جـــامـعـــة الـــشـرق الأوسـط
MIDDLE EAST UNIVERSITY
Amman - Jordan

A HADOOP MAPREDUCE IMPLEMENTATION OF C5.0 DECISION TREE ALGORITHM

انشاء و تطبيق خوزارمية شجرة القرارات " C5.0" باستخدام " Hadoop MapReduce "

Prepared By
Mamoun Abu-Lubbad

Supervisor
Dr. Bassam Al-Shargabi

Thesis Submitted In Partial Fulfillment of the Requirements of the Master Degree in Computer Science

Computer Science Department

Faculty of Information Technology

Middle East University

June, 2020

Authorization

I, Mamoun Fawaz Abulubbad authorize Middle East University to provide an electronic copy of my thesis to the libraries, organizations, or bodies and institutions concerned in research and scientific studies upon request.

Name: Mamoun Fawaz AbuLubbad

Date: 06 / 07 / 2020.

Signature : 

Thesis Committee Decision

This is to certify that the thesis entitled "A Hadoop MapReduce Implementation of C5.0 Decision Tree algorithm " was successfully defended and approved on 16-6-2020.

Examination Committee Members	Signature

Dr. Bassam Al- Shargabi (*Supervisor / Chairman*).....

Associate Professor, department of computer science

Middle East University

Dr. Sharefa Murad (*Internal Member*).....

Assistant Professor, department of computer science

Middle East University

(External Committee Member)

Prof. Wael Mardini (*External Member*).....

Professor, department of computer science

Jordan University of Science and Technology

Acknowledgement

I would like to thank Dr. Bassam Al-Shargabi, my supervisor, for his consistent support and guidance during the running of this thesis. And express my deep sense of gratitude to the group general manager of HijaziGhosheh company Dr. Hani Hijazi for encouraging and assist me to achieve my high education. And I would like to express my special gratitude to all the lecturers at the Faculty of Information Technology, university of the Middle East, and to all those who supported me in carrying out this work.

The researcher
Mamoun Abu-Lubbad

Dedication

To:

My parents and friends who helped me a lot in accomplishing this thesis within the required time, and also to my wife bara, with great love, she gave me the power and encouragement, and this work would not have been possible without her input.

Table of Contents

List of Figures

List of Tables

Table of Abbreviations

Abbreviations	**Meaning**
HDFS	Hadoop Distributed File System
MR	MapReduce
DT	Decision Tree

Table of Equation

A Hadoop MapReduce Implementation of C5.0 Decision Tree Algorithm

Prepared By: Mamoun Abu-Lubbad

Supervisor: Dr. Bassam Al-Shargabi

Abstract

Recently, many of the research institutes have been involving in boosting the accuracy and efficiency of different classification techniques. To date, a lot of enhancement efforts are spent in order to boost such techniques. In addition, the growing volume of data produced daily raises more issues that need to be resolved, which presents risks to the standard Decision Tree (DT) algorithms. Likewise, the process of generation DT is complicated and is time-consuming to complete the computation on one machine when the size of the datasets becomes big, and as the data can not keep the whole training dataset or most of it in memory on one machine. Some computations are transferred to the additional storage, which will lead to increasing the cost of input or output. In this thesis, the researcher will implement a standard DT algorithm C5.0 using Hadoop MapReduce and will compare the error-rate, leaf nodes, and rules with C4.5. The procedure used in this thesis is to transform the standard algorithm into steps of Map and reduce. In addition to implementing data structures to reduce the cost of communication and to proceed with comprehensive experiments on a vast dataset. The results of the study revealed that the MapReduce C5.0 tree is a fixed memory issue to enhance the execution time of the algorithm, and it is suitable for enormous data. The algorithm is characterized by being expandable in the cluster environment and is also characterized by time efficiency.

Keyword: Hadoop, MapReduce, Data Mining, Decision Tree, C5.0 .

انشاء و تطبيق خوزارمية شجرة القرارات " C5.0" باستخدام "Hadoop MapReduce "

اعداد: مامون ابو لباد

اشراف: الدكتور بسام الشربتجي

الملخص

في الآونة الأخيرة، يهتم المجتمع العلمي بكيفية زيادة دقة وأداء طرق التصنيف المختلفة، حيث تم تحقيق إنجازات كبيرة في هذا المجال حتى الآن. إلى جانب هذه التحديات، فإن الكمية المتزايدة من البيانات التي يتم إنشاؤها كل يوم تبرز المزيد من التحديات التي يجب التغلب عليها، والتي تظهر تحديات لخوارزميات شجرة القرار التقليدية. منها، نظرًا لأن حجم مجموعة البيانات يصبح كبيرًا للغاية، فإن عملية بناء شجرة قرارات يمكن أن يتم احتسابها في غضون فترة زمنية غير مقبولة على جهاز كمبيوتر واحد وهي عملية صعبة للغاية وتستغرق وقتًا طويلاً. لأنه لا يمكن الاحتفاظ بمجموعة بيانات بأكملها أو معظمها في الذاكرة على جهاز كمبيوتر واحد. لذالك يجب نقل بعض العمليات الحسابية إلى اجهزة التخزين الخارجي وبالتالي زيادة تكلفة الإدخال / الإخراج. و لتحقيق هذه الغاية، يقترح الباحث في هذة الرسالة تنفيذ خوارزمية شجرة قرار C5.0 باستخدام Hadoop MapReduce،

في هذة الرسالة، يقوم الباحث بتحويل الخوارزمية التقليدية إلى سلسلة من الخطوات و الاجراءات و كما يقوم ببناء بعض هياكل البيانات لتقليل تكلفة الاتصال. ويجري الباحث أيضًا تجارب عديدة على مجموعة بيانات ضخمة.التي تشير النتائج إلى أن خوارزمية المستخدمة لدى الباحث تتميز بتوفير الوقت وقابلية التوسع في البيئة الموازية.

الكلمات المفتاحية: Hadoop, MapReduce , استخراج البيانات , شجرة القرار , خوارزمية (C5.0)

Chapter One: Introduction

1.1 Overview

This chapter explains the need to extract a decision tree using the Hadoop MapReduce and describes the ability to produce a decision tree from the data gathered from a Hadoop Distributed File System (HDFS). The researcher sheds light on the background and importance of this study. This chapter includes definitions, introduction, problem statement, purpose, scope, limitation, and motivation of the thesis.

1.2 Definitions

Is this section we will define the key terms that are used on this thesis,

- Decision Tree (DT): It is one of the most famous data mining methods used. DT, is a structured tree splits that data as rules. The definition of DT in the computer system are some of the mathematical equations and computational processes executed on the data to find the hidden information. DT has three nodes, the root node, which is the start point of the DT, decision node, and the tree ends with the leaf node.
- Hadoop: It is a collection of software or programming modules which help us using a grid of commodity hardware to solve big data issue. Hadoop is an open-source framework; the main module for Hadoop is HDFS and MapReduce.
- MapReduce (MR): It is one of the core components of the Hadoop system, used for distributing the vast data to a small unit and store it on the HDFS, MR divides the data as Key and value, and it has two primary operations Map and Reduce.

- HDFS: It is the storage system for Hadoop; the design of the HDFS makes this system to be a highly efficient and scalable, and fully available system. The data store and distributed on many data servers called data nodes as small pieces. All these data nodes are controlled and managed by a master server called Name node.

1.3 Introduction

Day after day, the size of data, storage capacity, processing power, and availability of data is constantly increasing. In addition, the traditional storage management systems tools and data storage are unable to deal with the amount of data generated (Qasem, Sarhan, Qaddoura, & Mahafzah, 2018). To deal with this issue, the Hadoop framework was designed to solve such data problems. Hadoop is one of the most famous technologies or software programs intended to process and solve problems related to the large size of data in providing an effective data solution. The Hadoop system or framework contains two major components, namely HDFS and MapReduce. MapReduce is a programming model that was developed by Google, but now it is incorporated by the Apache (Yang & Hiong Ngu, 2017), it provides a framework for scalable distributive computing. MapReduce is hosted in two operations or stages which include Map stage, and Reduce stage. The Map stage refers to applying a process of input data, which changes over into key-value pairs. The second operation is the reduce stage, which takes the output of the Map stage as input. The reducer is to process all the data that comes from the mapper, after processing, a new set of outputs produced and stored in (HDFS). A strong feature or value of this programming model is that it avoids the complication of managing a cluster of distributive processing nodes (Polo, 2013), Hadoop MapReduce is considered the best solution to be used with data mining techniques when the data becomes bigdata, and when it is hard to process it of

a single computer. Data mining techniques are applied to raw data for extracting and finding useful information. The process of finding a model is to describe the data classes by using a classification algorithm, DT's are the most famous techniques for classifying and assisting the decision-making process in different data mining applications. DT's find the difficult or invisible information and the correlation between the enormous sets of data that are useful in decision making (Revathy & Lawrance, 2017). DT's are structured trees consist of three main parts: root node, decision nodes, and it ends with leaf nodes. The way from the root node to the leaf node forms is a decision rule to decide which class the new abilities and learning to (Dai & Ji, 2014).

To generate the DT, there are a lot of algorithms that should be used for that purpose. One of the DT algorithms is the C5.0 algorithm, which is an updated release of the C4.5 algorithm, C4.5 is an expansion of ID3. In addition, C5.0 is the algorithm for classification, which is improved to be used for big data. C5.0 are lease with improvement in memory, speed, and efficiency. In C4.5, all the errors are considered equally. The errors were not separated based on their importance or significance. The most exciting improvement in C5 over C4.5 depends on the size of their impact on the system; it treats all errors with individual classification. It creates classifiers that help to reduce the cost of misclassification rather than the high error toll (Revathy & Lawrance, 2017).

In this thesis, the researcher is implement the DT C5.0 algorithm using a Hadoop MapReduce to reduce communication cost of input and output when the data become huge and the memory not fit to hold all tanning dataset or part of it, which is affected to execution time and accuracy, afterward deploy it on a Hadoop cluster, to evaluate the performance and measure scalability Hadoop nodes with the execution time.

1.4 Problem Statement

As the amount of data produced daily is expanding very fast, several data mining methods is needed to learn from big data. Many data mining methods or algorithms are proposed up to now with the small/ and medium data sets. However, not many of them will be applied to the analysis of large data sets.

The main problems in learning from big data can be summarized as the following,

• **Memory restrictions:** It is hard to keep the whole training dataset or most of it in memory on a single computer.

• **Time complexity:** Completion of the computation process on a single computer within a tolerable time is difficult.

• **Data complexity**: The high dimensional and multi-modal features of the data that make a far-reaching influence on the performance and efficiency of research results.

However, due to the problem mentioned above, the researcher will be implementing a DT C5.0 algorithm using a Hadoop MapReduce. MapReduce is very suitable for distributed computing, which abstracts away from large numbers of challenges in parallelizing data management operations across a cluster of item machines.

1.5 Question of the study

- How can we implement the decision tree C5.0 algorithm using Hadoop MapReduce, regard to time producing tree and accuracy?

1.6 Purpose of the Study

The purpose of the thesis is to the speed-up growth of DT and reduce the error-rate classification prediction. The main objectives of this proposed work are:

- Implementing Decision Tree C5.0 algorithms using Hadoop MapReduce.
- Measuring and evaluating the execution performance after implementation of the DT C5.0 algorithm with MapReduce.
- Measuring and evaluating the error-rate during the classification process.

1.7 Scope of the study

The scope of this thesis is to implement a decision tree using a Hadoop MapReduce on a single node and cluster environment. Compare between C4.5 with C5.0, C5.0 with MapReduce C5.0 Tree based on the error-rate, execution time, and the number of leaf nodes on a single node, evaluate the execution time and scalability on the cluster. The classification algorithms used in this thesis is original C5.0.

1.8 Limitation of the Study

The work of this thesis is limited to implementing a decision tree original C5.0 algorithms using a Hadoop MapReduce v 3.0.2 under Ubuntu 18.4 LTS and evaluating the time execution on the cluster. Chapter 4, section 4.3. Of this thesis, provided considerable information about the

hardware used to achieve the desired goals. The researcher will only compare C4.5 with C5.0 and MapReduce C5.0 Tree on a single node and will evaluate the performance of the MapReduce C5.0 tree on a cluster environment.

1.9 Contribution and Important of the Study

The importance of this thesis stems from the implementation of the decision tree C5.0 algorithms using a Hadoop MapReduce. The researcher contribution in this thesis can be summarized as the following:

- The thesis implements data structures customized for a single node and cluster computing environment.
- The thesis proposes a MapReduce implementation of the original C5.0 algorithm.
- The thesis proves the efficiency of the approach used on C5.0 with extensive experiments on a vast dataset.

1.10 Motivation

The Motivation of this thesis comes from the famous quote, "we are drowning in data but starved for knowledge" for John Naisbitt, in his 1982 book Megatrends, while is written over 38 years ago, that sentence is true today, the amount of data produced daily is expanding very fast. And the data mining algorithms are needed to learn from big data. How we can find hidden information for these data to assists decision-making to solve problems is the motivations for the researcher in this thesis.

Chapter Two: Theoretical Background and Related Works

2.1 Introduction

This chapter will show a brief definition and theoretical background for the Hadoop framework and its components in addition to DT with an overview of the widely used and relevant big data and then literature review of related available works.

2.2 Hadoop Overview

Hadoop is a software framework (open source) designed for process large volumes of heterogeneous data sets through commodity hardware and computer clusters in a distributed manner using a simplified programming model. It provides a reliable system of shared storage and analytics. Hadoop has been released, based on Google's paper on the MapReduce, and it applies functional programming concepts. Hadoop was written among the highest-level Apache projects in the Java programming language (Yang & Hiong Ngu, 2017).

The design of the Hadoop is increasing, its capability of fault tolerance, distributed processing, and scalability. Hadoop is the solution to Big Data problems. It is the technology that provides bigdata analyzes through a distributed computing framework. Furthermore, store massive datasets on a cluster of commodity hardware in a distributed manner (Purdila & Pentiuc, 2014).

The next subsections cover the Hadoop architecture and its component.

2.2.1. Hadoop Architecture

Hadoop has a master-slave topology. On this topology, we can have many slave nodes and one master node. The primary function of master nodes is to define the task and distribute it on slave nodes. The master nodes store the metadata of the data stored on the slave nodes, while slave nodes store the actual data (Bikku, Sambasiva Rao, & Akepogu, 2016) Figure 2.1 illustrate the Hadoop master-slave architecture.

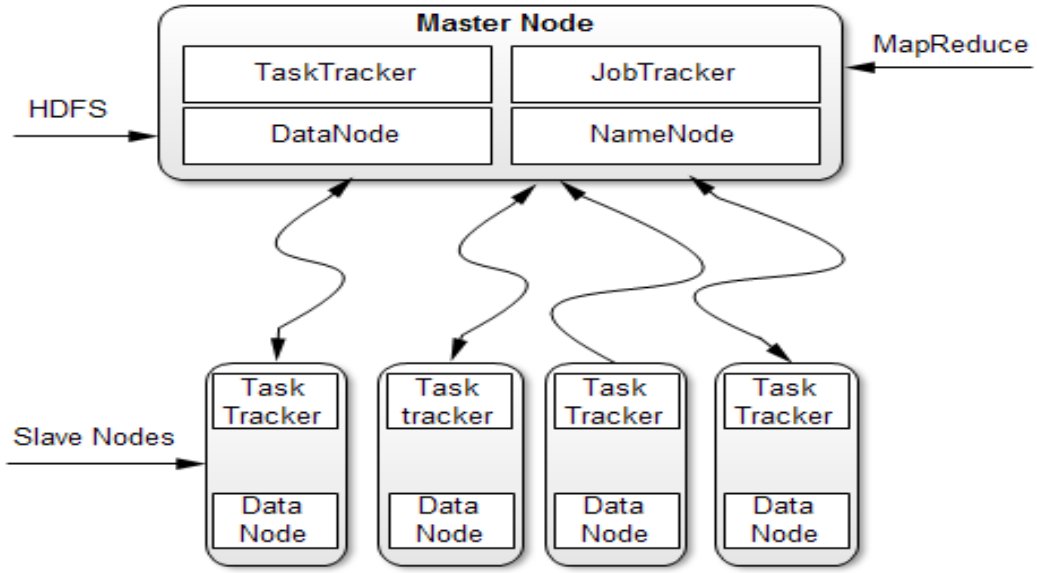

Figure 2.1 the Hadoop Master-Slave Architecture(Kebande & Venter, 2015).

This design or topology is objective to deal with large data sets, portability crosswise over heterogeneous hardware and software platforms, fault tolerance. Hadoop Architecture has two main components. They are:

- HDFS.
- MapReduce.

In the next subsections, the researcher explain the MapReduce and HDFS storage solution for the Hadoop framework.

2.2.1.1. Hadoop HDFS

HDFS, it is a data storage solution; it is considered one of the significant feature for Hadoop. HDFS divides data into small pieces; each piece is called blocks stored in distributed algorithms or methods. It has got two running services. One for a master node called name node and other for slave nodes called data node (Hu & Dai, 2014) Figure 2.2 shows the HDFS architecture.

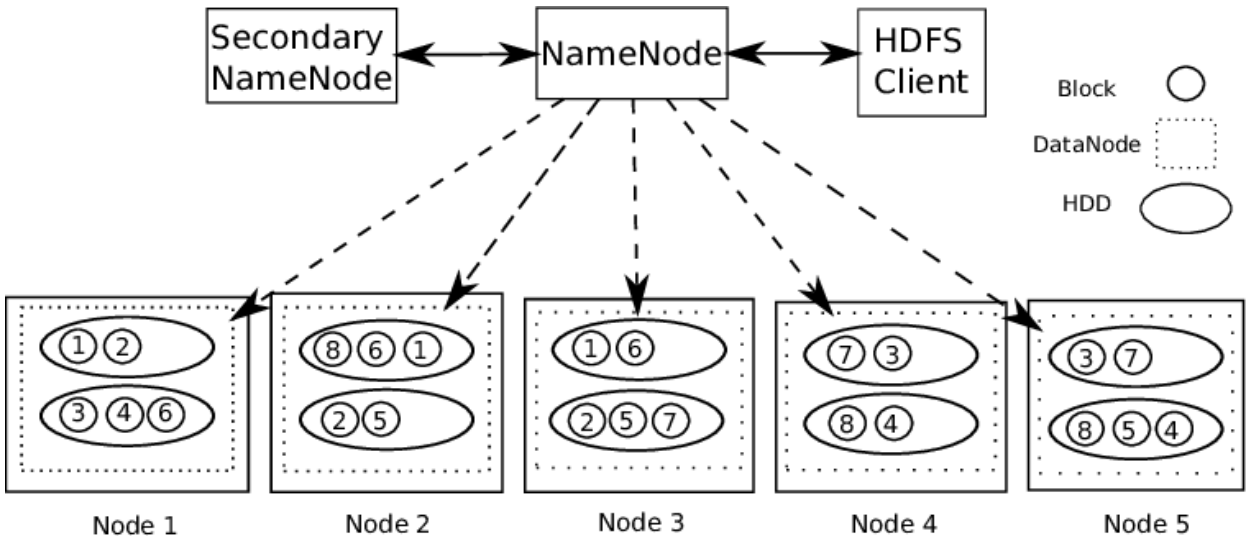

Figure 2.2 HDFS architecture with default data placement policy (Krish, Anwar, & Butt, 2014).

HDFS has the architecture of a Master-Slave. The service name is Name Node running under the master server or node. It is used for managing file access by the client and namespace management. Data Node service is running on slave nodes. It is used for storing the actual data submitted by the client. Inside, a file is split into many data blocks and placed on a group of

slave machines. Any changes that maybe happened in the file will be done by the name node (Hu & Dai, 2014). For example, renaming or indexes files, opening, and closing action will be managed by the name node. This data node creates, deletes, and replicates blocks on-demand from the name node. Java programming language is the local language of HDFS.

2.2.1.2. MapReduce

MapReduce is a programming model that started from google paper to solve the parallel and distributed vast amounts of data problems (terabyte data sets) on commodity hardware clusters at the same time. MapReduce is composed of two operations (or stages). The first one is Map, the Map or mapper job is to process the input data. Usually, data stored in the HDFS (Wu et al., 2009). The procedure to enter the input data to the map function is line by line. The Map generates many small chunks of data and processes them. Reduce is the second operation; it is represented by the shuffle stage and reduce stage. The job of the reducer is to process all the data that comes from mapper. After processing, a new set of outputs produced for us and stored in (HDFS). Figure 2.3 explains the map-reduce programming model.

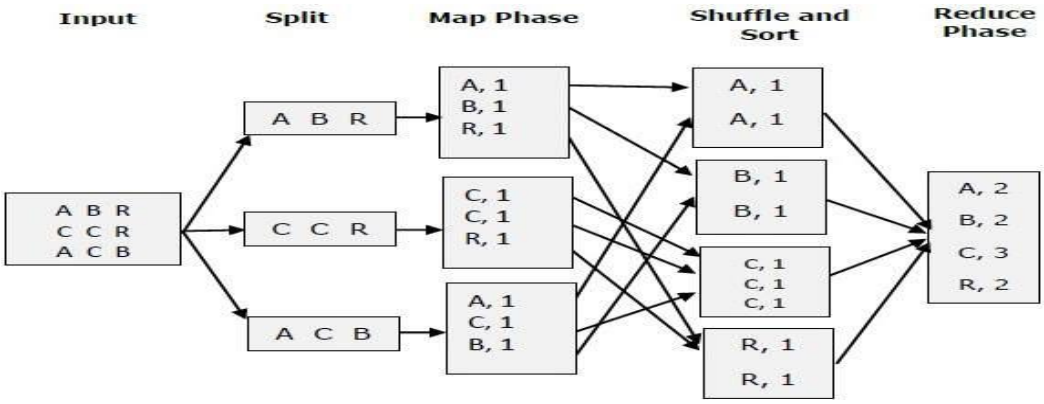

Figure 2.3 MapReduce Programming Model (Tutorialspoint, 2019).

MapReduce and HDFS run in the same node-group. That means the computing nodes and storage nodes are working together. This style of design enables the system to schedule tasks quickly so that the entire cluster is used efficiently (Polo, 2013).

2.3 Decision Tree

A DT is structure tree, with root, decision and leaf node, DT split the data on a set of rules for example, (if the income > 10 K and age > 18 so he can buy a car, if the income > 10 K and age < 18 he can not buy a car), the DT will start with the root node and end with a leaf node, in the decision node we can be split to two or more instances and sometimes decision node (Yang & Hiong Ngu, 2017), to clear the DT Tree concept figure 2.4 provide an example of a DT where the square indicate to the leaf node and the circle indicates to decision node. We have three classes (ages, gender, and so on) and the following rule (Dai & Ji, 2014).

Rule 1 – if the age < 20, can not buy a car.

Rule 2- if the age > or equal 20, and the gender is female Then Yes, can buy a car.

Rule 3- if the age > 20 and the Criteria x and is he has a license, then Yes.

Rule 4- if the age > 20 and the Criteria x and is he did not have a license, then-No.

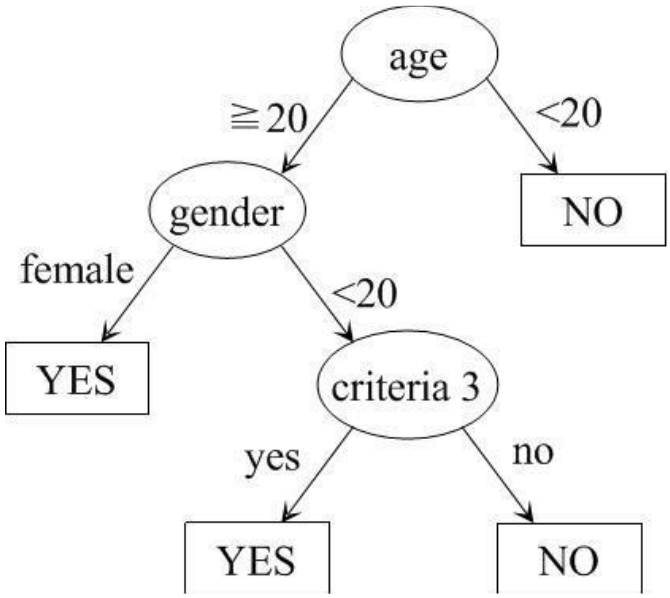

Figure 2.4. explanation of DT (Dai & Ji, 2014).

DT is a mathematical and computational process or method. To build a DT, we have to find the best split attribute. Once it found, the tree will be start generated, to root, decision, and leaf nodes. The DT procedure will be terminated when we find the leaf's node; otherwise, the calculator will repeat. If the DT is generated, the rules can also be generated. There are many algorithms used to generate a DT. In this thesis, the researcher will use the C5.0 algorithm, which is the latest update by Roos Quinlan, from Stanford University. The C5.0 update to deal with massive data and solve memory issues. In the next section, introduce the C5.0 algorithms with updated features.

2.4 C 5.0/ See5 Decision Tree Classification Algorithms

C5 algorithm is an update of C4.5. In C4.5, there no separation for any errors based on its importance or significance; all errors are taken equally. A clear improvement that comes in C5 on C4.5 is that it has handled all errors with individual classification depends on the magnitude of its impact on the system. C5 based on building classifiers that help reduce the misclassification cost function. This characteristic of C5 is defined as variable misclassification costs (Lakshmi, Indumathi, & Ravi, 2016). Due to the size of the account, each case is of varying significance. This problem is treated very well in C5 by adding a characteristic attribute called case weight. By using this function (case weight), the C5 lower the cost of biased predictive miscalculation, and C5 contains far more data types than in C4.5 or any of the previous algorithms. It includes case labels with date, timestamp, in C5 class called "not applicable" as it identifies a new data type and encourages the inclusion as a function of some other feature of a new category. Many of C4.5's various components have been merged into C5, for example, cross-validation and sampling, making this algorithm more straightforward and more effective (Lumpur, 2018). This algorithm released two versions, one for UNIX named as C5 and other See5 for Windows (Lumpur, 2018) in this thesis; we will use C5.0 on Ubuntu OS version 18.4 LTS.

2.5 Review of Related work

In this section, the researcher is reviewing the most related study, or works used the Hadoop MapReduce to generate decision tree in different algorithms:

- A Study was presented by (Shirzad & Saadatfar, 2020), the authors show problems with one unsuccessful job execution of MapReduce; the unsuccessful jobs can lead to significant resource waste. The authors attempt to predict the future of MapReduce work on the open cloud Hadoop cluster using their log files. The authors compared the learning methods. They showed that C5.0 algorithms had the best results.
- A Study was presented by (Revathy, Balamurali, & Lawrance, 2019), the authors analyze the agricultural Corp pest dataset using Hadoop MapReduce based on the C5.0 algorithm. The research methodology was used to classify a Corp pest dataset, and the result for this study is going on a single node.
- A Study was presented by (Rajeswari & Suthendran, 2019), the authors evaluate the selection process based on statistical techniques called Chi-square MapReduce and C4.5. The Chi-square is found irrelevant data, and therefore the resulting accuracy is not as expected as the authors aimed.
- A Study was presented by (Yang & Hiong Ngu, 2017), the authors have evaluated the efficiency of Hadoop implementation of DT C4.5 using AWS service. The authors evaluated the execution time with the number of CPU cores performed by the mapper / and reducer and the execution time with the size of the data input.
- A Study was presented by (Cui, Yang, & Liao, n.d.) 2017, where the authors have proposed a new algorithm for DT learning in the Hadoop MapReduce framework name it as PDTSSE.
- A Study was presented by (Purdila & Pentiuc, 2014)where the authors introduced a new algorithm for DT called MR-Tree, which can be used to learn Dt's using massive datasets and runs on the Hadoop platform. The problem of the proposed algorithm, as the authors

declare, is that it uses a MapReduce iteration to find and pick the best attribute to split each tree node, which means it can take time and memory for trees of big or large data.

- A study was presented by (Dai, Wei; Ji, Wei, 2014) the authors aim to execute a standard DT algorithm, C4.5, depend on the MapReduce programming model. And Propose and used modified data structures for cluster computing environments, and propose MapReduce use of C4.5 algorithms with Hadoop MapReduce.
- A study was presented by (Wu, Gongqing; Li, Haiguang; Hu, Xuegang; Bi, Yuanjun; Zhang, Jing; Wu, Xindong,2009) where the authors propose a new approach Called MReC4.5 used parallel and distributed classification. The authors show that increase the number of nodes would enhance the accuracy of the classification method, and model-level serialization operations render MReC4.5 classifiers.

Table 2.1 summarized the methods of most of the related works compared to the proposed method proposed in this thesis.

Authors, Year	Purpose and, the algorithm used	Comparison of the proposed method
(Revathy et al., 2019)	The purpose of the study is to analyze the agricultural Corp pest dataset using Hadoop MapReduce based on the C5.0 algorithm, and this study is implemented on a single node.	In this thesis, the researcher is implementing the DT C5.0 using Hadoop MapReduce, for the generic dataset and execute it on a single node and cluster of nodes.
(Yang, Tianyi Hiong	The purpose of the study is to	In this thesis, the researcher is using

Ngu, Anne Hee, 2017)	evaluate the efficiency of the Hadoop implementation of DT C4.5 using AWS service.	the C5.0 algorithm. The implementation used a data structure to reduce memory overhead.
(Purdila, V; Pentiuc, S,2014)	This study introduced a new algorithm for DT called MR-Tree, which can be used to learn Dt's using massive datasets and runs on the Hadoop platform. The problem of the proposed algorithm, as the authors declare, is that it uses a MapReduce iteration to find and pick the best attribute to split each tree node, which means it can take time and memory for trees of big or large data.	In this thesis, the researcher is using the C5.0 algorithm, and execute it on single and cluster of nodes.
(Dai, Wei; Ji, Wei,2014)	this study aims to execute a standard DT algorithm, C4.5, depend on the MapReduce programming model. And Propose and used modified data structures for cluster computing	In this thesis, the researcher is using the C5.0 algorithm, and execute it on single and cluster of nodes.

	environments, and propose MapReduce use of C4.5 algorithms with Hadoop MapReduce.	
(Cui, Yan; Yang, Yuanyang; Liao, Shizhong, 2017)	This study is proposed a new algorithm for DT learning in the Hadoop MapReduce framework named PDTSSE.	In this thesis, the researcher is using the C5.0 algorithm, and execute it on single and cluster of nodes.
(Wu, Gongqing; Li, Haiguang; Hu, Xuegang; Bi, Yuanjun; Zhang, Jing; Wu, Xindong,2009)	This study proposed a new approach Called MReC4.5 used parallel and distributed classification. The authors show that increasing the number of nodes would enhance the accuracy of the classification method, and model-level serialization operations render MReC4.5 classifiers	In this thesis, the researcher is using the C5.0 algorithm, and execute it on single and cluster of nodes.

Chapter Three: Methodology and the Proposed Approach

3.1 Introduction

This chapter presents a description of the proposed methodology, proposed approach. Sections 3.2 covers the methodology. Section 3.3 explains the proposed approach.

3.2 Methodology

In big data, the dataset is enormous, and typically, the data needed to build decision trees is complex. The following steps are summarized our methodology to implement a decision Tree C5 using Hadoop MapReduce,

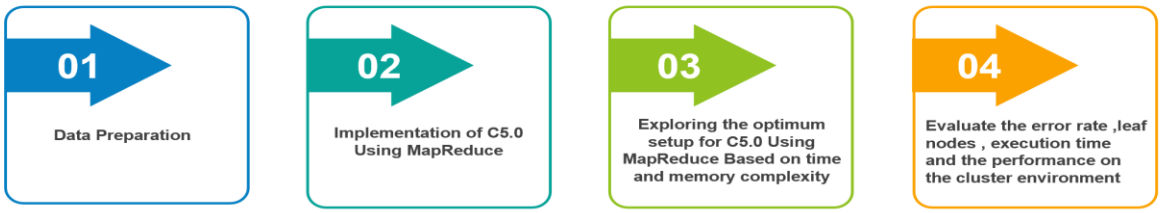

Figure 3.1 Methodology steps

Step 1 – Data Preparation (implement data structure over HDFS, to arrange and manage data that will help to grow the decision tree), Step 2- Implementation of C5.0 Using Hadoop MapReduce, Step 3- Exploring the optimum setup for C5.0 using MapReduce based on Time and Memory Complexity, Step 4- Evaluate the error rate, leaf nodes, execution time and the performance on the cluster environment.

3.3 The proposed approach

In this section, the researcher presents the proposed approach for MapReduce implementation of the C5.0 algorithm. First, the researcher introduces data structures used in this thesis, and afterward, present the MapReduce execution of C5.0 as a process of Map and Reduce function. The data structure and algorithms used in this section are used before on different studies (Dai & Ji, 2014). The approach used in this thesis used different algorithm library files than other studies, used different implementation platforms, and use a new table called intermediate and update on usage of count table mentioned in section 3.3.1.

3.3.1 Data Structures

In big data, the data need to build DT is complicated, and the data are massive. The standard algorithm can not store the whole training data over the memory, and it will take a troubled time to complete the classification process of the data. As the data can not fit in the memory, this will lead to transforming some computational to external storage, which will increase the cost of input and output. So, it's essential to use robust data structures to support this implantation with a distributed parallel algorithm. The main suppose that the memory is not fit to match the entire dataset.

The structure of data we used contains the following tables,

1- Property table, which contains and stores the attributes data details.
2- Count table, on this table, the save and count the number of all instances and count the instances of the attribute with unique classes
3- Intermediate table, this table will have the result of entropy, gain and gain ratio

4- Rule Table, this table saves the connection details of the tree nodes, like the connection between the decision node and its sub-section (nodes).

Note that for efficiency purposes, the dataset is vertically partitioned into a distributed parallel data nodes to optimize the protection of the characteristics of the localization.

In the proposed approach, we split the data set by similarly assigning attributes to multiple nodes. For example, if we have X attributes, Y nodes, so there is [X / Y] on the first Y-1 nodes, and the rest of the attributes are stored on the last node.

3.3.2 MapReduce Implementation

The entire process consists of four steps to implement a DT C5.0 Algorithm using the data structure mention on section 3.3.1,

Step 1- Preparing the data

Step 2- Selection of attributes

Step 3- Update the connection of nodes.

Step 4 -Developing the tree.

3.3.3 Preparing the data

At the beginning of the algorithm (Data Preparation), we need to change the dataset or the relational table based on the data set to the proposed data structure we mentioned in section 3.3.1. Now, in the algorithms 1 (Data Preparation), we have two procedures MAP_APP and REDUCE_ATT, the MAP_ATT, transforms the recording instance into a table of property (A)

as Key, for example (A1 (C=1,2,3,4,5….N)) and the row_id and class label (C) as the values, and on the REDUCE_ATT procedure will count the number of instances with a unique class label (cnt).

Algorithm 1 Data Preparation
1: **procedure** MAP_ATT (row_id, (a_1,a_2,....a_M,c))
2: emit(a_j,(row_id,c))
3: **end procedure**
4: **procedure** REDUCE_ATT ((a_j, (row_id,c)))
5: emit (a_j, (c, cnt))
6: **end procedure**

Where the *emit()* indicate save to the result from the procedure to HDFS table , *row_id* indicate to instance ID, *c* indicate to class label, *a* indicate to attributes, *cnt* = count the number of instances with a unique class label.

3.3.4 Selection of attributes

In this step, we will find the best split or attribute, in the algorithms 2 (Selection of attributes), we have a MAP procedure, and two Reduce procedures, the reduce procedure start taking the number of instances in every attribute or value pair to reach the maximum size of record for the specified attribute like Ac, Then, After the MAP_COMP procedure complete the compute Ac Information and splitting information, the REDUCE_COMP procedure compute the information gain ratio. Finally, the Ac with the best Gain Ratio counting will be considered as the best attribute for splitting.

Algorithm 2 selection of attributes
1: procedure REDUCE_POP ((a_j,(c,cnt)))
2: emit(a_j, all)
3: end procedure
4: procedure MAP_COMP ((a_c,(c,cnt,all))
5: compute Entropy(a_c)
6: compute Info(a_c)= $\frac{cnt}{all}$ Entropy(a_c)
7: compute SplitInfo(a_c)=– $\frac{cnt}{all}$log $\frac{cnt}{all}$
8: emit(a_c,(Info(a_c),SplitInfo(a_c))
9: **end procedure**
10: procedure REDUCE_COMP ((a_c,(Info(a_c),SplitInfo(a_c)))
11: emit (a_c,GainRatio(a_c))
12: end procedure

Where the *info* indicate information gain, and *all* count the total number of instances.

3.3.5 Update

After we find the best split, In this stage we will to update the rule table and save the node's connection, in the algorithm 3 (Update Table) we have two Map procedure, MAP_UPDATE_COUNT, which will update the split attribute to the intermediate table after reading a record from the property table with a key and value. And The MAP_RULE Procedure will set a node_identification or ID on the best split a rule value to verify and validate that records with the same values are split into the partition.

1: Algorithm 3 Update Tables
2: procedure MAP_UPDATE_COUNT((a_{best},(row_id,c)))
3: emit (a_{best},(c,cnt'))
4 : end procedure
5 :procedure MAP_RULE((a_{best},row_id))
6: compute node_id=rule(a_{best})
7: emit(row_id,node_id)
8 : end procedure

Where the *node_id* indicate DT node.

3.3.6 Developing the tree.

After we update the rule table, in this step or stage we need to generate the DT based on the datastore on the rule table, Algorithm 4 (Tree Developing) has a one MAP procedure, at the procedure we building the links between nodes,

1: algorithm 4 Tree Developing
2: procedure MAP ((a_{best}, row_id))
3: compute node_id=rule(a_{best})
4: **if** node_id is same with the old value **then**
5: emit (row_id, node_id)
6: **end if**
7: add a new subnode
8: emit(row_id, node_id,subnode_id)
9: end procedure

On this approach, the data preparation is a one-time job, and the other step is repeated until the tree generation or development.

Chapter Four: Experimental Design and Results

4.1 Overview

In this chapter, the researcher describes the experimental design and results using the proposed approach. Section 4.2 illustrative the datasets used in this thesis. Section 4.3 provides the computing environment and hardware specification used. Then, some comparisons on the datasets between the original C4.5 and C5.0, with and without MapReduce C5.0 Tree are presented in Section 4.4. Section 4.5 shows the confusion matrix. Section 4.6 shows the analysis of the cluster performance of the MapReduce C5.0 Tree.

4.2 Introduction

To be able to test the behaviors of the approach in real-world applications, we collect two tested datasets for the experimental studies in this thesis. As described in Table 4.1, The detailed information of the data sets in experiments,

First, Census Data set from the UCI archive, which has 200K of instances or cases. This data set is goals to predict if a person's income is higher than or less $50K. This data set has 40 attributes divided to 33 discrete or nominal and seven numerical (Kesorn & Chanmee, 2020). . This data set is used before on the for purpose of evaluating classification methods, and this is our goal from using such dataset it in this thesis to compare and verify the result of the proposed approach.

The second one is the Forest Cover dataset, also from the UCI archive, which has 600K records. This data set is published by US forest service. This data set has 21 attributes, 12 are numerical,

and two discrete, and seven classes. Also, this data set is used before with classification studies (Nathan & Scobell, 2012).

Table 4.1 the detailed information of the data sets in experiments

Attributes						
No	Dataset	Numerical	Nominal	Class	Instance	Size
1	Census	7	33	0	199,523	99 MB
2	Forest	12	2	7	581,012	71.6 MB

4.3 Computing environment

In our experiments on this thesis, the proposed MapReduce C5.0 Tree algorithm is implemented in the R language, rHadoop are used packages to integrate the R with the Hadoop and pass the data via MapReduce. C5.0 library is used for DT generation, MapReduce C5 Tree classification works well on large data sets. During the experiments and based on the proposed approach mention in chapter 3 on this thesis. The researcher evaluates the performance of the MapReduce C5.0 Tree classification system in a small cluster operating environment containing two host computers hosted 6 Virtual Machine. The detailed hardware specification of the cluster is given in Table 4.2.

Table 4.2 the hardware specification of the cluster hardware used

No.	Nodes	CPU, GHz	RAM, GB	OS	Bit
1	Master	Intel Xeon 2.00 GHz, 1 Core	16	Ubuntu 18.4	64

2	Slave01	Intel Xeon 2.00 GHz, 1 Core	8	Ubuntu 18.4	64
3	Slave02	Intel Xeon 2.00 GHz, 1 Core	8	Ubuntu 18.4	64
4	Slave03	Intel Xeon 2.00 GHz, 1 Core	4	Ubuntu 18.4	64
5	Slave04	Intel Xeon 2.00 GHz, 1 Core	2	Ubuntu 18.4	64
6	Slave05	Intel Xeon 2.00 GHz, 1 Core	16	Ubuntu 18.4	64

Our rules for virtual cluster specification are as follows to simulate the typical working environment in Hadoop MapReduce,

1) Virtual machine executes Map and Reduce functions, has a core per node.

2) Under vSphere System, each selected core of the same medium computing power is selected.

3) VMware vSwitch is chosen to be the network connection between nodes.

4.4 Comparisons between the C4.5 with C5.0, and C5.0 with and without MapReduce on Single node

In this section, we make some comparisons between C4.5, C5.0, and C5.0 with and without the MapReduce. On the unseen testing data set. The proposed MapReduce C5.0 Tree algorithm is performed on a single node with the number of processors implemented. We mainly focus on the comparisons about the error-rate, rules, and the number of leaf nodes as respectively shown in Tables 4.4 comparison between original c4.5 and C5.0, Tables 4.5 shown a comparison between C5.0 with and without MapReduce C5.0.

Table 4.3 Comparisons between C4.5 and C5.0

	C4.5				C5.0			
Dataset	Error Rate	Rules	Leaves	Time	Error Rate	Rules	Leaves	Time
Census Income	5.0%	190	264	2.782 Sec	4.9%	84	122	10 Sec
Forest cover	6.8%	5,316	10,167	30,115 Sec	5.9%	4269	9,185	270 Sec

Table 4.4 Comparisons between C5.0 and MapReduce C5.0

	C5.0				C5.0 MapReduce			
Dataset	Error Rate	Rules	Leaves	Time	Error Rate	Rules	Leaves	Time
Census Income	4.90%	84	122	10 Sec	4.0%	84	122	5 Sec
Forest cover	5.90%	4269	9,185	270 Sec	5.90%	4269	9,185	120 Sec

As seen in Table 4.3 and Table 4.4 we can conclude that the C4.5 and C5.0 algorithms can build classifiers which are either expressed DT's or rule sets. In C4.5, a slow execution in the ruleset and the memory usage are high. The improvement is clear based on results on table 4.3 (Comparisons between C4.5 and C5 algorithms), and this proves that C5.0 is memory enhancement a specially for a massive amount of data sets.

- For prediction accuracy: The MapReduce C5.0 Tree rules have lower error rates on forest datasets.
- For execution time: MapReduce C5.0 Tree is used less than C4.5, and its faster than the C4.5 algorithm. To generate a rule in C4.5, the model take more than 8 hours despite the fact that MapReduce C5.0 takes 3 minutes to do the DT Generation. And terminate the process successfully.

- For Memory: MapReduceC5.0 Tree is the needless size of memory than C4.5 when executing a job of rules for the forest dataset.

The main problem in C4.5 is fixed on the MapReduce C5.0 Tree which are the tree sizes and execution times, and memory, MapReduce C5.0 trees are smaller and faster than C4.5, which achieves our thesis goals and proves that the proposed approach even on a single node are outperformed.

4.5 Performance evaluation on cluster

In this section, we will measure and evaluate the performance of MapReduce C5 Tree in a cluster environment. The measurement of scalability evaluation is a performance with six nodes of the cluster, and performance with the size of training data sets used on both datasets we have in this thesis. As specified in section 4.3 computing environment, we have six nodes on two physical servers, and the data sets vary from 200 K to 600 K as to the number of instances.

We will verify the efficiency and scalability assessment on six nodes of the cluster and given specific training dataset. Figure 4.1 explains the execution time of the proposed MapReduce C5.0 algorithm with various numbers of cluster nodes for the census, forest cover dataset used in this thesis.

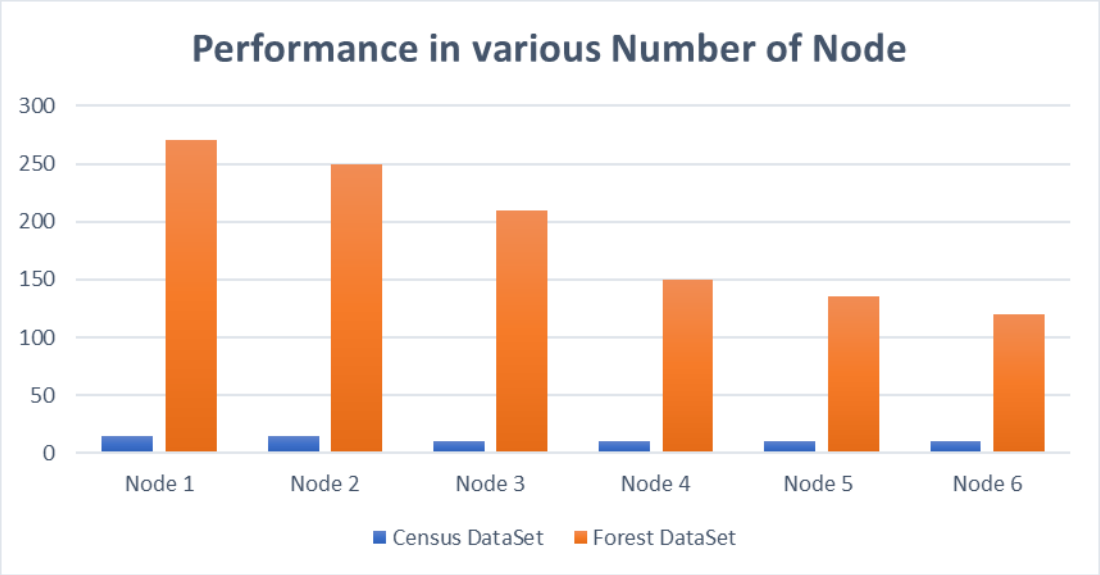

Figure 4.1. Performance on Various Numbers of Cluster Nodes for Census, Forest Dataset per Second

We can note that as the number of cluster nodes increases, the total execution time decreases. This indicates that increasing the algorithm's effectiveness depends on the increasing number of nodes used. Figure 4.2 provides the speed-up output of the training set instances as the number of cluster nodes increases, where speed-up is measured how much faster in an M computers system and can be expressed by equation (1).

Speedup (M) = executing time on one computer / executing time on M computers (1)

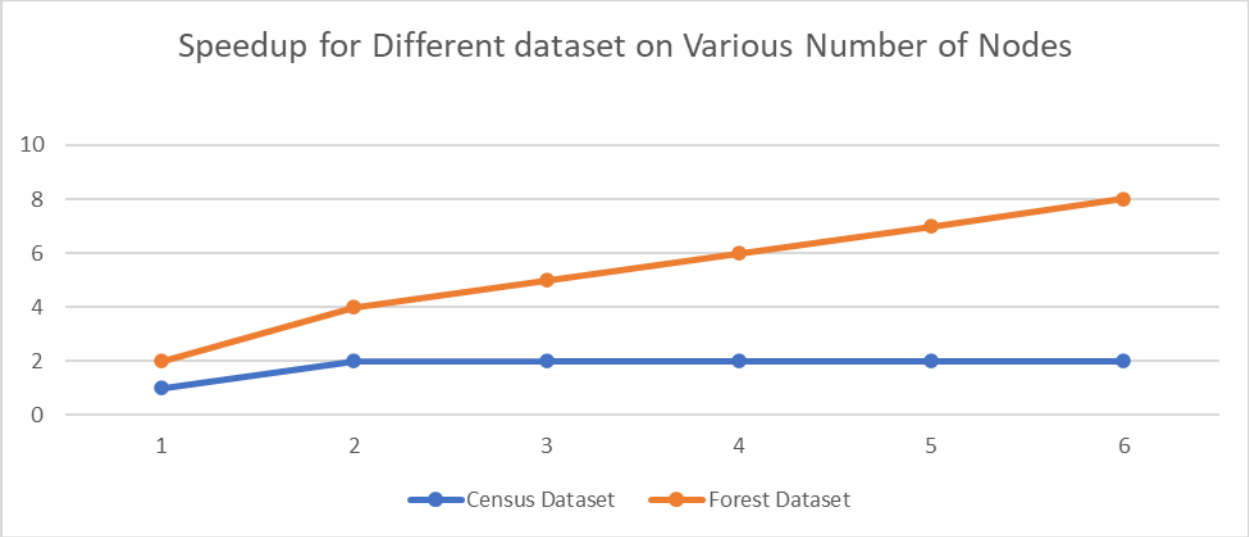

Figure 4.2. Speed-up for Different Training size on Various Number of Cluster Nodes

As depicted from Figure 4.1 and Figure 4.2 we conclude the following:

(1) We note that more Cluster nodes used, that will reflect the less of execution time.

(2) If there is a sufficient number of cluster nodes, despite an increase in the size of the datasets, this can bring us closer to optimum node efficiency.

4.6 MapReduce C5.0 Tree evaluation result

A confusion matrix is an N*N matrix, where N is the number of class labels. That provides values of portions of true-positive, negative, and false-negative, positive cases. The researcher used the confusion matrix to find the accuracy, precision value, and recall value, for the MapReduce C5.0 tree in census dataset used in this thesis.

The accuracy rate for the classification is calculated as the following, the No of true instances/total of all cases * 100 (Revathy et al., 2019). It is calculated using equation (2).

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \qquad (2)$$

Precision, it is defined as the percentage of selected entities that are correctly classified in class out of all the available entities in the dataset (Yuvaraj & SriPreethaa, 2019).It is calculated using equation (3).

$$\Pr ecision = \frac{TP}{TP + FP} \quad (3)$$

Recall, it is defined as the percentage of correct entities that are selected in class from all the available entities in the dataset, belonging to class(Yuvaraj & SriPreethaa, 2019). It is calculated using equation (4).

$$\mathrm{Re}call = \frac{TP}{TP + FN} \quad (4)$$

The census income data set are a 2×2 matrix; table 4.5 shows the confusion matric of census income dataset.

Table 4.5 confusion matrix for census income dataset

	Prediction		
Actual		TRUE	FALSE
	TRUE	185634	1506
	FALSE	6383	5999

It can be depicted from table 4.6, that the accuracy of implementing proposed approach for C5.0 using MapReduce is almost 93.79% with precision of 96.86 and recall of 96.67% for census dataset as illustrated in figure 4.3.

Table 4.6 Evaluation parameters of MapReduce C5.0 Tree for census income dataset.

Accuracy	Recall	Precision
93.79417 %	96.67581516 %	96.86953708 %

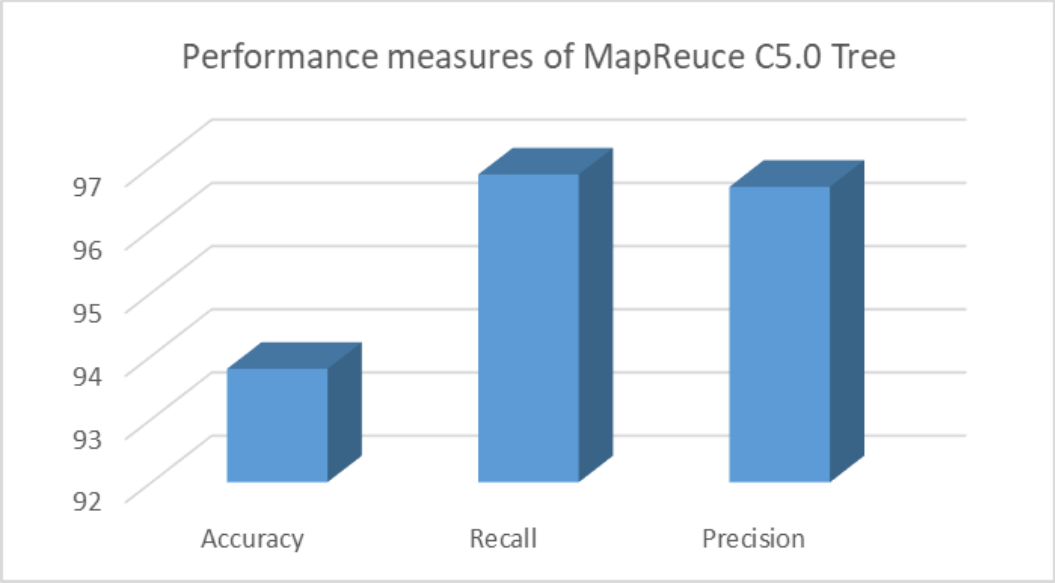

Figure 4.3. Performance measures of MapReduce C5.0 Tree for census income dataset.

Finaly, the challenges in C4.5 when the data become big, is fixed on the MapReduce C5.0 Tree which are the tree sizes, execution times, and memory, MapReduce C5.0 trees are smaller and faster than C4.5, which achieves our thesis goals and proves that the proposed approach.

- Accuracy: The MapReduce C5.0 Tree have lower error rates.
- Scalability, the number of nodes, reflect the less of execution time. Which indicate the scalability of MapReduce C5 Tree.

- Execution time: MapReduce C5.0 Tree is used less than C4.5 and C5, The model take more than 8 hours despite the fact that MapReduce C5.0 takes 3 minutes to do the DT generation. And terminate the process successfully.
- Memory: MapReduceC5.0 Tree is less size of memory than C4.5 when executing a job.

Chapter Five: Conclusions and Future Work

5.1 Conclusion

In conclusion, there is no doubt that the size of data, storage capacity, processing power, and availability of data are becoming increasing. However, research of this thesis aimed to implement a C5 DT algorithm using a Hadoop MapReduce and to overcoming memory restriction and time complexity when the dataset is huge. Therefore, the results of this thesis concluded that the classical algorithm is considered one of the most approaches used in small or medium datasets. And, there are several challenges for data mining algorithms to be performed on big data. Accordingly, the researcher made his efforts to implement a DT using the C5.0 algorithm to solve the challenges of massive data, and the effect on memory usage and time complexity. In addition, MapReduce C5.0 trees are smaller and faster than C4.5 is proving that the proposed approaches outperformed the sequential, even a single node.

5.2 Future work

In future works, we recommend using Hadoop MapReduce and Yarn to further research other typical data mining and machine learning algorithms.

6 References

Bikku, T., Sambasiva Rao, N., & Akepogu, A. R. (2016). Hadoop based feature selection and decision making models on big data. *Indian Journal of Science and Technology*, *9*(10), 660–665. https://doi.org/10.17485/ijst/2016/v9i10/88905

Cui, Y., Yang, Y., & Liao, S. (n.d.). *PDTSSE : A Scalable Parallel Decision Tree Algorithm Based on MapReduce*. 400–406.

Dai, W., & Ji, W. (2014). A mapreduce implementation of C4.5 decision tree algorithm. *International Journal of Database Theory and Application*, *7*(1), 49–60. https://doi.org/10.14257/ijdta.2014.7.1.05

Hu, P., & Dai, W. (2014). Enhancing fault tolerance based on hadoop cluster. *International Journal of Database Theory and Application*, *7*(1), 37–48. https://doi.org/10.14257/ijdta.2014.7.1.04

John Naisbitt. (1981). No Title. Retrieved from https://ericbrown.com/drowning-in-data-starved-for-information.htm

Kebande, V., & Venter, H. S. (2015). A functional architecture for cloud forensic readiness large-scale potential digital evidence analysis. *European Conference on Information Warfare and Security, ECCWS*, *2015-Janua*(September), 373–382.

Kesorn, K., & Chanmee, S. (2020). *Data Quality Enhancement for Decision Tree Algorithm using Knowledge-Based Model*. *20*(March), 259–277. https://doi.org/10.14456/cast.2020.15

Krish, K. R., Anwar, A., & Butt, A. R. (2014). HatS: A heterogeneity-aware tiered storage for hadoop. *Proceedings - 14th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2014*, (June), 502–511. https://doi.org/10.1109/CCGrid.2014.51

Lakshmi, B. N., Indumathi, T. S., & Ravi, N. (2016). A Study on C.5 Decision Tree Classification Algorithm for Risk Predictions During Pregnancy. *Procedia Technology*, *24*, 1542–1549. https://doi.org/10.1016/j.protcy.2016.05.128

Lumpur, K. (2018). *Decision Tree Algorithms C4 . 5 and C5 . 0 in Data Mining : A Review Faculty of Computer Science and Information Technology , University of Malaya*. *11*(1), 1–8.

Nathan, A. J., & Scobell, A. (2012). Classification of Forest Cover Type Using Random Forests Algorithm. In: Kolhe M., Tiwari S., Trivedi M., Mishra K. (eds) Advances in Data and Information Sciences. Lecture Notes in Networks and Systems, vol 94. Springer, Singapore. *Foreign Affairs*, *91*(5), 287. https://doi.org/10.1017/CBO9781107415324.004

Polo, J. (2013). Big data processing with mapreduce. *Big Data Computing*, 295–313. https://doi.org/10.1201/b16014

Purdila, V., & Pentiuc, S. (2014). MR-Tree - A Scalable MapReduce Algorithm for Building Decision Trees. *"Journal of Applied Computer Science & Mathematics," 16*(8), 16–19.

Qasem, M. H., Sarhan, A. A., Qaddoura, R., & Mahafzah, B. A. (2018). Matrix multiplication of big data using MapReduce: A review. *Proceedings of 2nd International Conference on the Applications of Information Technology in Developing Renewable Energy Processes and Systems, IT-DREPS 2017*, *2018-Janua*, 4–9. https://doi.org/10.1109/IT-DREPS.2017.8277807

Rajeswari, S., & Suthendran, K. (2019). Feature Selection Method based on Fisher's Exact Test for Agricultural Data. *International Journal of Recent Technology and Engineering*, *8*(4S2), 558–564. https://doi.org/10.35940/ijrte.d1104.1284s219

Revathy, R., Balamurali, S., & Lawrance, R. (2019). Classifying agricultural crop pest data using hadoop MapReduce based C5.0 algorithm. *Journal of Cyber Security and Mobility*, *8*(3), 393–408. https://doi.org/10.13052/jcsm2245-1439.835

Revathy, R., & Lawrance, R. (2017). Comparative Analysis of C4.5 and C5.0 Algorithms on Crop Pest Data. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(1), 50–58. Retrieved from www.ijircce.com

Shirzad, E., & Saadatfar, H. (2020). Job failure prediction in Hadoop based on log file analysis. *International Journal of Computers and Applications*, *0*(0), 1–10. https://doi.org/10.1080/1206212X.2020.1732081

Tutorialspoint. (2019). No Title. Retrieved from https://www.tutorialspoint.com/map_reduce/map_reduce_quick_guide.htm

Wu, G., Li, H., Hu, X., Bi, Y., Zhang, J., & Wu, X. (2009). MReC4.5: C4.5 ensemble classification with MapReduce. *4th ChinaGrid Annual Conference, ChinaGrid 2009*, (August), 249–255. https://doi.org/10.1109/ChinaGrid.2009.39

Yang, T., & Hiong Ngu, A. H. (2017). Implementation of Decision Tree Using Hadoop MapReduce. *International Journal of Biomedical Data Mining*, *06*(01), 1–4. https://doi.org/10.4172/2090-4924.1000125

Yuvaraj, N., & SriPreethaa, K. R. (2019). Diabetes prediction in healthcare systems using machine learning algorithms on Hadoop cluster. *Cluster Computing*, *22*. https://doi.org/10.1007/s10586-017-1532-x