两篇论文

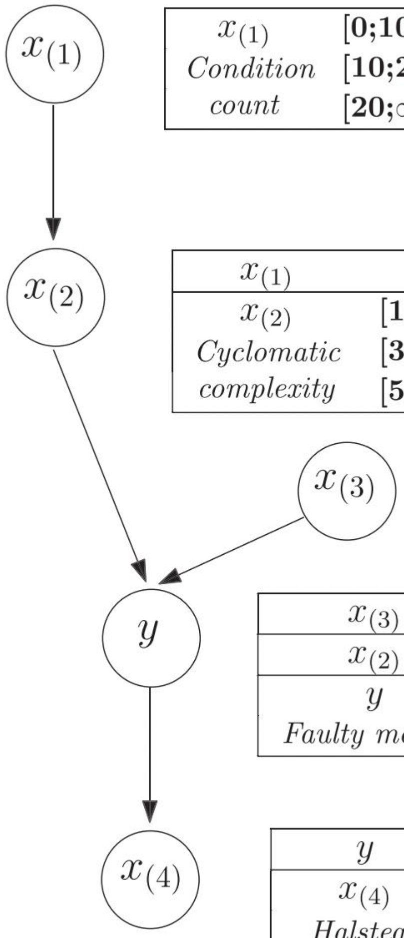
SCIENCE AND TECHNOLOGY

《Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers》	2013	TSE
《On the relative value of data resampling approaches for software defect prediction》	2019	ESE

GRADUATION DEFENSE

Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers

2013 TSE



$x_{(1)}$ Condition count		
	[0;10)	0.30
	[10;20)	0.45
	[20;∞)	0.25

$x_{(1)}$		**[0;10)**	**[10;20)**	**[20;∞)**
$x_{(2)}$ Cyclomatic complexity	**[1;3)**	0.40	0.20	0.55
	[3;5.5)	0.55	0.25	0.35
	[5.5;∞)	0.05	0.55	0.10

$x_{(3)}$ LOC comment		
	[0;17)	0.30
	[17;∞)	0.70

$x_{(3)}$		**[0;17)**			**[17;∞)**		
$x_{(2)}$		**[1;3)**	**[3;5.5)**	**[5.5;∞)**	**[1;3)**	**[3;5.5)**	**[5.5;∞)**
y Faulty module	**0**	0.95	0.88	0.85	0.90	0.87	0.91
	1	0.05	0.12	0.15	0.10	0.13	0.09

y		**0**	**1**
$x_{(4)}$ Halstead error estimate	**[0;0.125)**	0.25	0.25
	[0.125;0.2)	0.15	0.50
	[0.2;0.285)	0.10	0.05
	[0.285;∞)	0.50	0.20

Classification example

Condition count, $x_{(1)} \in [20;\infty)$
Cyclomatic complexity, $x_{(2)} \in [3;5.5)$
LOC comment, $x_{(3)} \in [17;\infty)$
Halstead error estimate, $x_{(4)} \in [0.2;0.285)$

Joint probability (Eq. 2)

$$P\left(y=0, x_{(1)} \in [20;\infty), x_{(2)} \in [3;5.5), x_{(3)} \in [17;\infty), x_{(4)} \in [0.2;0.285)\right) = 0.87 \cdot 0.25 \cdot 0.35 \cdot 0.70 \cdot 0.10 = 0.00533$$

$$P\left(y=1, x_{(1)} \in [20;\infty), x_{(2)} \in [3;5.5), x_{(3)} \in [17;\infty), x_{(4)} \in [0.2;0.285)\right) = 0.13 \cdot 0.25 \cdot 0.35 \cdot 0.70 \cdot 0.05 = 0.00040$$

Conditional probability

$$P\left(y=0 | x_{(1)} \in [20;\infty), x_{(2)} \in [3;5.5), x_{(3)} \in [17;\infty), x_{(4)} \in [0.2;0.285)\right) = \frac{0.00533}{0.00533+0.00040} = 0.93$$

$$P\left(y=1 | x_{(1)} \in [20;\infty), x_{(2)} \in [3;5.5), x_{(3)} \in [17;\infty), x_{(4)} \in [0.2;0.285)\right) = \frac{0.00040}{0.00533+0.00040} = 0.07$$

Fig. 2. Bayesian network classification by example.

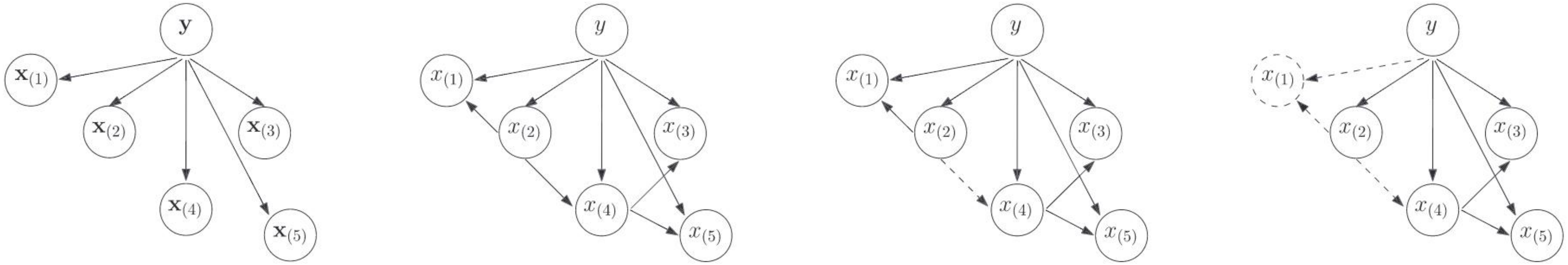
Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers

2013 TSE

"Simple models are often not outperformed by more complex ones and that in such cases, the simpler model should be selected."

Augment Naive Bayes

Inspired several modifications to relax the conditional independence assumption.



(a) Naive Bayes network

(b) Selective Tree Augmented Naive Bayes network

(c) Selective Forest Augmented Naive Bayes network

(d) Selective Forest Augmented Naive Bayes with Discarding network

Fig. 3. Examples of Bayesian network structures.

Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers

2013 TSE

TABLE 1
Augmented Naive Bayes Approach: Different Operators

Dependency Discovery Operators	**Description**
Selective Augmented Naive Bayes (SAN)	Operator which connects the class node with the attributes it depends on. Starting with an empty set, SAN greedily searches for possible arcs from the class variable y to other variables $x_{(j)}$ optimizing a certain quality measure, see Table II. The selected variable together with an associated arc are added to the network which is then passed to one of the *augmenting* operators. The latter establishes the dependencies among *all* variables $x_{(j)}$, irrespective of their connection with the class node.
Selective Augmented Naive Bayes with Discarding (SAND)	Operator which connects the class node with the attributes it depends on, as the SAN operator does. The difference, however, lies in that SAND will discard all variables which are not dependent on the class node before passing the network to one of the *augmenting* operators. As a result, the discarded variables are not part of the network; the difference between a network resulting from the SAN operator and SAND operator is illustrated in Fig. 3c and 3d respectively. Dashed lines indicate absent arcs or nodes in a network.
Augmenting Operators	**Description**
Tree-Augmenter	Operator which builds the maximum spanning tree among a given set of attributes. The algorithm is based on a method developed by Chow and Liu [22], but differs in the way how the mutual information is calculated. Sacha uses the conditional or unconditional probability of $x_{(j)}$ and $x_{(j')}$ depending on whether there exists an arc between the class node and the attribute (see formula 5). The resulting network can be regarded as a generalization of the network obtained using a TAN classifier, *not* requiring all variables to be connected with the class variable.
Forest-Augmenter	Operator which is also used to establish dependencies between attributes, but allowing for more flexibility. The forest-augmenter can create dependencies between variables in the form of a number of disjoint trees not requiring the existence of an undirected path between two attributes that does not pass through the class node. The difference between both augmenting operators is shown in Fig. 3b and 3c.

Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers

2013 TSE

Compared Method

- TAN: Tree Augmented Naive Bayes,
- FAN: Forest Augmented Naive Bayes,
- STAN: Selective Tree Augmented Naive Bayes,
- STAND: Selective Tree Augmented Naive Bayes with Discarding,
- SFAN: Selective Forest Augmented Naive Bayes,
- SFAND: Selective Forest Augmented Naive Bayes with Discarding.

- K2
- MHHC

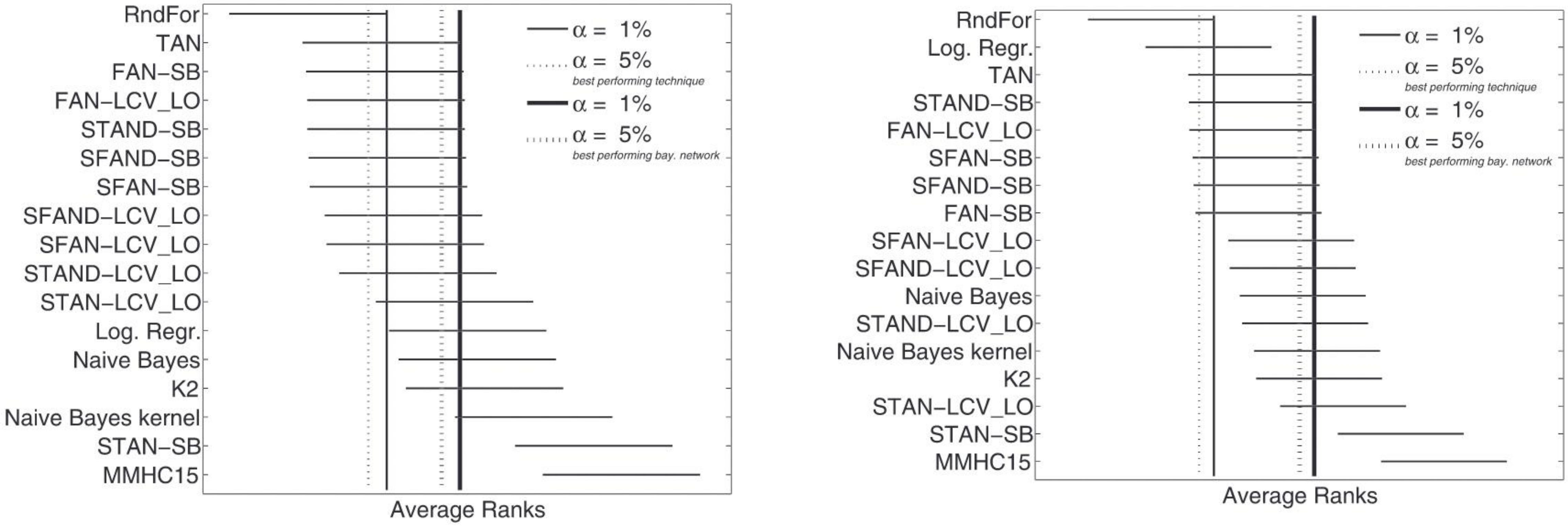
TABLE 2
Augmented Naive Bayes Approach: Different Quality Measures

Quality Measures	**Description**
Standard Bayesian measure (SB)	This *global* quality measure was first proposed by [13] and is proportional to the posterior probability distribution $p(G, \Theta \vert D_{trn})$ with an added penalty term for network size. The network size or dimensionality is defined as the number of free parameters required to fully specify the joint probability distribution, $P_B(x_{(1)}, ..., x_{(n)})$.
Local Leave-One-Out-Cross Validation (LOO)	This *local* quality measure calculates iteratively the class probability conditional on the data, $P(y \vert x_{(1)}, \ldots, x_{(n)})$, using all observations minus one to estimate all parameters. The remaining observation is then used to assess the network quality in the class node [83].

all of the above procedures adopt a quality measure to assess the fitness of a network given the data.

Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers

2013 TSE



(a) AUC

(b) H-measure with $\beta(2,2)$

Fig. 5. Ranking of software fault prediction models for (a) the AUC and (b) H-measure with $\beta(2,2)$ using the posthoc Nemenyi test.

On the relative value of data resampling approaches for software defect prediction

2010 ESE

Approaches

SMOTE

Safe-level SMOTE

Borderline-SMOTE

ADASYN

Random Over Sampling

Random Under Sampling

Pfp

the suitable ratio of defect and clean instances

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

$$Recall(pd) = \frac{TP}{TP + FN}$$

$$pf = \frac{FP}{TN + FP}$$

$$balance(bal) = 1 - \frac{\sqrt{(0 - pf)^2 + (1 - pd)^2}}{\sqrt{2}}$$

$$g - mean = \sqrt{\frac{TP}{TP + FN} * \frac{TN}{TN + FP}}$$

On the relative value of data resampling approaches for software defect prediction

2010 ESE

- RQ1: Is the Effect of Resampling Approaches on Defect Prediction Models Statistically and Practically Significant Regarding the Evaluation Metric and Dataset Used?

- RQ2: To What Degree (Percentage of Fault-Prone Modules Pfp) Should the Original Dataset be Resampled or Balanced?

- RQ3: Which are the High-performing Resampling Approaches for Defect Prediction?

On the relative value of data resampling approaches for software defect prediction

2010 ESE

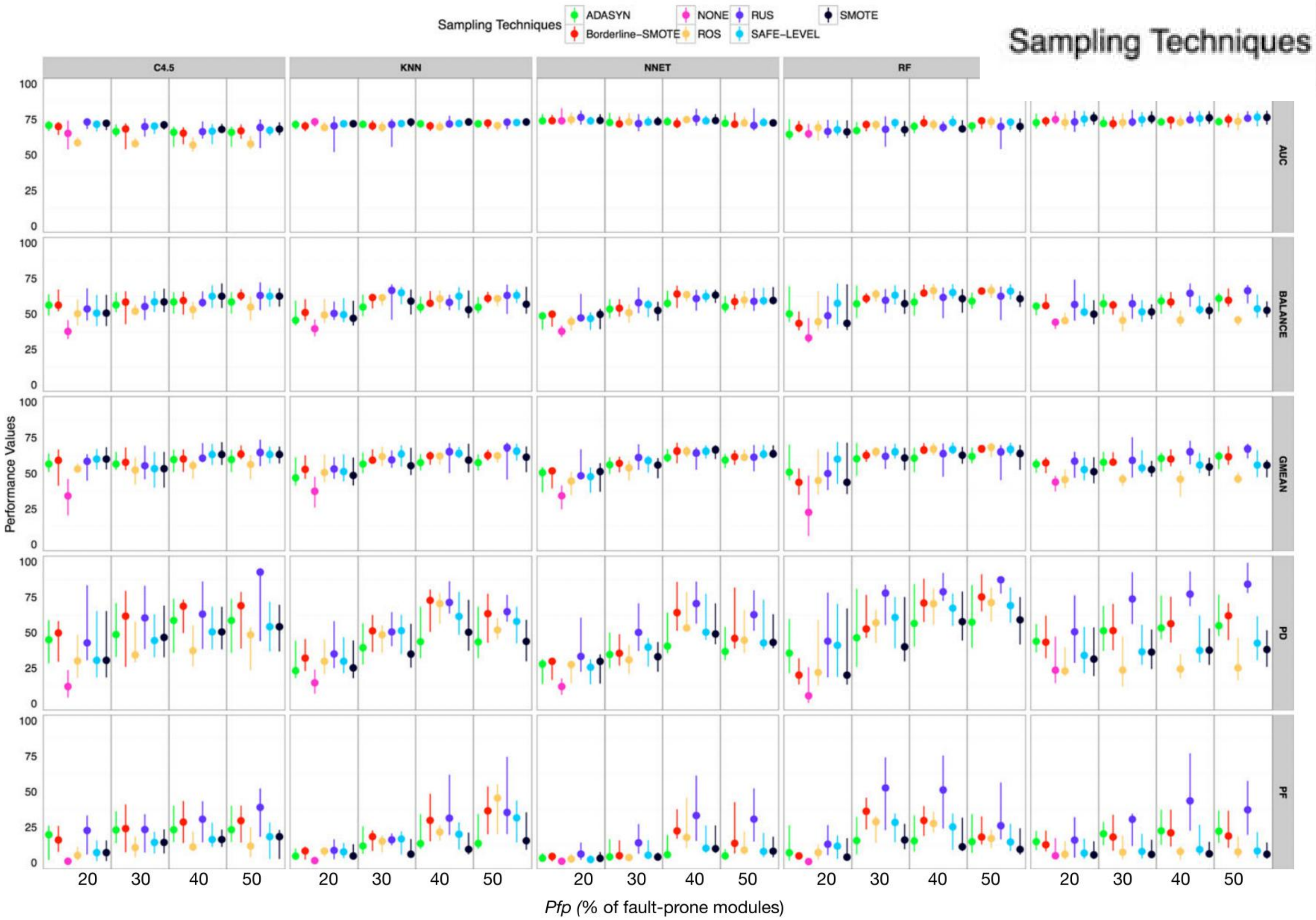


Fig. 6 Quartile charts of performance values for all sampling techniques on different Predictive Models at different (*Pfp*) values across all ten Process metrics datasets

On the relative value of data resampling approaches for software defect prediction

2010 ESE

AUC

#	Sampling	Wins	Losses	Wins-Losses
1	RUS	63	4	59
2	SMOTE	30	29	1
3	BORDERLINE	29	29	0
4	SAFE-LEVEL	31	31	0
5	ADASYN	24	27	−3
6	ROS	30	41	−11
7	NONE	19	65	−46

g-mean

#	Sampling	Wins	Losses	Wins-Losses
1	RUS	61	3	58
2	BORDERLINE	51	20	31
3	SAFE-LEVEL	58	28	30
4	ROS	65	40	25
5	SMOTE	44	29	15
6	ADASYN	35	45	−10
7	NONE	5	154	−149

balance

1	RUS	63	3	60
2	BORDERLINE	52	18	34
3	ROS	67	38	29
4	SAFE-LEVEL	53	29	24
5	SMOTE	45	30	15
6	ADASYN	34	47	−13
7	NONE	4	153	−149

pd

1	RUS	143	0	143
2	ROS	87	51	36
3	BORDERLINE	69	43	26
4	SAFE-LEVEL	56	60	−4
5	SMOTE	41	62	−21
6	ADASYN	34	65	−31
7	NONE	5	154	−149

pf

1	NONE	94	6	88
2	SMOTE	30	20	10
3	SAFE-LEVEL	16	23	−7
4	ROS	11	25	−14
5	BORDERLINE	19	35	−16
6	ADASYN	17	34	−17
7	RUS	10	54	−44

Table 5 Performance in Terms of wins, losses, and wins-losses aggregated across all prediction models and 20 datasets per each performance measure

Higher wins and wins-losses indicate higher performance, where as lower losses indicate higher performance. The ranks are ordered by wins-losses