The Model For Privacy Implications Of Qulity Metrics Flow Results In Data Mining

P Rajesh Kumar, Reddy sager A.C
[1]Computer Applications In Jawaharlal Nehru Technological University, Hyderabad
[2]Jawaharlal Nehru Technological University, Anantapur

Abstraction

Privacy-preserving data mining has concentrated on obtaining valid results when the input data is private. An extreme example is Secure Multiparty Computation-based methods, where only the results are revealed. Data mining on metrics has become important due to the fact that there are volumes of metrics functionalities are now available holding a wealth of valuable result information. Several metrics have been proposed for recognition of relationships between elements of two objects. Many of these methods select a number of such metrics and combine them to extract existing mappings. We present a method for selection of ore effective metrics –based on data mining techniques. Furthermore, by having a set of metrics, we suggest a data-mining-like means for combining them into a better alignment. Do the results themselves violate privacy? This paper explores this issue, developing a framework under which this question can be addressed. Metrics
Are proposed, along with analysis that those metrics are consistent in the face of apparent problems. By the following objects we explored various distance metrics and their behavior and developed a new distance metric. And it provides an efficient way of computation using P-trees.

Key words: *Privacy, Inference, computer machinery and intelligence (CMI)*

1. Introduction

Metrics are objects that represent business measures and key performance indicators. They are calculations to be performed on data stored in the database. Data Mining has emerged at the confluence of artificial intelligence, statistics, and databases as a technique for automatically discovering summary knowledge in large datasets. Data mining first requires understanding the data available, developing questions to test, and finally drawing conclusions from data analytic results. Metrics are some parameters or measures of quantitative assessment used for measurement or comparison in a given context. A metric for all practical purpose is just a variable. It needs to be clearly defined. The number of metrics needs to be kept under control to ensure that the measuring task is achievable. It is thus reasonable to expect that as the context changes, the metrics would change. Literature has not defined Data mining metrics as such. Data mining metrics may be defined as a set of measurements which can help in determining the efficacy of a Data mining Method / Technique or Algorithm. They are important to help take the right decision as like as choosing the right data mining technique or algorithm. Data mining metrics generally fall into the categories of accuracy, reliability, and usefulness. Accuracy is a measure of how well the model correlates an Outcome with the attributes in the data that has been provided.

This paper presents a start on methods and metrics for evaluating the privacy impact of data mining models. While the methods are preliminary, they provide a cross-section of what needs to be done, and a demonstration of techniques to analyze privacy impact. Work in privacy-preserving data mining has shown how to build models when the training data is kept from view; the full impact of privacy- reserving data mining will only be realized when we can guarantee that the resulting models do not violate privacy. Since the classifier uses some public information as input, it would appear that

The insurer could *improve* an estimate of the disease probability by repeatedly probing the classifier with the known public information and "guesses" for the unknown information. At first glance, this appears to be a privacy violation. Surprisingly, given reasonable assumptions on the external knowledge available to an adversary we can *prove* the adversary learns nothing new.

We assume that data falls into three classes:

• **Public Data :**(*P*) This data is accessible to every one including the adversary.
• **Private/Sensitive Data :**(*S*) We assume that this kind of data must be protected: The values should
Remain unknown to the adversary.
• **Unknown Data :**(*U*) This is the data that is not known to the adversary, and is not *inherently* sensitive. However, before disclosing this data to an adversary (or enabling an adversary to estimate it, such as by publishing a data mining model) we must show that it does not enable the adversary to discover sensitive data.

2. THE MODEL FOR PRIVACY IMPLICATIONS OF QULITY METRICS FLOW RESULTS IN DATA MINING

2. METRICS Architecture

The METRICS architecture shown in Figure 1 is a specific implementation of a distributed, client-server information gathering system. The EDA tools, which are the data sources, have a thin transmitter client embedded in script wrappers surrounding the tool or actually embedded (as an API) inside the tool's executable for more flexibility. Both the wrapper mode transmitter and the API mode transmitter allow transparent data collection within design processes. The tools – which can be located anywhere on an intranet or even the internet – broadcast in real-time as they run using standard network protocols to a centralized server which s at-tached to a *data warehouse*. The messages transmitted are encoded in industry-standard XML format, which is straightforwardly and robustly read, written and stored directly by data warehouses. Once the data is stored, reports or determining-based predictions can be generated.

These can be accessed from standard web browsers run from any authorized remote machine. Further details of METRICS are given in [10].Wrapper/ Embedded

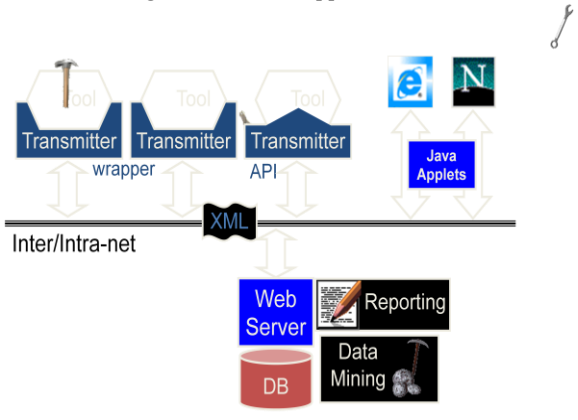


Figure 1: METRICS architecture.

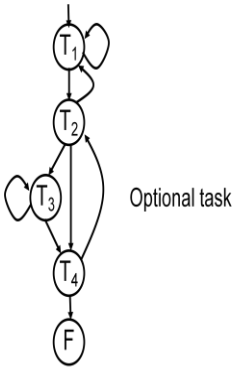
2.1Extensions to the METRICS Architecture

In this paper, we focus on two recent extensions to the METRICS architecture. First, we propose a chemo that is able to capture historical flow information, even for highly iterative or ECO-oriented flows with multiple potential "backward edges". The previous METRICS architecture lacked information related to design flows, and hence it was not possible to optimize design flows. Our new schema allows us to predict, for example, how many times a certain tool or optimization sub flow must be repeated before an acceptable result is obtained. Second, we integrate off-the-shelf data mining techniques and assess the ability of such tools to identify variable sensitivities, predict quality of results, estimate tool runtimes, and in general guide designers in making correct design process decisions. Our paper is organized as follows. Section 2 reviews previous and related works. In Section 3, we describe the schema that is used to maintain historical flow data.

3. FLOW METRICS

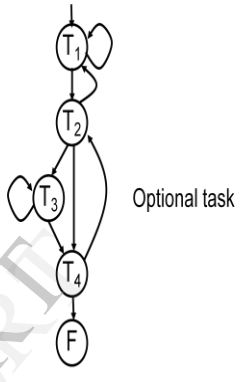
- Tool metrics alone are not enough
  - Design process consists of more than one tool
  - A given tool can be run multiple times
  - Design quality depends on the design flow and methodology (the order of the tools and the iteration within the flow)
- Flow definition
  - Directed graph $G(V,E)$
    - $V \equiv T \cup \{ S, F \}$
    - $T \equiv \{ T_1, T_2, T_3, \ldots, T_n \}$ (a set of tasks)
    - $S \equiv$ starting node, $F \equiv$ ending node
    - $E \equiv \{ E_{s1}, E_{11}, E_{12}, \ldots, E_{xy} \}$ (a set of edges)
  - E_{xy}
    - $x < y \rightarrow$ forward path
    - $x = y \rightarrow$ self-loop

$x > y \rightarrow$ backward path



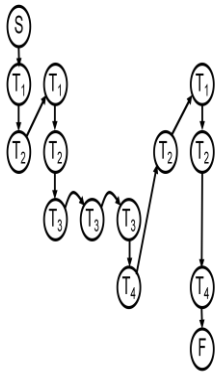
Task sequence: $T_1, T_2, T_1, T_2, T_3, T_3, T_3, T_4, T_2, T_1, T_2, T_4$

Figure 2: Integration of data mining tools within the METRICS architecture



Task sequence: $T_1, T_2, T_1, T_2, T_3, T_3, T_3, T_4, T_2, T_1, T_2, T_4$

3.1 FLOW TRCKING



Run No	Current Task	TASK_NO T1	T2	T3	T4	FLOW_SEQUENCE
1	T1	1	-	-	-	1
2	T2	1	1	-	-	1/1
3	T1	2	-	-	-	2
4	T2	2	1	-	-	2/1
5	T3	2	1	1	-	2/1/1
6	T3	2	1	2	-	2/1/2
7	T3	2	1	3	-	2/1/3
8	T4	2	1	3	1	2/1/3/1
9	T2	2	2	-	-	2/2
10	T1	3	-	-	-	3
11	T2	3	1	-	-	3/1
12	T4	3	1	-	1	3/1/0/1

Optimization of Incremental Multilevel FM Partitioning

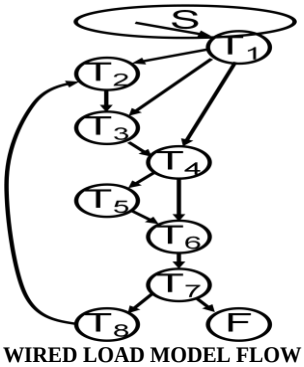
- Motivation: Incremental Net list Partitioning
  - net list ECOs are made; want top-down placement to remain similar to previous result
  - good approach [CaldwellKM00]: "V-cycling" based multilevel Fiduccia-Mattheyses

- What is the best tuning of the approach for a given instance?
  - Break up the ECO perturbation into multiple smaller perturbations?
  - #starts of the partitioned?
  - Within a specified CPU budget?

- Given: initial partitioning solution, CPU budget and instance perturbations (ΔI)
- Find: number of parts of incremental partitioning and number of starts
  - T_i = incremental multilevel FM partitioning
  - Self-loop $\rightarrow$ multistate
  - $n \rightarrow$ number of breakups ($\Delta I = \delta_1 + \delta_2 + \delta_3 + ... + \delta_n$)

Wire Load Model Flow

- WLM flows for finding the appropriate role of WLM
  - T_1 = synthesis & technology mapping
  - T_2 = load wire load model (WLM)
  - T_3 = pre-placement optimization
  - T_4 = placement
  - T_5 = post-placement optimization
  - T_6 = global routing
  - T_7 = final routing
  - T_8 = custom WLM generation



WIRED LOAD MODEL FLOW

Usage of Data mining Tools

Data mining has been used to extract common patterns from large datasets in many domains. These patterns are en used for prediction of future data/results. We now review recent integration of a Commercial data mining tool into the METRICS infrastructure, and some early observations concerning this integration.

3.2 Integration with Data mining Tools

When metrics data is sent through the transmitter, the data is stored in a centralized database. A Java interface built on top of this database is used both for receiving the metrics and for generating reports. In similar fashion, another Java interface is created for the purpose of data mining. This *data mining interface* (DMI) enables communication between data mining tools and the database. It also provides an interface for users to control the data mining process (see Figure 5).

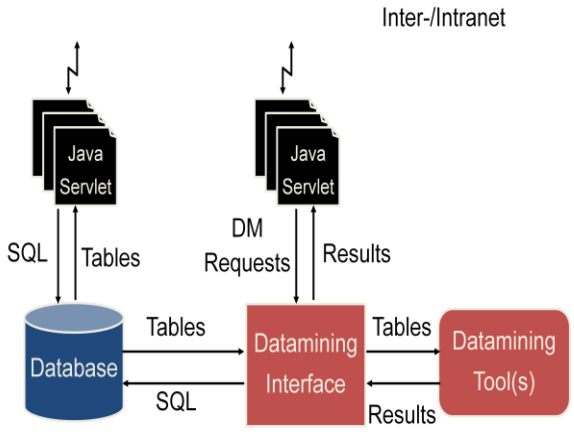


Figure 5: Integration of data mining tools within the METRICS architecture

Our current implementation of METRICS uses *CUBIST* [6] as the data mining tool. This tool is chosen because it produces a single *absolute* number as its result (e.g., number of CPU seconds, or number of passes to reach timing closure), in contrast to tools such as *C5* or *CONTIN* that will only produce categories (without building a model from those categories). CUBIST furthermore returns a set of rules that define a prediction model. To run this tool, we need to provide (i) a list of parameters with the attributes of their values (e.g., continuous numbers, discrete values, etc.), (ii) a dataset for training, and (iii) (for evaluation) a different dataset for testing. The tool uses the training set for the generation of its (piecewise-linear) rule-based predictive model, and it uses the test set to check the accuracy of the model. Our CUBIST integration allows users to select a variable as the prediction target and a set of variables as the inputs for the prediction. The selection can be done over the internet via an HTMLform. Once the users submit their selections, the DMI sends a query to the database, results of which are passed to the data mining tool. At the end, the DMI passes the data mining results to the users through web sites. Several additional tasks can also be assigned to the DMI, e.g., data cleanup, value transformations, variable reductions, etc.

4. Example Applications

The basic usage of the DMI and data mining tool integration is in creation of predictors/estimators from collected data. Other benefits can include:

Parameter sensitivity analysis, i.e., analysis of which input parameters have the most impact on tool results. As data mining tools give insights on how design tools behave when certain changes are made to specific parameters, we are able to use the design tools more effectively (e.g., preventing runtime wastage due to tweaking of knobs that don't matter).

Field of use analysis, i.e., analysis of the (runtime, capacity, quality) limits at which the tool will break. In our interactions within the METRICS community, we have found high interest in analysis of "sweet spots", i.e., the ranges of input attributes for which a given tool will give

best results. To perform sweet spot analysis, we need to run the tools with sufficiently many different input designs; whether these must be real, or can be "mutated" from real or randomly generated, is an open issue.

Process monitoring, i.e., analysis of potential or likely outcomes of the current design process (while the process is still running). Since tool and flow metrics are sent directly to the database in real time, data mining tools can calculate possible outcomes and QOR metrics for the design process online. Given the computed predictions, designers can decide whether they should stop the current process (e.g., given high likelihood of bad results) or let it run to completion. Such process monitoring can potentially shorten the design cycle by reducing time spent on doomed tool runs.

Resource monitoring, i.e., analysis of resource demands for given tasks. We can use data mining tools to identify unsafe resourcing conditions, e.g., if we run design tools on machines with too-small memory or disk. Again, design cycle time can be improved by preventing runs on ill-configured machines. Most of these example applications require tighter integration (additional interfaces and controls) between design tools and data mining tools. Some design checks and resource checks can be integrated with available CAD frameworks, e.g., the "flow manager "could check if the tools will run on the given machine and if the design inputs are in the field of use for the tools. Web-based monitoring can also be implemented to monitor the current process. In the next section, we present sample experimental results from the first type of integration – using data mining tools to generate runtime and QOR predictors – which has been developed for our METRICS architecture.

4.1 Experimental Results

Our METRICS data warehouse has been set up on a server with Oracle8i database, Java servlets, and Apache web server as shown in Figure 1. Integration of the *Cubist* data mining tool [6] has been performed as shown in Figure 5. Two different types of experiments are performed: (i) flow optimization, and (ii) data mining

1.7 Flow Experiments

Our flow experiment simulates the optimization of a design process. Our "process", or "flow", is built around an incremental multilevel Fiduccia-Mattheyses hyper graph partitioned who solves the incremental net list partitioning problem. Given an initial partitioning instance *Iinit* , an initial solution to that instance *Sinit*, a perturbation D*I*, and a CPU budget, we seek to optimize the use of a V-cycling based incremental multilevel FM partitioner.5 In other words, we wish to tune the application of the incremental partitioned so that it returns the best possible solution quality (in terms of minimizing the number of nets cut) within the prescribed CPU budget. With this experiment, we can find the flow tuning that gives the best final solution *Sf inal* for the final instance *I f inal*, which is derived from *Iinit* by applying the perturbation D*I*. The instance is perturbed by changing the weights of various hyper edges (signal nets). The number of nets that are reweighted is the size of perturbation. For purposes of the incremental optimization, the instance perturbation can be broken down into several smaller perturbations, i.e., D*I* =d*I*1+d*I*2+: : :+d*In* (the "breakup"), and various numbers of multistate can be applied to each resulting instance. The best result from the multistate on one instance is used as the starting point for all starts on the next instance. The quality of the result is based on the final instance (*I f inal*). Figure 6 illustrates the flow setup. We run our experiments on 8 standard test cases in the modern
Partitioning literature – the ibm01-06, ibm08 and ibm10 instances of [1]. For each test case, we run 10000 different combinations of D*I*, CPU budget, number of breaks, and number of starts (per break). Once the data are collected, we run the data mining tool to generate rules that give us the optimized flow for a given design with the specified perturbation size and CPU budget, i.e., the data mining tool will predict the number of breaks ("num inc parts") and the number of starts ("num starts"). Table 2 shows the first five out of the 30 rules produced by CUBIST.

foreach testcase
foreach D*I*
foreach *CPUbudget*
foreach breakup (*n* = number of parts)
Icurrent = *Iinitial*
Scurrent = *Sinitial*
for *i* = 1 **to** *n*
Inext = *Icurrent* +d*Ii*
run incremental multilevel FM partitioner
on *Inext* to produce *Snext*
if *CPUcurrent* > *CPUbudget* **then** break
Icurrent = *Inext*
Scurrent = *Snext*
end for
Save number of cuts
end foreach
end foreach
end foreach
end foreach
Figure 6: Flow setup for multilevel FM partitioner.

Flow	Sequence	WLM
F1	T1,T2,T3,T4,T5,T6,T7	Structural
F2	T1,T2,T3,T4,T5,T6,T7	Statistical
F3	T1,T2,T3,T4,T5,T6,T7	Custom*
F4	T1,T4,T5,T6,T7	N/A
F5	T1,T2,T3,T4,T5**,T6,T7	Statistical
F6	T1,T2,T3,T4,T6,T7	Statistical

* Custom WLMs are generated using flow F1 with additional task T8
** Special T5 run without logic restructuring

Design	Flow	CPU	Slack
Design04	F1	210.00	-0.104
	F2	335.77	-0.280
	F3	157.29	-0.158
	F4	649.05	-0.280
	F5	245.85	-0.719
	F6	53.17	-11.599
Design08	F1	930.12	-0.430
	F2	780.77	-0.479
	F3	384.60	-0.396
	F4	1442.58	-0.578
	F5	751.37	-0.441
	F6	287.69	-3.027
Design12	F1	560.91	-0.760
	F2	746.05	-0.230
	F3	409.98	-1.000
	F4	615.70	-0.640
	F5	834.05	-0.740
	F6	189.46	-14.310

5. THE MODEL FOR PRIVACY IMPLICATIONS OF DATA MINING RESULTS

To understand the privacy implications of data mining results, we first need to understand how data mining results can be used (and misused). As described previously, we assume data is Public, *U*nknown, or *S*ensitive. We now discuss additional background leading toward a model for understanding the impact of data mining results on privacy. We assume an adversary with access to *P*ublic data, and Polynomial-time computational power. The adversary may have some additional knowledge, possibly including *U*nknown and *S*ensitive data for some individuals. We want to analyze the effect of giving the adversary access to a classifier *C*; specifically if it will improve the ability of the adversary to accurately deduce *S*ensitive data values for individuals that it doesn't already have such data for.

5.1 Access to Data Mining Models

If the classifier model *C* is completely open (e.g., a decision tree, or weights in a neural network), the model description may reveal sensitive information. This is highly dependent on the model. Instead, we model *C* as a "black box": The adversary can request that an instance be classified, and obtain the class, but can obtain no other information on the classifier. This is a reasonable model: We are providing the adversary with *access* to *C*, not *C* itself. For example, for the proposed CAPPSII airline screening module, making the classifier available would give terrorists information on how to defeat it. However, using cryptographic techniques we can provide privacy for all parties involved: Nothing is revealed But the class of an instance. (The party holding the classifier need not even learn attribute values.) Here, we will only consider the data mining results in the form of classification models. We leave the study of other data mining results as future work.

5.2 Basic Metric for Privacy Loss

While it is nice to show that an adversary gains no privacy violating information, in many cases we will not be able to say this. Privacy is not absolute; most privacy laws provide for cost/benefit tradeoffs when using private information. For example, many privacy laws include provisions for use of private information "in the public interest"[6]. To tradeoff the benefit vs. the cost of privacy loss, we need a metric for privacy loss. One possible way to define such a metric for classifier accuracy is using the Bayesian classification error.

Suppose for data ($x1, x2, \ldots, xn$), We have classification problems in
Which we try to classify xi's into m classes which we labeled as $\{0, 1, \ldots, m-1\}$.
For any classifier C: $xi \_\rightarrow C(xi) \in \{0, 1, \ldots, m-1\}$, $i = 1, 2, \ldots, n$,
We define the classifier accuracy for C as: $m-1$ Xi=0 $Pr\{C(x) \_= i|z = i\}$ $Pr\{z = i\}$.

Does this protect the individual? The problem is that *some* individuals will be classified correctly: If the adversary can predict those individuals with a higher certainty than the accuracy, then the privacy loss for those individuals is worse than expected. Tightening such bounds requires that

5.3 Possible Ways to Compromise Privacy

The most obvious way a classifier can compromise privacy is by taking *P*ublic data and predicting *S*ensitive values. However, there are many other ways a classifier can be misused to violate privacy. We break down the possible forms a classifier that could be (mis)used by the adversary can take.

1. $P \rightarrow S$: Classifier that produces sensitive data given public data. Metric based on accuracy of classification.sup i „$Pr\{C(X) \_= Y | Y = i\} - 1$

2. $PU \rightarrow S$: Classifier taking public and unknown data into sensitive data. Metric same as above.

3. $PS \rightarrow P$: Classifier taking public and sensitive data into public data. Can adversary determine value of?
Sensitive data. (May also involve unknown data, but this is a straightforward extension.)

4. The adversary has access to *S*ensitive data for some individuals. What is the effect on privacy of other?
Individuals of classifiers as follows.

(a) $P \rightarrow S$: Can the adversary do better with such a classifier because of their knowledge, beating the expectations of the metric for 1.
(b) $P \rightarrow U$: Can giving the adversary a predictor for *U*nknown data improve its ability to build a
Classifier for *S*ensitive data.

5.4 Practical Use

For most distributions it is difficult to analytically evaluate the impact of a classifier on creating an inference channel. An alternative heuristic method to test the impact of a classifier is described in Algorithm 1. We now give experiments demonstrating the use, and results, of this approach.

Algorithm 1 Testing a classifier for inference channels
1: Assume that *S* depends on only *P*, *U*, and the adversary has at most *t* data samples of the form (*pi*, *si*).

2: Build a classifier *C*1 on *t* samples (*pi*, *si*).

3: To evaluate the impact of releasing *C*, build a classifier *C*2 on *t* samples (*pi*, *C* (*pi*), *si*).

4: If the accuracy of the classifier *C*2 is significantly higher than *C*1, conclude that revealing *C* creates a inference channel for *S*.

We tested this approach on several of the UCI datasets. We assumed that the class variable of each data set is private, treat one attribute as unknown, and simulate the effect of access to a classifier for the unknown. For each nominal valued attribute of each data set, we ran six experiments. In the first experiment, a classifier was built without using the attribute in question. We then build a classifier with the unknown attribute correctly revealed with probability 0.6, 0.7, 0.8, 0.9, and 1.0. For example, for each instance, if 0.8 is used, the attribute value is kept the same with probability 0.8; otherwise it is randomly assigned to an incorrect value. The other attributes are unchanged. In each experiment, we used C4.5 with default options given in the Weka package [17]. Before running the experiments, we filtered the instances with unknown attributes from the training data set. Ten-fold cross validation was used in reporting each result. Most of the experiments look like the one shown in Figure 1 (the credit-g dataset). Giving an adversary the ability to predict unknown attributes does not significantly alter classification accuracy (at most 2%). In such situations, access to the public data may be enough to build a good classifier for the secret attribute; disclosing the unknown values to the adversary (e.g., by providing a "black box" classifier to predict unknowns) does not really increase the accuracy of the inference channel.

In a few data sets (credit-a, kr-vs-kp, primary-tumor, splice, and vote) the effect of providing a classifier on some

6. CONCLUSIONS

- Extensions to current METRICS system is presented
- Complete prototype of METRICS system is working at UCLA with Oracle8i, Java Servlet and XML (other working prototypes are installed at Intel and Cadence)
- METRICS wrappers for Cadence, Synopsys and UCLA tools and flows
- METRICS system is integrated with Cubist data mining tool and NELSIS flow manager
- A complete METRICS system can be installed on a laptop and configured to work behind firewalls

Increases in the power and ubiquity of computing resources pose a constant threat to individual privacy. Tools from privacy-preserving data mining and secure multi-party computation make it possible to process the data without with disclosure, but do not address the privacy implication of the results. We have defined this problem and explored ways that data mining results can be used to compromise privacy.
We gave definitions to model the effect of the data mining results on privacy, analyzed our definitions for a Mixture of Gaussians for two class problems, and gave a heuristic example that can be applied to more general scenarios. We have looked at other situations, such as a classifier that takes sensitive data as input (can sampling the classifier with known output reveal correct values for input?) and privacy compromise from participating in training data. We are working to formalize analysis processes for these situations.
We plan to test our definitions in many different contexts. Possible plans include a software tool that automatically assesses the privacy threat due to the data mining result based on the related training instances and the private data. We also want to augment existing privacy-preserving algorithms
so that the output of data mining is guaranteed to satisfy the privacy definitions, or the algorithm terminates without generating results. Finally, we want to be able extend the formal analysis to more complex data models using tools from statistical learning theory.

References

[1] C. J. Alpert, "The ISPD98 Circuit Benchmark Suite", *Intl. Symp. on Physical Design*, 1998.
[2] K. O. ten Bosch, P. Bingley, and P. van der Wolf, "Design Flow Management in the NELSIS CAD Framework", *Proc. ACM/IEEE Design Automation Conf.*, 1991, pp. 711-716.
[3] F. Bretschneider, C. Kopf, and H. Lagger, "Knowledge-Based Design Flow Management", *Proc. IEEE Intl. Conf. on Computer-Aided Design*, 1990, pp. 350-353.
[4] A. E. Caldwell, A. B. Kahng and I. L. Markov, "Improved Algorithms for Hypergraph Bipartitioning", *Proc. Asia and South Pacific Design Automation Conf.*, Jan. 2000, pp. 661-666.
[5] A. Casotto and A. Sangiovanni-Vincentelli, "Automated Design Management Using Traces", *IEEE Trans. on Computer-Aided Design of Integrated Circuit and Systems*, vol. 12, August 1993, pp. 1077-1095.
[6] `http://www.rulequest.com/cubist-info.html`
[7] S. Baeder, Notes from DAC'96 Birds of a Feather meeting, *personal communication*, 2000.
[8] A. H. Farrahi, D. J. Hathaway, M. Wang, and M. Sarrafzadeh, "Quality of EDA CAD Tools: Definitions, Metrics and Directions", *Proc. IEEE Intl. Symp. on Quality Electronic Design*, March 2000, pp. 395-405.
[9] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From Data Mining to

Knowledge Discovery: An Overview", *Advances in Knowledge Discovery and*
Data Mining, AAAI Press, 1996, pp. 1-34.
[10] S. Fenstermaker, D. George, A. B. Kahng, S. Mantik and B. Thielges,
"METRICS: A System Architecture for Design Process Optimization", *Proc.*
ACM/IEEE Design Automation Conf., June 2000, pp. 705-710.
[11] E. W. Johnson and J. B. Brockman "Towards a Model for Electronic Design
Process Refinement", *Computers in Industry*, vol(30), 1996, pp. 27-36.
[12] E. W. Johnson, J. B. Brockman, and R. Vigeland, "Sensitivity analysis of iterative
design processes", *Proc. of Intl. Conf. on Computer Aided Design*, San Jose,
1996, pp. 142-145.
[13] E. W. Johnson, *Analysis and Refinement of Iterative Design Processes*, Ph.D.
Thesis, Computer Science and Engineering Dept., Univ. of Notre Dame, 1996.
[14] E. W. Johnson and J. B. Brockman "Measurement and Analysis of Sequential
Design Processes", *ACM Transaction on Design Automation of Electronic Systems*,
Vol. 3(1), January 1998, pp. 1-20.
[15] M. Keating, "Measuring Design Quality by Measuring Design Complexity",
Proc. IEEE Intl. Symp. on Quality Electronic Design, 2000, pp. 103-108.
[16] G. Karypis and V. Kumar, "hMetis: A Hypergraph Partitioning Package Version
1.5", *user manual*, June 23, 1998.
[17] G. Karypis and V. Kumar, "Multilevel *k*-way Hypergraph Partitioning", *Proc.*
ACM/IEEE Design Automation Conf., 1999, pp. 343-348.
[18] `http://www.numetrics.com`
[19] M. Shin and A. L. Goel, "Knowledge Discovery and Validation in Software Metrics
Databases", *Proc. of SPIE Data Mining and Knowledge Discovery: Theory,*
Tools, and Technology, April 1999, pp. 226-233.
[20] N. Whitaker, "Classification of Electronic Design Data", *Comp. Science Dept.*
Tech. Report, University of Manchester, Document No. STEED/T1/02/2, 1998.
[21] G. Ben-Yaacov, L. Bjork, and E. P. Stone, "Advancing Customer-Perceived
Quality in the EDA Industry", *Proc. IEEE Intl. Symp. on Quality Electronic*
Design, March 2000, pp. 411-414.

AUTHORS

First Author –P RAJESH KUMAR received the MCA POST GRADUATE In computer applications in Jawaharlal Nehru Technological University, Hyderabad. He is currently working as Assistant Professor, in Siddharth institute of Engineering & Technology College, putter, Andhra Pradesh, India. His field of interest is data mining and data ware house

Second Author – Reddy sager A.C received the B.Tech. Degree in Computer Science from Jawaharlal Nehru Technological University, Anantapur, and M.Tech. Degree in Computer Science from Jawaharlal Nehru Technological University, Anantapur. He is currently working as Assistant Professor, in Siddhartha institute of Engineering and Technology College, putter, Andhra Pradesh, India, India. His field of interest is data mining, database technologies, information retrieval systems.