Which: A Stochastic Best-First Search

Zachery Milton

April 17, 2008

Committee:
Tim Menzies, Ph.D.
Arun Ross, Ph.D.
Katerina Goseva-Popstojanova, Ph.D.

Motivation

- Treatment learning[1] fills a unique niche in data mining.
  - Previous work in treatment learning has only used one heuristic
  - No real validation tests made with this type of learning
- Previous work with software project defect detection has not used treatment learning.
  - These studies either use classic machine learners[2-5] or statistical models[6-9].
  - Results with these experiments do not give amount of code that needs to be searched.

Difficulties with Prior Results in Software Defect Detection

- The evaluations in most of these experiments are given in probability of detection and probability of false alarm.
- However, these results give no insight to the amount of code scanning required for these results.
  - Effort is defined as the fraction of the code scanned over the total code in the project.

Treatment Learning

- The process of creating a rule, or *treatment*, that predicts one specific class in a data set. When this is applied to the data set, a subset is created that will have a majority of the desired class in it.

$$R_x : x = a_1 \wedge y = a_2 \wedge b = a_3$$

Lift

- Lift is an evaluation heuristic TAR3[1] that uses to score its rules.

$$Lift_{Rx} = \frac{\sum_{i=1}^{n} f(i) * 2^{i}}{\sum_{i=1}^{N} F(i) * 2^{i}}$$

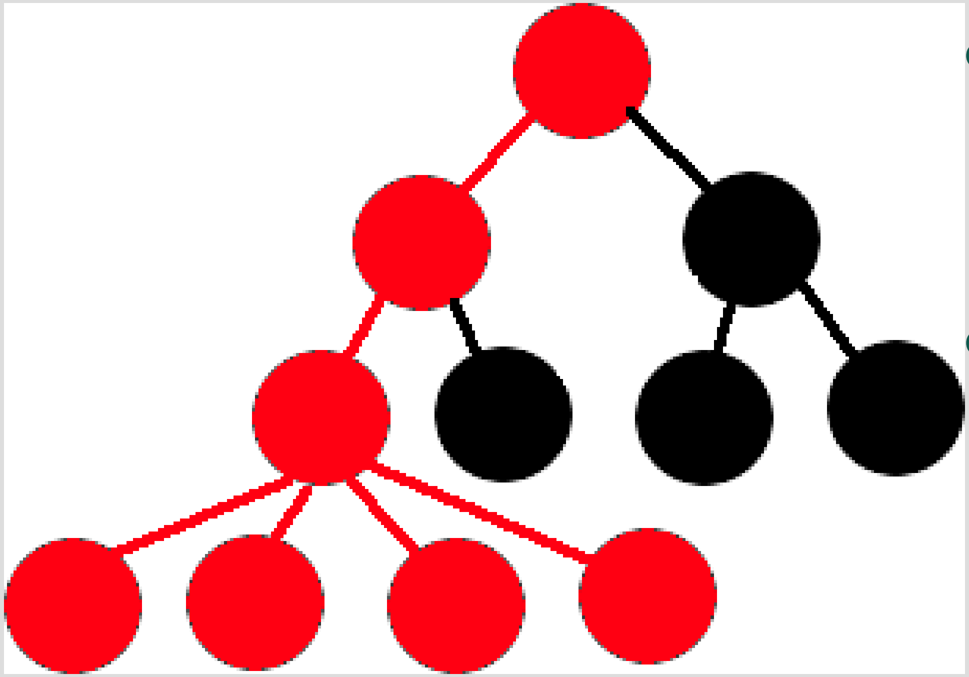
- The lift of a rule is equal to the sum of all classes in the subset over the sum of all classes in the entire set.

Which

Which

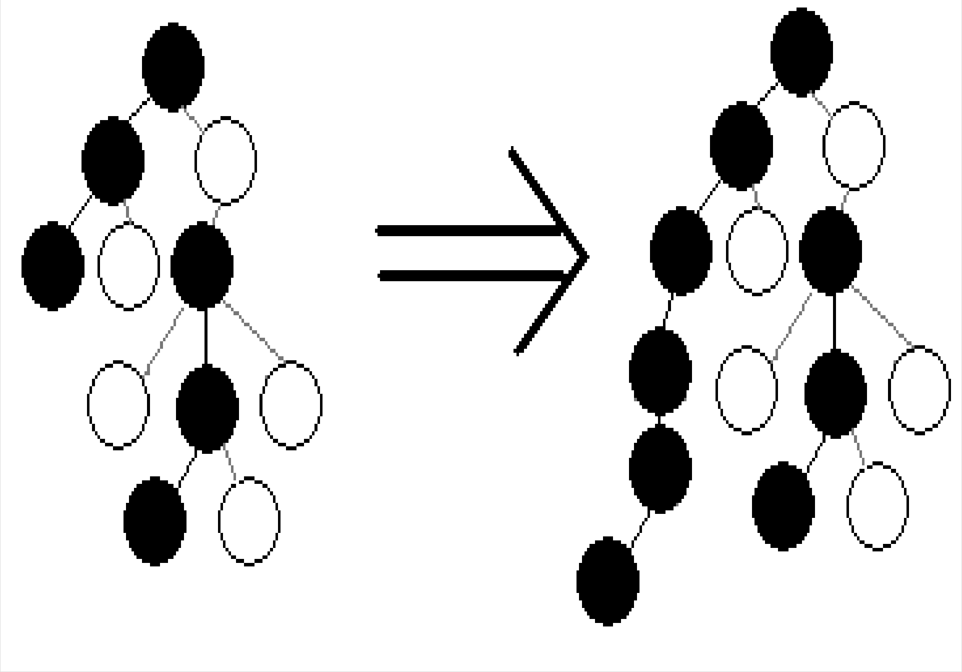
- A stochastic best-first search.
- Implemented as a Linked List.
- Learns rules by stochastically traversing the search space.
- A treatment learner that does not have a hard-wired evaluation heuristic.

Which
A Stochastic Best-First Search(BFS)



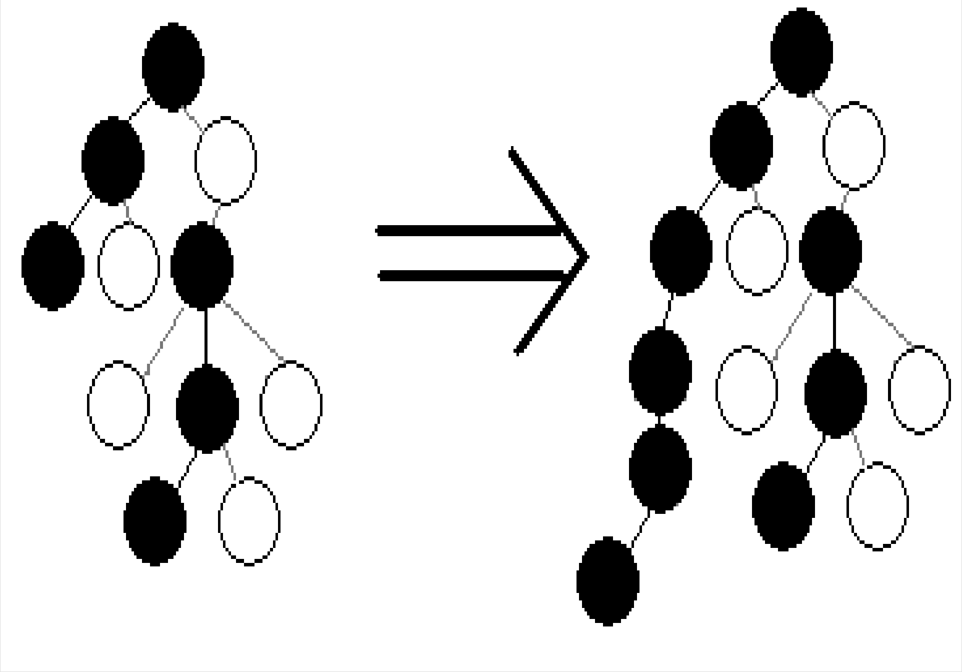
- In a BFS[10], layers are expanded and nodes are scored.
- The top N scoring nodes are then expanded another layer.

Which
A Stochastic Best-First Search(BFS)



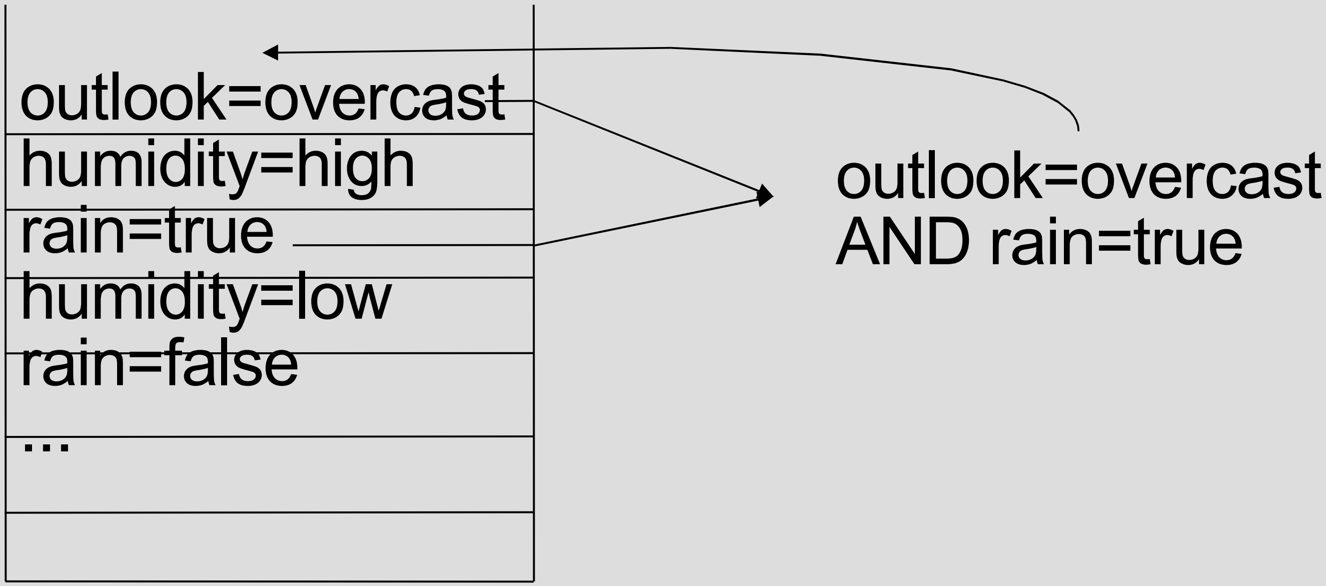
- In Which's search, subtrees are grafted based on how well they score.
- This process is called pickTwo.

Which
A Stochastic Best-First Search(BFS)



- An important difference from the standard BFS is that Which does not expand all nodes at the current level.
- Nor does it expand levels one at a time.
- This allows it to jump around the search space to find an optimum rule faster.

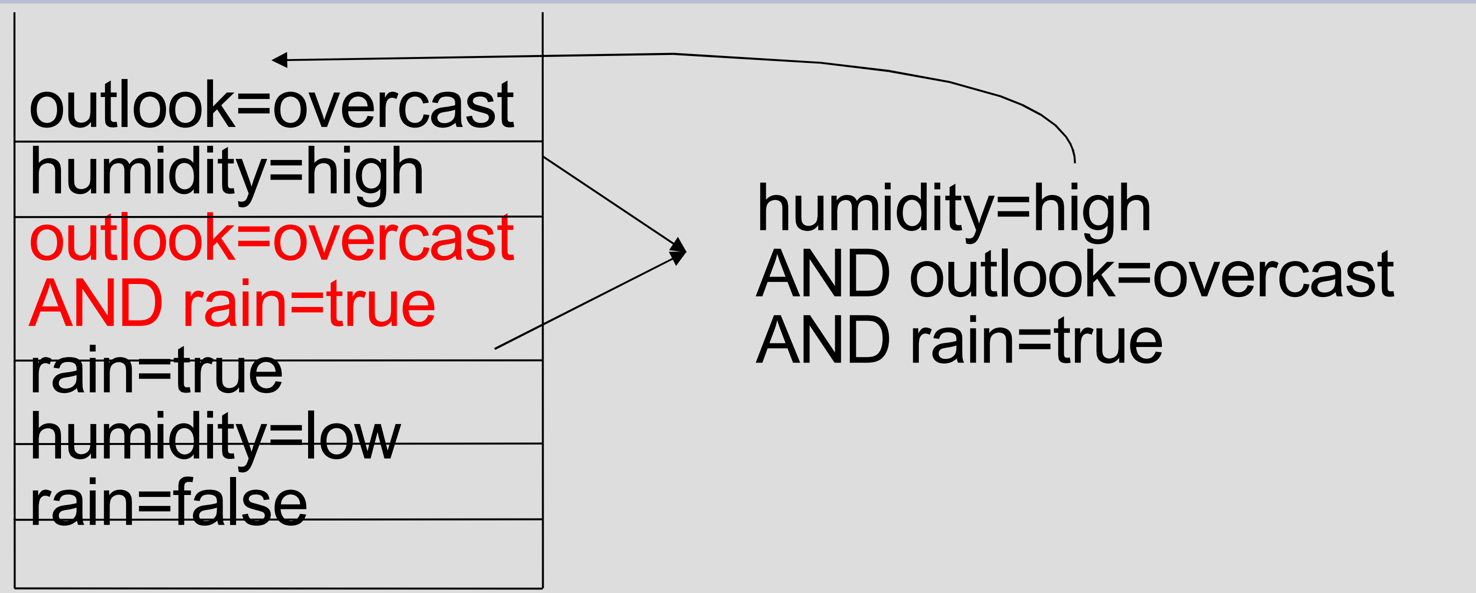
Which
A Linked List



- Instead of storing the tree and grafting subtrees, Which stores all explored paths as sets of attribute-value pairs and creates new rules by the union operation.
- Which can create new rules as both conjunctions and disjunctions.

$$(x=a_1 \vee x=a_2) \wedge (y=b_1 \vee y=b_2) \wedge (z=c_1)$$

Which
A Linked List



- Each new rule created is inserted into the list according to its score.
- If the new rule scores well, its chances of being picked are high.

Which

Learns rules by stochastically traversing the search space.

Score		Value		Probability
100		100		0.45
100		200		0.91
10		210		0.95
3	=>	213	=>	0.97
3		216		0.98
2		218		0.99
2		220		1

- In order to determine which rules are to be combined, Which stochastically picks two rules based on probability.
- Two random numbers are generated and they are compared to the probabilities in the stack.

Which

Learns rules by stochastically traversing the search space.

```
for r in rules
        sum = sum + r.score
        scores[r] = scores[r-1] + r.score
end for
for r in rules
        r.score = r.score/sum
end for
x = random(0,1)
for r in rules
        if x < scores[r] return r
        else x = x   scores[r]
        end if
end for
```

- This algorithm will create the probability table and select a rule index from the table.
- Two rule indexes are selected in this way and their corresponding rules are combined.
- This new rule is placed in the list in a position dependent on its score.

Which

A treatment learner that does not have a hard-wired evaluation heuristic.

- The method that Which scores is user-definable.
- This allows for Which to create rules that are specific to the domain it is being used in.
- Some example heuristics:
  - J48: $$H_{J48} = -\log_2\left(\frac{p}{p+n}\right)$$
  - Precision $$H_{precision} = \frac{p}{p+n}$$
  - Ripper $$H_{Ripper} = \frac{p-n}{p+n}$$
  - Balance $$H_{balance} = \frac{\sqrt{PD^2 * \alpha + (1-PF)^2 * \beta + (1-Effort)^2 * \gamma}}{\alpha+\beta+\gamma}$$

- J48 and Ripper are discussed in detail in [11] and [12] respectively

Which

A treatment learner that does not have a hard-wired evaluation heuristic.

- The classic machine learners of J48 and Ripper have hard-wired internal evaluations heuristics.
  - This creates a general process that might not perform as well as an internal evaluation heuristic that is specific to an application.
  - In other words, the class learners create rules on P that are evaluated on Q.
- Which circumvents this problem by allowing user-defined evaluation heuristics.
  - Which creates rules on Q that are evaluated on Q.

Which
Algorithm

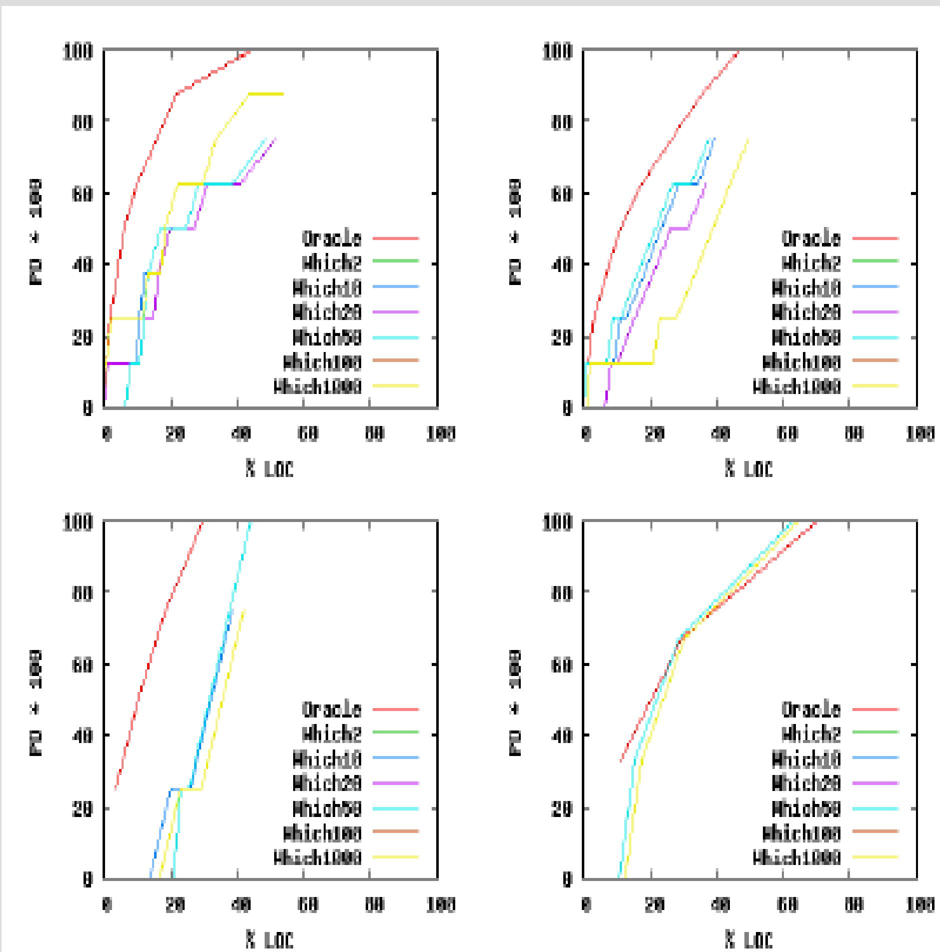
```
read test file
for each attribute a
      for each value a[v]
          score a[v]
          place a[v] in list

for i = 1 to C
      pick rule indexes r1, r2
      newR = union(r1,r2)
      score newR
      place newR in list

print list.top
```

Experiments

- Three major sections
  I. Which Parameters
  II. Which and TAR3
  III. Which on Defect Detection Data
  IV. Micro-Sampling and Which

Category I
Which Parameters

Category I
Which on Parameters



- Experiments were done on Which's maximum list size being set to a finite number.
- The numbers chosen were:
  - 10, 20, 50, 100, 1000, ∞
- Experiments were ran on defect detection data
- The results on the left show that in most cases the stack size being finite did not change the overall performance of Which
- Recommendation: Stack size set in range [50,100]

Category I
Which on Parameters

Stack Size	Data	Instances	Time(seconds)
∞	ar3	63	1.1
50	ar3	63	0.9
1000	ar3	63	0.6
∞	pc1	1109	9.2
50	pc1	1109	9.8
1000	pc1	1109	9.4
∞	mc2mod	403	3.5
50	mc2mod	403	3.3
1000	mc2mod	403	3.8

- The algorithm given earlier shows an extreme dependence of the run time on the size of the list.
- This table shows the different run times of the Which algorithm on 3 different sets.

Category II
Comparison of Which and TAR3

Category II
Comparison of Which and TAR3

- Which was ran using the lift heuristic that TAR3 uses.
- Run on 14 UCI[22] data sets using 10x10 cross-validation with 5 different discretization policies.
  - 2bins, 4bins, 8bins, 16bins, 32 bins equal frequency
- Results show that Which wins 30.5% of the time and ties TAR3 57.7% of the time.
  - Which does as well or better than TAR3 88.2% of the time.

Category II
Comparison of Which and TAR3 Results

File	Win	Loss	Tie	Sum
colic	9	0	1	9
kr-vs-kp	9	1	0	8
breast-can	8	0	2	8
vowel	9	1	0	8
anneal	7	0	3	7
autos	3	2	5	1
audiology	0	0	10	0
vote	0	0	10	0
primary-tu	2	2	6	0
segment	0	0	10	0
weather	0	0	10	0
weather.no	0	1	9	-1
heart-h	1	4	5	-3
heart-c	1	6	3	-5
Totals	49	17	73	42

File	Win	Loss	Tie	Sum
colic	7	0	3	7
breast-can	7	1	2	6
kr-vs-kp	7	1	2	6
vowel	5	4	1	1
audiology	0	0	10	0
vote	0	0	10	0
segment	0	0	10	0
anneal	3	3	4	0
weather	0	0	10	0
autos	2	3	5	-1
heart-c	1	2	7	-1
weather.no	0	1	9	-1
heart-h	1	4	5	-3
primary-tu	1	5	4	-4
Totals	34	24	82	10

File	Win	Loss	Tie	Sum
kr-vs-kp	10	0	0	10
breast-can	7	0	3	7
vowel	7	2	1	5
colic	4	1	5	3
segment	3	0	7	3
vote	2	1	7	1
heart-h	2	1	7	1
audiology	0	0	10	0
anneal	3	3	4	0
primary-tu	2	3	5	-1
autos	0	1	9	-1
weather	0	1	9	-1
weather.no	0	1	9	-1
heart-c	2	5	3	-3
Totals	42	19	79	23

File	Win	Loss	Tie	Sum
kr-vs-kp	10	0	0	10
colic	9	0	1	9
breast-ca	8	0	2	8
segment	7	0	3	7
anneal	3	0	7	3
heart-c	0	0	10	0
vowel	0	0	10	0
weather	0	0	10	0
primary-t	2	3	5	-1
audiology	0	1	9	-1
weather.n	0	1	9	-1
heart-h	0	2	8	-2
autos	0	2	8	-2
vote	0	2	8	-2
Totals	39	11	90	28

File	Win	Loss	Tie	Sum
colic	10	0	0	10
kr-vs-kp	10	0	0	10
anneal	8	0	2	8
breast-ca	7	1	2	6
segment	5	0	5	5
vote	1	0	9	1
audiology	0	0	10	0
vowel	0	0	10	0
autos	1	2	7	-1
heart-c	0	1	9	-1
weather	0	1	9	-1
primary-t	1	4	5	-3
Totals	43	9	68	34

Bins	Win	Loss	Tie	Sum
2	84	32	84	52
4	74	33	93	41
8	55	24	81	31
16	39	11	90	28
32	43	12	85	31
Totals	207	80	392	137

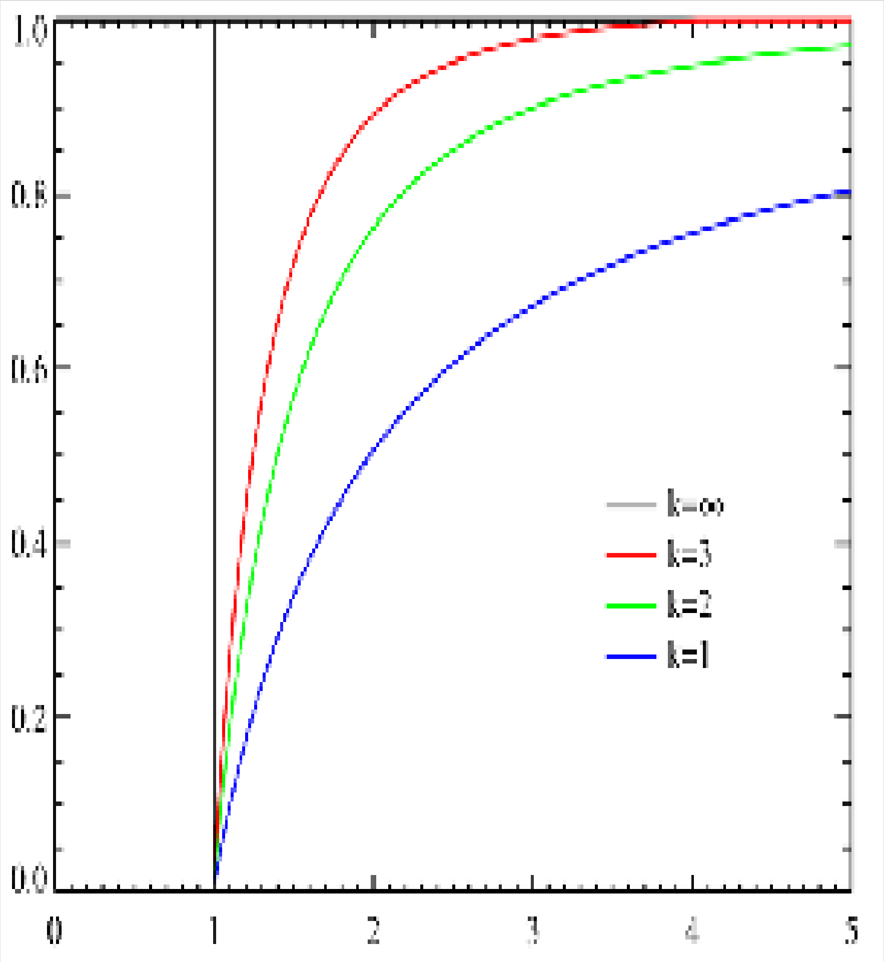
Category III

Which on Defect Detection Data

Category III
Critical Information

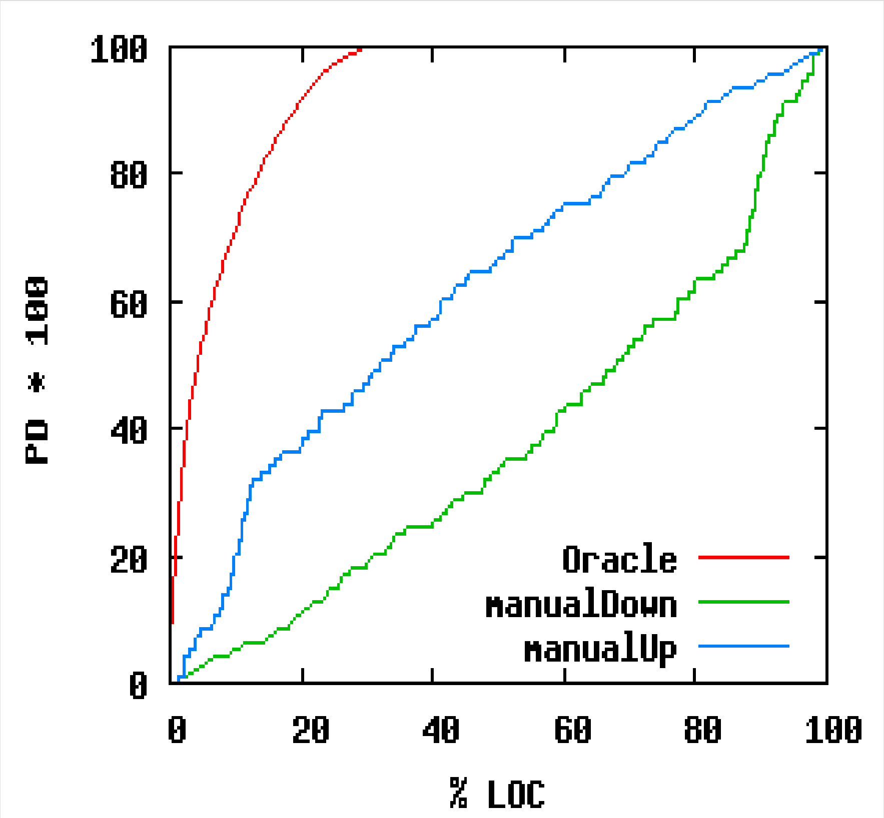
- Distribution of faults in software projects.
- Receiver-Operating Characteristic Curves
- The “Koru” Diagram

Category III
Fault Distribution



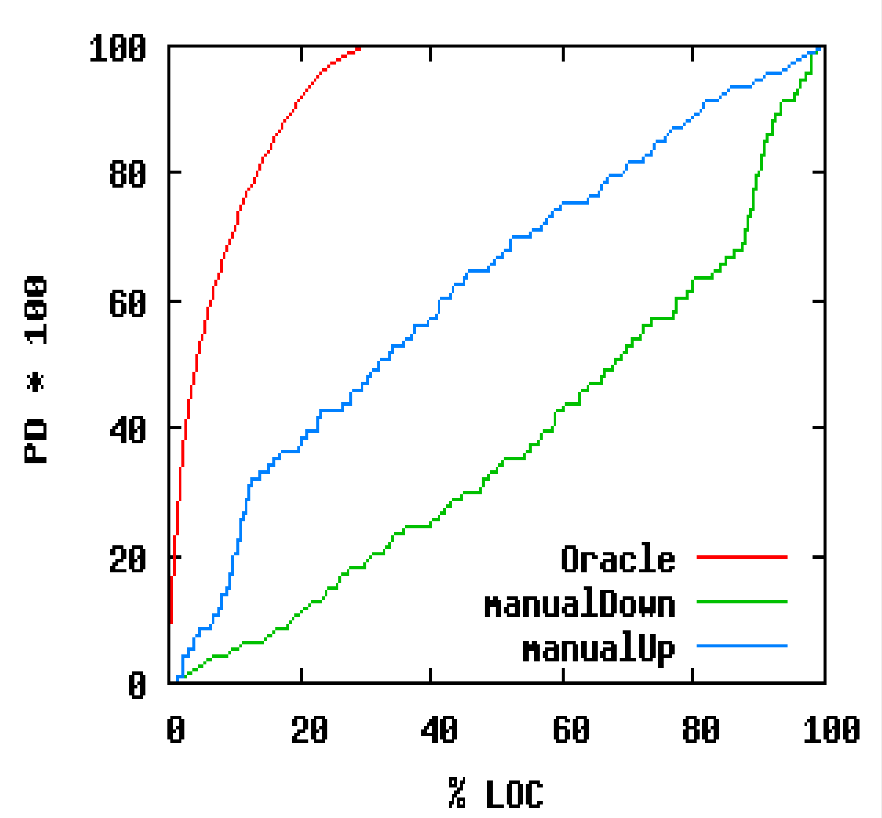
- Several studies have been made on the distribution of faults in a software project[13-17].
- This distribution closely follows the Pareto Distribution[18], or “80-20” rule.
  - This says that 80% of the faults lie in 20% of the code.

Category III
The “Koru” Diagram[19]



- Specialized diagram to illustrate the “path” a detector takes over the data set to reach its final score.
- Created by taking the subset of instances the detector classified as true and sorting them by their LOC attribute.
- Starting from (0,0), increment the PD if a true was properly classified as true and increment the %LOC at every instance.

Category III
The "Koru" Diagram
Special Detectors



- Oracle
  - The perfect detector
  - Classifies all defective modules perfectly.
- ManualDown
  - Sort the instances in descending order by LOC and classify all as true
- ManualUp
  - Sort the instances in ascending order by LOC and classify all as true.

Category III
The “Koru” Diagram

- Evaluation Metric
  - The area under the detector's path divided by the area under the oracle's path.
- The evaluation Metric is used to see how much like the oracle's path the detector's path is.
- Extend the end point (x,y) of all detectors to the point (100,y).
  - Ensures the oracle has the most area.

Category III
The "Koru" Diagram

- Which's heuristic in this category is balance.
  - Balance attempts to create rules that are close as possible to (0,100) in the Koru Diagram.
  - For these experiments:
    - $\alpha = 1$
    - $\beta = 1000$
    - $\gamma = 0$
- PD and effort too closely correlated to take both into account.
- These scalars make low PFs much more preferred than higher PDs

Category III
Experiments

- Which on MDP[20] data.
- Which on Turkey data.
- Which on AT&T data.

Category III
Experiments on MDP Data

- These experiments were done on 7 project files with 10x3 cross-validation
- The classic machine learners of J48, NaiveBayes, and Ripper are represented here.
- Results divided up into 3 classes:
  1. Which > manualUp > classic learners
     - 5/7
  2. Which > all
     - 1/7
  3. ManualUp > classic learners > Which
     - 1/7

Category III
Experiments on MDP Data
Results

learner	mean	Equal	Description
which2	68.1		
manualUp	59.8	=	
nBayes	52.1	<	
manualDown	47.6	<	
which8	11.4	<	
jRip	5.8	=	
j48	0.1	=	
which8loc	0.0	<	
which4loc	0.0	=	
which4	0.0	>	
which2loc	0.0	<	

learner	mean	Equal	Description
which2	76.0		
manualUp	67.6	<	
nBayes	61.9	<	
which4	52.9	<	
manualDown	43.3	<	
j48	27.8	<	
jRip	21.3	<	
which8loc	0.0	<	
which8	0.0	<	
which4loc	0.0	<	
which2loc	0.0	=	

learner	mean	Equal	Description
which2	87.3		
nBayes	64.2	<	
manualUp	64.2	=	
which4	47.8	<	
manualDown	47.6	=	
which8	46.7	=	
j48	23.1	<	
jRip	17.7	<	
which2loc	6.6	<	
which8loc	0.0	<	
which4loc	0.0	<	

learner	mean	Equal	Description
manualUp	74.3		
nBayes	55.9	<	
manualDown	42.8	<	
j48	43.7	=	
jRip	28.5	<	
which8	21.9	<	
which4	5.6	<	
which8loc	0.0	<	
which4loc	0.0	>	
which2loc	0.0	=	
which2	0.0	>	

- Each row of the table on the left is a different category.

Category III
Experiments on Turkey Data

- The Turkey data consisted of 3 data sets with a 10x3 cross-validation.
- The classic machine learners of J48, NaiveBayes, and Ripper are represented.

Category III

Experiments on Turkey Data

Results

learner	mean	Equal	Description
manual	54.7		
launam	52.6	<	
which8	42.7	<	
which4	41.8	<	
which2	40.4	>	
nBayes	34.4	<	
j48	47.8	>	
jRip	0.2	<	
which2loc	0.1	>	
which8loc	0.0	<	
which4loc	0.0	–	

learner	mean	Equal	Description
which2	58.6		
nBayes	56.2	<	
launam	56.2	–	
manual	56.5	–	
which2loc	55.9	–	
which4	49.3	<	
jRip	47.3	–	
which8	42.6	<	
j48	38.8	–	
which8loc	0.0	<	
which4loc	0.0	–	

learner	mean	Equal	Description
which4	71.4		
manual	69.6	–	
which2	67.6	–	
nBayes	54.1	<	
j48	56.1	–	
launam	56.5	<	
jRip	55.0	>	
which8loc	0.0	>	
which8	0.0	–	
which4loc	0.0	–	
which2loc	0.0	–	

- Results show that Which wins 2/3 and loses to manualUp and manualDown in the other case.

Category III
Experiments on AT&T Data.

- AT&T data consists of one project with 35 releases.
  - Experiments were done using releases (1-(n-1)) for train and release n for test.
- Classic learners of J48, NaiveBayes, and Ripper are represented.

Category III
Experiments on AT&T Data.
Results

learner	mean	Equal	Description
manualUp	60.4		
nBayes	48.5	<	
which10	45.9	<	
manualDown	44.7	<	
micro10010	33.3	<	
jRip	32.6	<	
j48	30.7	=	

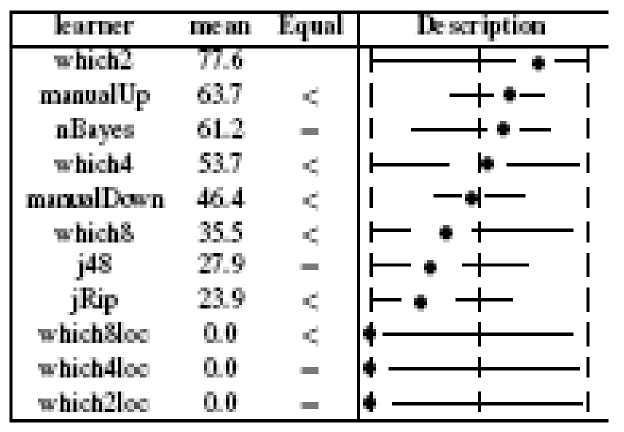
Release 4

learner	mean	Equal	Description
manualUp	63.1		
which10	51.9	<	
nBayes	48.5	=	
manualDown	48.1	=	
j48	30.7	<	
micro10010	28.1	>	
jRip	27.8	<	

Release 24

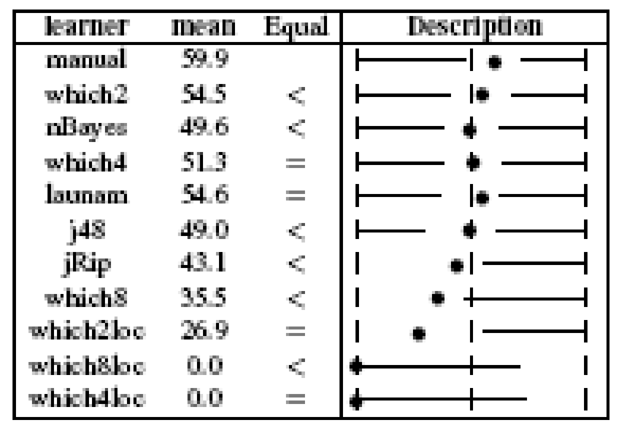
- Typically performs better than the classic machine learners but loses to manualUp and/or manualDown.

Category III
Overall Results



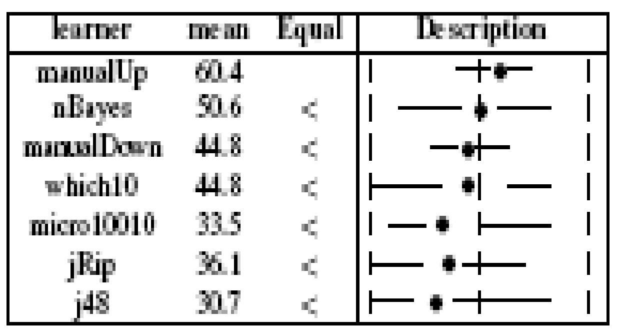
learner	mean	Equal	Description
which2	77.6		
manualUp	63.7	<	
nBayes	61.2	=	
which4	53.7	<	
manualDown	46.4	<	
which8	35.5	<	
j48	27.9	=	
jRip	23.9	<	
which8loc	0.0	<	
which4loc	0.0	=	
which2loc	0.0	=	

MDP



learner	mean	Equal	Description
manual	59.9		
which2	54.5	<	
nBayes	49.6	<	
which4	51.3	=	
launam	54.6	=	
j48	49.0	<	
jRip	43.1	<	
which8	35.5	<	
which2loc	26.9	=	
which8loc	0.0	<	
which4loc	0.0	=	

Turkey



learner	mean	Equal	Description
manualUp	60.4		
nBayes	50.6	<	
manualDown	44.8	<	
which10	44.8	<	
micro10010	33.5	<	
jRip	36.1	<	
j48	30.7	<	

AT&T

Category IV
Micro-Sampling and Which

- Micro-Sampling:
  - Create an even distribution of a binary class set with N instances in the set.
- Micro-Sampling experiments done on MDP and Turkey data sets using a 10x3 cross-validation.

Category IV
Micro-Sampling and Which Results

learner	mean	Equal	Description
which2	70.9		
micro20	67.0	<	
micro100	65.0	<	
micro30	64.1	=	
micro50	64.8	=	
micro75	64.1	=	

- Results show that while Which2 is best performer, it is hardly a stand-out performance.
- Interesting that Micro20, the most limiting policy, performs better than the other micro-sampling policies.
- All perform very close to the same.

Findings

- Treatment learning does indeed work over cross-validation.
- Which, a simpler algorithm than TAR3, still performs just as well or better 88% of the time.
- Treatment learners have a place in the field of defect detection in software projects.
- Creating rules with the true goal in mind drastically improves performance.
- Micro-Sampling is an effective sampling policy that does not hurt the performance of Which.

Future Work

- Improving the run-time of the Which algorithm.
- Fine-tuning the Which configuration parameters.
- Converting Which to a classification learner.
- Discovering better scoring heuristics for Which to learn with in defect detection data.

References

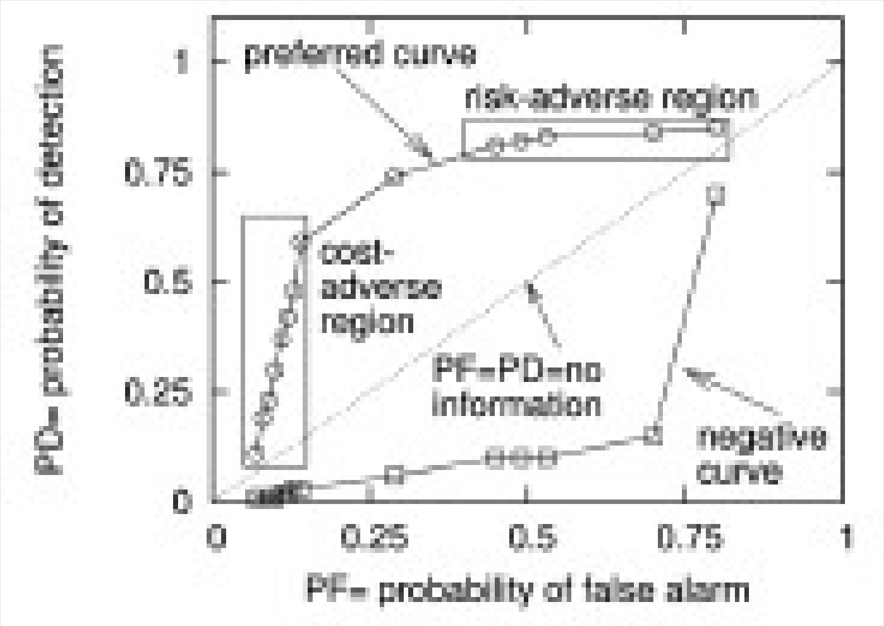
- [1]Ying Hu. Treatment learning: Implementation and application. Master’s thesis, University of Brittish Columbia, Brittish Columbia, Canada, May 2003.
- [2]Lan Guo, Yan Ma, Bojan Cukic, and Harshinder Singh. Robust prediction of fault-proneness by random forests. In ISSRE ’04: Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE’04), pages 417–428, Washington, DC, USA, 2004. IEEE Computer Society.
- [3]Tim Menzies, Jeremy Greenwald, and Art Frank. Data mining static code attributes to learn defect predictors. IEEE Transactions on Software Engineering, 33(1):2–13, 2007.
- [4]Tim Menzies, Justin Di Stefano, Kareem Ammar, Kenneth McGill, Pat Callis, Robert (Mike) Chapman, and John Davis. When can we test less? In METRICS ’03: Proceedings of the 9th International Symposium on Software Metrics, page 98, Washington, DC, USA, 2003. IEEE Computer Society.
- [5]Tim Menzies and Justin S. Di Stefano. How good is your blind spot sampling policy? Hase, 00:129–138, 2004.
- [6]Erik Arisholm and Lionel C. Briand. Predicting fault-prone components in a java legacy system. In ISESE ’06: Proceedings of the 2006 ACM/IEEE international symposium onEmpirical software engineering, pages 8–17, New York, NY, USA, 2006. ACM.
- [7]Robert M. Bell. Predicting the location and number of faults in large software systems. IEEE Trans. Softw. Eng., 31(4):340–355, 2005. Member-Thomas J. Ostrand and Fellow-Elaine J. Weyuker.
- [8]John C. Munson and Taghi M. Khoshgoftaar. The detection of fault-prone programs. IEEE Trans. Softw. Eng., 18(5):423–433, 1992.
- [9]Thomas J. Ostrand, Elaine J. Weyuker, and Robert M. Bell. Where the bugs are. In ISSTA ’04: Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis, pages 86–96, New York, NY, USA, 2004. ACM.
- [10]Judea Pearl. Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference. Morgan Kaufmann, September 1988.
- [11]J. Ross Quinlan. C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

References
(continued)

- [12] William W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, Proc. of the 12th International Conference on Machine Learning, pages 115–123, Tahoe City, CA, July 1995. Morgan Kaufmann.
- [13]Les Hatton. Reexamining the fault density-component size connection. IEEE Softw., 14(2):89–97, 1997.
- [14]J. Juran. Juran’s Quality Control Handbook. pub-mcgraw-hill, 4th edition, 1988.
- [15]Thomas J. Ostrand and Elaine J. Weyuker. The distribution of faults in a large industrial software system. SIGSOFT Softw. Eng. Notes, 27(4):55–64, 2002.
- [16]T.-J. Yu, V. Y. Shen, and H. E. Dunsmore. An analysis of several software defect models. IEEE Trans. Softw. Eng., 14(9):1261–1270, 1988.
- [17]Hongyu Zhang. On the distribution of software faults. IEEE Trans. Soft. Eng., xxx(xxx):1–2, 2008.
- [18]M. O. Lorenz. Methods of measuring the concentration of wealth. Puiblications of the American Statistical Association, 9(70):209–219, 1905.
- [19]A. Gunes Koru, Dongsong Zhang, and Hongfang Liu. Modeling the effect of size on defect proneness for open-source software. In PROMISE ’07: Proceedings of the Third International Workshop on Predictor Models in Software Engineering, page 10, Washington, DC, USA, 2007. IEEE Computer Society.
- [20]Tim Menzies. Promise data. http://promisedata.org.
- [21]C. Blake and C. Merz. Uci repository of machine learning databases, 1998.

Experiments

Receiver-Operating Characterstic Curves



$$PD = \frac{p}{P}$$

$$PF = \frac{n}{N}$$

- Creates a space for evaluating the performance of a learner.
- Measures the true positive rate compared to the false positive rate, or PD compared to PF.