Classifying Sybil in MSNs using C4.5

Anand Chinchore
Advanced Analytics Institute
University of Technology Sydney
Sydney, Australia
Anand.Chinchore@uts.edu.au

Guandong Xu
Advanced Analytics Institute
University of Technology Sydney
Sydney, Australia
Guandong.Xu@uts.edu.au

Frank Jiang
Advanced Analytics Institute
University of Technology Sydney
Sydney, Australia
Frank.Jiang@uts.edu.au

***Abstract* – Sybil detection is an important task in cyber security research. Over past years, many data mining algorithms have been adopted to fulfill such task. Using classification and regression for sybil detection is a very challenging task. Despite of existing research made toward modeling classification for sybil detection and prediction, this research has proposed new solution on how sybil activity could be tracked to address this challenging issue. Prediction of sybil behaviour has been demonstrated by analysing the graph-based classification and regression techniques, using decision trees and described dependencies across different methods. Calculated gain and maxGain helped to trace some sybil users in the datasets.**

Keywords—Classification and regression, C4.5; sybil detection, mobile social network, entropy model, decision tree, random forest, behaviour analytics.

I. INTRODUCTION

A huge amount of data is being collected and stored in databases across the world and its space stations, and this trend continues to increase year upon year. So much valuable knowledge is hidden in these database, it is practically impossible to mine them without an automatic extraction method. Over past years, many algorithms, called nuggets, have been created to extract this knowledge using various methodologies such as classification, association rules, clustering, and many more.

Sybil detection is an important topic in cyber security research. The evolution of sybil defense protocols have leveraged the structural properties of the social graph, with an underlying distributed system, to identify sybil identities. Researcher team first clarified the deep connection between sybil defense and the theory of random walks which led to a community detection algorithm that, for the first time, offered provable guarantees in the context of sybil defense [2]. Proposed research the sybil guard approach explains, sybil guard, is a new protocol for defending against sybil attacks without relying on a trusted central authority [4]. Sybil guard exploits this property to bind the number of identities a malicious user can create. The researchers proved the effectiveness of sybil guard both analytically and experimentally [5]. Network of friends contain the honest devices, and its networks of foes contain the suspicious devices. With the help of these two networks, the device is then able to determine whether an unknown individual is carrying out a sybil attack or not [6]. Mining (Social) Network Graphs to Detect Random Link Attacks research mine the social networking graph extracted from user interactions in the communication network to find RLAs and formally define RLA and show that the problem of finding an RLA is (theory) NP-complete [7]. Discussing defending sybil attacks in specific types of MSNs based on the past focus researchers. Research proposed a security mechanism to detect and eliminate sybil nodes [8]. Researched on sybil attacks and their defense in the IoT proposed survey sybil attack and defense system in IoT. Their research explained about the types of sybil attacks considering sybil attacker's capabilities. Also, the research presented some sybil defense schemes, with a social graph based sybil detection, behaviour classification based sybil detection and mobile sybil detection with the comprehensive comparisons [9]. Sybil attackers frequently change their pseudonyms to cheat other users. Researcher investigated the contact statistics of the used pseudonyms and detected sybil attackers by comparing the contact statistics of pseudonyms from normal users and that from sybil attackers [10].

II. CLASSIFICATION MODELS FOR SYBIL DETECTION

Classification consists of predicting a certain outcome, based on a given set of inputs. Typically, an algorithm processes a training set, containing a set of attributes and the respective outcome, to discover the relationships between the attributes that make the outcome possible. The algorithm is then given an unseen dataset, called the prediction set, which contains a similar set of attributes without the outcome. The algorithm then analyses the input and attempts to produce a prediction.

Classification models help to predict categorical class labels, which may be discrete or nominal. Constructed models classify data based on a training set and use the resulting class labels values as attributes with which to classify new data.

Decision trees are a very popular method of classification for supervised learning with nominal classes that have dependent labels. Using decision trees to predict nominal class behaviour given a simple 'yes' or 'no' is straightforward. However in some cases, nominal classes can specify more than two options, and hence which kind of entities responded to which options can be classified and/or predicted.

The first step in the development of this classification model was to evaluate the dataset using a decision tree.

A decision tree was built by first searching for users identified as suspicious. These users were split into the mostly evenly divided groups, given a set of observations and their features. Currently mixed up column 1 or User 1 and column 2 or User 2 connections with other respective columns of starting time, ending time including, etc.

This decision tree helped to classify the model according to previous observations, but dividing the nodes into group presented new questions:

1. Does this user connect to specific user regularly or do they connect randomly to anyone?

2. Is there a chance that this user might connect to each and every available node?

3. If they connect regularly, do they reconnect with friends or to other random nodes?

The decision trees generated by C4.5 can be used for classification and regression. The C4.5 algorithm is developed by Ross Quinlan, which use to generate decision trees. It is an extension of Quinlan's earlier ID3 algorithm. C4.5 often referred to as a statistical classifier.

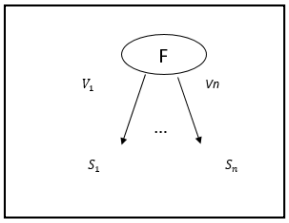
At each node of the tree, C4.5 chooses the data attribute that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (ie Kullback–Leibler divergence) and the difference in entropy (ie information theory). The attribute with the highest normalised information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sub-lists.

A. The C4.5 Algorithm

This algorithm has a few base cases. All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class. None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class. Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

•The algorithm operates over a set of training instances, C.

•If all instances in C are in class P, create a node P and stop, otherwise select a feature or attribute F and create a decision node.



•Partition the training instances in C into subsets according to the values of V.

•Apply the algorithm recursively to each of the subsets C.

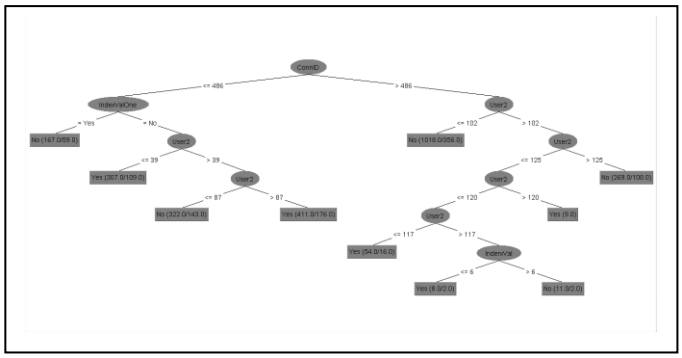


Fig. 1. Decision tree process

The research explain why this approach works, and why it is better for finding sybil users and their behaviour.

Further research idea helps to effectively view the data from many angles. It also eliminates insignificant features and nodes to improve prediction, help build a robust model, and more easily draw parallels with other datasets. However, the current research scope does not consider several crucial details. Each connection set requires an individual model, and observations from honest users and their connection times are irrelevant, so should be discarded. Building model of many nodes at once presents some challenges. We are going to parallelised model building by group it by first column single node in future research.

B. Decision trees, entropy and gain

In this section, decision trees are used to split data in to different forms and a number of trees are built based on information available in the dataset.

Decision trees work very differently than naïve Bayes. When predicting a sybil user using a decision tree, all the nodes and their possible connections need to be observed. For example, the Infocom06 dataset records 98 users, connecting to over 4000 users multiple times, which generates more than 200,000 connections. The set also includes the length of time the nodes were connected and its index values.

To predict whether or not a user is sybil, each user and their connected node are grouped. This procedure is repeated for each node in User1 column. The Manual intervention is required to determine how many other attempts to predict how many other nodes the selected node is connected to. Other factors for consideration include: whether the selected node is connected to a limited or large number of other nodes; whether those nodes have further connections that are limited or large; and whether the selected node is connected to other nodes for a specific or non-specific time.

Trees were selected for further random forest processing, based on these determinations. The key to generating a decision tree for each node is to glean what type of connections the node has made and why it made those connections. Examining each one of these attributes, such as User1 column and User2 column node, start and end time of connection, index values, etc. and try to meet the best interest possible

about assuming its connection, whether the selected node is suspicious and what other factors may influence the consideration of a node as sybil.

Generally speaking, attributes in the available data are examined and then used to split the data into subsets. For example, the number of connections a given node has made could be: specific (ie regular/honest); limited (ie only few connections); or large (ie connect to most of available or more than selected average). Additionally, those three subsets will contain further subsets of information.

If subset is pure then it is honest. That means if the node had specific connections, and connected over an unspecified time, the process will stop. Otherwise it will continue to investigate the behaviour and try to further split the data – a sort of variation on a divide and conquer algorithm.

The new datasets were tested for predictions next, to see which examples fell into which subsets. The dominant class is then used for that subset. For example, if the selected node always connects to the average number of nodes for an unspecified time, then is it certain that the selected node is honest and the branch is split at higher level displaying remaining values.

If the selected node is categorised as suspicious or considerably suspicious, the node is divided into even more branches based on the start and end time of its connections. These branches of the decision tree: have a lower connected node count; and/or, are less than or equal to the time difference. The tree will split further again if both values are true, otherwise the node will be deemed honest.

The algorithm stops when two pure subsets are found, because there is no need for further division after another subset is deemed pure, and there is no need to further investigate sybil activity. The next node is selected and the tree building process is restarted. The node is considered to be honest if there is a high probability of a low connection count, and the node has connected with other nodes for an undefined specific time. If a node in the second branch of the tree is not pure, the algorithm checks for index value and will repeat the procedure. If the second branch is also not found to be pure, then based on column six of Infocom06 (connection id) value of 0's count (>= avg 0's count), it will further branch the tree and the node will certainly be considered sybil.

Finding the split:

Number of branches *fi* – each feature in dataset

Let *fi* be the feature with the greatest gain

Create a decision node that splits on *fi*

For each split *Spl* on *fi*, *FindSplit(Spl)*

```
Algorithm 1 - C4.5:
1   Split (node, (number of branch (fi)))
2   A <- the best attributes for splitting the fi
3   Decision attributes for this node <- A
4   For each value of A, create new child node
5       Split training fi to child nodes
6       Do each child node / subset
7       If subset is Pure: STOP
8       Else
9           Split (child_node, (subset of fi)
10  Consider node: Considerably suspicious / honest
11  Repeat
12  If subset is Pure: STOP
13      Consider node: Honest
14      Consider node: Suspicious
15  Repeat
16  If subset is Pure: STOP
17  Else
18  Consider node: Sybil and user for RF (Random forest)
```

Selecting the best attribute: Nodes can be connected based on: time of connection; index value; or connection id's 0's count. There are pros and cons associated with each attribute. The purity of splits have to be measured, so the ideal choice is the attribute with all pure subsets to help reduce the dataset and eliminate data for further examination.

Some cases have a 50% chance of being pure or impure. So split the generated subset in pure side is good and not to split the generated subset likewise have complete impurity. A matrix that can measure both the purity of the subset and its uncertainty is required. The uncertainty value is a measure of probability that, after the data of a particular subset is split, a random item within that subset is positive or negative. A completely uncertain number, with a 50/50 chance of being positive or negative, demonstrates that the node is honest or suspicious.

We cannot use a posterior probability – the probability that an observation will fall into a group before the data is collected – because the subset needs to be symmetrical. It means, a pure subset or honest node has low regular connection with uncertain frequency of time connections which is good similarly pure subset or honest node have high connection with uncertain frequency of time connections. So it can't be a probability of positive. It must be some that is symmetric of positive side and negative side.

C. Entropy

Calculating entropy is a way to measure the uncertainty of a class in a subset of *fi*. Entropy is defined as:

$H(S) = -p(+)\log_2 p(+) - p(-)\log_2 p(-)$ *bits*

- *S* is subset of training example
- *p(+) / p(-) … % of positive / negative examples in S*

Entropy calculations are based on binary values of yes and no, or 1's and 0's and. The original dataset did not have any text values.

Hence, if the impure subset = 1 bit:

H(S) = - (number of purity / total number) log2 (number of purity / total number) - (number of impurity / total number) log2 (number of impurity / total number) = 1 bit

If the subset is pure it = 0 bits:

H(S) = - (number of purity / total number) log2 (number of purity / total number) - (number of impurity / total number) log2 (number of impurity / total number) = 0 bits

Entropy tells us how pure and impure is one set and subset.

Now the information must be segregated from several different subsets, because the attribute selected for the split has different values. A not-so-simple average is used. The aim is to have as many items considered to be honest as possible in pure subset, and a drop in entropy after the split is expected:

$$Gain\ (S,A) = H(S) - \sum_{V \in value\ (A)} \frac{|Sv|}{|S|} H(Sv)$$

This is taken in to account when adding the entropy value, by putting a weight on each entropy. A weight is put on each subset, which is the size of that subset divided by the overall number of *fi* there are at this split node.
V is a possible or particular value of A
S is a set of fi (example) {X}
Sv subset where X_A= V ,which is all the fixed time or all the maximum connections or all the connection id's 0 count.

This is the entropy of those subsets and the weight indicates what proportion of the items failed in to the rather fixed time that is no time difference or all the maximum connections that is highest number of connectivity. So items failed in information gain calculation to fixed time is multiplied by the how pure was the fixed time or all the maximum connections.

If the resulting subset is large and pure then it's good and if the resulting subset is small and impure then it's bad. If the result is good if it returns a large pure subset, any sized then it is good or if we get small or big impure subset it is bad.

In summary, the average purity is weighted by the size of the set's average purity after the split on attribute A, because there were some positive and some negative splits.

Looking at, the difference in entropy before and after the split is the determining point at which interpret whether we are sure how much we are certain before split and how much more certain after the split. That is which node is considered as honest or which going to be consider as, suspicious. This is called information gain.

III. EXPERTIMENTS

A. *Processing method*

The part of the research contributes to classification and regression models using the C4.5 algorithm for outcome generation. To prepare, the dataset was split based on the User1. There are 98 users in the first column, each connecting to many other users in the second column (User2). The split carries information in both columns.

The splitting process*:* To split the dataset an automated function in R was used to build the data frame. The complete original dataset was saved in to one frame, and a new variable for data frame was created.

A specific number of users was selected from column one of the dataset. Data with all columns related to the specific user number was fetched and exported to the group of data into new files and saved with a user number for future recognition.

B. *Binary count of columns: Total count for each node and connection*

After the split process in number of datasets, the total number of specific dataset user connections to other users was calculated. For example, calculating the entropy for the first level 1(split), requires knowledge of how many times node 1 connected to the second column of node 3. This brings us the total count of node 1 and 3 connection.

The process is as follows: fetch the split dataset for a specific user in one frame; create a new variable for the data frame; use the inbuilt count function to access User1 and User 2 columns of the dataset; return the count of each pair of connections; fetch data from User1 and User2 related to the specific user number selected with its count; export the group of data into a new file, and save the pair count value for future recognition; with its connection frequency between User 1 and User 2. The same process repeated for time difference, index value and connection id columns by selecting specific columns and values.

Entropy generation and analysis: Analysis of user vs TimeDiff

The following formula shows entropy generation for the first node, based on user count and time difference, and is calculated with the help of User1 and User2 pair and other similar node total in dataset with TimeDiff and its frequency.

then further calculate entropy for both side nodes.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

As specified, the entropy model has a right side and a left side. The left side tends toward 'yes' or calculation of leave node entropy (to 0) which helps to eliminate the complications in the end results and predictions. The right side tends toward 'no' or leave entropy to non-zero value (an entropy value >0).

Algorithm 2: Entropy generation per user - TimeDiff datasets:

1 Infocom06tbUsers <- Read dataset – Total user frequency count and save in data frame
2 N1 =Calculate length of dataset
3 Infocom06tbTimeDiff <- Read dataset – user frequency count based on TimeDiff and save in data frame
4 N2 =Calculate length of dataset
5 Data1 = Null #empty data frame to save new data
6 Loop L1 Infocom06tbUsers until N1 count
7 Access and save each row and column element in to variables
8 Loop L2 Infocom06tbTimeDiff until N2 count
9 Access and save each row and column
10 Check L2 column 2 user with L1 column 2 user are equal
11 Check L2 Column 2 user with next L2 Column 2 user are equal
12 Check L2 Column 2 user same as next L2 Column 2 user then
13 Calculate Entropy
14 Else Entropy set to 0
15 Bind Data in Data1 frame
16 Repeat all from same pairs in Infocom06tbUsers
17 Save and write Data 1 in NewFile

Repeating this algorithm for all users and time difference frequency count datasets. This generates new files for the pair with User1, User2, Frequency 1, and EntropyTimeDiff.

With the modification, analysis for time difference vs index value and analysis for index value and vs connection id is calculated.

C. *Information gain analysis and generation*

The process gain calculation, with the help of entropy, only uses an entropy calculation based on time difference, because there is no need to calculate gain for second and third stage of

entropy when the first stage result is 0. It happened because of entropy single binary value of 0.

Hence, the information gain model for the time difference tables is:

$$Gain\ (S, A) = H(S) - \sum_{V \in value(A)} \frac{|Sv|}{|S|} H(Sv)$$

Where,
S – user TimeDiff total entropy
A – selected attribute

Some of the returned gain values were either 0 or NA because it could not find second value and equation generate 0 results.

Based on the calculated information gain for each table and each node, the single maximum gain and their node number is calculated to help predict sybil users.

Algorithm 3: Calculate information gain for each node and dataset
1 Infocom06tbUsersIG <- Read dataset – Total user frequency count and save in data frame
2 I1 =Calculate length of dataset
3 Infocom06tbTimeDiffIG <- Read dataset – user frequency count based on TimeDiff and save in data frame
4 I2 =Calculate length of dataset
5 IGaintbUser = Null #empty data frame to save new data
6 Loop G1 Infocom06tbUsersIG until I1 count
7 Access and save each row and column element in to variables
8 Loop G2 Infocom06tbTimeDiffIG until I2 count
9 Access and save each row and column
10 Check G2 column 2 user with Last G2 column 2 user are not equal
11 Check G2 column 2 user with G1 column 2 user are equal
12 Check G2 column 2 user with next G2 column 2 user are equal
13 Calculate Gain
14 Bind Data in Data1 frame
15 Repeat all from same pairs in Infocom06tbUsersIG
16 Save and write IGaintbUser list in NewFile

D. maxGain analysis

maxGain is the maximum calculated value of the information gain of a single node among other nodes. maxGain, in this research was calculated based on User1 users or split datasets that we have generated to calculate the information gain for each node. The max function of R was used capture the max value of a single dataset including its other row values.

The results were saved in a new file, one by one, as the maxGain for each information gain dataset was calculated.

Algorithm 4: Calculate maxGain
1 Infocom06tbUsersIG <- Read dataset – Total user frequency count and save in data frame
2 M1 =Calculate length of dataset
3 IGainMaxtbUser = Null #empty data frame to save new data
4 maxGain is max value of Infocom06tbUsersIG Gain column
5 Loop Gn Infocom06tbUsersIG from 1 to M1 count
6 Access and save each row and column element in to variables
7 Check Gn Gain with maxGain are equal then
8 Set other values of that row to variables
9 Bind Data in IGainMaxtbUser frame
10 Repeat all from same pairs in Infocom06tbUsersIG\
11 Save and write IGainMaxtbUser row to list in File

IV. CONCLUSION

Modelling classification and regression for sybil detection is a very challenging task. Existing research has only made partial progress toward modeling classification for sybil detection and prediction. This research paper has proposed incremental progress for how sybil activity could be tracked to address this challenging issue. Prediction of sybil behaviour of has been demonstrated by analysing the graph-based classification and regression techniques, using decision trees and described dependencies across different methods.

Calculated gain and maxGain helped to trace some sybil users in the datasets. Decision trees were generated in R using our designed algorithm for each node. Along with observation of the charts and the behaviour of the classification model for Users 1, 4, 8, 12, 16, 18, 19, 42 and 66 were able to predict the behaviour of sybil users. The results were compared to trees generated by WEKA's inbuilt C4.5 algorithm to help evaluate and refine our algorithm. Analysis shows that the trees that mostly fall on the right side have negative leaves and a higher value of suspicious entropy compared to other leaves at the same level. This observation provides confidence that the research results are reasonably accurate, and experimentally prove how and why sybil attacks can be modelled for classification.

V. FUTURE WORK

Based on current predictions, some honest nodes are categorised as sybil attackers. Future research will continue to investigate and refine node identification in mobile social networks.

Random forest processing and the Hadoop system could also be further explored. Generating a random forest using a scoring model via cascading, and its deployment within a Hadoop system are natural next steps.

Future studies will also elaborate on the parallelised model building technique: using training data; grouping observations based on users; and generation of a behavioural model for each group.

Incorporating naïve Bayes and k-nearest neighbour techniques would also increase the scope of this research.

REFERENCES

[1] Abaya, S. A., Gerardo, B. D., "An education data mining tool for marketing based on C4.5 classification technique", e-Learning and e-Technologies in Education (ICEEE), Second International Conference, pp. 289-293, 2013

[2] Alvisi, L., Clement, A., Epasto, A., Lattanzi, S., Panconesi, A., "SoK: The Evolution of Sybil Defense via Social Networks : Security and Privacy (SP)" , 2013: In: IEEE Symposium on security and privacy, pp. 382 -396, 2013

[3] Behera, G., "Privacy preserving C4.5 using Gini index", Emerging Trends and Applications in Computer Science (NCETACS), 2nd National Conference, pp. 1-4, 2011

[4] Haifeng, Yu, Gibbons, P. B., Kaminsky, M., Feng, X., "Sybil limit: A Near-Optimal Social Network Defense Against Sybil Attack"', Networking, IEEE/ACM Transactions, vol- 18(3), pp. 885-898,2010

[5] Haifeng, Yu, Kaminsky, M., Gibbons, P. B., Flaxman, A. D., "Sybil guard: Defending Against Sybil Attacks via Social Networks", Networking, IEEE/ACM, vol -16(3), pp.576-589, 2008

[6] Quercia, D. and Hailes S., "Sybil Attacks Against Mobile Users: Friends and Foes to the Rescue", INFOCOM, Proceedings IEEE, 2010

[7] Shrivastava, N., Majumder, A., Rastogi, R., "Mining (Social) Network Graphs to Detect Random Link Attacks", Data Engineering, ICDE, IEEE 24th International Conference, pp. 486-495c, 2008

[8] Yan, S., Lihua, Y., Wenmao, L., "Defending sybil attacks in mobile social networks", Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference, 163-164, 2014

[9] Zhang, K., Liang, X. Lu, R., Shen, X., "Sybil Attacks and Their Defenses in the Internet of Things", Internet of Things Journal, IEEE, vol.- 1(5), pp. 372-383, 2014

[10] Zhang, K. Liang, X. Lu, R. Yang, K., Shen, X., "Exploiting mobile social behaviors for Sybil detection", IEEE Conference on Computer Communications (INFOCOM), Kowloon, pp. 271-279, 2015

TABLE I. SYBIL USERS INFORMATION GAIN

User1	User2	Frequency	Entropy	Gain	maxGain
1	41	7	0.210429	0	0.391827
4	146	7	0.210429	0	0.391827
8	44	9	0.201169	0	0.384668
12	386	6	0.211632	0	0.384001
16	211	7	0.210429	0	0.391827
18	136	7	0.210429	0	0.391827
19	290	7	0.210429	0	0.391827
42	39	7	0.210429	0	0.391827
66	63	7	0.210429	0	0.391827

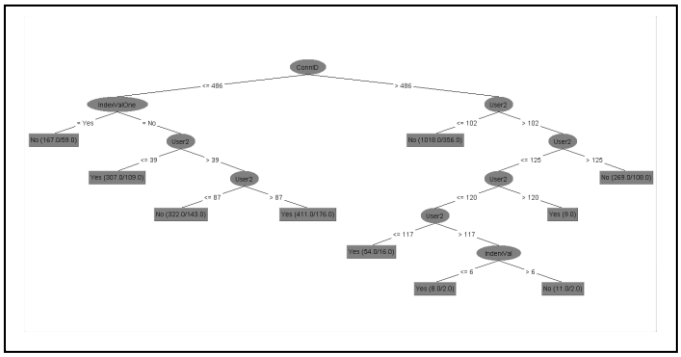


Fig. 2. User 1 decision tree

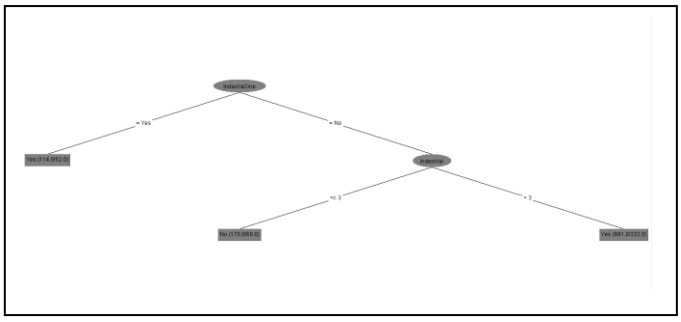


Fig. 3. User 4 decision tree

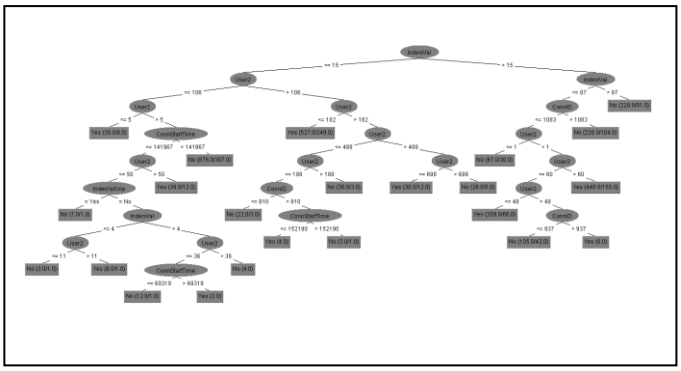


Fig. 4. User 8 decision tree