When Good Data Goes Bad

Tim Menzies (tim@menzies.us)
Lane Department of Computer Science and Electrical Engineering
West Virginia University
http://menzies.us/pdf/06good2bad.pdf

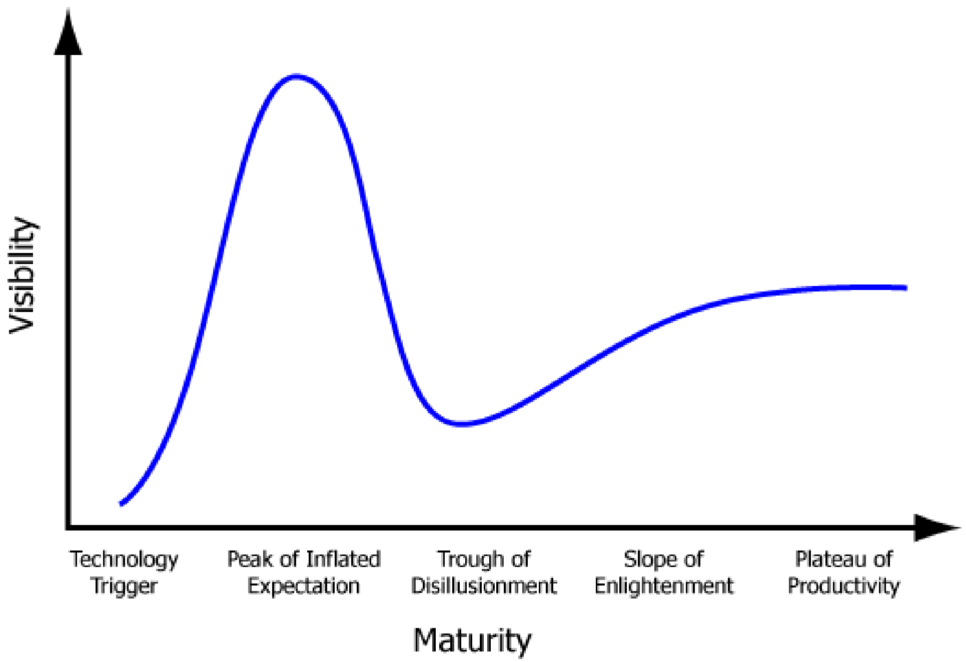
August 22, 2007

Executive Summary

- Data mining NASA project data
- Five examples where data mining found clear quality predictors
  - for effort
  - for defects
- In only one of those cases is that data source still active.
  - All that dead data.
- What to do?

"Don't let it eat away at you. You ex wasn't that smart. She said you'd rot in Hell. You, my friend, are not rotting."

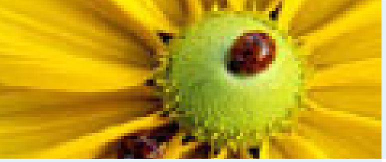
Background: AI- it works



- 1980s: AI summer
- 1980s (late): bubble bursts, AI winter
- 1990s: great success with planning, scheduling, data mining
- 2000s: many successes of AI (data mining) for SE

- This talk: AI really works (5 success stories with NASA data)
- Still, main problem is organizational, not technological
  - Despite clear success, $\frac{4}{5}$ of those data sources have vanished
  - What to do?

Eg #1: text mining @ NASA



What is data mining?

- Diamonds in the dust
- Summarization: not 1000 records, but 3 rules
- Example #1:
  - text mining issue reports
  -
- 901 NASA records, PITS issue tracker: {severity, free text}

severity	frequency
1 (panic)	0
2	311
3	356
4	208
5 (yawn)	26

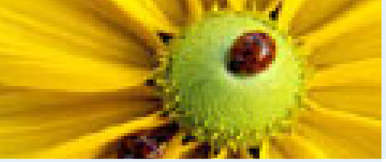
- All unique words, sorted by "magic" (see below)
- Rules learned from N best
- Severity 2 predictors: 10*{(train,test) = (90,10)%}

N	a=recall	b=precision	$F = \frac{2*a*b}{a+b}$
100	0.81	0.93	0.87
50	0.80	0.90	0.85
25	0.79	0.93	0.85
12	0.74	0.92	0.82
6	0.71	0.94	0.81
3	0.74	0.82	0.78

Rules (from N=3 words):

if (rvm $\leq$ 0) & (srs = 3) $\rightarrow$ sev=4
else if (srs $\geq$ 2) $\rightarrow$ sev=2
else $\rightarrow$ sev=3

- Diamonds in the dust
  - Not 9414 words total
  - or 1662 unique words
  - but 3 highly predictive words



Dumb Luck?

- Nope.
- In four other case studies, learning from just the top 3 terms ...
  - 10*{(train,test) = (90,10)%}
  - yields probability of detection of highest severity class of 93% to 99.8%.
- (Note: ignoring real rare classes.)

Project "b": 984 records

```
a   b   c   d   <-- classified as
1 380   0   0 |   a = _4
1 520   0   0 |   b = _3 pd=99.8%
0  59   0   0 |   c = _5
0  23   0   0 |   d = 2
```

Project "d": 180 records

```
  a   b   c   <-- classified as
157  23   0 |   a = _4
  9 121   0 |   b = _3 pd=99.4%
  6   1   0 |   c = _5
```

Project "c": 317 records

```
  a   b   c   <-- classified as
  9 121   0 |   b = _3 pd=93.1%
157  23   0 |   a = _4
  6   1   0 |   c = _5
```

Project "e": 832 records

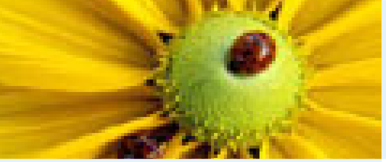
```
a   b   c   d   <-- classified as
0  23   0   0 |   a = _2
0 498   0  18 |   b = _3 pd=96.5%
0  34   0   7 |   c = _5
0 178   0  65 |   d = _4
```

How is this Possible?

- Feature subset selection (FSS) (Hall and Holmes [2003], Miller [2002])
  - $y = \beta_0 + \beta_1 f_1 + \beta_2 f_2 + \beta_3 f_3 ...$
  - Variance in y reduced by pruning some f_i
  - But don't prune too much:
    - e.g. $\forall f_i, y = \beta_0$
  - Analogous argument holds for other representations.
- Q: How to do FSS?
- A1:Apply domain knowledge
  - e.g., in text mining, TF*IDF:
    - *term frequency inverse document frequency*
    - Frequent terms, but only in a small number of documents
  - TF*IDF $= F[i,j] * log(IDF)$
    - $F[i,j]$ = frequency of word i in document j
    - IDF = #documents / (#documents with i)
    - Above study used top 100 $TF * IDF$ words.

Less is More

- Q: How to do FSS?
- A2: Exponential time FSS
  - Try all 2^F subsets of F on a *target learner*
  - Possible, with small feature sets, with some heuristic search
    - best first search, STALE=5 (Kohavi and John [1997]).
- A3: Linear time FSS (not as thorough as 2^F):
  - Sort F, somehow;
  - Try first $1 \leq f \leq F$ features
  - Above study sorted top 100 TF*IDF terms on infogain
    - Initially:
      - $H(C) = -\sum_{c \in C} p(c) log_2 p(c)$.
    - After seeing feature f:
      - $H(C|f) = -\sum_{x \in f} p(x) \sum_{cinC} p(c|x) log_2 p(c|x)$
    - So $InfoGain = H(C) - H(C|f)$



Less is more (2)

- Over-fitting avoidance
  - After learning “it” ...
  - ... Try throwing away bits of “it”
- e.g. RIPPER (Cohen [1995]),
  - If you learn a conjunction, prune with greedy back select
  - If you learn a set of rules, prune with greedy back select
  - For the surviving rules, try replace it with..
    - a dumb alternative
    - or a carefully selected modification
  - Very fast: $O(m(log(m))^2)$ for m examples
  - Often produces smaller theories than other methods
    - as above:

$$\begin{array}{ll} \text{if (rvm} \leq 0) \ \& \ (\text{srs} = 3) & \longrightarrow \text{sev=4} \\ \text{else if (srs} \geq 2) & \longrightarrow \text{sev=2} \\ \text{else} & \longrightarrow \text{sev=3} \end{array}$$

Eg #2: effort estimation @ NASA

Effort estimation

- NASA COCOMO data (Boehm et al. [2000])
- Results from IEEE TSE (Menzies et al. [2006]).
- learners for continuous classes
- A study of 160 effort estimation methods
- 20 * { pick any 10, train on remaining, test on 10 }

	$100 * (pred - actual)/actual$		
	50% percentile	65% percentile	75% percentile
mode= embedded	-9	26	60
project= X	-6	16	46
all	-4	12	31
year= 1975	-3	19	39
mode= semi-detached	-3	10	22
ground systems	-3	11	29
center= 5	-3	20	50
mission planning	-1	25	50
project= gro	-1	9	19
center= 2	0	11	21
year= 1980	4	29	58
avionics monitoring	6	32	56
median	-3	19	39

- i.e. usually, very accurate estimates

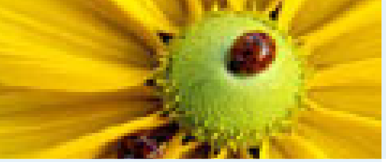
Eg #3: severity prediction @ NASA

SILAP: Early Life cycle Severity Detection

- NASA defect data: 5 projects, (Menzies et.al 2007)
- SILAP: predict error {potential, consequence} from project description

Derived	Raw features
co = Consequence	am =Artifact Maturity
dv = Development	as =Asset Safety
ep = Error Potential	cl =CMM Level
pr = Process	cx =Complexity
sc = Software Characteristic	di =Degree of Innovation
	do =Development Organization
	dt =Use of Defect Tracking System
	ex =Experience
	fr =Use of Formal Reviews
	hs =Human Safety
	pf =Performance
	ra =Re-use Approach
	rm =Use of Risk Management System
	ss =Size of System
	uc =Use of Configuration Management
	us =Use of Standards

```
function CO( tmp)    { tmp=0.35*AS + 0.65 *PF; return (round((HS) < tmp) ? HS : tmp)
function EP()        { return round(0.579*DV() + 0.249*PR() + 0.172*SC())}
function SC()        { return 0.547*CX + 0.351*DI + 0.102*SS  }
function DV()        { return 0.828*EX + 0.172*DO }
function PR()        { return 0.226*RA + 0.242*AM + formality()  }
function formality() { return 0.0955*US+ 0.0962*UC+ 0.0764*CL + 0.1119*FR +0.0873*DT + 0.0647*RM}
```



2^F FSS + RIPPER + SILAP data (211 components on 5 projects)

	features	$\lvert F\rvert$	_12	_3	_4	_5	$X = \left(\sum x\right)/4$
			severity predictions a=pd, b=prec $x = 2ab/(a+b)$				
A	all - L1 - L2 - group(6)	8	0.97	0.95	0.97	0.99	0.97
B	all - L1 - L2 - group(5 + 6)	7	0.95	0.94	0.97	0.96	0.96
C	all - L1 - L2 - group(4 + 5 + 6)	6	0.93	0.95	0.98	0.93	0.95
D	all - L1 - L2	16	0.94	0.94	0.93	0.96	0.94
E	all - L1 - L2 - group(3 + 4 + 5 + 6)	4	0.93	0.97	0.90	0.87	0.92
F	{co*ep, co, ep}	3	0.94	0.84	0.55	0.70	0.76
G	L1	1	0.67	0.69	0.00	0.46	0.45
H	just "us"	1	0.64	0.60	0.00	0.00	0.31
I	L2	1	0.57	0.00	0.32	0.00	0.22

Rules from "E":

rule 1	if	$uc \geq 2 \wedge us = 1$	then	severity = 5
rule 2	else if	$am = 3$	then	severity = 5
rule 3	else if	$uc \geq 2 \wedge am = 1 \wedge us \leq 2$	then	severity = 5
rule 4	else if	$am = 1 \wedge us = 2$	then	severity = 4
rule 5	else if	$us = 3 \wedge ra \geq 4$	then	severity = 4
rule 6	else if	$us = 1$	then	severity = 3
rule 7	else if	$ra = 3$	then	severity = 3
rule 8	else if	true	then	severity = 1 or 2

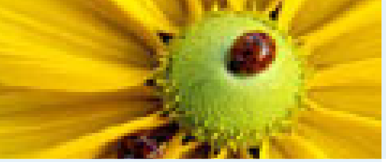
FSS to mature business knowledge

2005: Delphi session results:

goal	feature	weight	filter	weight* filter	contribution to goal
consequence	hs	0.35	1	0.350	35%
(co)	pf	0.65	1	0.650	65%
error	ex	0.828	0.579	0.479	47%
potential	cx	0.547	0.172	0.094	9%
(ep)	do	0.172	0.579	0.100	9%
	di	0.351	0.172	0.060	6%
	am	0.242	0.249	0.060	6%
	ra	0.226	0.249	0.056	5%
	us	0.0955	0.249	0.024	2%
	uc	0.0962	0.249	0.024	2%
	fr	0.119	0.249	0.030	2%
	dt	0.0873	0.249	0.022	2%
	ss	0.102	0.172	0.018	1%
	cl	0.0764	0.249	0.019	1%
	rm	0.0647	0.249	0.016	1%

2007: Features seen in 10 FSS (90% samples):

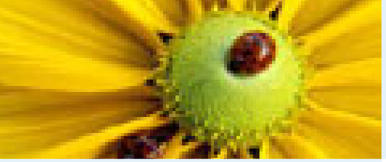
group	feature	notes	number of times selected
1	us	use of standards	10
2	uc	config management	9
	ra	reuse approach	9
	am	artifact maturity	9
3	fr	formal reviews	8
	ex	experience	8
4	ss	size of system	7
5	rm	risk management	6
6	cl	CMM level	5
	dt	defect tracking	5
	do	development organization	4
	di	degree of innovation	4
	hs	human safety	3
	as	asset safety	2
	cx	complexity	2
	pf	performance	1



Eg #4: defect prediction @ NASA

Defect predictors from Static code measures

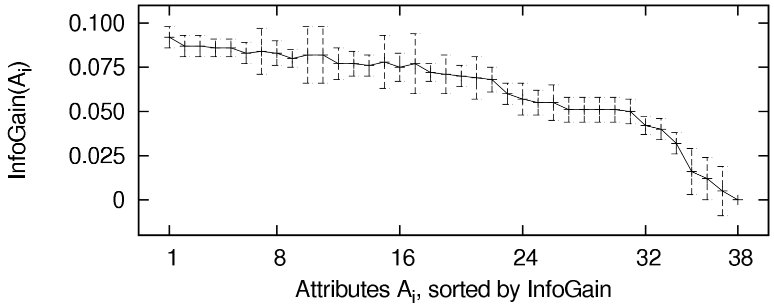
- IEEE TSE (Menzies et al. [2006])
- Modules from eight NASA projects, MDP, described using LOC, McCabe, Halstead metrics
- New methods
  - Shoot-out between :
    - Bayesian;
    - simple rule learners; e.g. $v(g) \geq 10$
    - complex tree learners; C4.5
  - Simple pre-processor on the exponential numerics
    - $num = log(num < 0.000001?0.000001 : num)$
- Prior state(s)-of-the-art, percentage of defects found:
  - IEEE Metrics 2002 panel: manual software reviews finds $\approx 60\%$
  - Raffo: industrial reviews finds TR(min,mod,max) = TR(35, 50, 65)%
  - My old data mining experiments: prob {detection,false alarm}={36,17}%



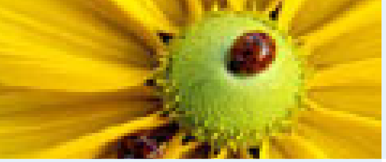
Results: 10 * { randomize, 10 * { (train,test) = (90,10)% }}

- Bayes + logging beats prior state of the art
- mean prob {detection,false alarm}={71,25}%
- Again, no one "best" theory

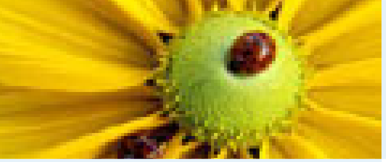
data	N	% pd	% pf	selected features	fss method
pc1	100	48	17	3, 35, 37	$O(2^F)$
mw1	100	52	15	23, 31, 35	$O(F)$
kc3	100	69	28	16, 24, 26	$O(F)$
pc2	100	72	14	5, 39	$O(F)$
kc4	100	79	32	3, 13, 31	$O(F)$
pc3	100	80	35	1, 20, 37	$O(F)$
pc4	100	98	29	1, 4, 39	$O(F)$
all	800	71	25		



ID	frequency	what	type
1	2	loc_blanks	locs
3	2	call_pairs	misc
4	1	loc_code_and_command	locs
5	2	loc_comments	locs
13	1	edge_count	misc
16	1	loc_executable	locs
20	1	I	H (derived Halstead)
23	1	B	H (derived Halstead)
24	1	L	H (derived Halstead)
26	1	T	H (derived Halstead)
31	2	node_count	misc
35	3	μ_2	h (raw Halstead)
36	1	μ_1	h (raw Halstead)
37	2	number_of_lines	locs
39	2	percent_comments	misc



Eg #5: more defect prediction @ NASA



Yet more detect prediction

- (Song et al. [2006])
- NASA SEL defect data: than 200 projects over 15 years.
- Predicting defects accuracy is very high (over 95%),
- false-negative rate is very low.

Conclusions

In summary...

- Five NASA data sources
  - Eg #1: text mining a NASA issue database (PITS)
  - Eg #2: effort estimation from NASA data (COCOMO)
  - Eg #3: early life cycle severity prediction (SILAP)
  - Eg #4: defect prediction from NASA static code data (MDP)
  - Eg #5: defect prediction (NASA SEL)
- All of which yield strong predictors for quality (effort, defects)
- Only one of which is still active (PITS)
- What went wrong?
- What to do?

Why is this Data Being Ignored?

- Group 1 : easy to explain
  - NASA SEL : Technology used in case study #5 very new
  - PITS :
    - Accessing PITS data was hard- required much civil servant support
    - No one was crazy enough to try text mining on unstructured PITS issue reports.
  - SILAP :
    - Newest data set of all the above
    - Never explored before since not available before
    - Data collection stopped since IV&V business model changed (now focused on model-based early lifecycle validation).
- Group 2 : harder to explain
  - MDP : Much interest across the agency (at GRC, JSC) in MDP (and associated tools).
  - COCOMO: well-documented, cheap to collect, many tools available
  - Maybe the answer lies in NASA culture:
    - NASA's centers compete for resources.
    - Reluctance to critically evaluate and share process information.

What to do?

- Stop *debating what data to collect*
  - Many loosely-defined sources will do: COCOMO, SILAP, defect reports
- Stop *debating how to store data*
  - Comma-separated or ARFF format or XML , one per component, is fine
- Stop *hiding data*
  - Create a central register for all NASA's software components
  - Register = component name and "part-of" (super-component)
  - Features extracted from all components, stored at a central location
  - All reports have anonymous join key to the central register
  - Make the anonymous data open source (lever the data mining community)
- Stop *ignoring institutional data*
  - Active repository, not data tomb
  - Success measure: not data in, but conclusions out
- Stop *publishing vague generalities*
  - Rather, publish *general methods* for building *specific models*
  - Open research question: how much data is enough to learn local model?

Questions? Comments?

References

- RIPPER
  - W.W. Cohen. Fast effective rule induction. In *ICML'95*, pages 115–123, 1995. Available on-line from `http://www.cs.cmu.edu/~wcohen/postscript/ml-95-ripper.ps`
- FSS
  - M.A. Hall and G. Holmes. Benchmarking attribute selection techniques for discrete class data mining. *IEEE Transactions On Knowledge And Data Engineering*, 15(6):1437– 1447, 2003. Available from `http://www.cs.waikato.ac.nz/~mhall/HallHolmesTKDE.pdf`
  - Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2): 273–324, 1997. URL `citeseer.nj.nec.com/kohavi96wrappers.html`
  - A. Miller. *Subset Selection in Regression (second edition)*. Chapman & Hall, 2002. ISBN 1-58488-171-2
- COCOMO
  - Barry Boehm, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford K. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani, and Chris Abts. *Software Cost Estimation with Cocomo II*. Prentice Hall, 2000
  - Tim Menzies, Zhihao Chen, Jairus Hihn, and Karen Lum. Selecting best practices for effort estimation. *IEEE Transactions on Software Engineering*, November 2006. Available from `http://menzies.us/pdf/06coseekmo.pdf`
- MDP
  - Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. *IEEE Transactions on Software Engineering*, January 2007. Available from `http://menzies.us/pdf/06learnPredict.pdf`
- SEL
  - Qinbao Song, Martin Shepperd, Michelle Cartwright, and Carolyn Mair. Software defect association mining and defect correction effort prediction. *IEEE Trans. Softw. Eng.*, 32(2):69–82, 2006. ISSN 0098-5589