Optimized Random Forest Classifier for Drone Pilot Identification

Aysha Khaled Alharam[1] and Abdulhadi Shoufan[2]

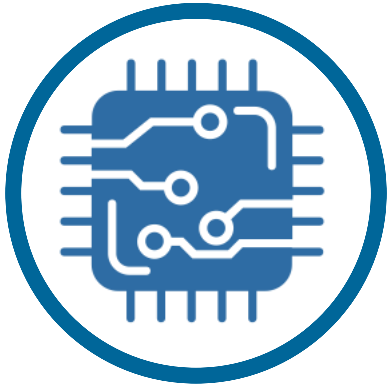
[1]National Space Science Agency Manama, Bahrain
[2]Center for Cyber-Physical Systems, Khalifa University, Abu Dhabi, UAE
Electrical and Computer Engineering Department, Khalifa University, Abu Dhabi, UAE

2020 IEEE International Symposium on Circuits and Systems
Virtual, October 10-21, 2020

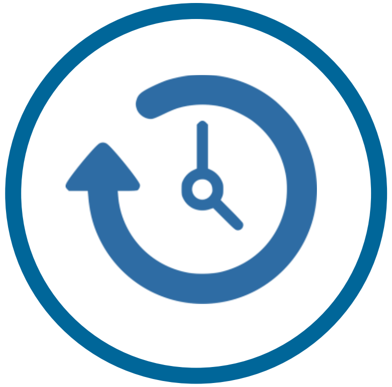
Research Motivation

- Random Forest (RF) is very important machine learning Algorithm
- It has high classification performance compared to other machine learning schemes
- RF needs improvement in its timing performance for real time applications
- Limitation of real-time systems:

Power Consumption



Hardware Complexity



Time Performance

Related Work

02

- Saqib et al. proposed a pipelined hardware architecture
- It aims to improve the throughput of decision tree classification
- This architecture processes each tree level in one pipelining stage
- Number of pipelining stages is equal to the number of tree levels excluding the leaves' level

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.
IEEE TRANSACTIONS ON COMPUTERS

Pipelined Decision Tree Classification Accelerator Implementation in FPGA (DT-CAIF)

Fareena Saqib, Aindrik Dutta, Jim Plusquellic, Philip Ortiz, Marios S. Pattichis
ECE Department, University of New Mexico
fareenasaqib1@gmail.com, (aindrik, jimp, pgortiz, pattichis) @unm.edu

Abstract - **Decision Tree Classification (DTC) is a widely used technique in data mining algorithms known for its high accuracy in forecasting. As technology has progressed and available storage capacity in modern computers increased, the amount of data available to be processed has also increased substantially, resulting in much slower induction and classification times. Many parallel implementations of decision tree classification algorithms have already addressed the issues of reliability and accuracy in the induction process. In the classification process, larger amounts of data require proportionately more execution time, thus hindering the performance of legacy systems. We have devised a pipelined architecture for the implementation of axis parallel binary decision tree classification that dramatically improves the execution time of the algorithm while consuming minimal resources in terms of area. Scalability is achieved when connected to a high speed communication unit capable of performing data transfers at a rate similar to that of the decision tree classification (DT) engine. We propose a hardware accelerated solution composed of parallel processing nodes capable of independently processing data from a streaming source. Each engine processes the data in a pipelined fashion to use resources more efficiently and increase the achievable throughput. The results show that this system is 3.5 times faster than the existing hardware implementation of classification.**

Keywords: Data Mining, Decision Tree Classification (DTC), Hardware Implementation, FPGA

I. Introduction

The process of converting unidentified or unprocessed data into actionable information that is important and valuable to the user is known as data mining [1]. Recent advances in technology and ever increasing demands for analyzing larger datasets have created abundant opportunities for algorithmic and architectural development and innovations. Hence data mining algorithms have become increasingly significant and complex. Similarly there is a great demand for faster execution of these algorithms, leading to efforts to improve execution time and resource utilization.

Decision Tree Classification (DTC) is a widely used classification technique in data mining algorithms. It has applications in daily life; for example, the detection of spam e-mail messages. It is also used in highly sophisticated fields of medicine and astronomy. Several diverse predictive models in classification algorithms including artificial neural networks [2], decision trees [3] and support vector machines [4] have also been previously described in the literature. A number of solutions have also been suggested for hardware implementation by various authors [5-7]. Decision tree classification techniques categorizes each data records/tuples, having set of attributes/properties into subgroups or classes. Assigning of a category or class to each input dataset consists of a two-step process in DTC.

The initial step is induction which involves construction of the decision tree model, where internal nodes and leaves constitute a decision tree model. Each internal node has a characteristic splitting decision and splitting attribute, while the leaves have particular category classification. Construction of a decision tree model from a training dataset/tuple constitutes of two phases. A splitting attribute and a split index are chosen by the model during the first phase. While during the second phase sorting of the tuples among the child nodes is performed based on the decision made in the first phase. This repetitive process is continued till the depth of the tree reaches a desired level. At this point, the decision tree can be used to predict the class of an input tuple which has not been classified yet.

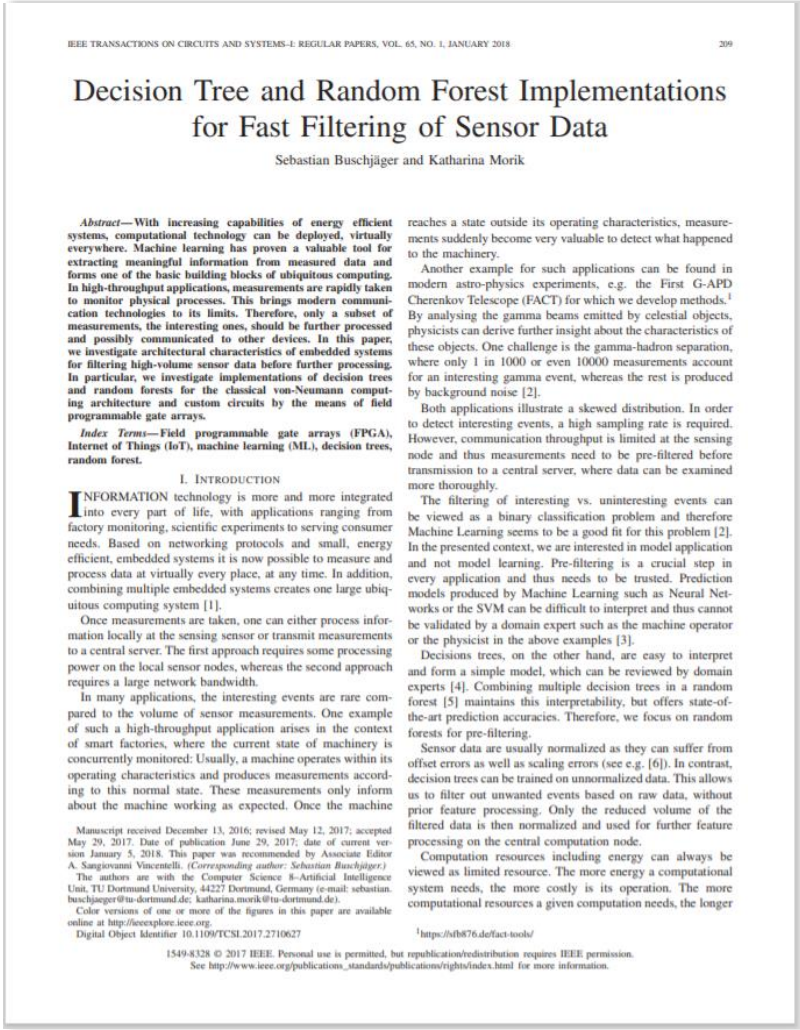
The second step is the classification that includes application of the decision tree model to the test dataset to predict its respective class. The primary goal of such a classification algorithm is to utilize the given training dataset to construct a model which subsequently can be used to sort unclassified datasets into one of the defined classes [8]. Breiman et al [9] presented decision trees approximately two decades ago, and described the decision trees as rooted tree structures, with leaves representing classifications and nodes representing tests of features that lead to those classifications. The accuracy of decision trees has been shown to be better or comparable to other models including artificial neural networks, statistical, and genetic models. The prediction in the classification process commences at the root, and a path to a leaf is followed by using the decision rules governed at each internal node. The characteristic class label to the leaf is then assigned to the incoming tuple.

DTC continues to function at high accuracy even in analysis of large data sets. Current technology advancements in data extraction and storage permit large amount of historic data to be preserved and utilized for data analysis and creation of more realistic classification rules. The property of DTC to function at high accuracy even when handling in large data sets makes it an appealing tool.

Digital Object Indentifier 10.1109/TC.2013.204

Related Work

- Buschjager and Morik proposed an FPGA solution to accelerate binary tree classifications
- It represents the tree as a Boolean function in the disjunctive normal form (sum of products)
- It evaluates all nodes in a single clock cycle
- This requires a full evaluation of all tree nodes and branches at the same time.



IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS–I: REGULAR PAPERS, VOL. 65, NO. 1, JANUARY 2018 209

Decision Tree and Random Forest Implementations for Fast Filtering of Sensor Data

Sebastian Buschjäger and Katharina Morik

***Abstract*—With increasing capabilities of energy efficient systems, computational technology can be deployed, virtually everywhere. Machine learning has proven a valuable tool for extracting meaningful information from measured data and forms one of the basic building blocks of ubiquitous computing. In high-throughput applications, measurements are rapidly taken to monitor physical processes. This brings modern communication technologies to its limits. Therefore, only a subset of measurements, the interesting ones, should be further processed and possibly communicated to other devices. In this paper, we investigate architectural characteristics of embedded systems for filtering high-volume sensor data before further processing. In particular, we investigate implementations of decision trees and random forests for the classical von-Neumann computing architecture and custom circuits by the means of field programmable gate arrays.**

***Index Terms*—Field programmable gate arrays (FPGA), Internet of Things (IoT), machine learning (ML), decision trees, random forest.**

I. Introduction

INFORMATION technology is more and more integrated into every part of life, with applications ranging from factory monitoring, scientific experiments to serving consumer needs. Based on networking protocols and small, energy efficient, embedded systems it is now possible to measure and process data at virtually every place, at any time. In addition, combining multiple embedded systems creates one large ubiquitous computing system [1].

Once measurements are taken, one can either process information locally at the sensing sensor or transmit measurements to a central server. The first approach requires some processing power on the local sensor nodes, whereas the second approach requires a large network bandwidth.

In many applications, the interesting events are rare compared to the volume of sensor measurements. One example of such a high-throughput application arises in the context of smart factories, where the current state of machinery is concurrently monitored: Usually, a machine operates within its operating characteristics and produces measurements according to this normal state. These measurements only inform about the machine working as expected. Once the machine reaches a state outside its operating characteristics, measurements suddenly become very valuable to detect what happened to the machinery.

Another example for such applications can be found in modern astro-physics experiments, e.g. the First G-APD Cherenkov Telescope (FACT) for which we develop methods.[1] By analysing the gamma beams emitted by celestial objects, physicists can derive further insight about the characteristics of these objects. One challenge is the gamma-hadron separation, where only 1 in 1000 or even 10000 measurements account for an interesting gamma event, whereas the rest is produced by background noise [2].

Both applications illustrate a skewed distribution. In order to detect interesting events, a high sampling rate is required. However, communication throughput is limited at the sensing node and thus measurements need to be pre-filtered before transmission to a central server, where data can be examined more thoroughly.

The filtering of interesting vs. uninteresting events can be viewed as a binary classification problem and therefore Machine Learning seems to be a good fit for this problem [2]. In the presented context, we are interested in model application and not model learning. Pre-filtering is a crucial step in every application and thus needs to be trusted. Prediction models produced by Machine Learning such as Neural Networks or the SVM can be difficult to interpret and thus cannot be validated by a domain expert such as the machine operator or the physicist in the above examples [3].

Decisions trees, on the other hand, are easy to interpret and form a simple model, which can be reviewed by domain experts [4]. Combining multiple decision trees in a random forest [5] maintains this interpretability, but offers state-of-the-art prediction accuracies. Therefore, we focus on random forests for pre-filtering.

Sensor data are usually normalized as they can suffer from offset errors as well as scaling errors (see e.g. [6]). In contrast, decision trees can be trained on unnormalized data. This allows us to filter out unwanted events based on raw data, without prior feature processing. Only the reduced volume of the filtered data is then normalized and used for further feature processing on the central computation node.

Computation resources including energy can always be viewed as limited resource. The more energy a computational system needs, the more costly is its operation. The more computational resources a given computation needs, the longer

Manuscript received December 13, 2016; revised May 12, 2017; accepted May 29, 2017. Date of publication June 29, 2017; date of current version January 5, 2018. This paper was recommended by Associate Editor A. Sangiovanni Vincentelli. *(Corresponding author: Sebastian Buschjäger.)*

The authors are with the Computer Science 8–Artificial Intelligence Unit, TU Dortmund University, 44227 Dortmund, Germany (e-mail: sebastian.buschjaeger@tu-dortmund.de; katharina.morik@tu-dortmund.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2710627

[1] https://sfb876.de/fact-tools/

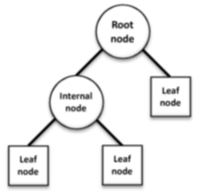
Methodology

Simulated Data

- Using MATLAB
- Real Flight data of piloted drone

Data Pre-processing

- Splitting the trees to find non-threshold trees.



FPGA Implementation

- Spartan6 FPGA used for hardware implementation

Data Explanation

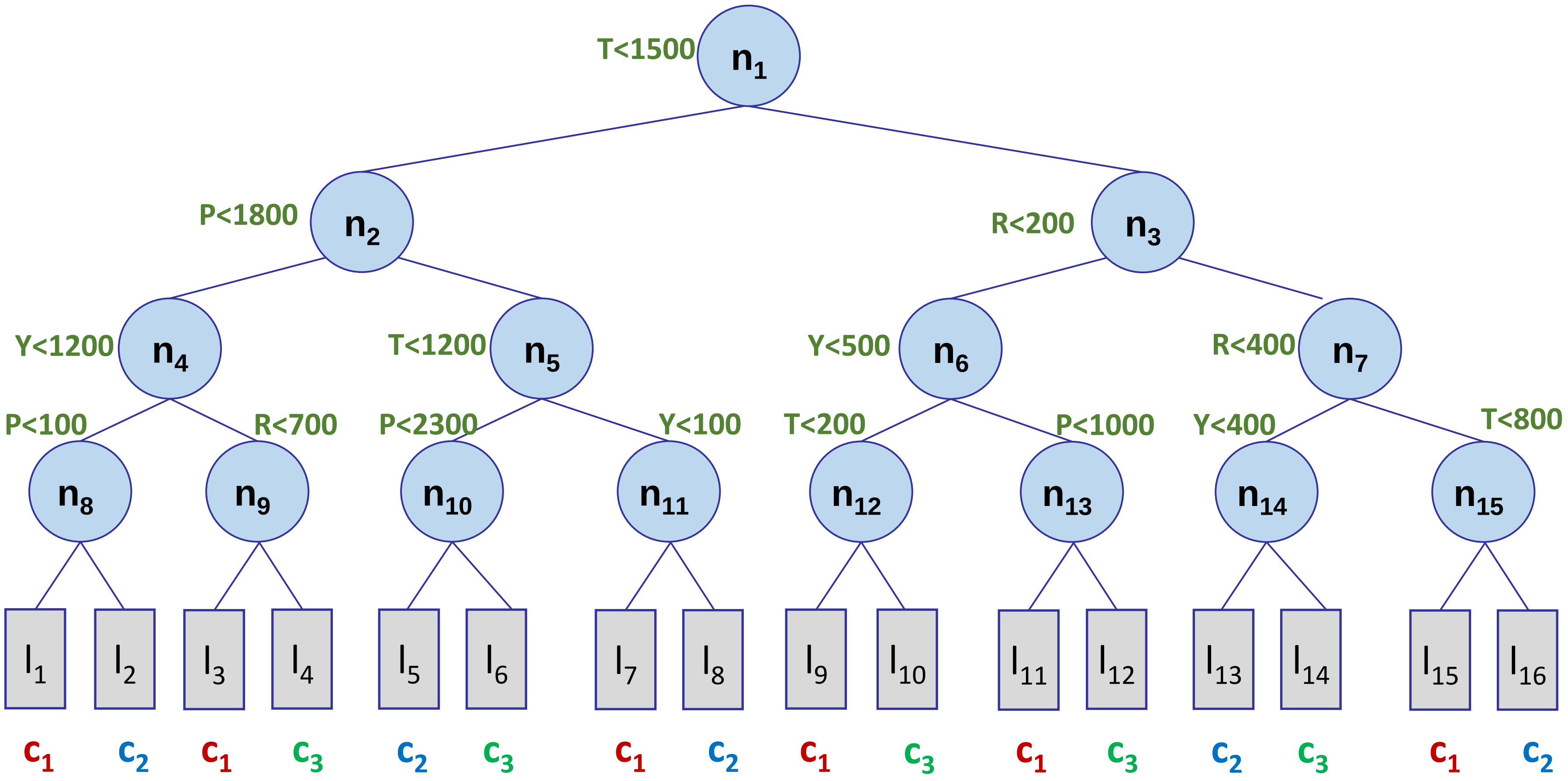
- Remote control signal of piloted drone
- Frequency of analysis 10 Hz
- Digital numbers vary between 0 and 4096
- It has 4 features and 3 classes

Time	T	P	R	Y	Class label

Feature	Description
T	Thrust
P	Pitch
R	Roll
Y	Yaw

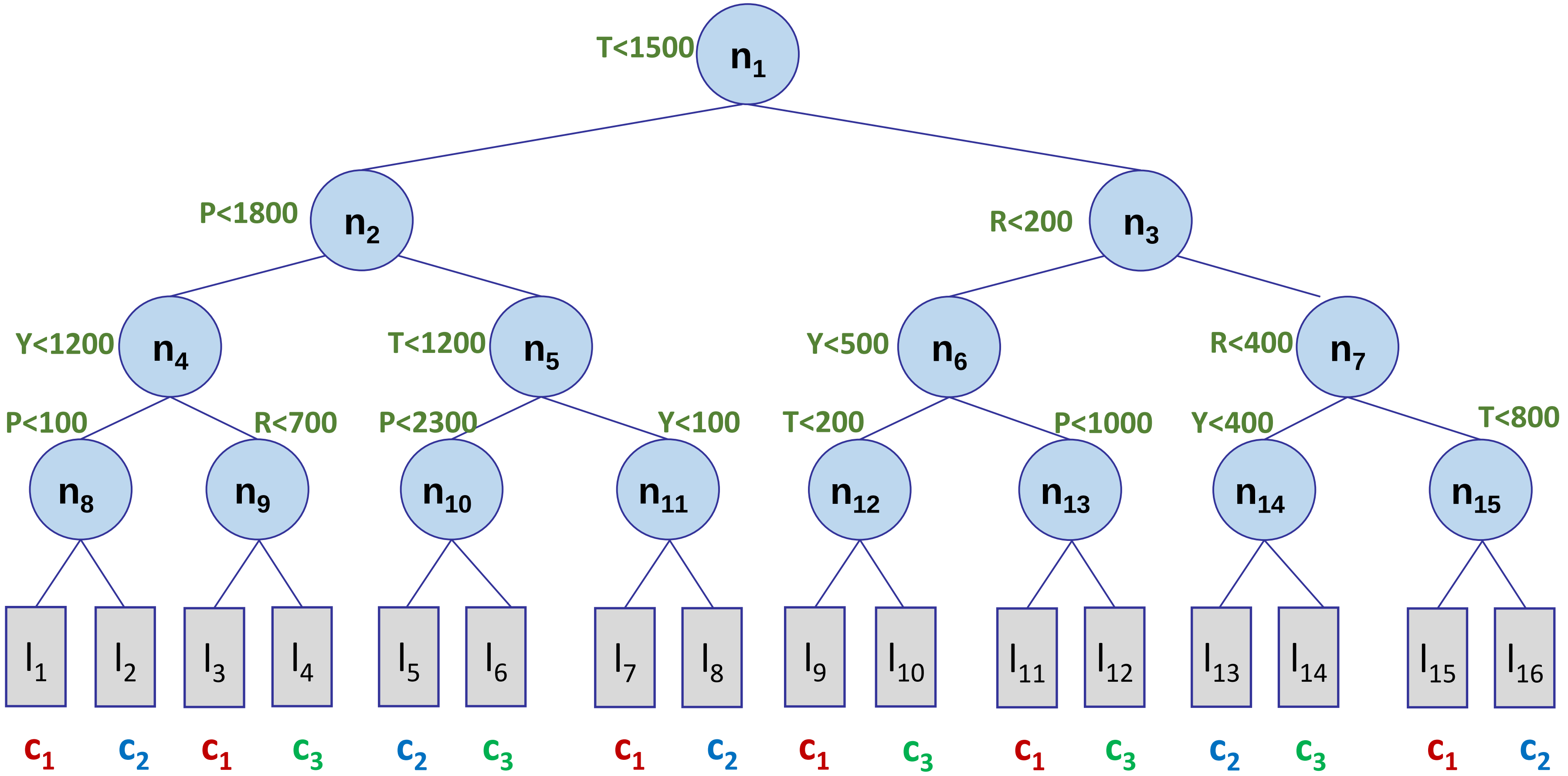
Proposed Algorithm

- Threshold value for each feature is the minimum and maximum node value

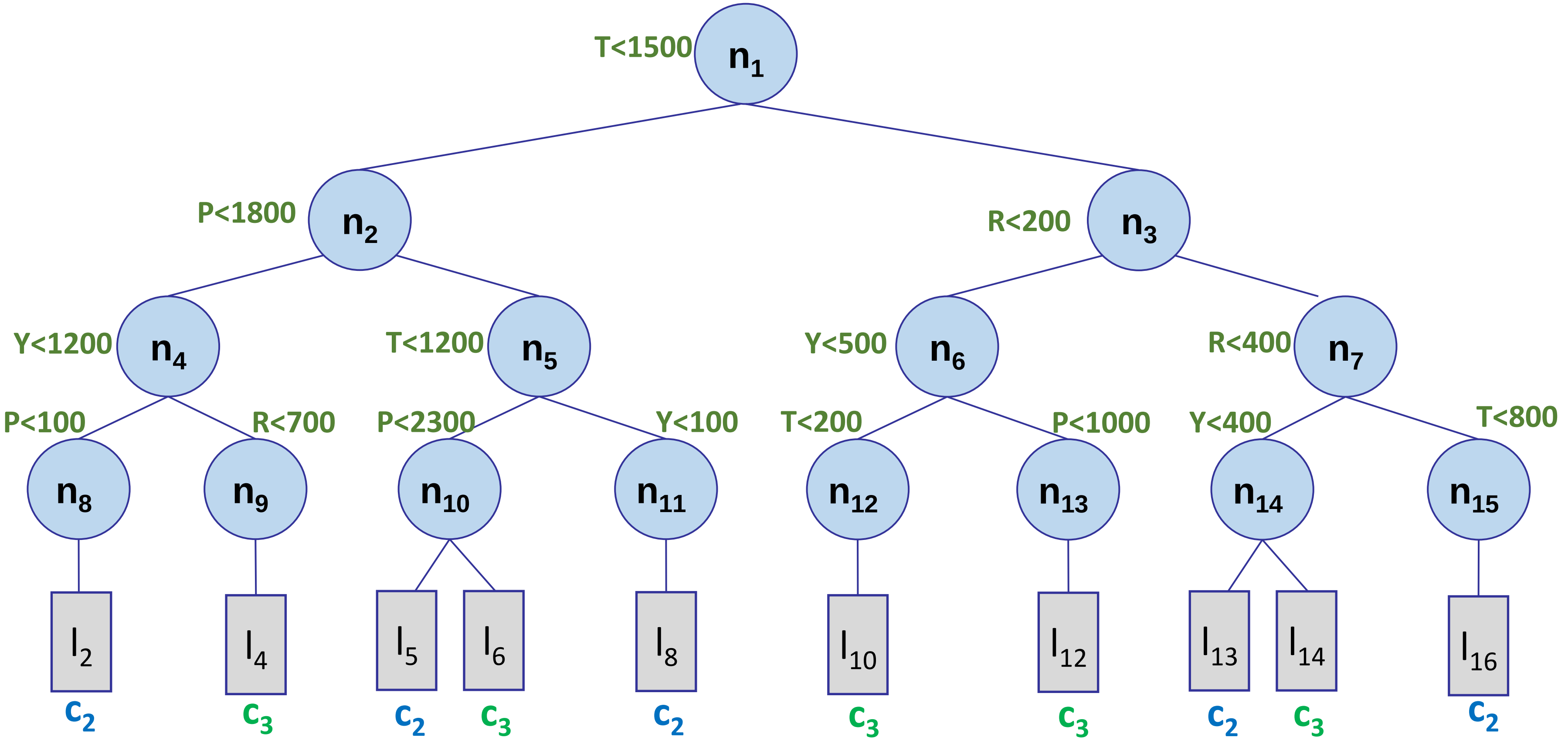


Proposed Algorithm

- Eliminate the most represented class

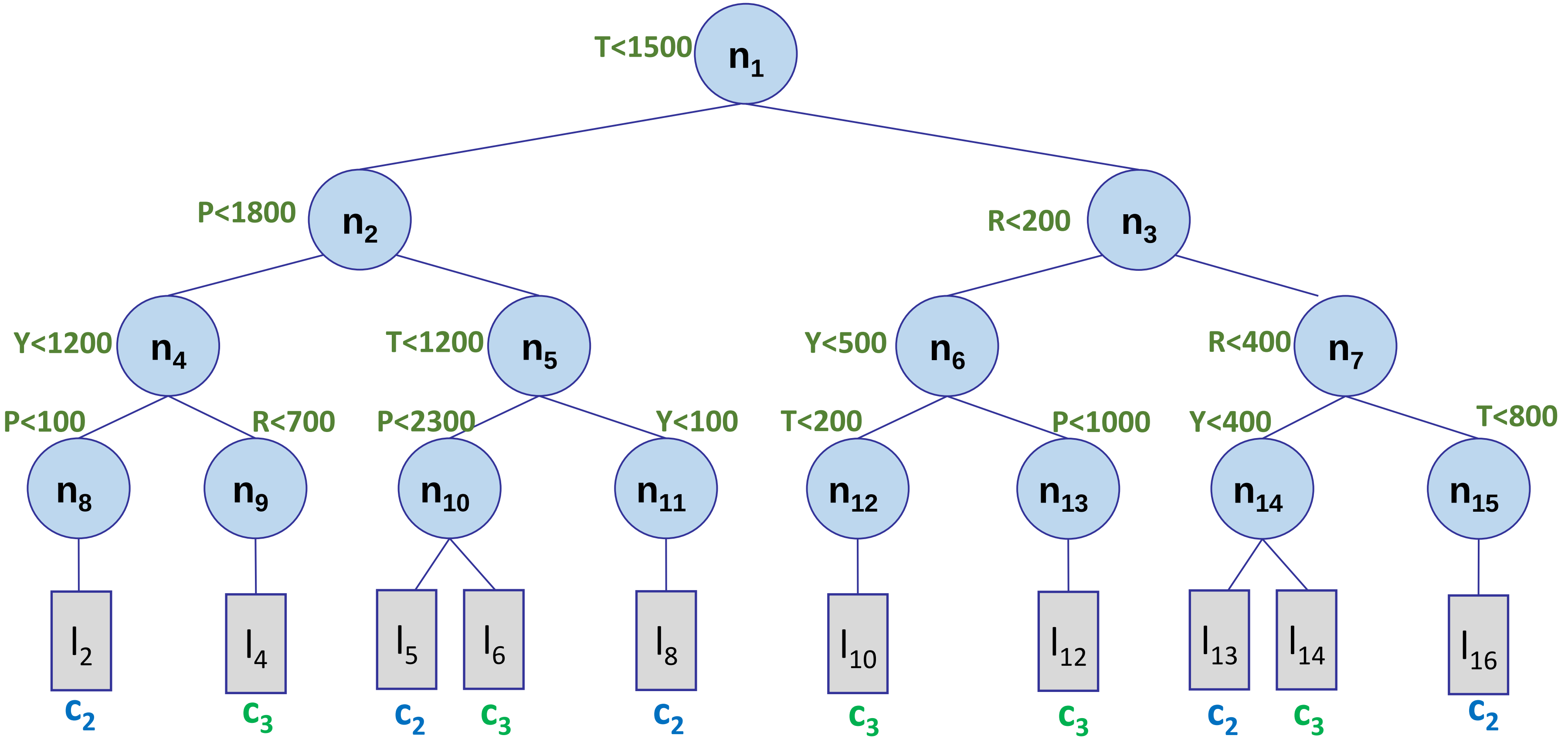


Proposed Algorithm



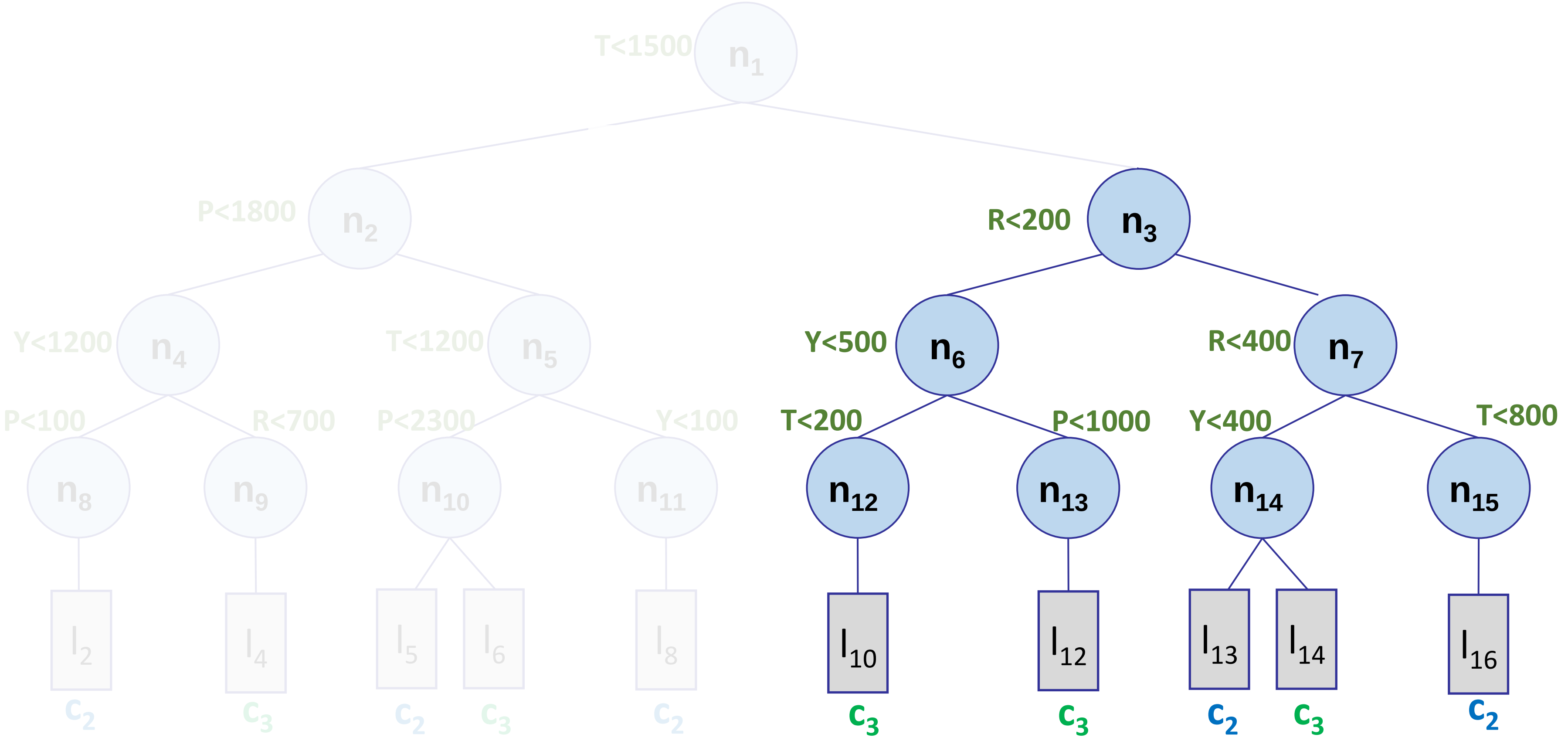
$C_2 = n_1 n_2 n_4 n_8' + n_1 n_2' n_5 n_{10} + n_1 n_2' n_5' n_{11}' + n_1' n_3' n_7 n_{14} + n_1' n_3' n_7' n_{15}'$
$C_3 = n_1 n_2 n_4' n_9' + n_1 n_2' n_5 n_{10}' + n_1' n_3 n_6 n_{12}' + n_1' n_3 n_6' n_{13}' + n_1' n_3' n_7 n_{14}'$
$C_1 = (C_2 + C_3)'$

Proposed Algorithm



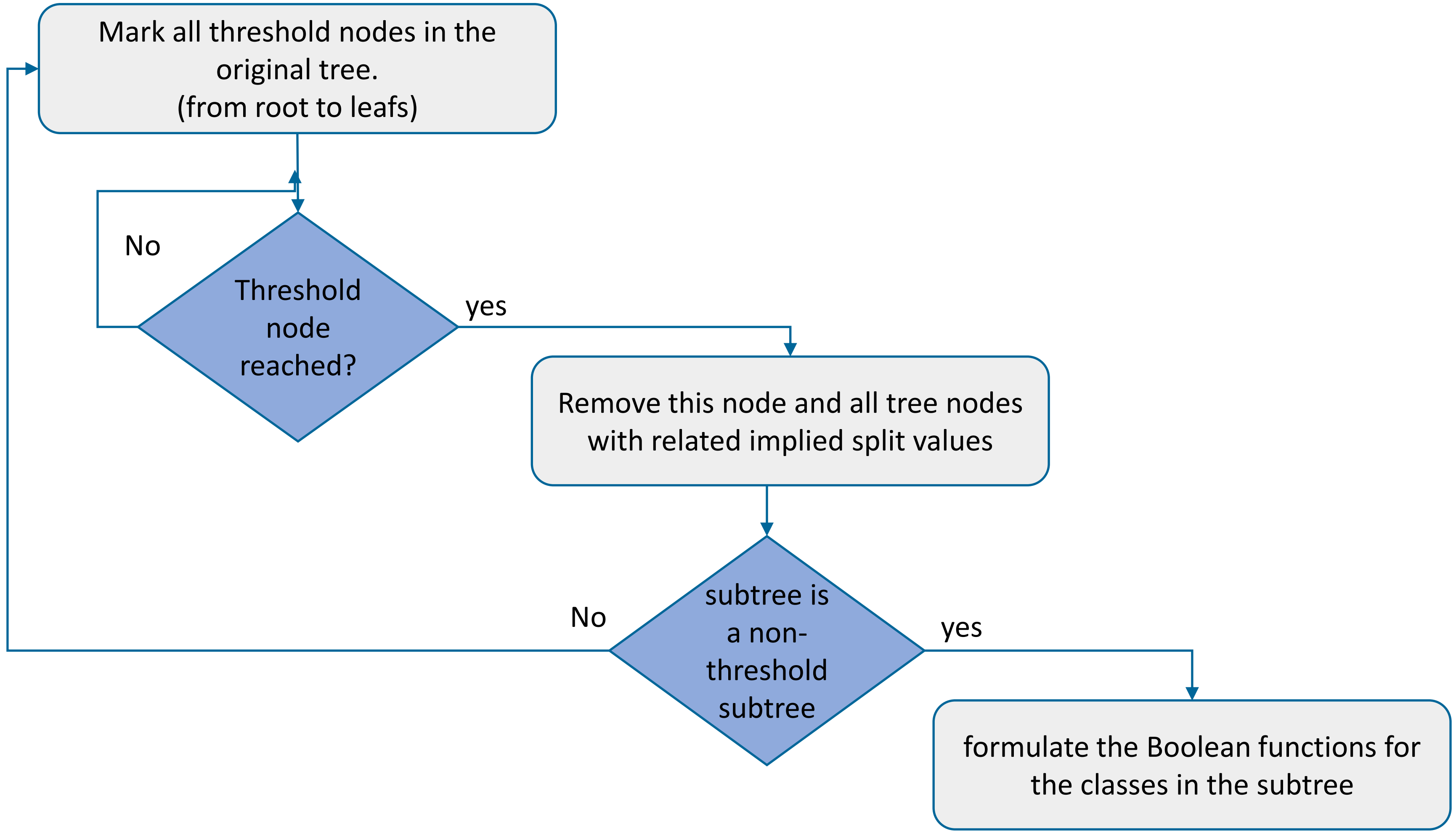
$C_2 = n_1 n_2 n_4 n_8' + n_1 n_2' n_5 n_{10} + n_1 n_2' n_5' n_{11}' + n_1' n_3' n_7 n_{14} + n_1' n_3' n_7' n_{15}'$
$C_3 = n_1 n_2 n_4' n_9' + n_1 n_2' n_5 n_{10}' + n_1' n_3 n_6 n_{12}' + n_1' n_3 n_6' n_{13}' + n_1' n_3' n_7 n_{14}'$
$C_1 = (C_2 + C_3)'$

- **Example:** if T≥1500



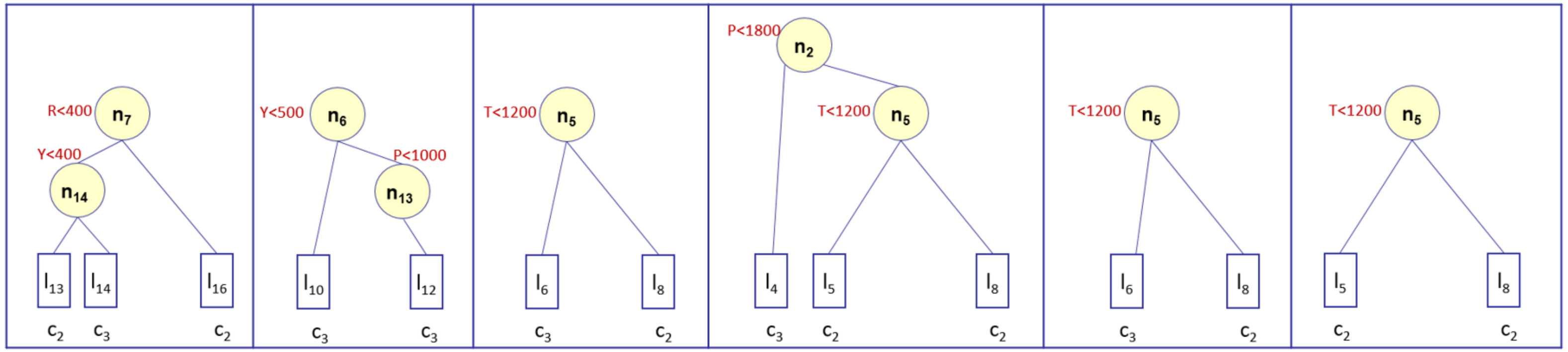
Proposed Algorithm

- **<u>Recursive Algorithm:</u>** To generate non-threshold subtrees

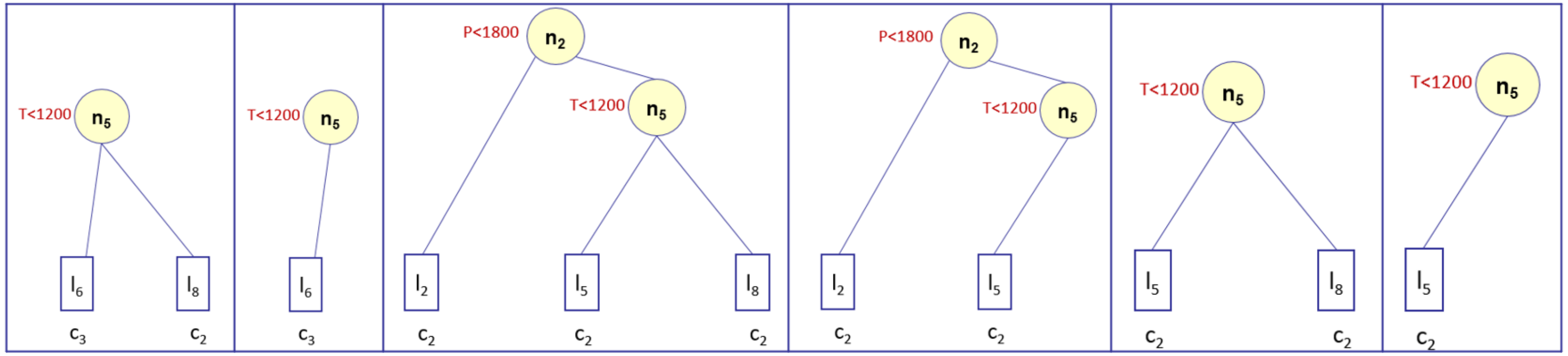


Proposed Algorithm

- **Non- Threshold trees examples:**



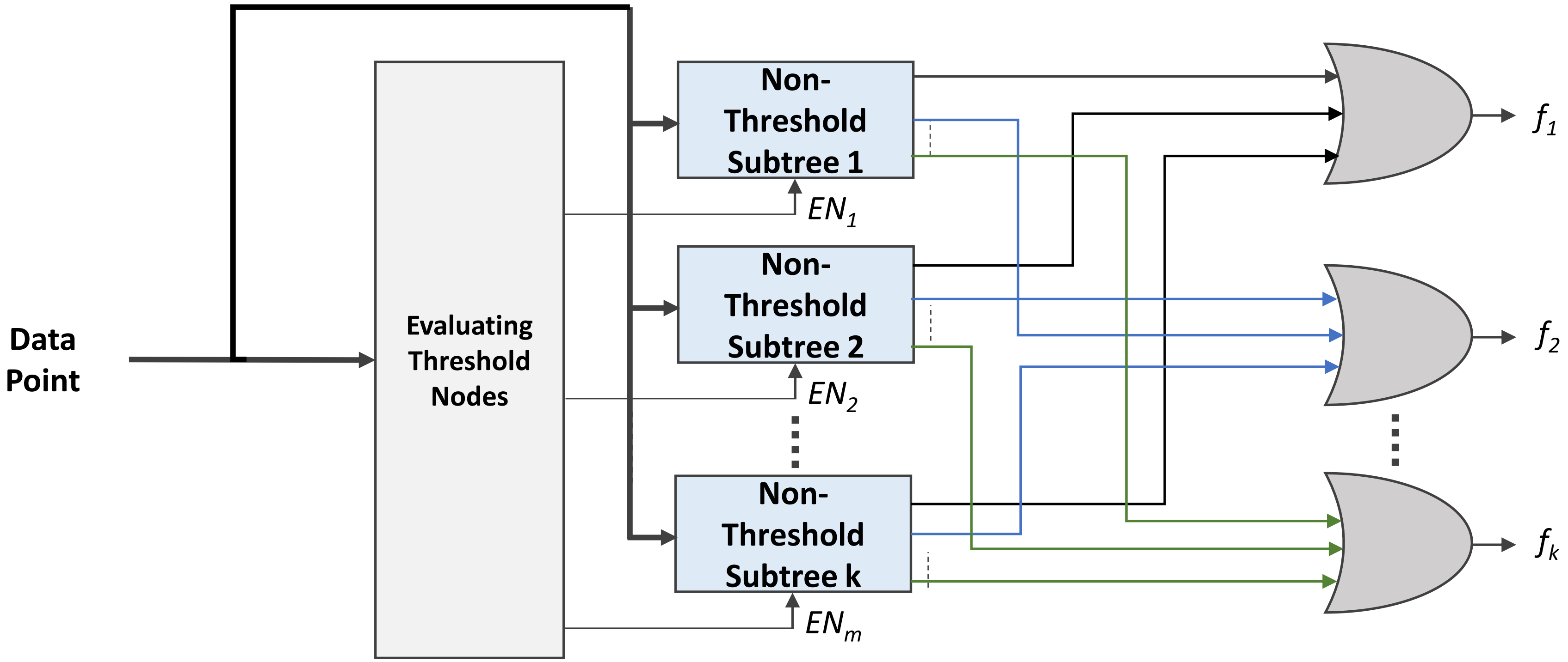
	n_3	n_1	n_1, n_{10}	n_1, n_9	n_1, n_9, n_{10}
$f_2 = n_7n_{14} + n_7'$	$f_2 = 0$	$f_2 = n_5'$	$f_2 = n_2'n_5 + n_2'n_5'=n_2'$	$f_2 = n_5'$	$f_2 = 1$
$f_3 = n_7n_{14}'$	$f_3 = n_6 + n_6'n_{13}'$	$f_3 = n_5$	$f_3 = n_2$	$f_3 = n_5$	$f_3 = 0$



n_1, n_4	n_1, n_4, n_{11}	n_1, n_4, n_{10}	n_1, n_4, n_{10}, n_{11}	n_1, n_4, n_8	n_1, n_4, n_8, n_{11}
$f_2 = n_5'$	$f_2 = 0$	$f_2 = n_2 + n_2'n_5+n_2'n_5'=1$	$f_2 = n_2 + n_2'n_5$	$f_2 = 1$	$f_2 = n_5$
$f_3 = n_5$	$f_3 = n_5$	$f_3 = 0$	$f_3 = 0$	$f_3 = 0$	$f_3 = 0$

The Hardware Architecture

- The hardware architecture for a real-time classifier.



Results

- Resource utilization comparing to previous algorithm from literature

	Previous Algorithm	Proposed Method	Difference Percentage(%)
Number of tested Nodes	611	320	**47**
Number of branches	216	132	**38.9**
Number of Lookup tables	97	82	**15.5**

Conclusion

Proposed solution has the following properties:

- Optimizes the real-time behavior of a random forest classifier
- Works only for trees with numerical data but not for nominal or binary data
- Compares maximum and minimum thresholds

Future work:

- We will test different data sets type
- Test for different number of class
- Also we will select different threshold values (ex: mean value)