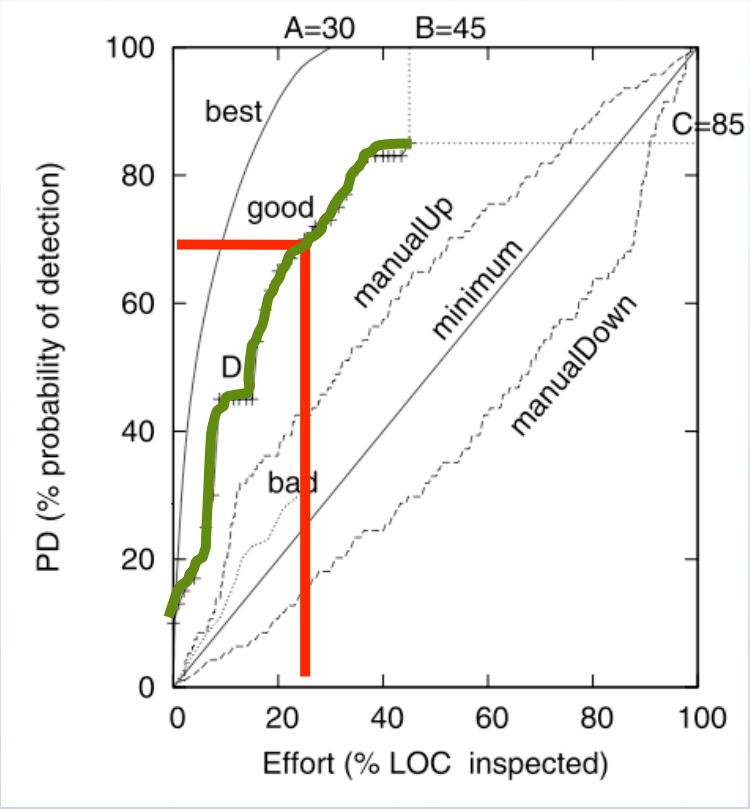
Our repo

- No restrictions on research questions
- Send us something,
  - We'll place it on-line
- Let many groups analyzing that data
  - Extensive list of conclusions
- Still an active, on-going research initiative.

Some results:

Learning to Organize Testing

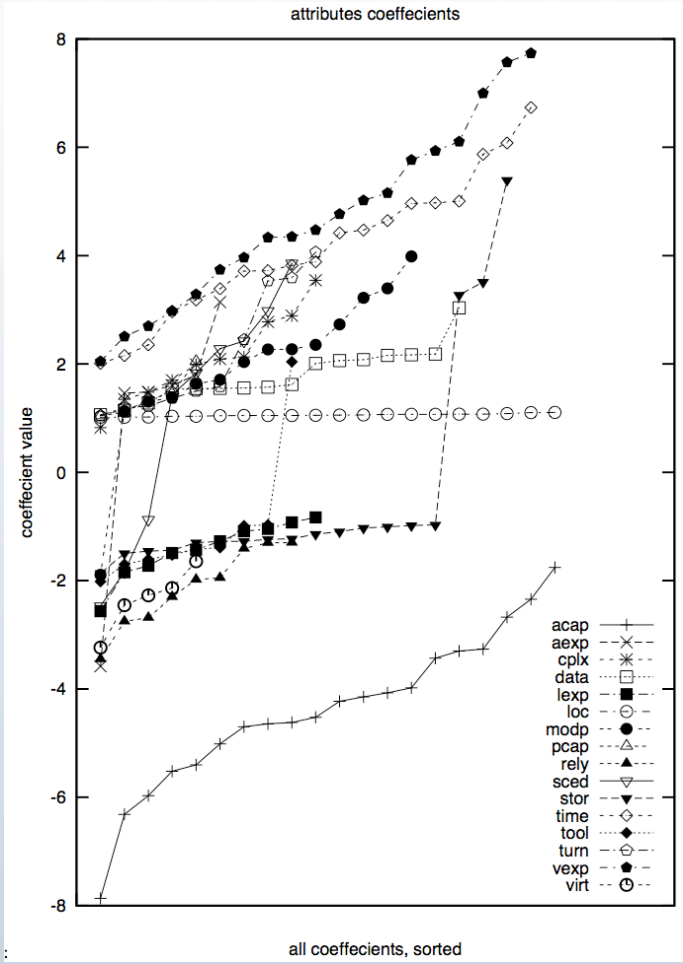
- Inspect X% of the code, find Y% of the bugs
  - But which X%?
  - Ask a data miner
- Typical results:
  - [Ostrand10]: X= 20% of files , Y= 75 to 93%
  - [Tosun10] : X= 25% . Y= 88%
  - [Menzies10a] X= 25%, Y= 70%
    - Or any other point you like



Conclusion Instability

20 * 90% sub-samples

- Linear regression on COCOMO data
- Wild variance in the learned model
  - Specifically, in the variable coefficients



Locality

[Brady10]: case-based reasoning to tame instability

- Seek best changes to a project
  - Tames conclusion instability, but conclusions project specific
- Beware the snake oil salesman telling you that "it worked there, it will work here"
- Best to assess policies w.r.t. local data
- (Are you collecting local data?)

cases	query	acap 3	apex 3	ltex 3	ltex 4	plex 3	pmat 3	pmat 4	sced 2	sced 3	stor 3	time 3	tool 3	# of Changes
nasa93	ground					100%	55%				85%			3
nasa93	flight					95%	70%				100%			3
nasa93	osp	95%	90%										100%	3
nasa93	osp2				100%			80%	85%					3
coc81	flight						60%					65%		2
coc81	osp2			55%	55%		65%			100%				4
coc81	ground						80%					100%		2
coc81	osp						65%			65%				2
Overall:		12%	11%	7%	19%	24%	49%	10%	11%	21%	23%	21%	13%	

How to find significant data

Some data is better than others

- **Relevancy filtering**
  - Given a test set T
  - Relevant = the k-th nearest neighbors in the train set
  - Only train on this "relevant" set

- **Data subset selection**
  - Reveal the core content

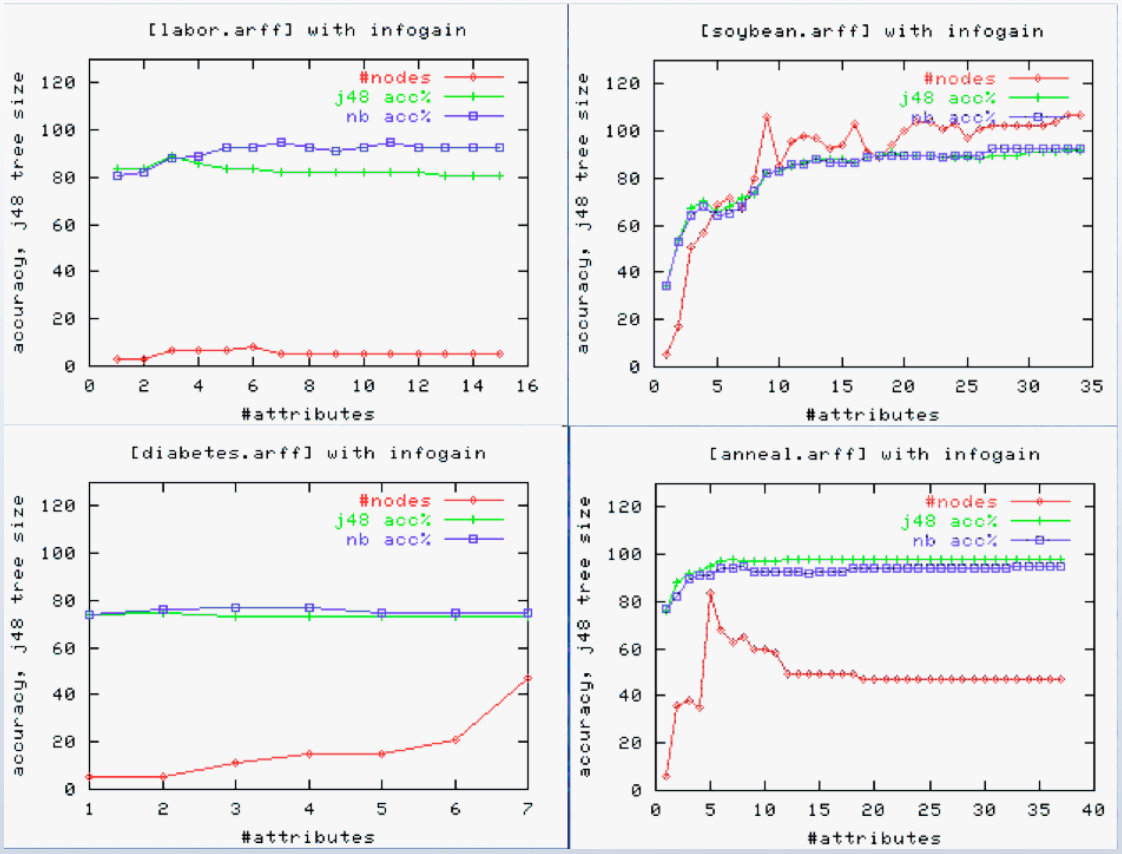
Relevancy filtering:

How to use other people's data

- **Learning**
  - (a) software development estimation or
  - (b) software defect predictors
- Suppose you don't have local data
  - Find relevant data from other companies
- Works as well as if you had local data
  - For defect prediction [Turhan09]
  - For effort estimation [Menzies10]
- **An experiment with N contractors**
  - Imported:
    - train on them, test on me
  - Local:
    - train on one, test of one
  - Relevant:
    - find nearest neighbors in them, train on just those
    - test on me
- Pd = detection, pf = false alarm
  - Imported pd,pf = 94 (!) , 68
  - Local: pd, pf = 75,29
  - Rlevant: pd,pf = 69,27

Feature selection: strip away spurious details

X-axis
sorted by entropy
sum(-p*log(p))

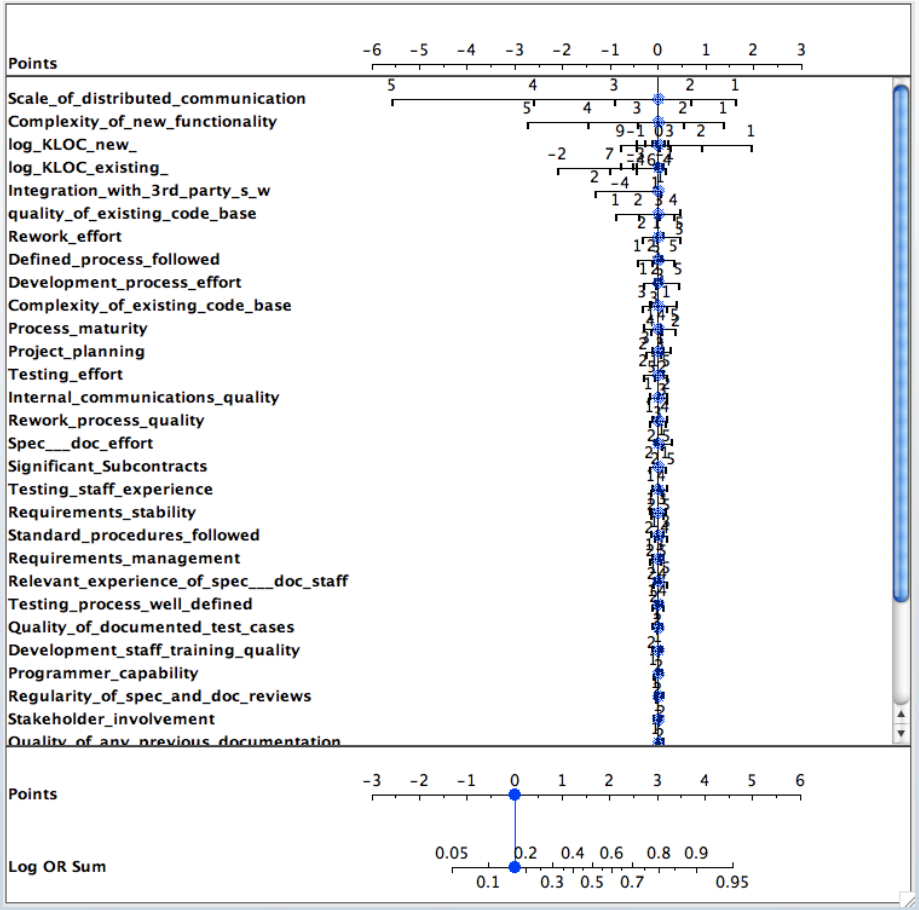


Useful = 6/16, 10/35, 1/7, 5/45 = 38%, 26%, 14%, 11%

Select features with Nomograms

- Nomograms [Mozina04]:
  - Discretize every feature.
  - For all pairs of target / other classes of size C1, C2 count frequency of range N1, N2 in each class
  - Log(odds ratio) = log((N1/C1) / (N2/C2))
  - If positive, then more frequent in target

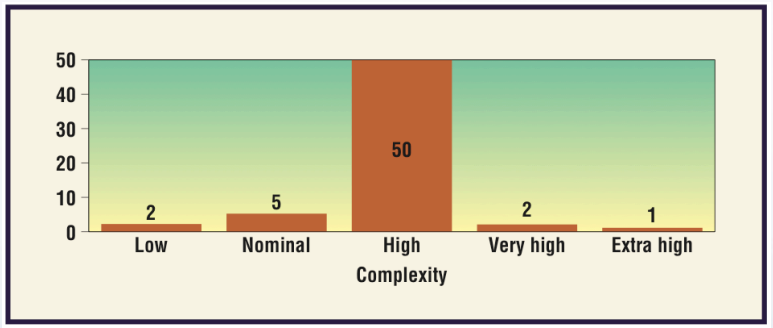
- Data from Fenton's Bayes Nets
  - [Fenton08]
  - Target class: worst defects
  - Useful subset of attributes: 7%
  - And of those attributes,
    - Only a few ranges matter

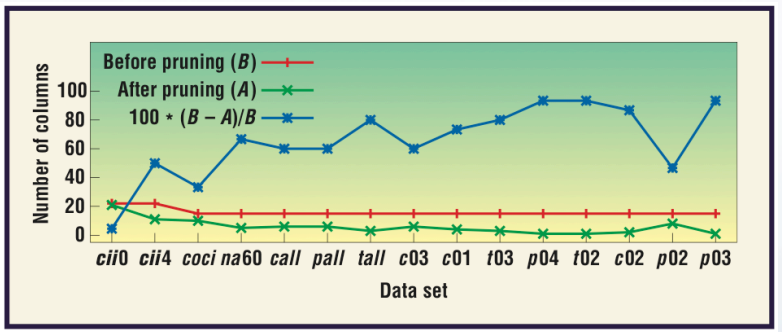


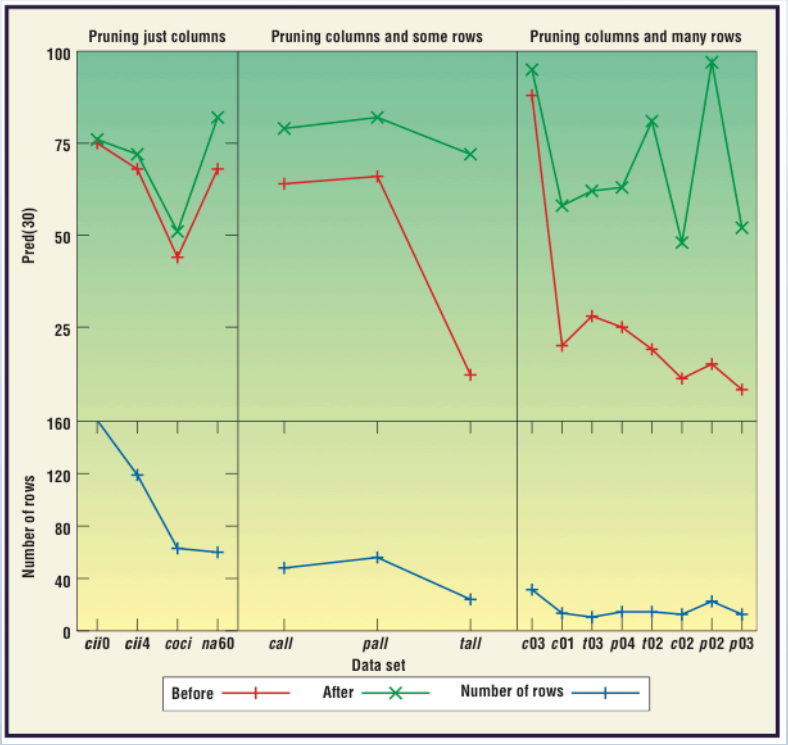
Caution

- **Simple entropy, nomograms,**
  - running in linear time
  - not the best feature selector
- **Best = WRAPPER**
  - A search through all subsets of F features
  - Warning: may be impractical for large data sets
- **Mixed strategies**
  - Send out the scouts (e.g. nomograms) to quickly prune
  - Before firing the big guns (WRAPEPR)

Example 1: Select columns with WRAPPER for effort estimation







- Reference: [Menzies05]
- Useful = 65%

Example 2: Select columns with WRAPPER for defect prediction

- 10-way cross-val, LSR, on [Bno8]

Defects =
82.2602 * S1=L,M,VH +
158.6082 * S1=M,VH +
249.407 * S1=VH +
41.0281 * S2=L,H +
68.9153 * S2=H +
151.9207 * S3=M,H +
125.4786 * S3=H +
257.8698 * S4=H,M,VL +
108.1679 * S4=VL +
134.9064 * S5=L,M +
-385.7142 * S6=H,M,VH +
115.5933 * S6=VH +
-178.9595 * S7=H,L,M,VL +
...
[50 lines deleted]

R^2 = 0.45 (mean on x-val)

- WRAPPER:
  - Search for attributes that matter, 10 times on 90% of the data
  - Only build model from attributes selected at least 50% of the time.

Defects =
876.3379 * S7=VL +
-292.9474 * D3=L,M +
483.6206 * P5=M +
5.5113 * KLoC +
95.4278

Uses 4/53 = 8% of the data

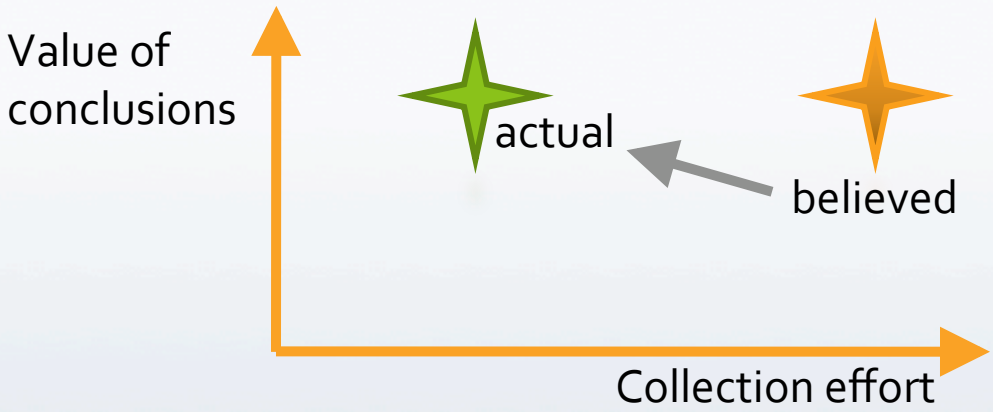
R^2=0.98 (mean on x-val)

Implications for data quality planning

- A warning: it may be a waste of effort to:
  - Expend great effort to precisely define all possible data points,
  - Then spend much time collecting data according to those definitions.
- Rather, more effective to:
  1. Quickly implement a draft definition/ collection plan.
  2. As soon as any data is available, apply feature selection.
  3. Only elaborate data that feature selection reports are significant.

Summary

Explaining the surprising conclusions of PROMISE



- Given the local conditions of a particular project,
  - Only a small number of features are insignificant:
  - E.g. 38%, 26%, 14%, 11%, 63%, 7%, 7%
- So in a large pool of messy data
  - They may exist a subset that is still useful for prediction
- So real-world data is good, providing you get enough of it, and throw most of it away before you learn

Open issues in data quality (for software engineering)

- Conclusion Instability
- Locality
- Best way to find significant data
  - How to import data from other sites
  - Relevancy filtering
  - Feature selection

References

[Basili02] Victor Basili, Roseanne Tesoriero, Patricia Costa, Mikael Lindvall, Ioana Rus, Forrest Shull, and Marvin Zelkowitz. Building an experience base for software engineering: A report on the first cebase eworkshop. In in Profes (Product Focused Software Process Improvement, pages 110–125, 2001.

[Bn08] http://icde.googlecode.com/svn/trunk/share/data/arff/bn.arff

[Brady10] Adam Brady and Tim Menzies. Case-Based Reasoning vs. Parametric Models for Software Quality Optimization, PROMISE'10

[Fenton08] Project Data Incorporating Qualitative Factors for Improved Software Defect Prediction Norman Fenton, Martin Neil, William Marsh, Peter Hearty, Lukasz Radlinski and Paul Krause., PROMISE 2008

[Gentleman07] R. Gentleman and D. Temple Lang. Statistical analyses and reproducible research. Journal of Computational and Graphical Statistics, 16(1):1–23, March 2007.

[Leisch02] F. Leisch. Sweave: Dynamic generation of statistical reports using literate data analysis. In W. Härdle and B. Rönz, editors, Proceedings in Computational Statistics (2002), pages 575–580. Physica 2002.

[Menzies05] Tim Menzies with Scott Chen and Dan Por and Barry Boehm, Finding the Right Data for Software Cost Modeling , IEEE Software Nov/Dec 2005

[Menzies06] T. Menies, K. Lum, and J. Hihn. The deviance problem in effort estimation. In *PROMISE, 2006*,

[Menzies07] T. Menzies with G. Boetticher, and T. Ostrand. PROMISE Repository of empirical software engineering data. West Virginia University, Department of Computer Science, 2007. http://promisedata.org/.

[Menzies10] Ekrem Kocaguneli, Gregory Gay, Tim Menzies, Ye Yang, Jacky W. Keung: When to use data from other projects for effort estimation. ASE 2010: 321-324

[Menzies10a] Tim Menzies, Zach Milton Burak Turhan, Bojan Cukic et al. Defect prediction from static code features: current results, limitations, new approaches Automated Software Engineering, Dec 2010

[Mozina04] Martin Mozina,, Janez Demsar, Michael Kattan, and Blaz Zupan, Nomograms for Visualization of Naive Bayesian Classifier

[Ostrand10] Programmer-based Fault Prediction Thomas J. Ostrand, Elaine J. Weyuker, Robert M. Bell. PROMISE'10

[Tosun10] Tosun, A., Bener, A.: AI-based software defect predictors: Applications and benefits. In: IAAI'10 (2010)

[Turhan09] Turhan, B., Menzies, T., Bener, A., Distefano, J.: On the relative value of cross-company and within-company data for defect prediction. Empir. Softw. Eng. 68(2), 278–290 (2009). Available from http://menzies.us/pdf/08ccwc.pdf

Questions?
Comments?