Extraction of Frequent Patterns from Web Logs using Web Log Mining Techniques

Rakesh Kumar
Research Scholar, Dept. of Computer Science & Applications, Kurukshetra University, Kurukshetra, India

Kanwal Garg, PhD.
Assistant Professor, Dept. Of Computer Science Applications, Kurukshetra University, Kurukshetra, India

Vinod Kumar
Research Scholar, Dept. Of Computer Science Applications, Kurukshetra University, Kurukshetra, India

ABSTRACT

World Wide Web is a huge repository of web pages and links. It provides profusion of information for the Internet users. The growth of web is tremendous as approximately one million pages are added daily. User's accesses are recorded in web logs. Because of the incredible usage of web, the web log files are growing at a faster rate and the size is becoming huge. Web data mining is the application of data mining techniques in web data. Web log Mining applies mining techniques in log data to extract the behaviour of users. Web log mining consists of three phases pre-processing, pattern discovery and pattern analysis. Web log data is usually noisy and ambiguous and pre-processing is an important process before web log mining. For discovering patterns sessions are to be constructed efficiently. This paper presents the existing work done to extracting patterns by using decision tree methodology in the technique of web log mining.

Keywords

Web Log mining, Web Log Files, World Wide Web (WWW), HTTP (Hyper Text Transfer Protocol) and CHAID (Chi-Squared Automatic Interaction Detection)

1. Introduction to Web Log Files

The huge advancements in World-Wide Web (WWW) technology and the ever growing popularity of the WWW, a huge number of Web access log records are being collected in the form of web log files (Osmar R. Zaiane, Man Xi and Jiawei Han, 2001). In Web log file there are three kinds of files recording the client visiting behaviors: Access Log, Refer Log, Agent Log, and for some systems Cookie Log is also recorded. Access Log, recording detailed visiting behavior of every client, is the main data resource for Web log mining. Refer Log keeps record of the information about page layout requested by client, such as client visiting date and period, visiting path pattern and etc for client recognition and path supplement. Cookie Log can have label number held by client for recognizing client and its conversation and etc. (GAO, 2010)

When any user agent (e.g., IE, Mozilla, Netscape, etc) hits an URL in a domain, the information related to that operation is recorded in an access log file. In the data processing task, the web log data can be preprocessed in order to obtain session information for all users. Access log file on the server side contains log information of user that opened a session. These records have seven common fields, which are: 1. User's IP address, 2. Access date and time, 3. Request method (GET or POST), 4. URL of the page accessed, 5.Transfer protocol (HTTP 1.0, HTTP 1.1), 6.Success of return code, 7.Number of bytes transmitted. (Veeramalai et.al. 2010)

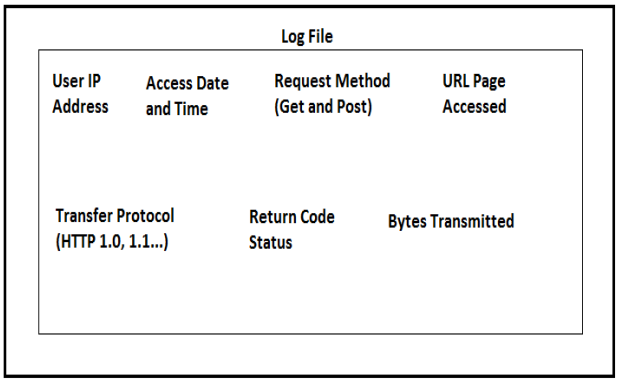

Fig 1: Log File Fields

This paper has been divided into five sections. Section 1 explores the web log files. Section 2 discusses about frequent patterns. Section 3 highlights the proposed strategy. Section 4 focuses on the analysis of frequent patterns. Section 5 finally concludes by discussing the outcome of study.

2. ABOUT Frequent Patterns

Frequent patterns are values or events type combinations that often occur together in the data. They provide information, which can be used to find rules or patterns of correlated or otherwise searched value combinations. A pattern is called frequent if the number of its occurrences in the data is larger than a given threshold.

There are two kinds of frequent patterns: frequent sets and frequent episodes. Frequent sets consist of value combinations that occur inside data records like log entries. Frequent episodes, on the other hand, describe common value sequences like log message types that occur in the network. (Kimmo Hatonen, 2009).

```
777;11May2000; 0:00:23;a_daemon;B1;12.12.123.12;tcp;;
778;11May2000; 0:00:31;a_daemon;B1;12.12.123.12;tcp;;
779;11May2000; 0:00:32;1234;B1;255.255.255.255;udp;;
780;11May2000; 0:00:38;1234;B2;255.255.255.255;udp;;
781;11May2000; 0:00:43;a_daemon;B1;12.12.123.12;tcp;;
782;11May2000; 0:00:51;a_daemon;B1;12.12.123.12;tcp;;
```

Fig 2: An example firewall log fragment (Kimmo Hatonen, 2009)

```
11234 NE321 8.7.1997 020936 1234 2 Link_failure
11234 NE321/TRX1 8.7.1997 020936 5432 1 Call_channel_missing
11234 NE321/TRX3 8.7.1997 020937 5432 1 Call_channel_missing
11234 NE321/TRX1 8.7.1997 020937 6543 3 Link_access_failure
11234 NE321/TRX3 8.7.1997 020939 6543 3 Link_access_failure
11234 NE321/TRX2 8.7.1997 020940 6543 3 Link_access_failure
12345 NE123 8.7.1997 020942 8888 2 XXX/YYY:_alarm_indication_signal_received
12345 NE123 8.7.1997 020942 8888 2 XXX/YYY:_alarm_indication_signal_received
```

Fig 3: An example alarm log fragment (Kimmo Hatonen, 2009)

2.1 Frequent sets

This section gives definitions for frequent sets in the context of the telecommunications network event logs.

Definition 3.1 (Items) Items is a finite set of items that are field: value pairs, i.e., Items= {A : ai, B : bj , C : ck, . . .}, where field is one of the fields in log entries and value attached to each field belongs to the domain of possible values of the field.Definition 3.2 (log entry) A log entry e is a subset of Items such that∀ F : u,G : v ∈ e : F _= G.

Definition 3.3 (log) A log r is a finite and non-empty multiset r = {e1, e2, . . . , en} of log entries (Kimmo Hatonen et. al. , 2003).

Definition 3.4 (itemset) An itemset S is a subset of Items. The main properties that an itemset has entries.

Definition 3.5 (itemset support) A log entry e supports an itemset S if every item in S is included in e, i.e., with respect to a given log are a set of entries in which it occurs, i.e., of which it is subset, and the number of those S ⊆ e. The support (denoted supp(S, r)) of an itemset S is the multiset of all log entries of r that support S. Note that supp(∅ , r) = r.

Definition 3.6 (itemset frequency) The absolute frequency of an itemset S in a log r is defined by freq(S, r) = |supp(S, r)| where |.| denotes the cardinality of the multiset (Kimmo Hatonen et. al., 2003).

Closed sets

A telecommunications network log can be seen as a sparse transactional database. For example, in firewall logs fields potentially have a very large set of possible values, e.g., the value of the Destination field that contains the requested destination address, can be any IP address in the Internet. However, probably in most of the entries, the field contains addresses of those servers in an intranet, which are hosting services like web and mail that are open to the Internet. (Jean-Fran et. al., 2001).

The Apriori algorithm works fine when the number of candidates is not too large. In a sparse database, the number of candidates usually starts to decrease after the frequent sets of size two or three have been computed. With data like firewall logs, which are dense, this does not happen. On the contrary, when there are many local correlations between field values, the number of candidates and frequent sets starts to expand quickly. This problem of a large number of closely related frequent sets can be solved with so-called closed sets, which can be used as a condensed representation of a set of frequent sets (Nicolas Pasquier et. al., 1999).

2.2 Frequent episodes

The notion of association rules was postulate for sequences by defining episode rules. Episode rule A ⇒ B describes association “if A occurs in a sequence also B occurs in the sequence”. The confidence of the episode rule gives a probability P(”B occurs in a sequence” | ”A occurs in the sequence”) The probability can be computed from data by computing frequent episodes, which reveal items occurring close to each other in a sequence and correspond to frequent sets (Kimmo Hatonen, 2009).

The sequences in the log domain consist of log event types — for example, alarm numbers or message texts — which are ordered by their recording or creation times. The patterns, the so-called episodes, are ordered or unordered sequences of entry types of log entries occurring within a time window of specified width. (Kimmo Hatonen, 2009).

3. Proposed Strategy

With respect to secondary literature review, this includes sources of information not available in the public but someone has to get in touch with companies or universities in order to get them. I got the Web log file from the Technical Support of Kurukshetra University’s Institute of Engineering Collage.

Then, I needed first of all to convert them from the log’s file format to a format that data mining software would understand.

The second step is to build a excel file by using MS. Access through which the data mining tool easily loaded and then making decision tree for the extraction of patterns.

The final form of research that, I was conducting interviews with members’ of the Technical Support Group and then discuss with my dissertation guide, this was essential in order for me to clarify some aspects of the web log file contents; an example of this web log file is given below:

2012-03-18 00:29:07 W3SVC49 PLESK-WEB14 208.91.198.202 GET /tap_pb.html - 80 - 180.76.5.51 HTTP/1.1 Mozilla/5.0+(compatible;+Baiduspider/2.0;++http://www.baidu.com/search/spider.html) - - uietkuk.org 200 0 0 8848 224 2168

2012-03-18 00:41:12 W3SVC49 PLESK-WEB14 208.91.198.202 GET /robots.txt - 80 - 77.75.77.17 HTTP/1.1 SeznamBot/3.0+(+http://fulltext.sblog.cz/) - - www.uietkuk.org 404 0 2 5394 197 2043

3.1 Algorithms for Creating Decision Trees

3.1.1 The CHAID Algorithm

The CHAID algorithm was devised by J.A.Hartigan in 1975 (Berry and Linoff, 1997). It tries to ’evaluate’ the quality of the tree so that no pruning will be needed. There is another difference concerning the way the splitting is performed; first of all, classes that correspond to the same value of the target field are grouped together. An example of that is that BMW, Mercedes and Audi would be grouped together as car brands whereas Boeing and Airbus would be grouped together as airline brands. This happens, though, if the proportion of classes it leads to have no significant difference- this is determined by the use of the X2 test. The mathematical formula for this test is given below: (the Ei represent the expected frequencies of classes and the Oi the ones that actually occurred) (Damianou, 1998).

$$X^2 = \sum_{i=1}^{n} \frac{(E_i - O_i)^2}{E_i^2}$$

Eq. 1: Equation X2 formula (Lekeas, 2000)

3.1.2 The CART Algorithm

The CART algorithm was devised in 1984 by L.Briemen and associates (Berry and Linoff, 1997). The product is a binary tree; this means that there are only two nodes for each split.
This is done with the aid of a training set – that is a set for which the target attribute's value is known. The measure we use for that split is diversity; despite the fact that many ways for calculating diversity exists there is a common interpretation for their values. In the case that its value is low, this means that most records fall under one class whereas if the value is high that means that all classes are represented. Aim is at each step to reduce the amount of diversity by the biggest possible amount. Try to do the split using each different field and then pick up the one that reduces diversity the most. Once the first split is done, there are two nodes as its product; each node is now considered to be the root of the tree and the search for the field that gives the best split is performed. This is how the tree grows until no field can be found that would reduce diversity significantly.
In order to first identify those branches that give the least additional predictive power; for that to happen, need to introduce another measure the adjusted error rate.
This is equal to AE(T)= E(T) +α * leaf_count(T) (Berry and Linoff, 1997)
In order to find the first subtree, to evaluate the adjusted error rate's values as α increase; when it becomes less or equal to the respective value for the whole tree the first candidate subtree is found and so on for other subtrees.

3.1.3 The QUEST Algorithm

QUEST was introduced by (Wei-Yin Loh and Yu-Shan Shih ,1997) and is an acronym for "Quick, Unbiased, Efficient, Statistical Tree." To become quick and unbiased, this algorithm selects an input variable to split on before searching for a split, thereby eliminating the time required for searching on most inputs and eliminating the bias towards nominal inputs inherent when relying on candidate splitting rules to select the input variable.
A simplified version of the QUEST input selection is as follows. For an interval input, perform a J-way ANOVA, where J is the number of target values. For a nominal input with M values, compute a chi-square statistic from the J by M contingency table. Select the input with the smallest Bonferroni adjusted p-value. If a nominal input is selected, it is transformed to an interval one. A linear discriminant analysis is then performed on the single selected input, and one of the discriminant boundary points becomes the split point.
Splits on linear combinations of inputs are also possible. QUEST searches for binary splits on nominal or interval inputs for a nominal target. Cases whose values are missing an input are excluded in calculations with that input. Surrogate rules assign such cases to branches. Recursive partitioning creates a large tree that is retrospectively pruned using cross validation.

3.2 Mathematical formulation for Decision Tree

Given training vectors $x_i \in R^n$, i=1,..., l and a label vector $y \in R^l$, a decision tree recursively partitions the space such that the samples with the same labels are grouped together. [2]

Let the data at node m be represented by Q. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets

$$Q_{left}(\theta) = (x, y)|x_j <= t_m$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta)$$

The impurity at m is computed using an impurity function $H()$, the choice of which depends on the task being solved (classification or regression)

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$

Select the parameters that minimises the impurity

$$\theta^* = argmin_\theta G(Q, \theta)$$

Recurse for subsets $Q_{left}(\theta^*)$ and $Q_{right}(\theta^*)$ until the maximum allowable depth is reached, $N_m < min\_samples$ or $N_m = 1$.

4. Interpretation of Patterns

4.1 Creating the Model for Decision Tree

The Decision Tree Procedure offers several different methods for creating tree models.

4.1.1 CHAID Tree Model

To run a Decision Tree analysis, from the menus choose:

Menu Tab
Analyze
Classify
Tree...

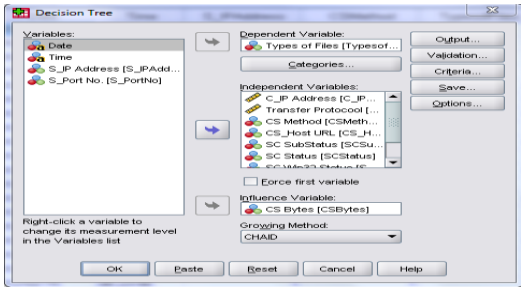

Fig 4: Decision Tree dialog box

- Select Types of Files as the dependent variable.
- Select all the remaining variables as independent variables.
- Select CS Bytes as influence variable.

At this point, you could run the procedure and produce a basic tree model, but we're going to select some additional output and make a few minor adjustments to the criteria used to generate the model.

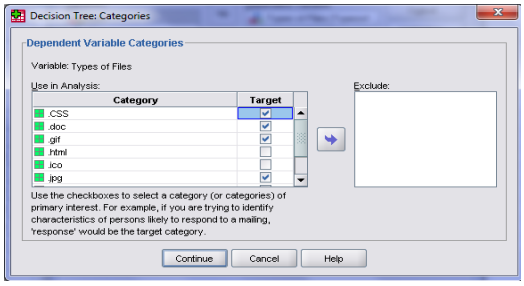

Fig 5: Selecting Target Categories

- Click the Categories button right below the selected dependent variable. This opens the Categories dialog box, where you can specify the dependent variable target categories of interest.
- Select (check) the Target check box for the .CSS, .doc, .gif, .jpg, .pdf and .png categories.
- Click Continue.

4.1.1.1 Specifying Tree Growing Criteria

- To keep the tree fairly simple, so limit the tree growth by raising the minimum number of cases for parent and child nodes.
- In the main Decision Tree dialog box, click Criteria.
- In the Minimum Number of Cases group, type 100 for Parent Node and 50 for Child Node.
- Click Continue.

4.1.1.2 Selecting Additional Output

- In the main Decision Tree dialog box, click Output. This opens a tabbed dialog box, where you can select various types of additional output.
- On the Tree tab, select (check) Tree in table format.
- Then click the Plots tab.

4.1.1.3 Saving Predicted Values

- You can save variables that contain information about model predictions.
- In the main Decision Tree dialog box, click Save.
- Select (check) Terminal node number, Predicted value, and Predicted probabilities.
- Click Continue.
- In the main Decision Tree dialog box, click OK to run the procedure.

4.1.2 Evaluating the Model

- This model results include:
- Tables that provide information about the model.
- Tree diagram.
- Charts that provide an indication of model performance.
- Risk Estimate and Classification.

Model Summary Table

TABLE 1. Model Summary

Specifications	Growing Method	CHAID
	Dependent Variable	Types of Files
	Independent Variables	C_IP Address, Transfer Protocool, CS Method, CS_Host URL, SC SubStatus, SC Status, SC Win32 Status, Time Taken, SC Bytes
	Validation	None
	Maximum Tree Depth	3
	Minimum Cases in Parent Node	100
	Minimum Cases in Child Node	50
Results	Independent Variables Included	SC Bytes, Time Taken, C_IP Address
	Number of Nodes	7
	Number of Terminal Nodes	4
	Depth	2

The model summary table provides some very broad information about the specifications used to build the model and the resulting model. Nine independent variables were specified, but only three were included in the final model. The variables for transfer protocols, cs method, cs_hosturl, sc substatus, sc status and sc win32status did not make a significant contribution to the model, so they were automatically dropped from the final model.

I. Tree Diagram

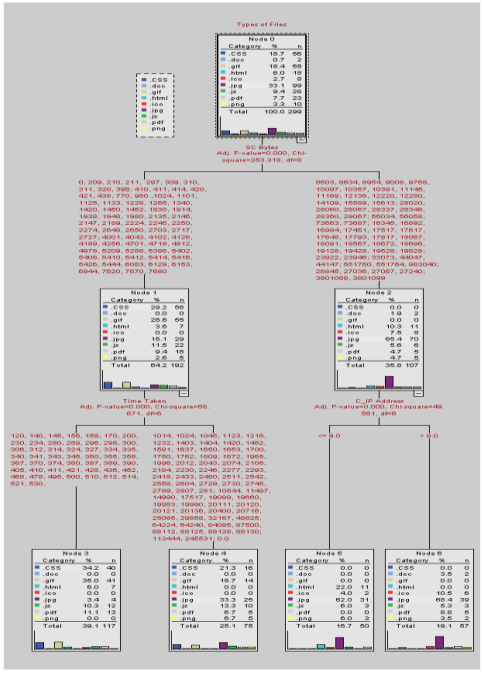

Fig 6: Tree Model

The tree diagram is a graphic representation of the tree model. This tree diagram shows that:

Using the CHAID method, SC Bytes is the best predictor of Types of Files.
For the node1 and node2, the next best predictors are Time taken and C_IP Address respectively.
The Tree Editor can hide and show selected branches, change colours and fonts, and select subsets of cases based on selected nodes.

II. Tree Table

TABLE 2. Gain Summary for Nodes

Node	N	Percent	Profit	ROI
3	117	39.1%	4.556	227.8%
4	75	25.1%	3.160	158.0%
5	50	16.7%	2.240	112.0%
6	57	19.1%	1.912	95.6%

Growing Method: CHAID
Dependent Variable: Types of Files

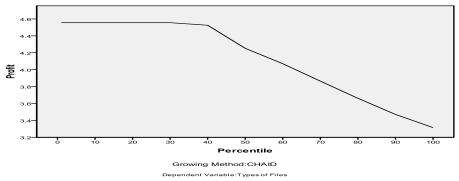

Fig 7: Profit Graph

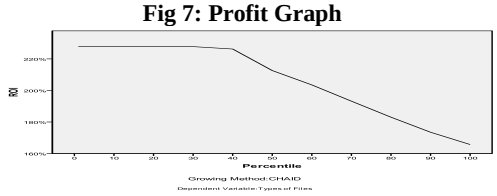

Fig 8: ROI Graph

Target Category: .CSS

TABLE 3. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
3	117	39.1%	40	71.4%	34.2%	182.5%
4	75	25.1%	16	28.6%	21.3%	113.9%
6	57	19.1%	0	.0%	.0%	.0%
5	50	16.7%	0	.0%	.0%	.0%

Growing Method: CHAID
Dependent Variable: Types of Files

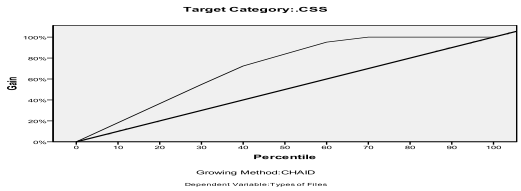

Fig 9: Gain Graph - .CSS

Target Category: .doc

TABLE 4. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
6	57	19.1%	2	100.0%	3.5%	524.6%
3	117	39.1%	0	.0%	.0%	.0%
4	75	25.1%	0	.0%	.0%	.0%
5	50	16.7%	0	.0%	.0%	.0%

Growing Method: CHAID
Dependent Variable: Types of Files

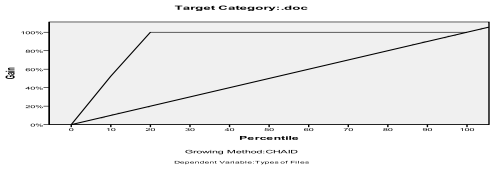

Fig 10: Gain Graph - .doc

Target Category: .gif

TABLE 5. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
3	117	39.1%	41	74.5%	35.0%	190.5%
4	75	25.1%	14	25.5%	18.7%	101.5%
6	57	19.1%	0	.0%	.0%	.0%
5	50	16.7%	0	.0%	.0%	.0%

Growing Method: CHAID
Dependent Variable: Types of Files

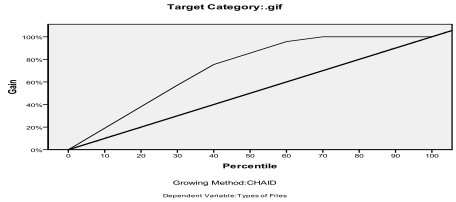

Fig 11: Gain Graph - .gif

Target Category: .jpg

TABLE 6. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
6	57	19.1%	39	39.4%	68.4%	206.6%
5	50	16.7%	31	31.3%	62.0%	187.3%
4	75	25.1%	25	25.3%	33.3%	100.7%
3	117	39.1%	4	4.0%	3.4%	10.3%

Growing Method: CHAID
Dependent Variable: Types of Files

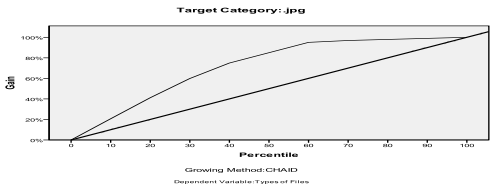

Fig 12: Gain Graph - .jpg

Target Category: .pdf

TABLE 7. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
3	117	39.1%	13	56.5%	11.1%	144.4%
6	57	19.1%	5	21.7%	8.8%	114.0%
4	75	25.1%	5	21.7%	6.7%	86.7%
5	50	16.7%	0	.0%	.0%	.0%

Growing Method: CHAID
Dependent Variable: Types of Files

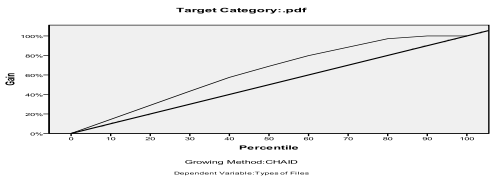

Fig 13: Gain Graph - .pdf

Target Category: .png

TABLE 8. Gains for Nodes

Node	Node		Gain		Response	Index
	N	Percent	N	Percent		
4	75	25.1%	5	50.0%	6.7%	199.3%
5	50	16.7%	3	30.0%	6.0%	179.4%
6	57	19.1%	2	20.0%	3.5%	104.9%
3	117	39.1%	0	.0%	.0%	.0%

Growing Method: CHAID
Dependent Variable: Types of Files

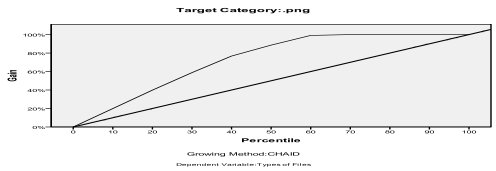

Fig 14: Gain Graph - .png

The gains for nodes table provide a summary of information about the terminal nodes in the model.

- Only the terminal nodes—nodes at which the tree stops growing—are listed in this table.
- Since gain values provide information about target categories, this table is available only if specified one or more target categories.
- Node N is the number of cases in each terminal node, and Node Percent is the percentage of the total number of cases in each node.
- Gain N is the number of cases in each terminal node in the target category, and Gain Percent is the percentage of cases in the target category with respect to the overall number of cases in the target category.
- For categorical dependent variables, Response is the percentage of cases in the node in the specified target category.
- For categorical dependent variables, Index is the ratio of the response percentage for the target category compared to the response percentage for the entire sample.

III. Gains Chart

Gains chart indicates that the model is a fairly good one. Cumulative gains charts always start at 0% and end at 100% as you go from one end to the other. For a good model, the gains chart will rise steeply toward 100% and then level off. A model that provides no information will follow the diagonal reference line.

IV. Risk Estimate and Classification

Table 9. Risk Estimate and Classification

Risk

Estimate	Std. Error
.548	

Growing Method: CHAID
Dependent Variable: Types of File

Classification

Observed	Predicted									
	.CSS	.doc	.gif	.html	.ico	.jpg	.js	.pdf	.png	Percent Correct
.CSS	40	0	0	0	0	16	0	0	0	71.4%
.doc	0	0	0	0	0	2	0	0	0	.0%
.gif	41	0	0	0	0	14	0	0	0	.0%
.html	7	0	0	0	0	11	0	0	0	.0%
.ico	0	0	0	0	0	8	0	0	0	.0%
.jpg	4	0	0	0	0	95	0	0	0	96.0%
.js	12	0	0	0	0	16	0	0	0	.0%
.pdf	13	0	0	0	0	10	0	0	0	.0%
.png	0	0	0	0	0	10	0	0	0	.0%
Overall Percentage	39.1%	.0%	.0%	.0%	.0%	60.9%	.0%	.0%	.0%	45.2%

Growing Method: CHAID
Dependent Variable: Types of Files

The risk and classification tables provide a quick evaluation of how well the model works.The risk estimate of 0.548 indicates that the category predicted by the model (Types of File) is wrong for 54.8% of the cases. So the "risk" of misclassifying Clients is approximately 29%. The results in the classification table are consistent with the risk estimate. The table shows that the model classifies approximately 45.2% of the types of Files correctly.

5. Conclusion

Pattern discovery is an important subfield of web mining that attempts to discover interesting (or high-quality) patterns from web log files. There are several efficient techniques to discover such patterns with respect to different interestingness measures. Merely discovering the patterns efficiently is rarely the ultimate goal, but the patterns are discovered for some purpose. One important use of patterns is to summarize data, since the pattern collections together with the quality values of the patterns can be considered as summaries of the data.

6. REFERENCES

[1] Berry Michael J.A. and Linoff Gordon, "Data Mining Techniques: For Marketing, Sales and Customer Support, John Wiley & Sons Ltd.", ISBN: 0-471-17980-9, (1997).

[2] Decision Tree http://scikit-learn.sourceforge.net/stable/modules/tree.html

[3] Damianou Charalambos," Statistics, Symmetria Publications", (1998).

[4] Gao, "Research On Client Behavior Pattern Recognition System Based On Web Log Mining"," Proceedings of the Ninth International Conference on Machine Learning and Cybernetics, Qingdao, 11-14 July 2010", 978-1-4244-6527-9/10, (2010).

[5] Jean-Fran et. al.,"Mining free-sets under constraints", In Michel E. Adiba, Christine Collet, and Bipin C. Desai, editors, "Proceedings of International Database Engineering & Applications Symposium (IDEAS'01)", pages 322 – 329, Grenoble, France, July (2001).

[6] Kimmo Hatonen, "Data mining for telecommunications network log analysis", (2009).

[7] Kimmo Hatonen et. al.,"Comprehensive log compression with frequent patterns, "Proceedings of Data Warehousing and Knowledge Discovery, 5th International Conference", (DaWaK 2003), volume 2737 of LNCS, pages 360 – 370, Prague, Czech Republic, September (2003).

[8] Lekeas, "Data mining the web: the case of City University's Log Files", (2000).

[9] Loh, W. and Shih, Y. (1997), "Split Selection Methods for Classification Trees," Statistica Sinica, 7, 815-840. Introduces the QUEST algorithm. Refer to http://www.stat.wisc.edu/~loh.

[10] Nicolas Pasquier et. al.,"Closed set based discovery of small covers for association rules", "In Christine Collet, editor, Proceedings of BDA'99", pages 361 – 381, Bordeaux, France, October (1999).

[11] Osmar R. Zaiane et. al. "Discovering Web Access Patterns and Trends by Applying OLAP and Data Mining Technology on Web Logs", (2001).

[12] Veeramalai et.al. "Efficient Web Log Mining Using Enhanced Apriori Algorithm with Hash Tree and Fuzzy", "International journal of computer science & information technology (IJCSIT) Vol.2, No.4, August 2010", 10.5121/ijcsit.2010.2406, (2010).