CS-584 Project Report

Implementation of Result Prediction by Analyzing Data in DotA2

Xin Su, Shiyang Li
Department of Computer Science
Illinois Institute of Technology

April 19, 2016

Abstract

In this short project proposal, an attempt was made to implement a model created by Filip[1] using machine learning and data analysing that can be used to predict the winning team in DotA2 games through the partial data collected with the game processing. The data collection is based on the WebAPI powered by *Valve* Developer [2]. The algorithms to implement the model include Decision Trees(DT) and Random Forest(RF), and compare the results with Support Vector Machine(SVM), Naive Bayes(NB) and AdaBoost(AB) algorithm.

1 Introduction

Right after the *AlphaGo*, developed by *Google DeepMind*, defeated professional player, the world is involving into astonishing that the power of machine learning and artificial intelligent. Indeed, machine learning, a field of computer science designed by human and mastered by systems, is dedicated to learning from the data given by specific task. DotA2 is one of the world popular Multi-player online battle arena(MOBA) competitive games, which attracts more than 12 million unique players each month[3] and holds over $18 million prize pool in the *2015 International DotA2 Championships.*[4]

As the prediction of physical sports, it is significant to predict the winning team in an ongoing game not only to make financial profit but also to build high intelligent system such as *AlphaGo*.

This document was compiled using LaTeX.

2 Related Work

Prediction of the traditional physical sport results has been conducted based on previous game results with related conditions during the game. The analysis mostly is focused on the player behavior. Previous work in predicting DotA2 is more based on the hero lineups picked in the game or the statistics of the players, such as the "Kills-Deaths-Assistants"(KDA) and "Gold-Per-Minute". Specifically, Kevin and Daniel built a recommendation engine for picking heroes in DotA2 with 69.8% accuracy by using logistic regression(LR) in 2013.[5] In 2014, Atish and Michael achieved 62% to predict winner with different team compositions with LR.[6] Kuangyan, Tianyi and Chao obtained 61% accuracy predicting the winning side of DotA2 with LR in 2015.[7]

3 Methodology

3.1 Data Collection

End-game statistics can be retrieved from WebAPI developed by *Valve*. Since the match detail data can be obtained from a unique match ID, the statistics retrieved from WebAPI is used to filter out the games that are with inadequate conditions, such as incomplete games. The valid data sets in this project are "$min\_players = 10$" and "$game\_mode = 1$", which means that the game is with 10 players and game mode is "All Pick", respectively. We retrieved match details by WebAPI with "game_duration" greater than 1080 second to filter out the short duration games. Finally, we selected objective features as "kills", "deaths", "assists", "last_hits", "denies", "gold_per_min" and "xp_per_min". The "radiant_win" is the true result, that is, if "radiant_win" is "true" then "radiant" wins, otherwise "Dire" wins.

Data sets of 1908 games were gathered, consisting games with AP(all pick) game mode played from 2016-04-16 to 2016-04-18 with duration at least 1080 second. And the data is prepared to 90 features with all the 10 players' 9 features above, that is, totally 90 features.

3.2 Algorithm

To gain a better performance model, experiments were conducted by attempting different algorithms, in which include DT and RF, and comparing with SVM, NB and AdaBoost algorithm, respectively. By selecting and modifying each general algorithm, models are created developed and im-

plemented. The accuracy of prediction from each model is the primary evaluation factor. Comparison among the accuracy of each prediction yields to the expected model which holds the one or two highest accuracy scores(or percentages), in which the precondition of each model is the same and with simple complexity. Further experiments are made to study and improve the stabilization of prediction performance.

3.2.1 Decision Trees

Decision Tree(DT) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that prodicts the value of a target variable by learning simple decision rules inferred from the data features.[8] The greedy decision tree learning is implemented. It starts with an empty tree, and one feature is selected to split data. Two problems raised to achieve the learning, first is feature split selection and second is stopping condition. In the first problem, given a subset of data(a node in tree), for each feature, split data of the subset according to the feature and then compute classification error split.[9] Choose the feature with lowest classification error. The classification error is calculated with

$$\begin{aligned} Error &= \frac{\#\ of\ mistakes}{\#\ of\ data\ points} \\ mistakes &= minority\ class \\ data\ points &= majority\ class + minority\ class \end{aligned}$$

To deal with real valued features, threshold split is applied. To find the best threshold split, considering a threshold between points, same classification error for any threshold split between two values of one certain feature. Mid-points are considered to obtain, where leads to finite numbers of splits. The threshold split selection algorithm is that sort the values of a feature, consider each split, compute classification error for threshold split and finally choose the split with the lowest classification error.

In the second problem, one stopping condition is when all data agrees on the true class value, which means nothing to split. Other stopping condition is when it is already splited on all features. To deal with overfitting in decision trees, early stopping for learning decision trees is applied. Deeper trees increase the complexity and lead to overfitting when training error goes too small while true error goes extraordinary huge. Limit depth of tree is included in the implementation by using classification error. Another early stopping condition is when no split improves classification error. In practical, a parameter is added to determine the stopping decision instead

of setting the decreasing size as zero. In addition, the third early stopping condition applied is that stopping if number of data points contained in a node is too small.

3.2.2 Random Forest

Random forests is a notion of the general technique of random decision forests that are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set.[10] A general voting mechanism is majority voting which uses the majority of the results generated by the decision trees as the predicted result. Another way is to add weights to all decision trees. AdaBoost algorithm is a popular ensemble algorithm to give weights to weak learns (classifiers).

3.2.3 AdaBoost

AdaBoost, short for "Adaptive Boosting", is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire who won the Gödel Prize in 2003 for their work.[11] It can be used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms ('weak learners') is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing (e.g., their error rate is smaller than 0.5 for binary classification), the final model can be proven to converge to a strong learner. The algorithm starts with same weight for all points $\alpha_i = 1/N$. For each weak classifier $t = 1\ to\ T$, learn $f_t(x)$ with data weights α_i, compute coefficient $\hat{w}_t$,

$$\hat{w}_t = \frac{1}{2} ln \left(\frac{1 - weight\_error\,(f_t)}{weight\_error\,(f_t)} \right)$$

and then recompute weights α_i,

$$\alpha_i = \begin{cases} \alpha_i e^{-\hat{w}_t} & \text{if } f_t(x_i) = y_i \\ \alpha_i e^{\hat{w}_t} & \text{otherwise} \end{cases}$$

Final model predicts by:

$$\hat{y} = sign\left(\sum_{t=1}^{T} \hat{w}_t f_t(x)\right)$$

3.3 Training and Validation Subsets

For each execution of the main experiments, 5 folds cross validation[12] is applied with data shuffling. For each fold calculation, confusion matrix[13], precision, recall[14], f_score[15] and accuracy are obtained for analyzing.

3.4 Performance Measurement

The confusion matrix is shown in Table 1, where TP, FP, FN and TN denote true positives, false positives, false negatives and true negatives, respectively.

		Prediction Value		
		Positive	Negative	Total
Actual Value	Positive	TP	FN	$TP+FN$
	Negative	FP	TN	$FP+TN$
	Total	$TP+FP$	$FN+TN$	N

Table 1: Confusion Matrix

Precision, recall, F-measure(F_β) and accuracy are then defined as follows, respectively.

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$F_\beta = (1+\beta) \cdot \frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} = \frac{TP+TN}{N}$$

where β is a positive real number in general, and $\beta = 1$ in this case.

4 Result and Discussion

4.1 Parameter Exploration

4.1.1 Decision Trees

To find the optimal parameters for DT, different maximum depths were trained and 5 folds cross validation test with the whole data sets and features with $min\_amt = 5$, $max\_depth = 5$ and $n\_folds = 5$. The confusion matrix, Accuracy, Precision, Recall, F-score and Total Accuracy of training, testing and total data sets are shown in following tables, respectively. We obtained great result with $Total\ Acc. = 0.967$. Figure 1 shows the changing between maximum depth and accuracy.

$$\begin{bmatrix} 548 & 24 \\ 30 & 924 \end{bmatrix} \begin{bmatrix} 576 & 24 \\ 26 & 900 \end{bmatrix} \begin{bmatrix} 577 & 20 \\ 29 & 900 \end{bmatrix} \begin{bmatrix} 552 & 23 \\ 27 & 925 \end{bmatrix} \begin{bmatrix} 567 & 21 \\ 28 & 911 \end{bmatrix}$$

$$\begin{bmatrix} 157 & 4 \\ 5 & 216 \end{bmatrix} \begin{bmatrix} 129 & 4 \\ 9 & 240 \end{bmatrix} \begin{bmatrix} 128 & 8 \\ 6 & 240 \end{bmatrix} \begin{bmatrix} 153 & 5 \\ 8 & 215 \end{bmatrix} \begin{bmatrix} 138 & 7 \\ 7 & 229 \end{bmatrix}$$

Table 2: Data Confusion matrix of DT with 5 Folds Cross Validation for Training(up) and Testing(down)

	Training		Testing	
Class	Radiant	Dire	Radiant	Dire
Accuracy	0.967	0.967	0.967	0.967
Precision	0.967	0.967	0.967	0.967
Recall	0.962	0.970	0.962	0.970
F-score	0.957	0.973	0.957	0.973
Total Acc.	0.967		0.967	

Table 3: Analysis Values for DT with 5 Folds Cross Validation

4.1.2 Random Forest

RF was implemented with $min\_amt = 0$, $max\_depth = 3$, $tree\_amt = 15$, $feature\_size = 0.1$ and $n\_folds = 5$. The confusion matrix, Accuracy,

$$\begin{bmatrix} 705 & 28 \\ 35 & 1140 \end{bmatrix}$$

Table 4: All Data Confusion matrix of DT

Values	Radiant	Dire
Accuracy	0.967	0.967
Precision	0.967	0.967
Recall	0.962	0.970
F-score	0.957	0.973
Total Acc.	0.967	

Table 5: Analysis Values for All Data DT

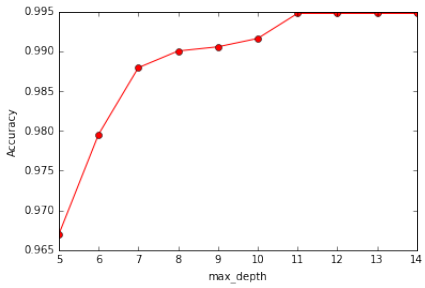


Figure 1: Maximum depth and Accuracy Plot

Precision, Recall, F-score and Total Accuracy of training, testing and total data sets are shown in following tables, respectively. We obtained great result with $Total\ Acc. = 0.979$ with Majority Voting(MV) and $Total\ Acc. = 0.969$ with AdaBoost(AB). In Figure 2, three parameters, maximum depth, amount of DT and feature size, were explored to achieve better prediction accuracy with MV method. By exploring AdaBoost method, Figure 3 was obtained to show the tendency of the accuracy changing along with those three parameters.

$$\begin{bmatrix} 560 & 32 \\ 8 & 926 \end{bmatrix} \begin{bmatrix} 554 & 27 \\ 5 & 940 \end{bmatrix} \begin{bmatrix} 558 & 23 \\ 4 & 941 \end{bmatrix} \begin{bmatrix} 568 & 20 \\ 4 & 935 \end{bmatrix} \begin{bmatrix} 551 & 39 \\ 3 & 934 \end{bmatrix}$$

$$\begin{bmatrix} 135 & 6 \\ 1 & 240 \end{bmatrix} \begin{bmatrix} 144 & 8 \\ 1 & 229 \end{bmatrix} \begin{bmatrix} 144 & 8 \\ 1 & 229 \end{bmatrix} \begin{bmatrix} 140 & 5 \\ 0 & 236 \end{bmatrix} \begin{bmatrix} 134 & 9 \\ 0 & 238 \end{bmatrix}$$

Table 6: Confusion matrix of RF with 5 Folds Cross Validation and MV for Training(up) and Testing(down)

	Training		Testing	
Class	Radiant	Dire	Radiant	Dire
Accuracy	0.978	0.978	0.980	0.980
Precision	0.978	0.978	0.980	0.980
Recall	0.952	0.995	0.951	0.997
F-score	0.971	0.983	0.973	0.984
Total Acc.	0.978		0.980	

Table 7: Analysis Values for RF with 5 Folds Cross Validation and MV

$$\begin{bmatrix} 697 & 36 \\ 4 & 1171 \end{bmatrix}$$

Table 8: All Data Confusion matrix of RF with MV

Values	Radiant	Dire
Accuracy	0.979	0.979
Precision	0.979	0.979
recall	0.951	0.997
F-score	0.972	0.983
Total Acc.	0.979	

Table 9: Analysis Values for All Data RF with MV

$$\begin{bmatrix} 569 & 21 \\ 3 & 933 \end{bmatrix} \begin{bmatrix} 575 & 23 \\ 7 & 921 \end{bmatrix} \begin{bmatrix} 542 & 22 \\ 15 & 947 \end{bmatrix} \begin{bmatrix} 573 & 26 \\ 8 & 920 \end{bmatrix} \begin{bmatrix} 559 & 22 \\ 12 & 934 \end{bmatrix}$$

$$\begin{bmatrix} 139 & 4 \\ 1 & 238 \end{bmatrix} \begin{bmatrix} 129 & 6 \\ 0 & 247 \end{bmatrix} \begin{bmatrix} 163 & 6 \\ 3 & 210 \end{bmatrix} \begin{bmatrix} 131 & 3 \\ 3 & 244 \end{bmatrix} \begin{bmatrix} 149 & 3 \\ 3 & 226 \end{bmatrix}$$

Table 10: Confusion matrix of RF with 5 Folds Cross Validation and AB for Training(up) and Testing(down)

	Training		Testing	
Class	Radiant	Dire	Radiant	Dire
Accuracy	0.979	0.979	0.983	0.983
Precision	0.979	0.979	0.983	0.983
Recall	0.961	0.990	0.970	0.991
F-score	0.973	0.983	0.978	0.986
Total Acc.	0.979		0.983	

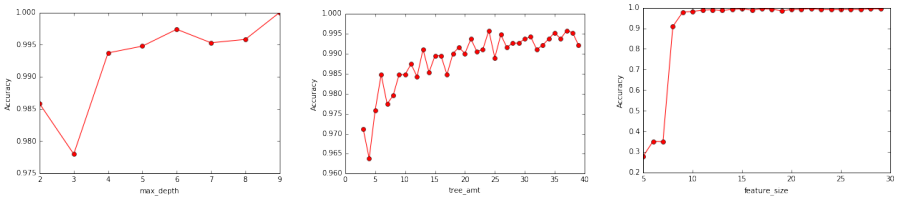
Table 11: Analysis Values for RF with 5 Folds Cross Validation and AB

$$\begin{bmatrix} 684 & 49 \\ 10 & 1165 \end{bmatrix}$$

Table 12: All Data Confusion matrix of RF with AB

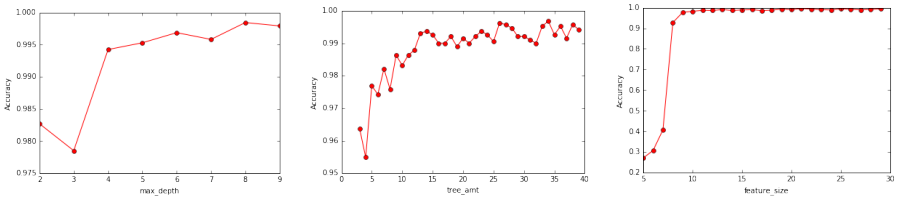
Values	Radiant	Dire
Accuracy	0.969	0.969
Precision	0.969	0.969
recall	0.933	0.991
F-score	0.959	0.975
Total Acc.	0.969	

Table 13: Analysis Values for All Data RF with AB



(a) DT Maximum Depth in RF and Acc. Plot (b) DT Amount in RF and Acc. Plot (c) DT Feature Size in RF and Acc. Plot

Figure 2: Three Parameters Exploration with MV Method Plots



(a) DT Maximum Depth in RF and Acc. Plot (b) DT Amount in RF and Acc. Plot (c) DT Feature Size in RF and Acc. Plot

Figure 3: Three Parameters Exploration with AB Method Plots

4.2 Classifier Comparison

The results show that all three algorithms (DT, RF(MB) and RF(AB)) are all very fast and the generated models are all fits very well with ideal accuracy. For the efficiency, these three algorithms runs relatively fast, because the RF uses DT as its basic unit and DT does not include too much matrix calculations. The RF runs a little slower than DT does, without losing accuracy. And from the result of precision for each class, it concludes that the DTs and RF are both fits each class perfectly. So it strongly proves that the model is stable. However, as the DT runs with pretty accuracy, that means the AdaBoost won't contribute much, and the result shows that exactly: it acquires almost the same accuracy by using or not using AdaBoost voting.

From the tree generated by DT ($max\_depth = 5$) (See attached figure), we can see that the first features used to split the data is the 6th feature which is the first player's "gold_per_min" feature, and second level uses two features: 1st feature which is the "kills" from the first player, 27th feature which is the "xp_per_min" from the third player. As we get result using deeper trees we can see these three kinds of features: "gold_per_min", "kills",

"xp_per_min" are more often to be chosen as the split features. This is easy to understand. From the game, getting more gold and experience brings the ability of growing faster more aggressively, while "killing" more hostages means hinder the development of the rival.

In addition, results obtained are compared with build-in Classifiers, such as SVM, Gaussian NB, DT(sklearn) and RF(sklearn) from Sklearn packages. Details are shown in Table 14.

Model	Accuracy	Duration
DT	0.99	$\leqslant$ 1s
RF(MV)	0.984	$\leqslant$ 1s
RF(AB)	0.982	$\leqslant$ 16s
DT(sklearn)	0.79	$\leqslant$ 0.5s
RF(sklearn)	0.985	$\leqslant$ 0.5s
SVM(sklearn)	0.99	$\leqslant$ 0.1s
GNB(sklearn)	0.99	$\leqslant$ 0.1s

Table 14: Analysis Values for All Data DT

5 Conclusion

The DT and RF algorithms are pretty neat. DT uses key features to separate the datasets recursively and makes the classification more "profound" as it goes deeper into the tree. RF solves the overfitting problem that DT may have. Both runs fast because of less manipulation of the dataset matrix. They are also easy to implement by using recursion. For this case in the DotA2 game. the features retrieved are very characterized and they are relatively independent to each other, therefore are very adaptive to the algorithms.

In the future, this model can be embedded into larger model to predict the game result in real time along the game timeline. Also, it can also be combined with mother models to classify player's role in the team so that can predict the game better.

References

[1] F. Johansson and J. Wikström, "Result prediction by mining replays in dota 2," Master's thesis, Blekinge Institute of Technology, SE-371 79 Karlskrona, Sweden, 2015.

[2] "The WebAPI Website," 2012. http://dev.dota2.com/showthread.php?t=58317 [Online; accessed 10-Mar-2016].

[3] "Dota2 Offical Blog," 2016. http://blog.dota2.com [Online; accessed 10-Mar-2016].

[4] "2015 International DotA2 Championships Overview," 2015. http://www.dota2.com/international/overview/ [Online; accessed 10-Mar-2016].

[5] D. P. Kevin Conley, "A Recommendation Engine for Picking Heroes in Dota2," 2013.

[6] M. P. Atish Agarwala, "Learning Dota 2 Team Compositions," 2014.

[7] C. M. Kuangyan Song, Tianyi Zhang, "Predicting the winning side of DotA2," 2015.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[9] C. G. Emily Fox, "Classification: A machine learning perspective," University of Washington, Coursera, 2016.

[10] Wikipedia, "Random forest — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=Random_forest&oldid=700408350 [Online; accessed 18-April-2016].

[11] Wikipedia, "Adaboost — wikipedia, the free encyclopedia," 2016. [Online; accessed 19-April-2016].

[12] Wikipedia, "Cross-validation (statistics) — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=Cross-validation_(statistics)&oldid=705824849 [Online; accessed 24-February-2016].

[13] Wikipedia, "Confusion matrix — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=Confusion_matrix&oldid=706277467 [Online; accessed 21-March-2016].

[14] Wikipedia, "Precision and recall — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=Precision_and_recall&oldid=704896315 [Online; accessed 21-March-2016].

[15] Wikipedia, "F1 score — wikipedia, the free encyclopedia," 2016. https://en.wikipedia.org/w/index.php?title=F1_score&oldid=703225135 [Online; accessed 21-March-2016].

```
@@@@@@@@@@@@ printing tree start @@@@@@@@@@@@
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=7, th=2.5)
                                                  depth = 5 (LEAF result = -1)
                                      depth = 3 (fid=12, th=363.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=15, th=4.5)
                                                  depth = 5 (LEAF result = 1)
                         depth = 2 (fid=1, th=5.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=3, th=12.5)
                                                  depth = 5 (LEAF result = 1)
                                      depth = 3 (fid=2, th=5.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=31, th=8.5)
                                                  depth = 5 (LEAF result = -1)
               depth = 1 (fid=26, th=367.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=2, th=5.5)
                                                  depth = 5 (LEAF result = -1)
                                      depth = 3 (fid=1, th=5.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=43, th=3.5)
                                                  depth = 5 (LEAF result = -1)
                         depth = 2 (fid=8, th=4.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=12, th=363.5)
                                                  depth = 5 (LEAF result = -1)
                                      depth = 3 (fid=36, th=3.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=27, th=307.5)
                                                  depth = 5 (LEAF result = -1)
depth = 0 (fid=5, th=351.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=4, th=29.0)
                                                  depth = 5 (LEAF result = -1)
                                      depth = 3 (fid=2, th=5.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=3, th=12.5)
                                                  depth = 5 (LEAF result = -1)
                         depth = 2 (fid=15, th=4.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=9, th=6.5)
                                                  depth = 5 (LEAF result = -1)
                                      depth = 3 (fid=51, th=6.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=35, th=4.5)
                                                  depth = 5 (LEAF result = 1)
               depth = 1 (fid=0, th=2.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=65, th=8.5)
                                                  depth = 5 (LEAF result = 1)
                                      depth = 3 (fid=2, th=5.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=3, th=12.5)
                                                  depth = 5 (LEAF result = -1)
                         depth = 2 (fid=1, th=5.5)
                                                  depth = 5 (LEAF result = -1)
                                            depth = 4 (fid=15, th=4.5)
                                                  depth = 5 (LEAF result = 1)
                                      depth = 3 (fid=23, th=5.5)
                                                  depth = 5 (LEAF result = 1)
                                            depth = 4 (fid=16, th=6.5)
                                                  depth = 5 (LEAF result = -1)
@@@@@@@@@@@@ printing tree end @@@@@@@@@@@@
```