Can Organizations Really Use Predictions?
Discussion paper: PROMISE '08

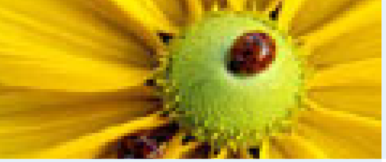
Tim Menzies
tim@menzies.us
Lane Department of Computer Science and Electrical Engineering
West Virginia University
http://menzies.us

May 5, 2008

We’re Learning. Who’s Listenning?

- This talk:
  - Data Mining for SE 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?
- How to insure that all our good work is not wasted?

Examples



Eg1: Text Mining NASA

- 901 NASA records, PITS issue tracker: {severity, free text}

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

- Top 100 unique words (selected by Tf*IDF); sorted by InfoGain
- Rules learned from N best (RIPPER Cohen [1995a]

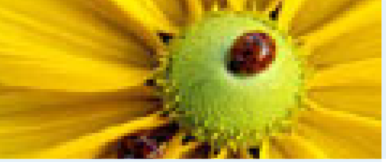
- Severity 2 predictors: 10*{(train,test) = (90,10)%}

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



Top 3 words

- In issue reports from four other projects:
  - 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

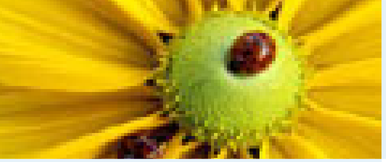
```
  a   b   c   <-- classified as
157  23   0 |   a = _4
  9 121   0 |   b = _3 pd=99.4%
  6   1   0 |   c = _5
```

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
```



Eg2: 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

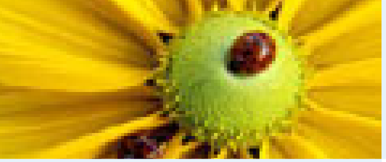
- i.e. usually, very accurate estimates

Eg3: 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}
```



Eg3: 2^F FSS + RIPPER + SILAP data (211 components on 5 projects)

severity predictions a=pd, b=prec $x = 2ab/(a+b)$

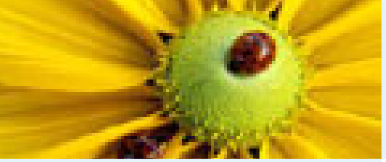
	features	$\|F\|$	_12	_3	_4	_5	$X = \left(\sum x\right)/4$
A	all - L1 - L2 - group(6)	8	0.97	0.95	0.97	0.99	0.97
B	all - L1 - L2 - group(5 + 6)	7	0.95	0.94	0.97	0.96	0.96
C	all - L1 - L2 - group(4 + 5 + 6)	6	0.93	0.95	0.98	0.93	0.95
D	all - L1 - L2	16	0.94	0.94	0.93	0.96	0.94
E	all - L1 - L2 - group(3 + 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

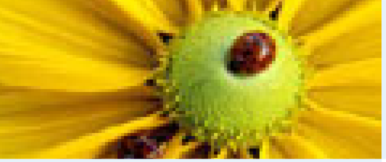
Eg4: Defect predictors from Static code measures

- 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}%

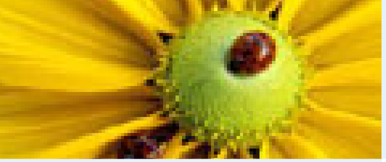


Eg5: Yet more detect prediction

- (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 (under 3%).
- Wow.



Conclusion

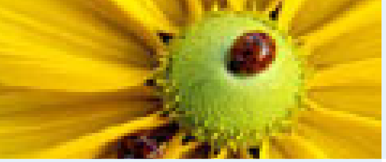


In summary...

- 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 <u>one</u> of which is still active (PITS)
- What went wrong?
- What to do?

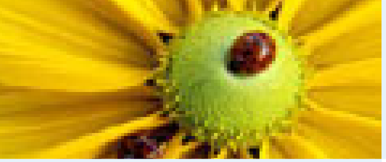
Why is this Data Being Ignored?

- 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

- Eg1: text mining
  - SEVERIS: T. Menzies and A. Marcus. Automated severity assessment of software defect reports. In *Submitted to ICMS'08*, 2008. Available from `http://menzies.us/pdf/08severis.pdf`
  - RIPPER: W.W. Cohen. Fast effective rule induction. In *ICML'95*, pages 115–123, 1995b. Available on-line from `http://www.cs.cmu.edu/~wcohen/postscript/ml-95-ripper.ps`
- Eg2: effort estimation
  - 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`
- Eg3: issue prediction
  - SILAP: T. Menzies, M. Benson, K. Costello, C. Moats, M. Northey, and J. Richarson. Learning better ivv practices. *Innovations in Systems and Software Engineering*, March 2008. Available from `http://menzies.us/pdf/07ivv.pdf`
- Eg4: static code
  - 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`
- Eg5: 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