Rapid Deployment of Data Mining for Engineering Applications

Nikhil Dakwala, Broadcom Corporation

Abstract. This paper enables rapid deployment of data mining to improve engineering efficiency and productivity. The paper presents the fundamentals of data mining: structured vs. unstructured data, supervised vs. unsupervised learning, cluster analysis, association learning, and decision trees. The paper explains algorithms, data structures, and fuzzy clustering to extract actionable intelligence. The paper provides pseudocode to mine Integrated Circuit (IC) test data and guides readers to practice mining skills on 2012 Medicare payments data online.

1. Introduction

Data Mining (DM) is the process of discovering knowledge hidden in the underlying data. This process is also called knowledge discovery in data (KDD). This hidden knowledge is the Actionable Intelligence, the critical knowledge that will guide the further course of action to quickly attain the desired goals. Data mining is as old as our civilization. Ancient Egyptians were the first known data scientists. They collected enough empirical evidence to link the arrival (clear visibility) of the star Sirius with the flooding of the river Nile. Based on these observations they derived actionable intelligence of when to plant crops and when to stay away from the river. Human history is full of engineers without degrees performing data mining to solve immediate problems at hand. This compelling history can be read online in reference [14].

Data mining was transformed into machine (artificial) intelligence as computers began to proliferate. Within the last decade, an explosion in the computing platforms has created a data deluge in a variety of industries. Most of this data is never analyzed and is simply archived and forgotten. While not all data is worth mining, embedded monitors, on-chip and onboard instruments, network of industrial sensors, etc. provide continuous real-time data. This type of data needs to be mined to improve performance, production costs, runtime reduction, and security for a wide range of end applications. An engineer is very likely to encounter data deluge which can be immediately mined to improve personal productivity and efficiency. An engineer is also unlikely to have easy and immediate access to Relational Database Management systems (RDBM) that offer built-in DM solutions. Being a quintessential knowledge worker, an engineer is likely to be intimately familiar with the underlying data and to possess the necessary technical skills for KDD. This paper shows how to develop an engineering DM solution without requiring expert DM knowledge or commercial DM software. For readers curious about commercial data mining, review the links provided by Microsoft in reference [12] and Oracle in [15].

The author's first introduction to data mining was in the form of machine intelligence while beginning Master's thesis in 1989 to encapsulate a diagnostic expert's knowledge in the form of heuristics that drove diagnostic software engine and on-chip test logic to detect faults in IC chips. Advances in IC fabrication around the year 2000 shrank chip size and simultaneously boosted their performance. Along came an explosion in the variety and quantity of defects. Consequently, when chips failed, they failed in large numbers. Out of necessity, data mining began to be deployed at several stages in the IC chip lifecycle. Reference [1] uses an engineer's knowledge, called pre-filtering, to detect physical defects in electric circuits. Reference [4] presents Bayesian statistics DM, which starts with a set of previously learned probabilities and dynamically updates them with new data. Reference [6] targets outliers to speed up Boolean state justification. The DM to target timing-critical circuits is presented in [2]. Reference [3] uses a support vector machine (SVM) DM utility from MilDe, a publicly available machine learning package. An excellent guide to DM is in reference [7], an open source book from MIT press, "Principles of Data Mining", by Hand, Mannila, and Smyth.

The author leveraged data mining to counter problems faced while testing IC chips before and after fabrication. Every DM term and solution in this paper is presented in context of testing IC chips. The IC test terminology has been greatly simplified, and the DM solutions are generic enough to be applicable to all fields of engineering. It is still important for the reader to thoroughly read section 2, which familiarizes the reader with the IC test knowledge required to understand DM techniques presented in the paper. The remaining sections are organized as follows. Section 3 explains fundamental DM terms, theory, algorithms, and processes. Section 4 presents details about software tools, pseudo-coded data structures, and algorithms for rapid deployment. Readers can experience DM by participating in the online mining of 2012 Medicare payments data as shown in section 5. While an engineer will be intimately familiar with the data he/she is dealing with, not all engineers are programmers. In such cases, hopefully there will be some programmers on the team. Otherwise, an aspiring engineer can obtain the required programming skills by following the methods and references provided in this paper. The software mentioned in this paper is available from the public domain. For the remainder of this paper, DM, DM software, and DM solutions are used interchangeably.

2. IC Testing 101

Figure 1 shows the IC test flow. Test patterns generated to test IC functionality are validated against the software model of the chip running under a software simulator. Failures at this stage could be due to incorrect test patterns or an incorrect software model. Once the test patterns pass in the software simulation, they are executed on the chip, utilizing a hardware chip tester. Failures at this stage can mainly be due to

manufacturing defects, incorrect test patterns, or a discrepancy between the software model and the actual chip.

IC chips undergo rigorous testing on the hardware tester at multiple voltage and temperature values. Depending on fabrication process variations, certain chips can exhibit sensitivity to different voltage and temperature values, i.e., they will fail at a voltage lower than 10% of the normal operating voltage, but otherwise function correctly at all other operating conditions. This is called low-voltage sensitivity. Consider the fact that a 12-inch diameter wafer can hold 100 or 1000 chips, depending on the size of each chip. When these chips fail, the volume of failure and debug data can become overwhelming.. DM is very useful in debugging the data deluge created due to process variation failures.

Another possibility of data deluge occurs when two different blocks in the chip are utilizing two independent clocks, and they each have multiple data registers. Each register can have one or more bits. It is possible that there is an unintentional path starting from a register bit in one clock domain, crossing into another clock domain, and ending at another register. This is called Clock Domain Crossing (CDC). Such clock domain interactions must be detected and handled through special synchronization circuits, or prevented altogether because they can cause data to be lost while travelling across clock domains running at different frequencies.

From a software perspective, a 32-bit register is a single entity. From a chip design perspective, each bit in this register is a unique entity. This allows us to identify and account for each piece of logic-circuit on the chip. This is similar to having a residential community where US mail can be delivered to each family living in each house by identifying them through a unique home address. An example below shows bit 39 of GPIO_1 register. The delimiter "/" represents a logical hierarchy in the chip. Starting from left to right, the logical hierarchy progresses from the input ports of the chip towards the output ports.

soc_top/core_1/i_biu/gpio_1/reg_39_/q
soc_top/core_2/i_biu/gpio_1/reg_39_/q

Thus, the example above shows two separate register bits with the same name, separately residing in core_1 and core_2.

2.1 Determining DM necessity

DM can be deployed for a variety of engineering requirements. Past data can be mined to extract useful features, discard least used features, and develop new products. Data from current processes can be mined to improve efficiency and productivity. The user needs to have some insight into data,-patterns to look for, process sigma variations to detect outliers, and a set of theories to prove or disprove. Contrary to popular definition of Big-Data, DM does not depend on the size of the underlying data. In fact, as will be explained in section 3, a smaller dataset is initially required to develop and test a DM solution. Once validated on small dataset, it is then applied to the larger set for knowledge discovery. Therefore, DM deployment depends on the user's need and aptitude.

3. Fundamentals of Data Mining

Figure 2 shows the DM flow.

First we will briefly describe each step shown in the DM flow above, followed by details in subsequent sections. Step

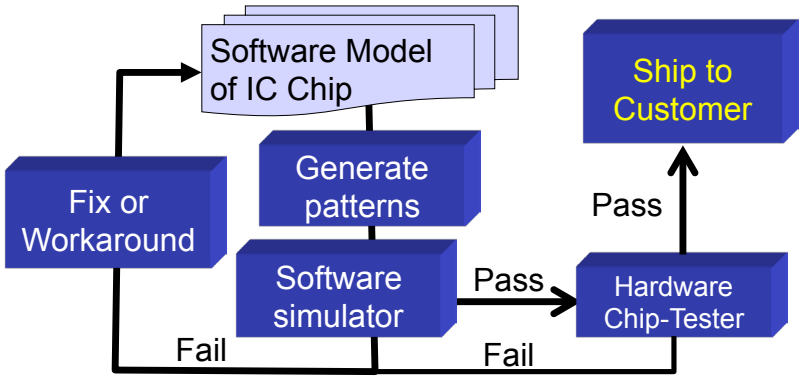

Figure 1. IC Test Flow

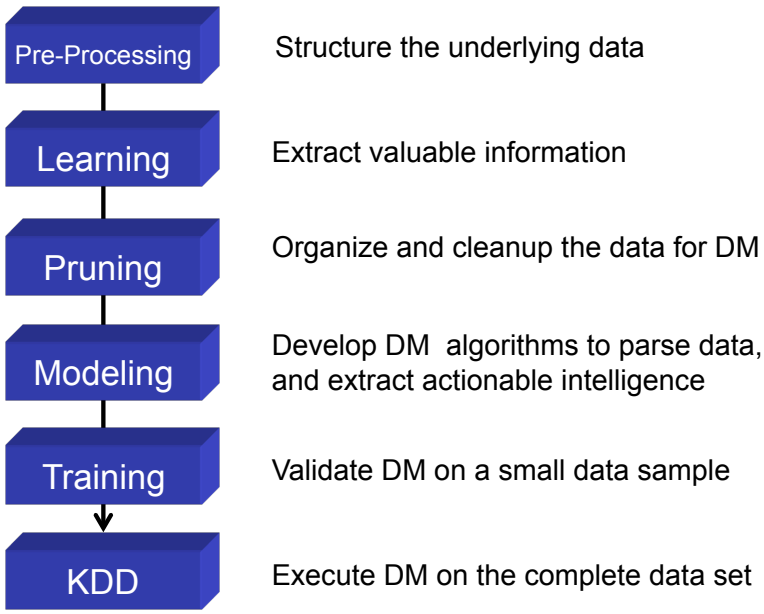

Figure 2. DM Flow

0 is pre-processing which is sometimes needed to properly structure the underlying data. This is explained in section 3.1. Once the data is properly structured, step 1 in developing a DM solution is to learn the underlying data, i.e., to understand the information it represents, its syntax, and semantics. This is explained in section 3.2. Another item that requires attention is data storage and retrieval from the database and hard disks. For engineering applications, we will assume that the data is stored on hard disks and is readily available. Step 2 is to prune the data by parsing/translating it into a format that the DM solution can work with (discarding useless data) and thereby saving compute resources. Pruning relies on the likelihood of relevant data features producing valid and usable results. Pruning is done in the software used to develop DM solution. Step 3 prepares DM models, i.e., algorithms to exploit relevant data features and relationships found through learning and pruning. Modeling is also done in DM software. Both pruning and modeling depend on user's programming skills, aptitude and are not addressed in

this paper. In step 4, the algorithms are proven, or trained, on a small sample size using model(s) that have been just developed. Step 4 is part of due diligence in any engineering process and hence not discussed in this paper. KDD, the final step targets the entire dataset to extract actionable intelligence. KDD is covered in sections 3.3, 3.4 and 3.5.

3.1 Structured vs. Unstructured Data

If your DM can easily extract the information you value from the underlying data, then the data is correctly structured. If, on the other hand, the underlying data is circumspect about the information you value, then it is unstructured data. Let us say that we have a DM to extract the population of each town in Texas by performing a Wikipedia query. Go to www.wikipedia.org and search for Old Dime Box, Texas. You will see the following:

"Old Dime Box is an unincorporated community in Lee County, Texas, United States. According to the Handbook of Texas, the community had an estimated population of 200 in 2000."

A Wikipedia search for another town, Muleshoe, Texas, reveals the following information:

"Muleshoe is a city in Bailey County, Texas, United States. The town of Muleshoe was founded in 1913 when the Pecos and Northern Texas Railway built an 88-mile (142 km) line from Farwell, Texas to Lubbock through northern Bailey County. In 1926, Muleshoe was incorporated. The population was 5,158 at the 2010 census. The county seat of Bailey County, it is home to the National Mule Memorial."

The two examples above show that the population-extraction-DM is dealing with unstructured data. Each town has its population listed with a variety of other details, in no particular order. In computer terms, there is no syntax or semantics that will enable pattern matching. Wikipedia solves this problem by also displaying a table on the far right of the search result screen. This table has the relevant information structured as shown below:

Old Dime Box
Unincorporated community
Location within the state of Texas
Coordinates: 30°22' 38" N 96°51' 47" W
Country United States
State Texas
County Lee
Population (2000)
• Total 200
Time zone Central (CST) (UTC-6)
• Summer (DST) CDT (UTC-5)
GNIS feature ID 1375270

The structured data above can be easily parsed by DM to extract the relevant information.

3.2 Supervised vs. Unsupervised Learning

A typical DM solution employs supervised or unsupervised learning processes to discover detailed information about the underlying data. Supervised learning extracts specific knowledge (patterns, relationships) from specific input data. In other words, an engineer looks at the data and either creates a customized DM or modifies an existing one. Unsupervised learning discovers hidden knowledge from structured and unstructured data. Unsupervised DM is mostly based on advanced neural network and artificial intelligence algorithms. In the author's view, it is impractical for rapid deployment in most engineering applications.

Supervised learning can only work on structured data. It relies on the engineer's intimate knowledge of the underlying problem and data, thereby jump-starting the learning process. By visual inspection of the problem data, an engineer can discern possible cause(s) or failure behavior based on which DM solution can be modeled. This paper only discusses supervised learning. Ambitious readers looking for a challenge can learn about unsupervised learning and a lot more complicated DM in the excellent online book listed in reference [13].

Now, let us travel back in time to Dime Box, Texas and supervise construction of a DM to extract its population from the structured data. Let us assume that we already have a list of all towns in Texas and an automatic method to communicate with the Wikipedia website. We can model our population-extraction-DM as follows:

- First line is the name of the town
- Ignore all lines, unless the line begins with the word "Population", followed by #decimal_year
- Obtain #decimal_population from the next line, after the word: "Total"

If at first you encounter an engineering application producing unstructured data, talk to the owner and reconfigure it to produce structured data. If the owner refuses or otherwise is incapable, you can employ regular expressions, parse out irrelevant information, and impose a structure on the data. Regular expressions will be demonstrated in section 4.

3.3 Knowledge Discovery: Cluster Analysis

Sections 3.3, 3.4, and 3.5 discuss knowledge discovery in data. Theory behind knowledge discovery presented in these sections is just enough to support practical deployment. The reader will find detailed theoretical background in reference [13]. Supervised knowledge extraction depends on the type of knowledge we are seeking and the type of relevant patterns we are able to discern in the data. Knowledge extraction is conducted using cluster analysis, association analysis, and decision trees. In the same order as the three techniques listed, the quality and quantity of KDD obtained, and DM difficulty progressively increases.

Cluster analysis organizes the data in groups or clusters such that each member of the cluster is closer to other members of the same cluster than it is to any other member in any other cluster. This is called supervised K-Nearest Neighbor (K-NN) clustering where users control the number and type of clusters.

For example, we can group Texas towns based on their counties: Lee, Travis, etc. We can also group them based on their population count being greater or smaller than a certain limit. Cluster analysis provides cluster statistics such as X% Texas towns belong to Lee county, Y% belong to Travis county, etc. The statistics can be organized in ascending or descending order for further analysis.

3.4 Knowledge Discovery: Association Analysis

Association analysis derives knowledge by associating data points from similar and different sources. Association requires an engineer to be familiar with the entire system in which the data is generated and consumed. To illustrate association in terms of IC failures, let us say that we have clustered IC chips failing at nominal and ±10% of nominal voltage. We can then associate the failing vectors with software simulation logs to determine the exact functionality being executed at the time of failure. Such associated data can bring out one or more functionalities during which most failures occur.

Going back to Texas, let us say that we perform similar population-extraction-DM on all Texas towns every year, going back 20 years. We can now associate the current year's data with the data from each previous year and learn when the population increase (or decrease) started.

It should be evident by now that supervised DM depends not only on the engineer's knowledge about the data, but also on what kind of information the engineer is looking for. If we are trying to solve IC chip failures, we need to have a theory of possible causes. Then we need to examine the data and detect patterns we can leverage to construct a DM to prove or disprove those theories. Clustering and association show data points that fit with our theories as well as the outliers that disagree with our thinking. The final piece of evidence, the actionable intelligence, comes out through decision trees, which are explained in the following section.

3.5 Knowledge Discovery: Decision Trees

Once the knowledge is clustered and associated, a tree consisting of a series of if-then-else decisions is constructed to draw actionable conclusions, which will prove/disprove the engineer's original theories. This process is called Decision Tree (DT) based DM. The elements of a supervised DT are decisions, conditions, errors, stubs, number of members or size of the dataset, and data attributes. A decision tree is a flowchart of conditions that the data has to satisfy for the engineer to reach a decision. Each condition causes a split in the DT based on it being satisfied or not satisfied. The engineer needs to be cognizant of the types of errors the data can have due to incorrect experiment settings or equipment malfunctions. Stubs are illegal conditions invalidating the complete dataset causing a dead end in the DT. Finally, the data itself has certain attributes, like the test results under high temperature, low voltage, etc.

Quantity and quality of compute power and storage is usually not a concern for engineering rapid deployment. In case resource planning is required, it is possible to estimate the data storage, runtime memory requirements, and the computational complexity. For a given dataset with m members, a attributes, d decisions, c conditions, e errors, and s stubs, the worst-case computation is represented by equation: d.m.a(e+s+2c). The equation to estimate data storage requirements is m(size_of_a). The equation for runtime storage estimates is m(size_of_d). As described in an earlier section, a DM solution can also have pruning, clustering, association, and sorting capabilities, which all affect runtime, speed, and storage. The author does not recommend drawing an actual decision tree because it is a distraction during rapid deployment. A decision table can be constructed in an Excel spreadsheet if needed. It is best to deploy it directly in software using in-built functions like if-then-else, foreach, while loops, regular expressions, and multidimensional arrays.

4. Data Mining Software

The reader can implement DM using any software the reader is familiar with. The author uses PERL software for DM. There are certain applications where the author also uses TCL software. Both PERL and TCL are included in Linux and Unix operating systems. A Windows version is available from www.activestate.com. PERL for Windows is also available at www.strawberryperl.com. DM software needs multidimensional arrays, regular expressions, and subroutines to enable fast pruning, data processing, and DT implementation. Regular expressions are essential for parsing and pruning to impose a structure on the underlying data which will enable knowledge discovery.

4.1 Fuzzy Clustering and Regular Expressions

DM is all about diving into the data and discovering pieces of knowledge. Fuzzy clustering is a contrarian technique which provides an eagle's eye point of view. It enhances cluster analysis by removing uniqifiers to expose global characteristics; e.g., removing register bit indices, circuit flattening uniquifiers, multicore identifiers and hierarchies, etc. to bring out commonality from the sea-of-uniqueness. Fuzzification is done through regular expressions. Most software languages support pattern matching and regular expressions. While the author is most comfortable with PERL, the reader can employ any other software language for regular expressions. For illustration, consider the two registers in Section 2.

```
soc_top/core_1/i_biu/gpio_1/reg_39_/q
soc_top/core_2/i_biu/gpio_1/reg_39_/q
```

Using PERL, regular expression prunes out every piece of information and extracts the register name:

```
/(\/)(\w+)(\/\w+\/q)/;
$register_name = $2;
```

These register names are then counted for their occurrence, stored in an array, sorted, and utilized for further knowledge discovery. The best way to learn PERL is to read the book by PERL's inventor, Larry Wall [16]. To learn TCL, visit the tutorial in [17]. Both these references contain good explanations on regular expressions and pattern matching.

4.2 DM pseudocode for rapid deployment

We will go back to Clock Domain Crossing (CDC) issues highlighted in Section 2 on IC testing.

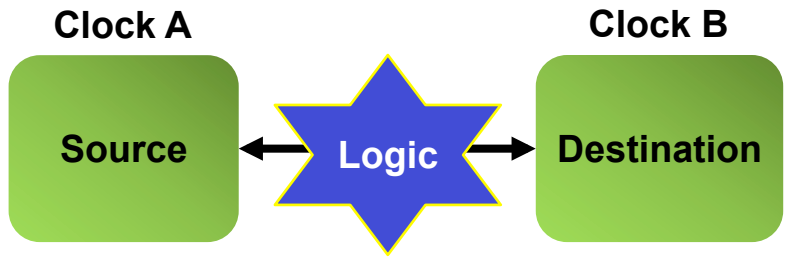

Figure 3. clock domain crossing

Figure 3 shows two clocks, A and B, that can be running at difference frequencies. For illustration purposes, clock A is the source of start points, i.e., individual bits of registers like GPIO, transmit data, receive data, etc. that can go through different logic circuits and end up at individual bits of other registers located in the clock B domain. In reality, clock B can also originate start points that end up in the clock A domain. Circuit tracing applications generate a full report of all such paths in the circuit with the following syntax:

<end_clock> <end_point_name>

<start_clock> <start_point_name>

Figure 4 shows the knowledge we seek.

The goal of mining CDC data is to identify abnormalities where a single start point reaches multiple endpoints across other clock domains. Similarly, identify a single endpoint acting as a terminator for multiple start points. The DM pseudocode is shown below. Note the following:

- It is based on the PERL syntax.
- Data structures are implemented using hash-tables (associative arrays).
- Algorithms are coded into subroutines controlled by variables, which will also control their respective placement in hash tables.
- Extracting relevant start, end–points, and clock names is done through regular expressions.

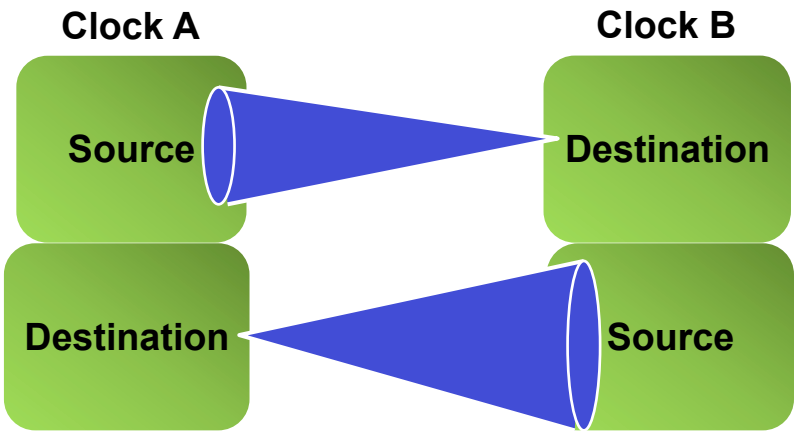

Figure 4. DM Goal: Identify cones

- None of the regular expressions and other software codes for parsing data, sorting hash tables, and compiling statistics are shown because they differ for each application.

```
##################################
#Main Code
#defaults and defines
#parse mandatory, optional arguments, individual files or file list
#open gzip'd file and non-zipped files
#parse data
#extract end-clock and end-point
&process_store_node ($clock, $end_point, "end_point");
#3rd argument is branch/flow control in hash storage

#extract start-clock and start-point
&process_store_node ($clock, $start_point, "start_point");
#
#sort and print the mined jewels
&process_sort_collection("end_point", "full");
&process_sort_collection("start_point", "full");
&process_sort_collection("end_point", "fuz");
&process_sort_collection("start_point", "fuz");
#
# Finish Main Code
##################################
sub process_store_node {
#store node according to flow control, update counts
#each node will have a counter and associated clock variables, #which are not shown here

#Data structure to store full nodes
  $dat_hash{full}{$flow_ctl}{$node};
  $dat_hash{full}{$flow_ctl}{$node}

#fuzzify, store node, update counts
$fuzed_node = &process_fuzzify_name($node);

#Data structure to store fuzzy nodes
  $dat_hash{fuz}{$flow_ctl}{$fuzed_node}
}#sub process_store_node
##################################
sub process_sort_collection {
#counts each hash entry, sort and print in descending order
} #sub process_sort_collection
##################################
sub process_fuzzify_name {
#remove uniquifiers: reg, inst, bit-slices, other unique traits using regular expressions shown earlier
} #sub process_fuzzify_name
##################################
```

Figure 5. shows an example of the actionable intelligence extracted from the CDC DM on a design the author worked on. The worst offenders are shown first followed by others in descending order. The first table lists the registers acting as the start points of paths crossing over to another clock domain. The second table lists registers acting as endpoints (terminations) for paths originating from another clock domain.

#starts	**Register name**
160656	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_rw_cssnoop_vld
42537	vaux_mdio_intf_mdio_slave_pci_addr
18240	vaux_cfg_multi_vf_cfg_vf_cfg_private_vf_flr_in_progress
15840	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_tl_ack

terminations	**Register name**
200836	core_tl_tl_isolate_cfg_tl_demux_cfg_bararr
19899	core_tl_trx_receive_trx_common_interfac_rx_data
18400	core_tl_trx_receive_trx_cmplctl_vf_bdf_table
12842	vaux_cfg_multi_vf_cfg_vf_cfg_dec_vf_cfg_cs_rd_data

Figure 5. Actionable Intelligence

5. Experience DM on 2012 Medicare payments

The Wall Street Journal (WSJ) and several other entities have been mining Medicare payments data made available by the USA Government to identify fraud and misuse. The raw government data can be obtained from www.cms.gov, at the Research & Statistics link. The WSJ has structured 2012 data and put it online. Reference [18] is one of several articles they have published with knowledge discovered from this data. The reader is cautioned not to jump to conclusions based on the billing data. To experience DM algorithms, go to this link: http://projects.wsj.com/medicarebilling/?mod=medicarein#

The data is grouped based on following clusters:

- Last name / Company name
- Specialty / Facility type
- City
- Location

Each of the above has been further clusterized in to several smaller clusters. For example, Specialty/Type has clusters like Psychiatry, Ambulatory services etc. Leave 1st three columns (name/type/city) empty and select "Foreign Country" as location and press the Search button. You will discover foreign individuals collecting from American Taxpayers. Several individual and company names will have a "+" sign to their left, which will expand when clicked, and will show further breakdown of charges.

To experience association analysis, leave name and city columns empty, select "slide preparation" facility type and "all" locations. Click on "state/country" columns to sort based on the State name. Now you have associated the most expensive providers of slide preparation services with the States in which they're located. A visual count will show 48% (12/25) of these providers reside in California.

Decision trees are constructed in software to sort through the clusters and their associations. Limited DT based DM can be performed by copying the "payment" column in to an Excel spreadsheet, and by identifying the average, median and outlier payments. Knowledge extracted through DT is presented in reference [18]

6. Conclusion

Author's primary objective in writing this paper was to enable DM deployment in different engineering applications. The success and speed of such deployment will depend on each application and the user's skills. . The DM pseudocode presented in section 4.2 is a good template to construct reusable DM software. For DM novices, start by learning Perl [16]. Focus on regular expressions, associative arrays and sorting. Author recommends starting with supervised data mining. Cluster analysis is the easiest form of data mining. Engineers with good software skills and basic DM knowledge can deep dive in associative analysis and decision trees. The reader will learn much more DM by practice rather than reading. Readers are encouraged to contact the author with questions, for more information and explanations as needed. Good Luck.

ABOUT THE AUTHOR

Nikhil Dakwala I firmly believe it is possible to improve personal efficiency and productivity through data mining. As mentioned in the paper, my first introduction to data mining was in the form of machine intelligence while working on my Master's thesis at SUNY Buffalo. After obtaining my MS EE in 1991, I joined Motorola's microcontroller division where the goal was to manually detect faults and achieve the highest quality at the lowest cost. Without any automated EDA solution, this experience resembled trench warfare, but it was also a data mining paradise. Out of necessity, I began using cluster fault analysis to detect the maximum number of faults for the maximum coverage increase in the least amount of time.

Since then, I have worked at various companies such as IBM, ARM, and startups. I have been a consultant and currently I am employed at Broadcom. When faced with roadblocks, I resort to data mining to gain deep insight into the data and chart the best course forward. I have conducted research on topographical analysis of silicon failures based on chaos theory. I have presented at IEEE ITC, and STC. Feel free to contact me with questions and DM ideas.

E-mail: ndakwala@broadcom.com

REFERENCES

1. Y. Hirano, et al, "Scrubber induced substrate cracks found by data mining", IEEE ISSM, 2005, pp. 257-259
2. J. Chen, et al, "Mining AC delay measurements for understanding speed-limiting paths", ITC 2010, p 18.3
3. S. Wang, et al, "Machine learning based volume diagnosis", DATE, 2009, pp. 902-905.
4. Daasch et al, "Die level adaptive test: real time test reordering and elimination", ITC 2011, p15.1
5. Caruana and Mizil, "Empirical comparison of supervised learning algorithms", ICML 2006.
6. W. Wu and M. Hsiao, "SAT-based state justification with adaptive mining of invariants", ITC 2008, p7.2
7. "Principles of Data Mining", by David Hand, Heikki Mannila and Padhraic Smyth, MIT Press open source
8. K. Baker and J. Beers, "Shmoo plotting: The black art of IC testing", ITC 1998, L2.3
9. P. Patten, "Divide and conquer based fast shmoo algorithms", ITC 2004, P8.3
10. L. Huisman et al, "Data mining IC fails with fail commonalities", ITC 2004, P232
11. N. Dakwala, "Data Mining Fail Data Through Cluster Analysis and Association Learning", ITC DATA 2011
12. Microsoft SQL Server: "Basic Data Mining Tutorial", http://technet.microsoft.com/en-us/library/ms167167.aspx
13. Pang, Steinback and Kumar: "Introduction to Data Mining", http://www-users.cs.umn.edu/~kumar/dmbook/index.php
14. Stephen Wolfram: "Advance of the Data Civilization", http://blog.wolframalpha.com/2011/08/16/advance-of-the-data-civilization-a-timeline/
15. Oracle "Data Mining Document Library", http://www.oracle.com/pls/db102/portal.portal_db?selected=6
16. Larry Wall: "Programming Perl"
17. TCL Tutorial: http://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html
18. The Wall Street Journal, "Taxpayers Face Big Medicare Tab for Unusual Doctor Billings," June 9 2014, http://online.wsj.com/articles/taxpayers-face-big-medicare-tab-for-unusual-doctor-billings-1402364264