DEVELOPMENT OF AN ENHANCED C4.5 DECISION TREE ALGORITHM USING A MEMOIZED MAPREDUCE MODEL

BY

Florence Oyebimpe PAUL
P15SCMT8031

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE,
AHMADU BELLO UNIVERSITY,
ZARIA, NIGERIA

November, 2019

TITLE PAGE

DEVELOPMENT OF AN ENHANCED C4.5 DECISION TREE ALGORITHM USING A MEMOIZED MAPREDUCE MODEL

BY

Florence Oyebimpe PAUL
P15SCMT8031

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF MASTER OF SCIENCE DEGREE IN COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE,
AHMADU BELLO UNIVERSITY,
ZARIA, NIGERIA

November, 2019

DECLARATION

I declare that the work in this dissertation entitled “Development of anEnhancedC4.5 Decision Tree Algorithm using a Memoized MapReduce Model” has been carried out by me in the Department ofComputer Science, under the supervision of Prof. A.A. Obiniyi and Dr. A.F. Donfack Kana. This dissertation has not been presented for another degree at any university. The information contained in the literature has been duly acknowledged in the text and a list of reference provided.

Paul Florence Oyebimpe _____ _____
Name of Student Signature Date

CERTIFICATION

This dissertation entitled “DEVELOPMENT OF ANENHANCEDC4.5 DECISION TREE ALGORITHM USING A MEMOIZED MAPREDUCE MODEL” by Florence Oyebimpe PAUL meets the regulations governing the award of the degree of M.Sc. in Computer Science of the Ahmadu Bello University (ABU), Zaria and is approved for its contribution to knowledge and literary presentation.

Prof. A. A. Obiniyi Chairman, Supervisory Committee	_____ Signature	_____ Date
Dr. A.F. Donfack Kana Member, Supervisory Committee	_____ Signature	_____ Date
Prof. S.E. Adewumi External Examiner	_____ Signature	_____ Date
Prof. S. B. Junaidu Head of Department	_____ Signature	_____ Date
Prof. S. Abdullahi Dean, School of Postgraduate Studies	_____ Signature	_____ Date

DEDICATION

This dissertation is dedicated to the Most High God, and to my beloved family. I want to appreciate my beloved husband Dr. Elaoyi David Paul and my wonderful children Eboh, Oyare and El'adaje for their love, support and encouragement throughout the period of this programme.

ACKNOWLEDGEMENT

My profound gratitude goes first to the Almighty God for the wisdom and grace He gave me to do this work. I appreciate my supervisors Prof. A.A. Obiniyi and Dr. A.F. Donfack Kana for their inputs, thorough supervision and efforts made to make sure this work was a success. My thanks also goes Dr. Oyelade, Chuka Ezeaku and Jerry for their assistance, and to other friends and colleagues of mine for their encouragements. May the Lord bless everyone.

ABSTRACT

Classification is a technique in data mining that focuses on prediction. It is used for classifying newly available data based on the data that is already known. The classical C4.5 decision tree algorithm is one of the popularly known decision tree algorithms that is used in classification. Its wide application is due to its ability to perform well in the prediction of unknown variables, ease of use and cost of implementation which is less expensive compared to other classification algorithms. However, C4.5 algorithm is limited by its complexities in computation which arises from large datasets. These limitations have led to an inefficient performance of the algorithm in terms of computing time, memory utilization and data complexity. Several researches have been carried out to handle these limitations, the parallelizing of the C4.5 algorithm using the Map Reduce programming model is one of such improvements. This involves the breaking down of a large dataset into smaller chunks of data and distributing them on multiple computers for parallel processing. However, the computational cost remains high due to high number of calculations in which a large number of them are repeated at each node due to the recursive nature of the algorithm. This research his aimed at further reducing computation time by using a memoized Map Reduce model. The memoization technique is incorporated into the model, to store the result of previous calculations used in the selection of the best attribute for partitioning larged a taset. When same calculations re-occurs, the cached result will be returned, thereby eliminating re-computation. The experiment was carried out on a single node hadoop cluster environment with datasets sourced online from University of California Irvine (UCI) machine learning, Kaggle and Data, gov repositories. Results were captured and evaluated based on the execution time. The summary of the result in the proposed system shows that there was a 10.80% decrease in time.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF APPENDICES

LIST OF ALGORITHMS

LIST OF ABBREVIATIONS

DP: Dynamic Programming

CART: Classification and Regression Tree

ID3: Iterative Dichotomiser

KDD: Knowledge Discovery in Databases

SVM: Support Vector Machines

KNN: K-Nearest Neighbor

HDFS: Hadoop Distributed File System

UCI Repository: University of California at Irvine Repository

CAT: Computerized Axial Tomography

CHAPTER ONE

1.0 INTRODUCTION

1.1 Background to the study

An increase in the growth and development of powerful computing and data storage technologies, has ledto a massive explosion of data, springing out of diverse facets of life such as medicine, sciences, finance,government and in many other fields.Jaseena and David (2014) pointed out that the rapid growth of data and information is posing a great need, and if they are not properly harnessed, they can become a big challenge. The need to discover knowledge from these databases led to the first workshop held at the International Joint Conference on Artificial Intelligence (IJCAI) in Detroit, where Gregory I. Piatetsky-Shapirocoined the phrase "Knowledge Discovery in Databases" (KDD), (Piatetsky-Shapiro, 1991) . KDD refers to the overall process of discovering useful knowledge from data. Data mining is one of the steps in this process and it involves the application of specific algorithms in the extraction of these patterns from data (Fayyad *et al.,* 1996).

1.1.2 Data mining

Data mining has attracted a great deal of attention in the information industry and in the society at large in recent years. This is due to the availability of huge amounts of data and the need to turn these data into useful information and knowledge. It is a process of identifying interesting patterns in databases that can be used in making decisions (Bose and Mahapatra, 2001; Song *et al*., 2018). Data mining uses different techniques to extract useful knowledge from data, these techniques are classified into supervised and unsupervised learning algorithms. A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples while unsupervised learning algorithm deals with unlabelled instances, and

the classes have to be inferredfrom the unstructured dataset(Sebastian, 2014). Classification is an example of a data mining technique.

1.1.3 Classification

Classification is a supervised learning technique, which focuses on prediction and it is used for classifying newly available data based on the data that is already known. It is a technique where the data is categorized into a number of classes, with the aim of finding the class to which a new data belongs.To achieve this aim, a two- step process is involved,the first beingcreation of a model by applying a classification algorithm for instance a decision tree algorithm on the training dataset and in the second step the model is tested against a predefined test dataset (Neelamegam and Ramaraj, 2013).

1.1.4 Decision tree

A decision tree is a classification system which generates a tree and set of rules, representing the model of different classes from a given dataset. It is commonly used among othe classification algorithms because of its ease of use and cost of implementation which is less expensive (Rokach and Maimon,2014).Decision trees can handle both categorical and numerical data, this is known as classification or regression models respectively. It breaks down a dataset into smaller subsets builds a decision tree with decision nodes and leaf nodes from it.

1.1.5 C4.5 decision tree algorithm

The classical C4.5 algorithm is an example of a decision tree, it is a classifier used in buiding decision trees and generating classification rules used in decision making. It is an extension of theIterative Dichotomiser 3(ID3) algorithm, which developed by Quinlan Ross 1986, it was rated one of the top ten most influential data mining algorithms in the research community (Wu *et al*., 2008). Research has shown that even

though the algorithm performs well when handling small datasets, it however suffers from the following bottleneckswhen handling large datasets, the capacityto store largedataset on a single computer is restricted, also the time it takes to complete computation is quite high. The complex nature of big data which results from the volume, speed and variety of data that is being generated daily, has made it difficult for the traditional system of processing to cope with the data complexity. These limitations have led to series of improvements on the algorithm, one of such is the parallelising of the C4.5 algorithm through the use of the MapReduce programming model (Dai and Ji 2014; Mu *et al*., 2017).

1.1.6 MapReduce

The MapReduce programming model is used for parallel and distributed processing of large datasets on clusters (Dean & Gemawat, 2008; Zhu *et al*., 2018). It is a fault-tolerant, simple, and scalable framework for data processing that enables its users to process huge amount of data (Wang *et al*., 2014). The works of Dai and Ji, (2014) and Mu *et al*., (2017) were able to handle the above mentioned bottlenecks by parallalizing the classical C4.5 algorithm through the use of the MapReduce paradigm. However Badgujar and Sawant, (2017), discovered that the gain ratio calculations which is a criterion for selection of the best attribute for splitting the dataset to form the nodes for the building of the decision tree are quite expensive. Due to the recursive nature of the algorithm, these calculations are often repeated and re-computed thereby increasing the execution time. The authors addressed this challenge, by introducing an approximation method to simplify the calculation process involved in the C4.5 algorithm.This research is aimed at handling the problem of repeated calcualations in the parallelised C4.5 algorithm by applying the technique which will optimize the algorithm by storing the

result of previous calculations in a cache and making it accessible for re-use whenever the need arises.

1.1.7 Memoization

Memoization is an optimization technique in Dynamic Programming (DP), whichuses a hash table to store the result of previous computations in a cache and in the course of processing if similar inputs are encountered.The stored result is accessed and re-usedinstead of re-computing it. The avoidance of re-computation will help to save time and optimize the algorithm for better performance. The idea of dynamic programming is to build an exhaustive table with optimal solutions to sub-problems, the use of a table helps to avoid re-computation and in so doing it shares computation by storing results and avoiding their re-computation (Pfenning, 2010). It is an approach that offers practical problem solving tools in many different application areas such as network optimization, decision analysis, inventory problems, artificial intelligence, computer science, agriculture, forestry, finance and medicine (Sniedovich, 2004).

1.2 Problem Statement

According to Badgujar and Sawant(2017), there are several calculationsin the selection of test attributes which best splits the dataset into partitions. Due to the recursive nature of the C4.5 algorithm manyof these calculations are repeated therefore leading to re-computation.Re-computing such calculations each time they are encountered, results in high computation time which slow down the construction of the decision tree and the classification rules generated by the tree,thereby making the algorithm less efficient. To improve the time efficiency of the algorithm, there is a need to use a technique that eliminates the recomputation of already encountered subproblems.

1.3 Motivation of Research

The wide application of C4.5 decision tree algorithm in different fields of study has encouraged several researches in its ability to perform well on different datasets. From literature (Han and Kamber, 2006; Badgujar and Sawant, 2017) it was discovered that the algorithm performs well when the dataset is minimal but when the dataset is large its performance is poor due to the time consumed in building the decision tree from which classification rules are generated for decision making(Dai and Ji, 2014).

Moreover, its ability to predict unknown classes from previously known classes stems from the fact that it selects the best attribute with the highest gain ratio in splitting the dataset into partitions. These selected attributes are used to create nodes which form the decision tree. Having carefully studied the computations involved in attribute selection, it was discovered that for each node in the selection of test attributes, there are logarithmic calculations involved, with calculations which have been previously performed (Badgujar and Sawant, 2017). This work proposes an enhancement of the algorithm by using the memoization technique,which will help to solve the problem of recomputation. This improvement is expected to reduce the time requirement and overcome the memory limitation of the C4.5 algorithm.

1.4 Aim and Objectives

The aim of the research is to enhance the C4.5 decision tree algorithm by reducing the computational overload using a memoized MapReduce model. The aim is expected to be achieved through the following objectives by;

i. designing an enhanced C4.5 algorithm;
ii. implementing the enhanced C4.5 algorithm;
iii. evaluating and comparing the enhanced C4.5 algorithm against a previous result.

1.5 Research Methodology

i. Designing an enhanced C4.5 algorithm involves the following steps;

   i. downloadingdatasets from UCI(University of Irvine) repository, Kaggle and Data.gov repositories;
   ii. designing a distributed data storage for storing the datasets;
   iii. designing a C4.5 memoized MapReduce flowchart;
   iv. designing a C4.5 memoized algorithm;
   v. designing a C4.5 memoized system architecture.

b. To implement the enhanced C4.5 algorithm, the following steps were taken;

   i. install Apace Hadoop on a single nodecluster,
   ii. store datasets on HDFS
   iii. implement memoized MapReduce algorithm using Java programming language.

c. To evaluate the enhanced C4.5 algorithm, the following steps were taken

   i. Results were compared with that of Mu *et al*., (2017) on large data sets and evaluation was measured based on processing time.

1.6Organization of Dissertation

This work consist of five chapters. Chapter one focuses on the generalbackground to the study, it also examines the research problem, the aim and objectives of the research and methods of achieving the objectives. Chapter two describes related works done byothers. Chapter three presents the proposed model, which includes the proposed algorithm, the flowchart and the system architecture. In chapter four the analysis and result of the implementation was

presented and discussed. Chapter 5presents the summary, conclusion and the recommended future work.

1.7 Contribution to Knowledge

a. This research was able to design and implement an enhanced C4.5 algorithm.

b. The enhanced C4.5 algorithm was improved using a memoized mapreduce model.

CHAPTER TWO

2.0 LITERATURE REVIEW

2.1 Knowledge Discovery in Databases (KDD)

KDD is a process that focuses on the overall discovery of knowledge from data, this includes how the data are stored and retrieved, as well as how algorithms can be scaled to large datasets and still run efficiently. In addition, the discovery of knowledge from data includes visualizing and interpreting results and how the overall communication between man and machine can be usefully modeled and supported (Fayyad *et al.*, 1996; Goebel, 2014).

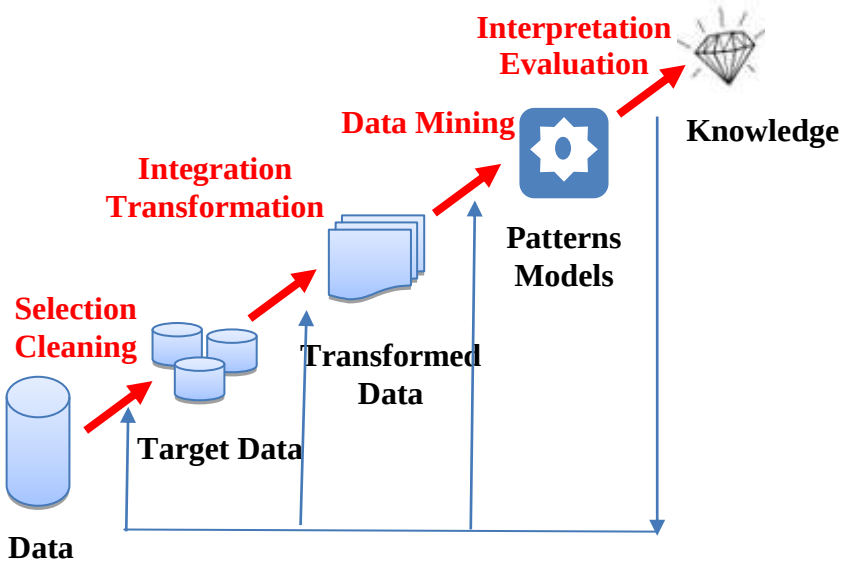

Figure 2.1: The KDD Process (Dhaenens and Jourdan, 2016)

This process is interactive and iterative,it involves the use of a database from which a target data is created by selecting a data set or data sample on which the knowledge discovery is to be performed. At this stage the data also undergoes cleaning and

processing which includes the removal of meaningless data known as noisy data and the type of strategy to use in handling missing data fields(Hiremath & Patil, 2016).

2.2 Data Mining

Data mining is a process that uses statistical, mathematical, artificial intelligence, and machine learning techniques to extract and identify useful information and subsequently gain knowledge from a large database (Turban *et al*., 2007). Vladan (2001) described it as the process of pattern discovery in a data set from which noise has been previously eliminated and has been transformed in such a way as to enable the pattern discovery process. It is also described as a process of nontrivial extraction of implicit, previously unknown and potentially useful information (such as knowledge rules, constraints, regularities) from data in databases (Piatetsky-Shapiro and Frawley, 1991;Hiremath & Patil, 2016). This process involves the use of specific algorithms to extract useful patterns from data. The information and knowledge gained can be used for applications ranging from market analysis, fraud detection, and customer retention, to production control and science exploration (Han and Kamber, 2006).

2.3 Data Mining Techniques

Data mining techniques like Classification, Clustering, Regression and Association Rules are used for knowledge discovery from databases.

a. Classification

Classification technique is a supervised learning algorithm that focuses on the prediction based on known properties. It can predict categorical class labels and classify data based on training set and class labels, hence, it can be used for classifying newly available data. It is an unavoidable part of data mining and is gaining more popularity (Chandgude and Pawar, 2015). Classification has several applications in fraud

detection, target marketing, performance prediction, manufacturing and medical diagnosis. An example of this technique is the Naïve Bayes algorithm, a statistical classifier which can be used for text classification (El- Kourdi *et al*., 2004; Mansour, 2018).

b. Clustering

Clustering is an unsupervised learning technique which organises an observed data into meaningful arrangements, groups or clusters that are initially unknown or not pre-defined. The technique is aimed at grouping a set of objects into k clusters such that objects in the same cluster are similar and very different from objects of other clusters. There is no provision of providing accurate characterization of unobserved samples that are generated from the same probability distribution (Rui and Donald, 2005; Zhao and Fränti, 2014). According to Yue *et al*., (2007) clustering analysis is concerned with the problem of decomposing or partitioning a data set into groups so that the points in one group are similar to each other and are as different as possible from the points in other groups. An example of this technique is the K-means clustering. Cluster analysis has been widely used in numerous applications, including market research, Agriculture(Majumdar *et al*., 2017), data analysis, and patternrecognition (Lin *et al*., 2017). The k-mode clustering algorithm was used the market segmentation analysis in E-commerce business (Kamthania *et al*. 2018).

c. Regression

Regression is also used in prediction analysis, but regression is used to predict a numeric or continuous value while classification assigns data into discrete categories. Unlike classification, regression takes a numerical dataset and develops a mathematical formula that fits the data. It analyzes the dependency of some attribute values, which are

dependent on the values of other attributes that are present in the same item (Sathiyapriya and Kanagaraj, 2018).Wang *et al*., (2018) proposed the use of regression method in fore-casting monthly electricity sales.Khoshgoftaar *et al*., (2002) proposed the use of regression trees to classify fault-prone software modules.

d. Association Rule

Association rules are if/then statements that help to uncover relationships between unrelated data in a database, these rules are used to find the relationships between the objects which are frequently used together. In association, a pattern is discovered based on a relationship between items in the same transaction, which is why the technique is also known as a relation technique. An example of the algorithm that uses this technique is the Apriori algorithm (Lahti, 2008).Association rules are applied in market basket analysis, cross-marketing, catalogue design, and loss-leader analysis (Kumbhare and Chobe, 2014).The market basket analysis is a process that analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets. The discovery of such associations can help retailers develop marketing strategies by gaining understanding into which items are frequently purchased together by customers. For instance, if customers are buying computers, how likely are they also to buy antivirus as they take their trip to the electronic shop. Such information can lead to increased sales by helping retailers doselective marketing and plan their shelf space (Han and Kamber, 2006).

2.4 Data Mining Algorithms

Data mining algorithms are well-defined procedures that takes data as input and produces output in the form of models or patterns. They are applied in pharmaceutical industries in detecting the adverse effect of pharmaceutical products (Subeesh, *et al*.,

2018).They can be used in fraud detection (Albashrawi, 2016; Nadarajan and Ramanujam, 2016). Several algorithms exist. Namely Neural Networks. SVM, and Decision Trees,

a. Neural Networks

Neural Networks or Neural nets are highly interconnected networks of relatively simple processing elements, or nodes that operate in parallel. They are designed to mimic the function of neuro-biological networks. It consist of an interconnected group of artificial neurons and processes information using a connectionist approach to computation. In most cases it is an adaptive system that changes its structure based on external or internal information that flows through the network during the learning phase (Lippmann, 1988; Shahid *et al*., 2019). Neural Networks are applicable in medical diagnoses, their ability to learn, make them suitable for diagnosing diseases (Al-Shayea, 2011) and recognising diseases from various scans for example, cardiograms, computerized axial tomography (CAT)scans and ultrasonic scans. Neural networks are ideal in recognising diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so that the details of how to recognise the disease are not needed(Saleh and Knieper, 2017).

b. **Support vector machines (SVM)**

SVM is a supervised learningtechnique which has been successfully applied in many application areas such as face recognition, bioinformatics and forecasting. It is one of the most successful classification algorithms in the data mining area, but its long training limits its use. The aim of SVM is to find optimal separating hyper plane by maximizing the margin betweenthe two classes, which offer the best generalization ability for future data(Cortes and Vapnik, 1995).Their computation and storage requirementsincrease

rapidly with the number of training vectors. SVM uses statistical learning theory to maximize generalization propertyof generated classifier model. The basic SVM takes a set of input data and predicts, for each given input, which of two possibleclasses forms the output, making it a probabilistic binary linear classifier. Support Vector Machines (SVM) are the classifierswhich were originally designed for binary classification (Chang and Lin, 2011). This technique is applied in medical diagnosis (Chu *et al.*, 2005).

c. Decision Trees

The Decision tree algorithms are commonly used because they are easily understood and cheap to implement. A decision treeis a classification system which generates a tree and set of rules, representing the model of different classes from a given dataset (Han and Kamber, 2006). A decision tree, Figure 2.2, is a flow chart like tree structure where each internal node denotes a test on an attribute, each branch represents an outcome of the test and leaf nodes represents the classes or class distributions, while the top most node in a tree is the root node. The root node which has no incoming edge while the internal nodes with exactly one incoming edge, are known as the decision nodes. At the training stage, each internal node split the instance space into two or more parts with the objective of optimizing the performance of a classifier. After that, every path from the root node to the leaf node forms a decision rule to determine which class a new instance belongs The decision tree performs the classification of a given data sample through various levels of decisions in order to reach a final decision. Most decision tree induction algorithms like ID3, C4.5 and CART algorithms are based on a greedy top down recursive partitioning strategy for tree growth. They use different variants of impurity measures to select an input attribute to be associated with an internal node, viz;

information gain (Barros *et al*., 2012), gain ratio (Wang *et al*., 2005), and distance-based measures (De Mántaras, 1991).

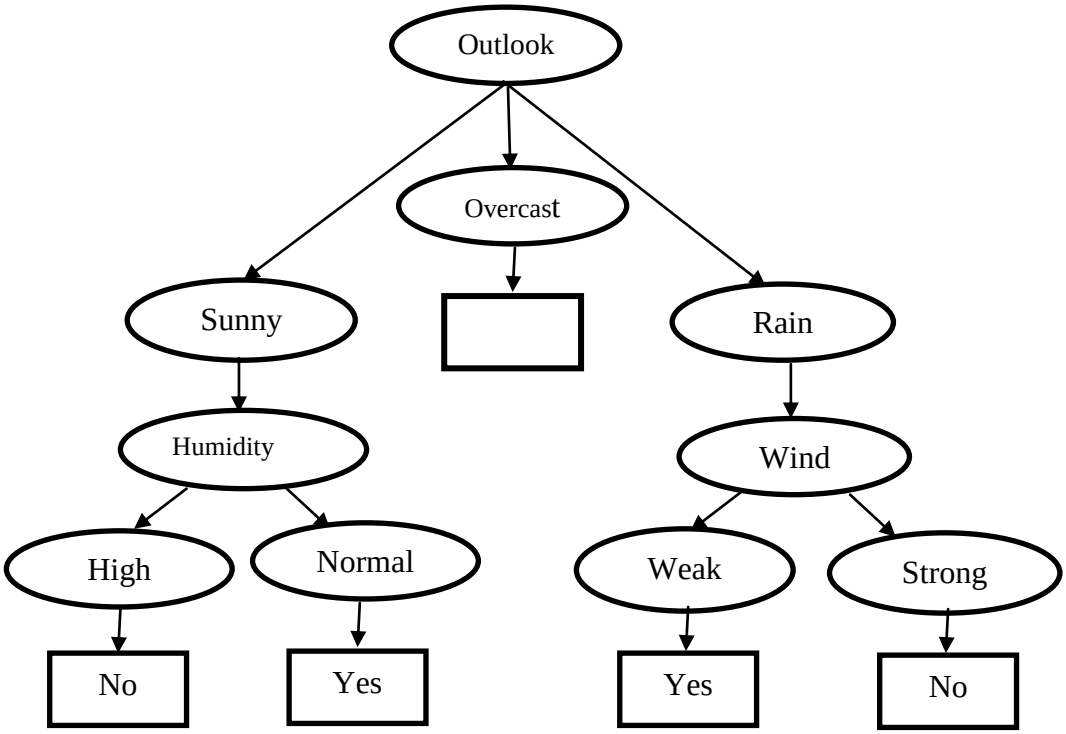

Figure 2.2: The Decision Tree (Lavrenco & Sutton., 2009)

The Iterative Dichotomiser 3 (ID3), C4.5, and CART are examples of this classifier.

i. CART

The CART algorithm was introduced by Breiman in 1984, it builds classification and regression trees for predicting continuous dependent variables (regression) and categorical predictor variables (classification) (Breiman, *et al*., 1984). The fundamental idea is to select each split of a subset so that the data in each of the descendant subsets are purer than the data in the parent subset. The impurity or purity measure used in building decision tree in CART is Gini Index (Sharma and Kumar, 2016).

ii. ID3

It is an algorithm invented by Ross Quinlan which is used to generate a decision tree from the dataset (Quilan, 1993). The ID3 algorithm is a classification algorithm based on Information Entropy and its basic idea is that all examples are mapped to different categories according to different values of the condition attribute set. The algorithm uses information gain (Barros *et al*., 2012), as an attribute selection criteria, usually the attribute that has the highest information gain is selected as the splitting attribute of current node. According to the different values of the attribute, branches can be created, and the process above is recursively done on each branch to create other nodes and branches until all the samples in a branch belong to the same class. In selecting the splitting attributes, the concepts of Entropy and Information Gain are used (Adhatrao *et al*., 2013).

iii. C4.5

C4.5 algorithm is an improvement of the ID3 algorithm, developed by Quinlan Ross. It is based on Hunt's algorithm and also like ID3, it is serially implemented. Pruning takes place in C4.5 by replacing the internal node with a leaf node thereby reducing the error rate (Podgorelec *et al.*, 2002). Unlike the ID3, the C4.5 accepts both continuous and categorical attributes in building the decision tree. It has an enhanced method of tree pruning that reduces misclassification errors, due to noise or too-much detail in the training data set. The C4.5 algorithm is a standard algorithm for inducing classification rules in the form of decision tree,it uses a divide-and-conquerapproach to growing decision tree. The algorithm is based on the information gain ratio which is evaluated by entropy, the information gain ratio measure is used to select the test features at each node in the tree and such a measure is referred to as a feature or attribute selection

measure. The attribute with the highest information gain ratio is chosen as the test feature for the current node (Seema *et al*., 2013). See appendix A for the C4.5 algorithm.

2.5 Application of the C4.5 Algorithm

Adeyemo *et al*., (2015) applied three different data mining algorithms, namely Multilayer Perceptron (MLP) Artificial Neural Network, ID3 and C4.5 Decision Trees algorithms in diagnosing patients with typhoid fever. Cinaroglu, (2016), used three algorithms Random Forest, CART and C4.5 respectively to predict the health expenditure of countries who are members of the Organization for Economic Co-operation and Development (OECD). Baskar *et al*., (2013) applied it in the prediction of soil fertility.It was used in predicting students' performance in examination (Adhatrao*et al*., 2013; Hamoud *etal*., 2018) and in the predictionof stock market prices to help investors decide the timing for buying and selling of stocks(Al-Radaideh *et al*., 2013).

2.6 Limitation of the Algorithm

However the algorithm is limited in its ability to handle large datasets. Research has shown that the C4.5 algorithm which is a standard algorithm has undergone a series of improvements in order to enhance its performance in terms of accuracy, execution time and memory utilization (Badgujar and Sawant, 2017; Mu *et al*., 2017; Dai and Ji,2014). The following are some of the improvements on the algorithm.Muslim *et al*., (2017) proposed an optimization of the C4.5 algorithm for the diagnosis of breast cancer. In order to improve the accuracy of the algorithm in diagnosing the disease.Their work combined the Particle Swarm Optimization (PSO) algorithm with C4.5 algorithm. The integration was used to optimise attribute selection in the C4.5 algorithm which led to the improvement of the algorithm. The original and improved algorithms were

evaluated on different datasets, results showed that the improved algorithm performed better than the original algorithm in terms of accuracy with 96.61% and 95.61% respectively when compared with each other.

Rajeshinigo and Jebamaler, (2017) also improved on the accuracy in prediction of the C4.5 algorithm by using the K-means clustering algorithm to transform continuous values into categorical values. Their work handled the setbacks caused by continuous values which is lowering the accuracy of the C4.5 algorithm and producing large trees which leads to post pruning as a result of the values being considered more than once while building the decision tree. The integration of the K-means clustering algorithm with the C4.5 algorithm resulted in a better performance of the algorithm compared to the original algorithm after evaluation with a percentage accuracy of 73% and 92% respectively. A novel algorithm proposed called the Very Fast C4.5 (VFC4.5) algorithm, which was meant to speed up and improve the performance of the C4.5 algorithm. The VFC4.5 was able to handle the weakness of the C4.5 algorithm in dealing with continuous values by introducing the statistical mean and median as an alternative to the original threshold searching process in the C4.5 algorithm. The result after evaluation showed that the enhanced algorithm performed better than the C4.5 algorithm by showing a significant speed up in the process of binarization. This is the process of transforming continuous and discrete attributes into one or more binary attributes(Cherfi *et al*., 2018).

An implementation of the C4.5 decision tree algorithm using the MapReduce programming model was proposed by Dai and Ji,(2014). The authors observed that as the size of dataset increases the process of building the decision tree consumes more time and the memory becomes too small for the data to fit in. This resulted in the movement of some computations to external storage leading to an increase in the cost of

input/output (I/O cost). In order to leverage these limitations, MapReduce was introduced as an improvement on the algorithm. Both algorithms were evaluated and compared, result showed that the improved algorithm did better than the original algorithm. As the dataset was increasing the execution time in the improved algorithm was reducing due to the parallel distribution of data on different computing nodes. Scalability was also achieved in their work, indicating that as the dataset increases and more nodes were added, the overall execution time also decreased thereby making the algorithm more efficient.

Badgujar and Sawant,(2017) proposed an Improved C4.5 Decision tree classifier algorithm for analysis of data mining application to solve the problem of low efficiency of computing time when C4.5 algorithm is used in mass calculations due to large dataset. This was done by improving the rule of C4.5 through the use of L'Hospital Rule which says if $\lim_{x\to c} f(x) = 0\ and\ \lim_{x\to c} g(x) = 0\ and\ \lim_{x\to c} \frac{f'(x)}{g'(x)} = L\ then\ \lim_{x\to c} \frac{f(x)}{g(x)} = L$ using an approximation method to simplify the calculation process by eliminating logarithmic calculation involved in calculating Gain-Ratio in the C4.5 algorithm and approximating them with less complex operations. This resulted in a faster construction of the decision trees the reduction in execution time of the improved algorithm when compared with the ID3 and original C4.5.

Mu *et al.*, (2017) proposed a parallelized C4.5 decision tree algorithm based on MapReduce (MR-C4.5-Tree) in order to solve the problem of memory restriction and time complexity due to large dataset. In achieving their aim the authors introduced two parallelized methods in building the tree nodes, this includes the MR-A-S which is an attribute selection method based on the MapReduce framework, it was used to compute

the information gain ratio based on the several subsets that are divided from the whole data by the Hadoop framework.

2.7 Hadoop

Hadoop is an open source, the most accessible and the most mature implementation of MapReduce. In Hadoop, data storage and computation both reside on the same machines within the cluster. This proximitymakes it possible for hadoop to efficiently process large volumes of data. It provides very simple and easy-to-use tools for the purpose of large-scale data storage and analysis and coordination of large number of machines. All the modules in Hadoop framework are designed to automatically handle hardware failures. This fault-tolerance is made possible by the automatic and efficient data replication in Hadoop and since it is written in Java, it can run on any platform running JVM.

2.8 Hadoop Distributed File System Architecture

The job submitted by the user is stored in Hadoop distributed file system (HDFS) which has the capability of storing large datasets, thereby handling the problem of limited memory space in storing large volume of datasets (Cao *et al*., 2016). The HDFS has a master-slave architecture, its cluster consists of a single Namenode known as the master and multiple datanodes known as the slaves (Mahmoud *et al*., 2018).

2.9MapReduce

Siddesh *et al*., (2016) discovered that due to the ever increasing size of the web there is a need for an efficient web crawling algorithm that will be able to crawl the web in a time-efficient manner. In order to handle this limitation,the MapReduce programmimg model was integrated into the web crawler for an efficient optimization of the algorithm. The user of the MapReduce library expresses the computation as two functions,Map and Reduce.The computation takes a set of input key/value pairs, and produces a set of

outputkey/value pairs. As shown in Figure 2.1, the input data is partitioned into splits of appropriate sizes while the Map functionwhich is written by the user, takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library groups together all intermediate values associated with the same intermediate key and passes them to the Reduce function. The Reduce function, which is also written by the user, accepts an intermediate key and a set of values for that key as input and merges these values together to form a possibly smaller set of values. Typically just zero or one output value is produced per reduce invocation. The intermediate values are supplied to the user's reduce function via an iterator,this allows the handling of lists of values that are too large to fit in memory (Dean and Ghemawat, 2008).

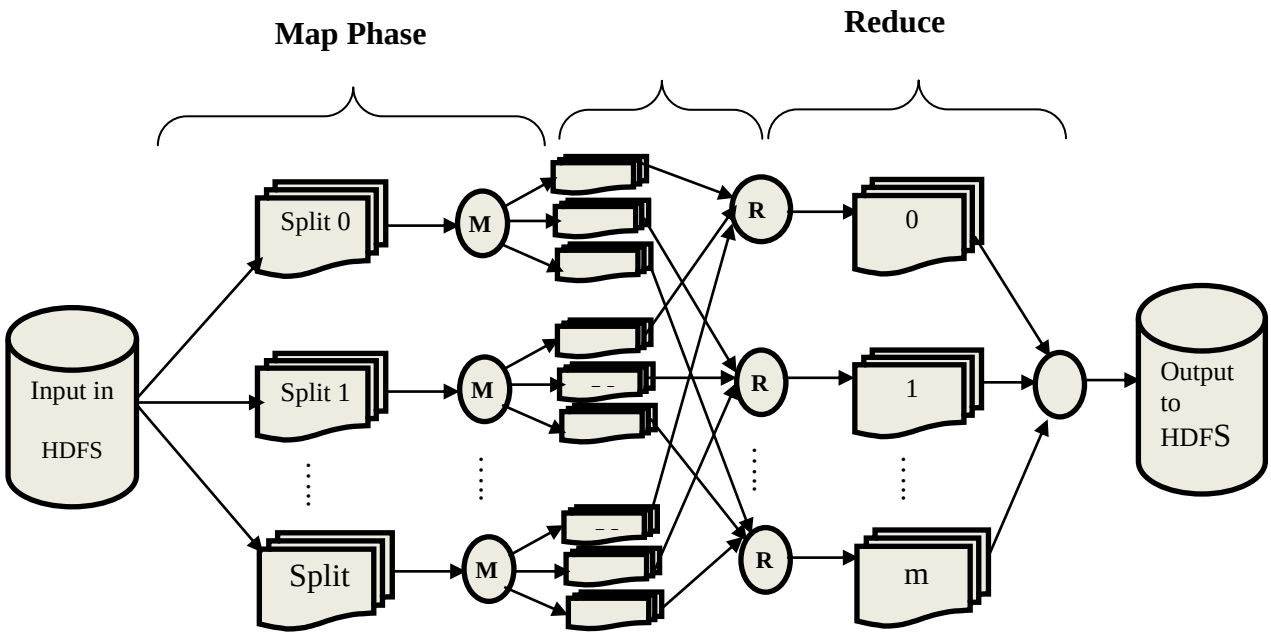

Figure. 2.3: MapReduce Programming Model, (Mu *et al*., 2017)

The Attribute Selection Algorithm (MR-A-S)in appendix B shows the selection of the best splitting attribute the order to find the target attribute from the subsets, the sum of the information gain ratio of the same attribute in the subsets was calculated and the attribute corresponding to the maximum sum was chosen as shown in, whereas in the traditional C4.5 decision tree algorithm the gain ratio is computed based on the whole dataset.As soon as the best splitting attribute and cut points are confirmed by the (MR-A-S), the data splitting method (MR-D-S) was used to partition the training dataset into subsets in a parallelized manner as shown in the Dataset Splitting Algorithm (MR-D-S) in appendix C. The Algorithm describes the data splitting process and also shows how the over-partitioning problem which is a general problem in decision trees is handled. This is done by setting some conditions to limit the size of the decision tree. For instance when the largest class probability in X_{id} is bigger than $\Theta \in [0, 1]$ or the size of X_{id} is smaller than N or the X_{id} belong to the same class then Xid will be created as a leaf node and not added to Ω for the next splitting (Mu *et al*., 2017).

Their work also handled the problem of over-partitioningwhich can lead to an over-fitted and redundant tree by setting some conditions to limit the size of the decision tree.

This researchis aimed at incoporating an optimization technique known as memoization into the work of (Mu *et al*., 2017) to handle the problem of repetitive calculation which occurs in the C4.5 algorithm due to its recursive nature. In the process of selecting the best attribute for splitting the dataset, memoization technique is used to store the result of previous computation in a cache for re-use, sothat when the same inputsare encountered again the cached result is returned, therebyeliminating re-computation and saving execution time.

2.10 Memoization

Memoization is a technique in Dynamic Programming (DP) which is an approach that offers practical problem solving tools in many different application areas such as network optimization, decision analysis, inventory problems, artificial intelligence, computer science, agriculture, forestry, finance and medicine (Sniedovich, 2004). The idea of dynamic programming is to build an exhaustive table with optimal solutions to subproblems. The use of a table helps to avoid re-computation in so doing it shares computation by storing results and avoiding their re-computation (Pfenning, 2010). The term "Memoization", was coined by Donald Michie (Michie, 1968) whois regarded as the father of memoization, because his work was the first to show the benefit of saving the result of a function in a table and re-using it as the need arises (Suresh *et al*., 2016).Jaffar *et al*., (2008) proposed an Efficient Memoization for Dynamic Programming by using interpolants in the generalization of the context of subproblems in order to address the problem of effective reuse of subproblem solutions in dynamic programming. The summary of the generalizations and its optimal solutions was stored in a memo table so that, when a subproblem with similar context is encountered in the search tree it can be reused. A major disadvantage of this technique is the overhead incurred on function calls. It trades memory for speed, it uses more memory by storing the results of previous computations in order to remove some code execution.

2.11 Gaps in literature

From literature, it is observed that most approaches suffer from memory size limitation when the data size is large (Siddesh *et al*., 2016; Dai and Ji, 2014) as well as computational time due to re-computation (Dai and Ji, 2014; Badgujar and Sawant, 2017) although some approaches attempts to address memory requirement issues through MapReduce (Dai and Ji, 2014; Mu *et al*., 2017), the computational time

remains high. The proposed solution of addressing computational time through L'Hospital rule still suffer from approximation as well as memory requirement on large dataset.

This work therefore proposes the enhancement of the MapReduce C4.5 decision tree algorithm by using memoization. This technique will save the result of previous computation such that when the same calculations are encountered there will be no need for re-computation, but the previously stored results can be accessed and reused. This enhancement will further reduce computation time of the algorithm in the construction of decision trees for large datasets, thereby making the algorithm more efficient.

CHAPTER THREE

3.0 METHODOLOGY

3.1 Introduction

Chapter three discusses the design of an enhanced C4.5 decision tree algorithm using a memoized MapReduce model.

3.2 Memoization

Memoization is an optimization technique in Dynamic Programming (DP), which uses a hash table to store the result of previous computations in a cache so that when similar inputs are encountered, the stored result is accessed and re-used instead of re-computing it.This technique is incorporated into the existing model, to store the result of previous calculations used in the selection of the best attribute for partitioning large dataset,and when same calculations are encountered again, the cached result is returned, thereby eliminating re-computation.

3.3 Experimental design

3.3.1 Experimental setup

Hadoop 2.6 was downloaded and installed on a single node hadoop cluster. In a single node cluster the namenode, datanode, jobtracker, and tasktracker all run on a single machine. The experiment was carried out on a dual core personal computer with the CPU running at a clock rate 1.70GHz to 2.40GHz. Java Development Kit (JDK) version 8 was installed on Linux Ubuntu with Secure Shell server SSH properly configured to manage the nodes on the cluster. Three virtual machines were setup with each of them having a database instance also known as nodes installed on them. Prior to installing the first virtual machines the Ubuntu ISO was downloaded, to be installed as the operating system that the machine will be working on. The machine was created using the Oracle virtual box on Ubuntu and given a name, this is because the other machines will be

cloned from it. In the process of installing the virtual machine, Ubuntu was selected as the operating system and 6 GB was allocated to the machine as the Memory (RAM) space. The virtual hard disk was also created and its allocation was done dynamically, this is because the virtual machine will need to take up as much space as it will need so the space on the laptop can be conserved. A maximum size of the 20.13GB was allocated to the hard disk, this is the size limit of the amount of file data the virtual machine will be able to store on the hard disk. In order to start the virtual machine Ubuntu has to be setup, to do this the Ubuntu ISO which was previously downloaded was installed. At the end of the installation a user name and password was created for the virtual machine.

The cloning of the other virtual machines from the first one required some commands that the system had to run, this the Ubuntu terminal. A check was done before cloning the virtual machines to ensure that everything was up to date by using the sudo command which allows the administrator to run any command. The command '*sudo apt-get update; sudo – get upgrade*' checks if everything is up to date. After the check, the SSH was also set up, this was done before the cloning of the virtual machines because it enables the software that will be installed on the virtual machine to have ssh access to all the nodes in the cluster as well as have ssh access to itself. The command '*sudo apt-get install openssh-server*' sets up the SSH server.

Cloning of the virtual machines involved setting up a network where the machines can communicate with each other effectively. A virtualbox network (vboxnet0) was setup to enable communication between the machines. Three machines were cloned one after the other, the first was named the master which is meant to have control over the cluster, also the mac address of all the network cards were reinitialised to ensure the network operates effectively and also fully cloned indicating that it is an independent copy of a

virtual machine that shares nothing with the parent virtual machine after the cloning operation. The other two virtual machines were also cloned in the same manner but each having different names that is Slave 1 and Slave 2 respectively with all three machines having the same user name and password. Next a passwordless ssh was set up to enable all nodes communicate with one another through it, this set up was done from the master virtual machine by running some commands which will connect it to the machines. The command 'ssh-keygen' creates some files inside a hidden ssh folder which can be accessed by the command 'ls-a .ssh', the files in the folder contain two keys. The first file has private key id_rsa which is kept on the master chine alone and the second file has a public key 'id_rsa.pub' which was installed on all three machines including the current using the command 'ssh-copy-id' followed by the IP addresses of each of the machines, for instance 'ssh-copy-id 10.71.34.104'.

3.4 Data collection

The data sets used for the experiment were sourced from online open repositories which include the UCI, Machine Learning Repository which is a collection of databases, domain theories, and data generators that are used by the machine learning community for the empirical analysis of machine learning algorithms. Three datasets were downloaded from the UCI repository namely diabetes, breast cancer and dermatology (Khan, n.d; Zwitter &Soklic 1988; llter & Guvenir, 1998) respectively,they are commonly used datasets in the machine learning community.Kaggle, is an online community of data scientists and machine learners, owned by Google limited liability company (Google LLC), a company that specializes in Internet-related services and products. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment. The National Basketball Association (NBA) of the United States historical dataset (Hallmark, 2018) was sourced from the Kaggle.com

website. The NBA is a professional basketball league in the US, it is one of the most premier in the world. The dataset contains five files. Fifa Worldcup dataset (Becklas, 2018).

The Labour market statistics, Dataset was also sourced from the Data.gov. site. The dataset includes the unemployment and employment rates, demand for labour, and changes in wages and salaries.The site is a website of the United States government, it is a repository for federal, state and local government information that are made available to the public in order to improve public access to high value machine readable datasets generated by the government. Table 3.1 shows the different datasets and their sizes.The files are simple text-based files, with the csv extension which indicates that they are comma separated files, an advantage of this file format type is that they are inherently splittable and supported by Hadoop framework. Data is laid out in lines, with each line being a record and terminated by a new line character (/n).

Table 3.1: Datasets from open sourced repositories

S/No	Dataset	Data size	Instances	No. of attributes	Data Type
1.	Diabetes	434KB	769	20	Categorical and integer
2.	Worldcups	26.4KB	21	10	multivariate
3.	Dermatology	366KB	366	33	Categorical and integer
4.	breast cancer	18.5KB	286	9	Categorical
5.	Nba_players_all	638KB	21	22	multivariate
6.	Nba_betting_Totals	9.73MB	131387	9	multivariate
7.	Nba_betting_money_line	7.50MB	125287	7	multivariate
8.	Nba_betting_spread	9.41MB	131691	9	multivariate
9.	Nba players_game_stat	193MB	1048576	33	multivariate
10.	Worldcupplayers	2.27MB	37785	9	multivariate
11.	Ici_dec_18qtr	4.31MB	21058	14	multivariate
12.	Ques-dec 18qtrs	37.7MB	188325	14	multivariate

3.3 Distributed Data storage

The Hadoop File System is designed for storing very large amount of files with streaming data access which implies that applications or commands are executed directly using the MapReduce processing model. The HDFS has a master-slave architecture, its cluster consists of a single namenode known as the master and multiple datanodes known as the slaves. The namenode maintains and manages the datanodes and allocates tasks to them.It is highly fault tolerant and stores huge amount of datasets and provides easy access to them. The files are stored across multiple nodes in an efficient order,these stored files are stored to avoid possible data losses in case of failure, and to make applications available for parallel processing. Data is mainly stored in HDFS's files and these files are separated into one or more segments and further stored in individual data node. The file segments are known as blocks,an HDFS block is the smallest unit of a data in a file system. A block size is 128MB which can be modified as per the requirements from the HDFSconfiguration.All blocks of the file are the same size except the last block, which can be either the same size or smaller. The files are split into 128 MB blocks and then stored into the Hadoop file system.

The Hadoop application is responsible for distributing the data block across multiple nodes The job submitted by the user is stored in the Hadoop's distributed file system, which has the capability of storing large datasets. The Namenode stores metadata such as the filename, number of block, location of block and block ID's. It regulates access to files by clients and manages the file system namespace by executing operations like opening, closing, and renaming files and directories. In order to read or write a file in HDFS, a client interacts with the namenode first, this is because it is the only place where metadata about datanodes are stored and it specifies the location of the slaves where the data is stored, so that the client can interact directly with the

specifieddatanode and read the data from there.Figure 3.1 shows how data is stored and distributed to different nodes on the cluster.

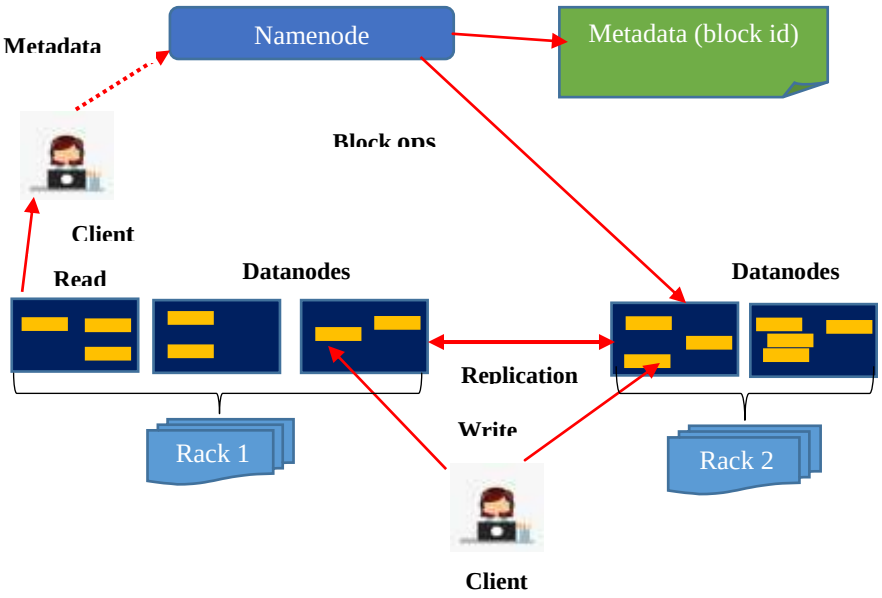

Figure 3.1.Data Storage in HDFS

3.5Memoized MapReduce Architecture

The input data is divided into input splits by the Hadoop framework and a map task is created per input split for the list of <key1, value1> entries in the split one by one to produce an intermediate list of <key2, list of value2>. In map stage, some classes are defined by the user. InputFormat, InputSplits, RecordReader, Mapper and Partioner.

a. InputFormat is a class that defines the source of input files, it defines how input files that are stored in HDFS are split and read. It defines the data splits and the size of the map tasks responsible for processing these splits, the record reader which is responsible for reading actual record from file is also defined in this class.

b. InputSplits defines a single unit of work for a single map task, it represents the data which is processed by an individual mapper. They are created by the Input format and divided into records, each of these records which is a key-value pair are processed by a mapper Therefore, the total number of map tasks is equal to the number of input splits.

c. The RecordReader communicates with the InputSplit and converts datainto key-value pair suitable for the mapper to read**.**By default it uses TextInputFormat to do this conversion. The TextInputFormat treats each line of each input file as a separate record and performs no parsing. The key is the byte offset of the beginning of the line within the file whiles the value represents the contents of the line with the line terminators excluded.

d. The Mapper task is the first phase of processing where each input record from the RecordReader is processed and an intermediate key-value pair is generated as output which the mapper stores on its local disk after the partioning has been done.

e. Partioner,partitions the intermediate key space per Reducer. Before writing the output of each mapper task, partitioning of the output takes place on the basis of the key and then sorting is done. This partitioning specifies that all the values for each key are grouped together.

At the reduce stage Reducer reduces a set of intermediate values which share a common key to a smaller set of values. Reducer has three primary phases which are shuffle, sort and reduce. Shuffle is the process of moving relevant partitions of map outputs to the reduce tasks using the Hypertext Transfer Protocol (HTTP). The set of intermediate keys on a single node is automatically sorted by Hadoop before they are presented to the Reducer. The number of reduce tasks is a user controllable parameter unlike the map

tasks. When number of reduce tasks is high, the shuffle is more complicated and will consumes more bandwidth. A Reducer instance is created for each reduce task and it merges the values which have the same key and the output is written back to the HDFS as defined by the OutputFormat.

Figure 3.2 shows the architecture for the proposed system. Memoization is incorporated at the map phase where the gain-ratio calculation is done for each attribute in order to determine the attribute with the maximum gain-ratio which will be selected for splitting the dataset.

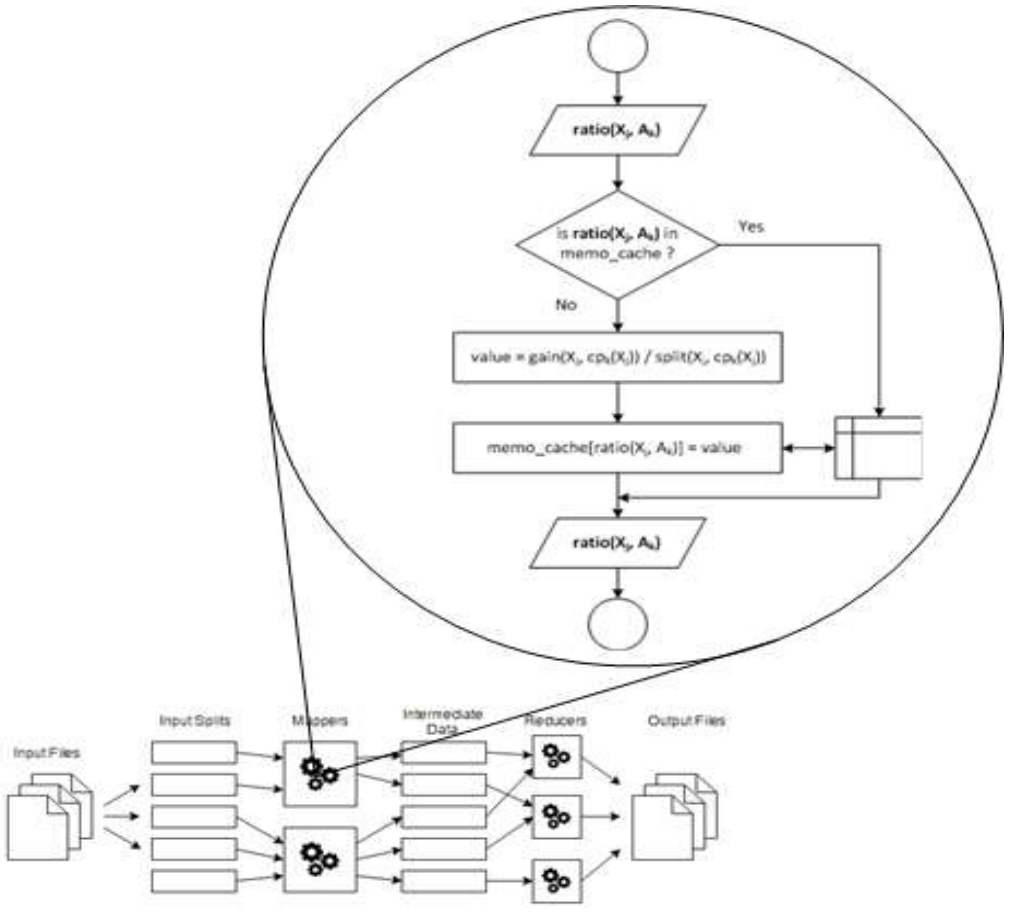

Figure 3.2**Memoized MapReduce Architecture**

3.6Memoized MapReduce Flowchart

The flowchart in Figure 3.3 shows the work flow in the proposed system. The input to the system is the attribute A_k (where k = 1,2, …….. n) in the m subsets. The algorithm checks the memoization cache to see if the input and corresponding result of calculation exists in the hashtable. If the result exists in the cache, it is returned and used, but if it does not exist, the gain-ratio is calculated by taking the ratio of the information gain to the information split for each value of an attribute and storing the result into the variable 'value'. The variable value is then stored in the memoization cache to be used when the need for it arises.

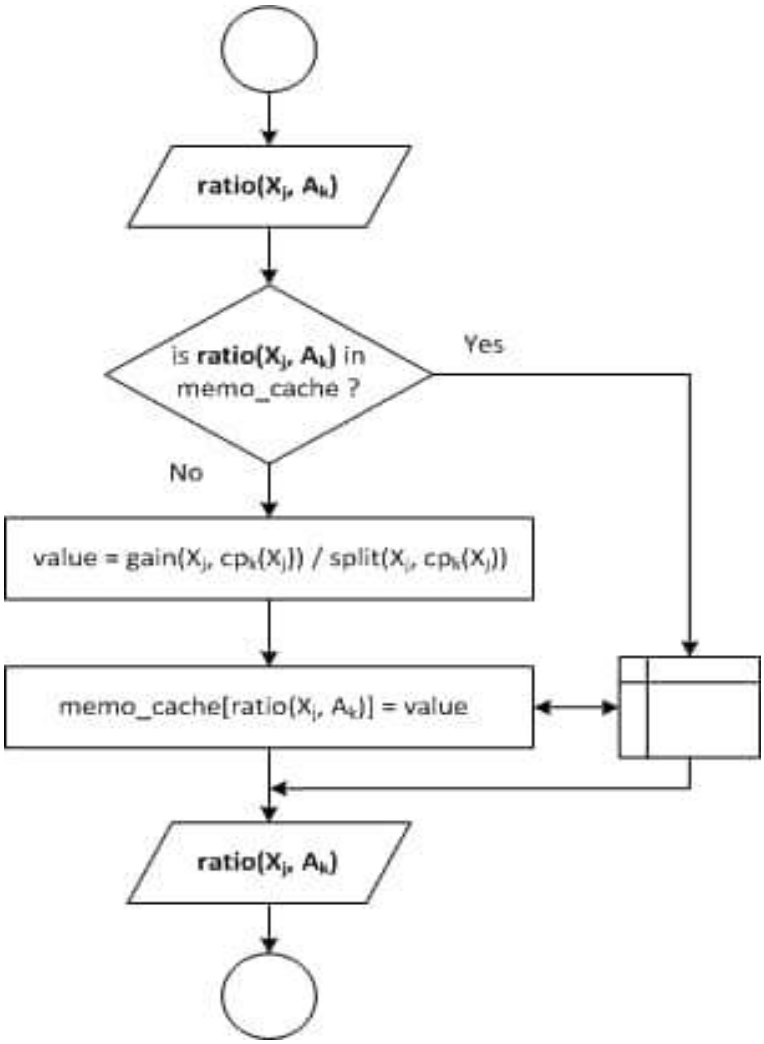

Figure 3.3: Memoized MapReduce Flowchart

3.7 Proposed Algorithm

The dataset is divided into several subsets by the Hadoop framework, and these subsets serves as input to the system.This is done in a key/value pair format, and for each of the attributes in m subsets, the algorithm checks if the value is numerical or not, before the check is done, the Memo_Cache function which will store the results of previous computations for the gain-ratio calculation is defined as shown in line 6 of algorithm 3.2.During the check, if the attribute has numerical values then they are sorted out first in ascending order and only the distinct values are retainedto select the candidate cut points. The information gain for the sorted values for each cut point in the m subsetsis calculated, and the cut point with the maximum gain is selected which is used to partition the subset.The split information is also calculated and the result of both computations is used to determine the gain ratio of each of the attributes. When the calculation is done for each of the attribute in m subset, the Memo_Cache function performs a check to determine whether the result is already saved or not, if it is not found, it calculates it and saves it by using an indexed function which makes accessing to the values stored in the hash table very fast.

In line 33 of the algorithm, which is the Select TaskReducer phase, the intermediate output of the map phase is sent as the input to the Reduce phase, where the sum of the information gain ratio of the same attribute in the subsets is calculated and the attribute corresponding to the maximum sum is chosen as the best splitting attribute. It is important to note that the difference between the C4.5 traditional algorithm and the MapReduce based algorithm is that the former computes the information gain ratio based on the whole dataset while the latter computes the information gain ratio based on the several subsets that are divided from the whole data set by the Hadoop framework.

Input: A training data X:

Output: A selected attribute A_k *and the cut points* $cp_{k^*}(X)$.

1: Initialise a Hadoop Job SELECTJOB:
2: Set SelectTaskMapper as the Mapper class
3: Set SelectTaskReducer as the Reducer class
4: Adjust the block size of HDFS until the dataset X can be split into m subsets $\{X_j\}_{j=1}^{m}$
5: **In the j − th SelectTaskMapper**:
Input: $X_j = (x_i)_{i=1}^{N}$, where x_i is the i − th instance with n attributes $(A_k)_{k=1}^{n}$
Output: $\langle key, value\rangle = \langle A_k, [Ratio_k(X_j, A_k), cp_k(X_j)]\rangle$
6: Memo_Cache = { }
7: **for** each attribute $A_k, k = 1,2, \ldots\ldots, n$ **do**
8: **if** A_k is numerical then
9: Sort its values $x_{1k}, \ldots\ldots, x_{Nk}$ and record as $x_{1k}^*, \ldots\ldots, x_{Nk}^*$
10: Find all cut points $cp_{ik} = \frac{x_{ik}^* + x_{i+1,k}^*}{2}, i = 1, \ldots\ldots, N-1$
11: **for** each cut points cp_{ik} **do**
12: Calculate the Information Gain:
13: $Gain(X_j, cp_{ik})$
$= Info(X_j) - \left[\frac{|x_{ij}^1|}{|x_j|} Info(X_{ij}^1) + \frac{|x_{ij}^2|}{|x_j|} Info(X_{ij}^2)\right]$ where
$X_{ij}^1 = \{x_i \in X_j | x_{ik} \le cp_{ik}\}, X_{ij}^2 = \{x_i \in X_j | x_{ik} > cp_{ik}\}$ and symbol |x| is size of x
14: **end**
15: Select the optimal cut point $cp_k(X_j) = cp_{i^*k}$ of A_k:
16: $i^* = \arg\max\{Gain(X_j, cp_{ik})\}_{i=1}^{N-1}$
17: Calculate the split information of $cp_k(X_j)$:
18: $Split(X_j, cp_k(X_j)) = -\left[\frac{|x_{i^*k}^1|}{|x_j|} \log_2 \frac{|x_{i^*k}^1|}{|x_j|} + \frac{|x_{i^*k}^2|}{|x_j|} \log_2 \frac{|x_{i^*k}^2|}{|x_j|}\right]$
19: **else**
20: Cut points: $cp_k = \bigcup_{i=1}^{C} \{x_{ik}^{'}\}$,
where $x_{ik}^{'} \in \{x_{1k}, \ldots, x_{Nk}\}$, C is the number of attribute values
21: Calculate the Information Gain:
22: $Gain(X_j) = Info(X_j) - \sum_{i=1}^{C} \frac{|x_j^i|}{|x_j|} Info(X_j^i)$ where X_j^i
$= \{x_i \in X | x_{ik} = x_{ik}^{'}\}$
23: Calculate the split information of cp_k
24: $Split(X_j, cp_k(X_j)) = -\sum_{i=1}^{C} \frac{|x_j^i|}{|x_j|} \log_2 \frac{|x_j^i|}{|x_j|}$
25: **end**
26: If Memo_Cache[A_k]not null
27: $Ratio(X_j, A_k) = Memo\_Cache[A_k]$
28: else
29: Calculate the gain ratio of A_k: $Ratio(X_j, A_k) = \frac{Gain(X_j, cp_k(X_j))}{Split(X_j, cp_k(X_j))}$

30: $\text{Memo\_Cache}[A_k] = \text{Ratio}(X_j, A_k)$
31:end
32: Mapper Output: $\langle \text{key}, \text{value} \rangle = \langle A_k, [\text{Ratio}_k(X_j, A_k), cp_k(X_j)] \rangle$
33: **end**
34: **In the SelectTaskReducer**:
Input: $\langle \text{key}, \text{value} \rangle = \langle A_k, \text{List}[\text{Ratio}_k(X_j, A_k), cp_k(X_j)] \rangle, j = 1,2, \dots, m$
Output: $\langle \text{key}, \text{value} \rangle = \langle A_{k^*}, [\text{Ratio}_{k^*}(X), cp_{k^*}(X)] \rangle$
35: **for** each attribute A_k **do**

36: $\text{Ratio}_k(X) = \sum_{j=1}^{m} \text{Ratio}_k(X_j, A_k)$
37: $cp_k(X) = \frac{\sum_{j=1}^{m} cp_k(X_j)}{m}$ (A_k is Numerical) or $cp_k(X) = U_{j=1}^{j=m} cp_k(X_j)$ (A_k is not Numerical)
38: **end**
39: Select the optimal index, where $k^* = \arg\max\{\text{Ratio}_k(X)\}_{k=1}^{n}$
40: **Reducer Output**: $\langle \text{key}, \text{value} \rangle = \langle A_{k^*}, [\text{Ratio}_{k^*}(X), cp_{k^*}(X)] \rangle$
41: **return** A_{k^*} and $cp_{k^*}(X)$.

Algorithm 3.1: Attribute Selection Algorithm (MR-A-S)

The **Input** at the beginning of the algorithm is the data stored in the Hadoop distributed file system (HDFS) where the file is broken down into blocks sized of 128MB and distributed across datanodes in the cluster.

The **Output** is the expected output after the best attribute which has the maximum gain ratio has been selected for splitting the dataset at each node for the construction of the decision tree.

In lines 1-4: During the job initialization, the job tracker creates an object to track the tasks and their progress. At this stage the map tasks for each input split are also created and the number of reduce tasks is also defined by the configuration mapred.reduce.tasks set by setNumReduceTasks method.

The Input in **Line 5** is the input to each individual mapper. When Hadoop submits a job, it splits the input data logically, this is also known as Input splits, and these splits are processed by each mapper. The InputFormat.getSplits() method is responsible for

generating the Input splits which are converted by the RecordReader into a key/value pair format which can be read by each mapper for processing. The number of mappers is equal to the number of Input splits created.

The relationship between the input at the beginning and the input in **line 5** is that the latter is a subset of the previous, which is m subsets as stated in **line 4**. The previous data is broken down into smaller chunks or pieces, and assigned to individual mappers for parallel processing. The output in **line 5** is the expected output from the map phase known as the intermediate output which is in a key/value pair format. It serves as Input to the reduce phase.

In line 6 the Memo_Cache is created for storing the results from gain ratio computation.

In Line 7, the **For** loop iterates on each of the attributes in the m subsets of the dataset, and line 8 checks if the values are numerical or nominal. If it is numerical (continuous values need to be converted to nominal values) then **Line 9** sorts out the values in ascending order that is, from the smallest to the highest.

Line 11 iterates on all values and separate dataset into two parts as instances less than or equal to current value, and instances greater than the current value. **Lines 12 and 13** calculates the gain for every step while **line 14** ends the loop. In Lines 15 and 16, the value which maximizes the gain or value with the highest gain is selected as the threshold. Lines 17 and 18 calculates the Split information of the optimal cut point. The Split information tries to measure how broadly and uniformly the attribute splits the data.

If the test is false then the statements in **lines 19 to 24** are executed. It calculates the information gain for each of the values of the attribute by taking the difference between the information needed to identify an element of m subsets and the information needed

to identify an element of m subsets after the value of attribute A_k has been obtained information.The Split information is also calculated.

The **end** in **line 25** terminates the **If then** statement.

In lines 26 to 30, the **If then else** statement checks the Memo_Cache for the result from the computation of gain ratio for each attribute. If the test is true then the result is fetched from the cache and returned, else if it is false then the gain ratio for each attribute is calculated and stored in the Memo_Cache. The **end** statement in **line 31** ends the **Ifthen** statement.

Line 32 is the intermediate result from each mapper which serves as input to the reducer after it has been shuffled and sorted out by key at the reduce phase and sent as input to the reducer. The **End** in **line 33** ends the **For** loop in line 7.

The input in **line 34** is the intermediate result from the mappers which has been shuffled and sorted by keys with a list of values associated with a particular key. While the output is the expected output from the reduce phase. The **For** loop in **line 35** iterates on each attribute which represents the key and does an aggregation or summation on the values associated to each key. The **end** in line 38 terminates the loop in line 35. **Line 39** selects the attribute with the maximum gain ratio that is used to split the current node on the decision tree. The output in **line 40 and 41** is the output from the reducer which has been reduced and is stored in the HDFS.

In line 6 the Memo_Cache is created for storing the results for gain ratio computation. **In lines 26 to 27** if the Memo_Cache is not empty then result is returned otherwise it is computed and stored in the cache. The Memo_Cache is updated in **line 30**. This

improvement is meant to save execution time, because each time same arguments in the computation of gain ratio are encountered, the stored result is fetched and reused and re-computation is avoided.

CHAPTER FOUR

4.0 IMPLEMENTATION, RESULTAND ANALYSIS

4.1 Introduction

Chapter four discusses the implementation details of the enhanced algorithm and the results of implementation. The system requirements of the model is as follows;

a. Host System

1. Windows 10 Pro Operating system

2. Intel Core i5-4210U CPU

3. 8.00 GB RAM

4. 64 bit processor pc or laptop (minimum operating frequency of 2.4GHz)

b. Virtual System

1. 6 GB RAM
2. 20.13 GB Hard Disk
3. Ubuntu 16.04 LTS (64-bit)Operating system
4. Netbeans IDE 8.1
5. Hadoop Framework Installation setup

4.2 Implementation details

a. **Programming Language**

The proposed model was implemented using the Java programming, which is an object oriented programming language which makes the creation of modular programs and reusable codes possible.Its platform-independence which is its ability to run the same program on many different computers has made it a preferred programming language

compared to others. Java has a common syntax as C and C++ because it inherits its syntax from it, its object model is adapted from the C++. In its response to the online environment, it supports distributed systems by providing features that streamlines programming for a highly distributed architecture.

4.3 Implementation Result

The proposed model was compared with the existing system with the execution time used as the performance metric for evaluating the result on different sizes of datasets.

4.4 Discussion of results

In order to evaluate the performance of the proposed algorithm a set of experiments were conducted using different sizes of datasets to measure the time it takes to run the proposed algorithm and that of the existing algorithm. Figure 4.1a, b and cshows the MapReduce log file statistics at different stages of processing.

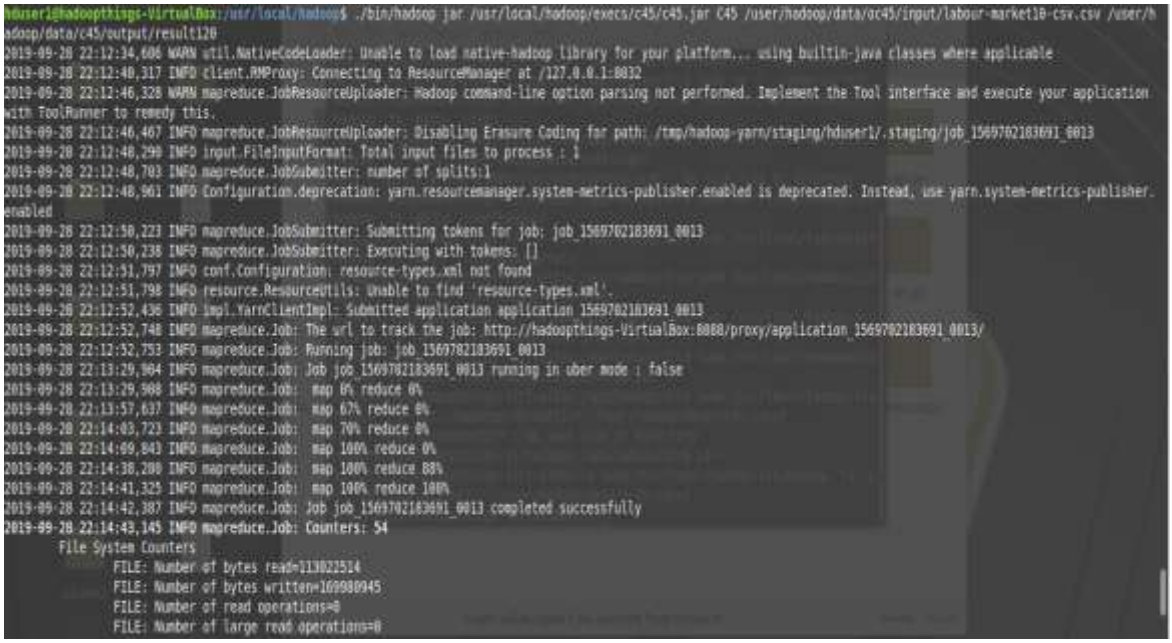

Figure 4.1a: Image of a Memoized MapReduce Process

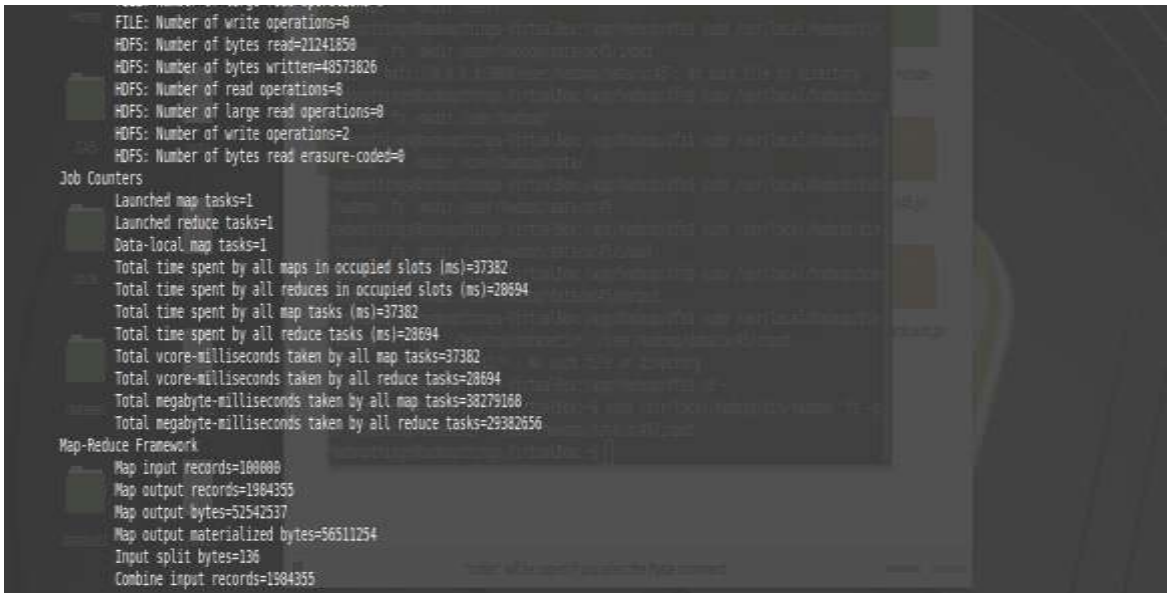

Figure 4.1b: Image of Memoized MapReduce Process

Figure 4.1c: Image of Memoized MapReduce Process

Table 4.1 shows the execution time in seconds (sec.) and Figure 4.2 shows the corresponding graph.

Table 4.1: Comparing execution time between the proposed system and the existing system.

S/No	Datasets	Time taken without Memoization (sec)	Time taken with Memoization (sec)	Percentage decrease in time
1.	Diabetes	4.02	4.22	- 4.7%
2.	Worldcups	3.31	1.6	51.66%
3.	Dermatology.csv	4.27	4.06	4.92%
4.	breast cancer	3.84	3.82	0.52%
5.	nba_players_all	5.66	5.26	7.07%
6.	Nba_betting_Totals	13.44	13.66	- 1.61%
7.	Nba_betting_money_line	13.79	12.4	10.08%
8.	Nba_betting_spread	13.87	14.24	- 2.66%
9.	Worldcupplayers	11.41	5.16	54.78%
10.	Ici_dec_18qtr	9.94	8.67	12.7%
11.	Ques-dec 18qtrs	19.99	19.26	3.65%
	Total	103.54	92.35	10.80%

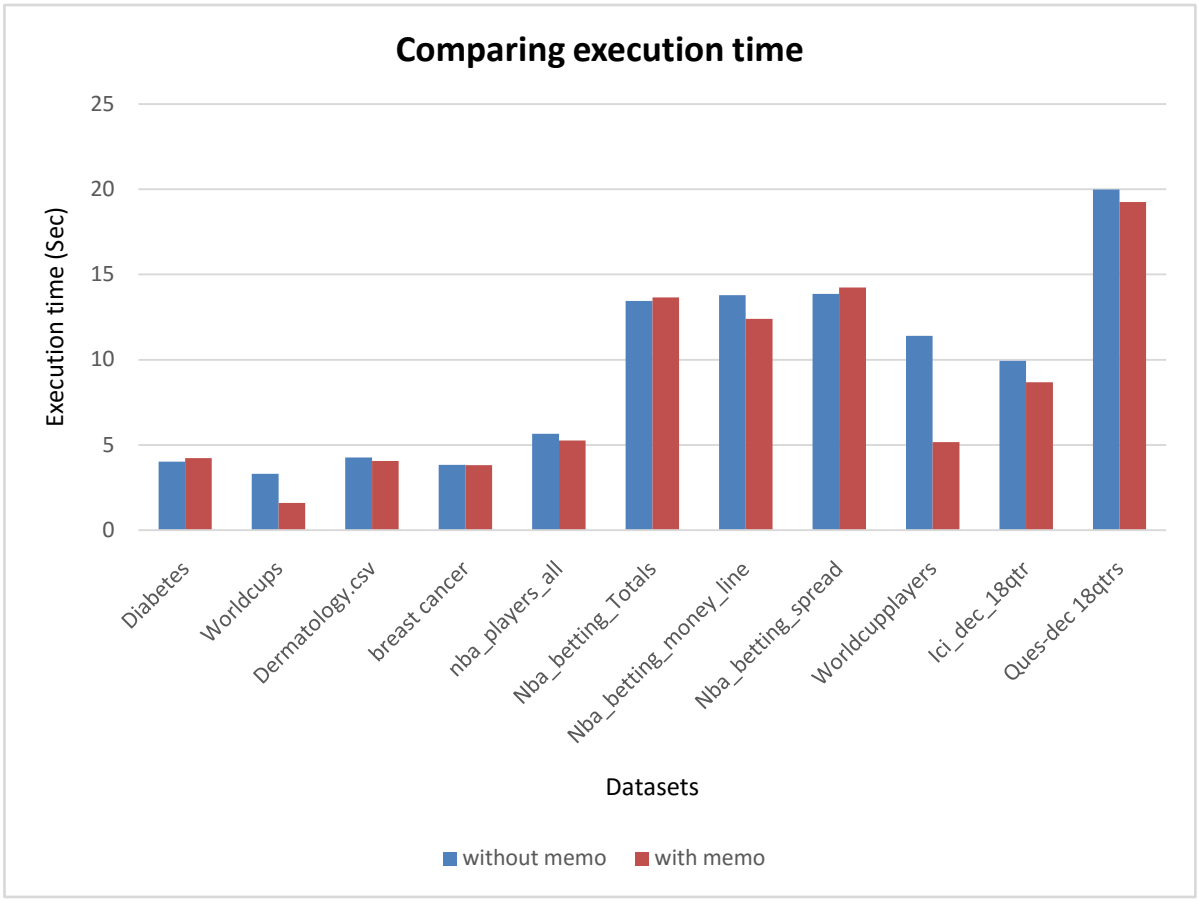

Figure 4.2: Comparing execution time

The result in table 4.1 shows the execution time (in seconds) of the existing system and the proposed system when applied on 12 datasets using a hadoop Mapreduce framework on a single node cluster environment..When compared with each other it was observed that the proposed system performed better than the existing system except on three datasets where the results were negative - 4.7 %, - 1.66 % and - 2.6 % indicating a percentage increase in the execution time of the proposed system.The graphical representation of the results is shown in Figure 4.2, with y axis showing the execution time in seconds and the x axis the different datasets. The red colour indicates the time taken with memoization, while the blue colour depicts the time taken without memoization

4.5 Summary of the Result

The comparison of the result showed that the proposed algorithm performed better in some cases though not all. The ability of memoization technique to store results of computation and make them available for reuse when the need arises without wasting time to re-compute them, has helped to improve the C4.5 decision tree algorithm.

CHAPTER FIVE

5.0 SUMMARY, CONCLUSION AND RECOMMENDATION

5.1 Summary

The C4.5 decision tree algorithm, though commonly used in classification, has a problem with the amount of time it takes in processing large datasets but the existing systemsolved this problem and improved the algorithm through the application of the MapReduce programming model. The proposed system applied memoization to the existing system to further improve it by reducing the execution time through the reuse of previously stored results when the need arises. Datasets were collected from online open sources and processed using the existing algorithm and the proposed algorithm on a single node cluster system. Evaluation of the proposed algorithm was done based on the execution time which was used as the performance metric.

5.2 Conclusion

The C4.5 decision tree algorithm is an algorithm which is widely used for classication and prediction in different applications, but its limitation in handling large datasets which results in high execution time has been one of the bottlenecks.Parallelising the C4.5 algorithm through the use of a MapReduce programming model has improved the algorithm by reducing the execution time.This research enhanced the existing system by further reducing the execution time by applying the memoization technique.The enhancement exhibited an average of 10.80% decrease in execution time when compared with the time taken by the existing system.

5.3 Recommendation

The future work should consider deploying this work on a multinode cluster for better performance when larger datasets are used.

REFERENCES

Adhatrao K., G. A. (2013). Predicting Students' Performance using ID3 and C4.5 Classification Algorithms. *International Journal of Data Mining & Knowledge Management Process, 3*(5), 39-52.

Albashrawi, M. (2016). Detecting Financial Fraud using Data Mining Techniques: A Decade Review from 2004 to 2015. *Journal of Data Science, 14*(2016), 553-570.

Al-Shayea, Q. K. (2011). Artificial Neural Networks in Medical Diagnoses. *International Journal of Computer Science Issues, 8*(2), 150-154.

Badgujar, G., & Sawant, K. (2017). Improved C4.5 Decision Tree Classifier Algorithm for Analysis of Data Mining Application. *International Journal for Research in Engineering Application & Management, 2*(10), 18-24.

Becklas, A. (2018). FIFA World Cup. *Kaggle Repository*. Kaggle Inc. Retrieved October 17, 2019, from https://www.kaggle.com/abecklas/fifa-world-cup

Bose, I., & Mahapatra, R. K. (2001). Business data mining - a machine learning perspective. *Information & Management, 39*(3), 211-225.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone , C. J. (1984). *Classification and Regression Trees.* Monterey: Brooks/Cole Publishing.

Cao, J., Cui, H., Shi, H., & Jiao, L. (2016). Big Data: A Parallel Particle Swarm Optimization-Back-Propagation Neural Network algorithm Based on MapReduce. *PLOS ONE, 11*(6), 1-17. doi:10.1371/journal.pone.0157551

Chandgude, N., & Pawar, S. (2015). A survey on diagnoses of diabetes using various classification algorithm. *International Journal on Recent Trends in Computing and Communication, 3*(12), 6706-6710.

Chang, C., & Lin, C. (2011). LIBSVM: A Library for Support Vector Machines. *ACM Transactions on Intelligent Systems and Technology, 2*(3), 1-27.

Cherfi, A., Nouira, K., & Ferchichi, A. (2018). Very Fast C4.5 Decision Tree Algorithm. *Applied Artificial Intelligence, 32*(2), 119–137. doi:10.1080/08839514.2018.1447479

Chu, F., Jin , G., & Wang, L. (2005). Cancer Diagnoses and Protein Secondary Structure Prediction Using Support Vector Machines: Tho. In L. Wang (Ed.), *Studies in Fuzziness and Soft Computing. Support Vector Machines: Theory and Applications.* (Vol. 177, pp. 343-363). Heidelberg, Berlin: Springer. doi:10.1007/10984697_16

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning, 20*(3), 273-297. doi:10.1007/BF00994018

Dai, W., & Ji, W. (2014). A MapReduce Implementation of C4.5 Decision Tree Algorithm. *International Journal of Database Theory Application, 7*(1), 49-60.

De Mántaras, L. R. (1991). A Distance-Based Attribute Selection Measure for Decision Tree Induction. *Machine Learning, 6*(1), 81-92. doi:10.1023/A:10226940

Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. *Proceedings of the 6th conference on Symposium on Opearting Systems Design & Implementation -, 6*, pp. 137-150. San Francisco, CA.

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Communications of the ACM, 51*(1), 107-113. doi:10.1145/1327452.1327492

Dhaenens, C., & Jourdan, L. (2016). *Metaheuristics for Big Data* (Vol. 5). Hoboken, NJ, USA: John Wiley and Sons Inc. doi:10.1002/9781119347569

El Kourdi, M., Bensaid, A., & Rachidi, T. (2004). Automatic Arabic document categorization based on the Naïve Bayes algorithm. *Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages*, (pp. 51-58). Geneva, Switzerland.

Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From Data mining to Knowledge Discovery in Databases. *AI Magazine, 17*(3), 37-54.

Frawley, W. J., Piatetsky-Shapiro, G., & Matheus, C. J. (1992). Knowledge discovery in databases: an overview. *AI Magazine*, *13*(3), 57-70.

Hallmark, E. (2018). NBA Historical Stats and Betting Data. *Kaggle Repository*. Kaggle Inc. Retrieved August 13, 2019, from https://www.kaggle.com/ehallmar/nba-historical-stats-and-betting-data

Hamoud, A. K., Hashim, A. S., & Awadh, W. A. (2018). Predicting Student Performance in Higher Education Institutions Using Decision Tree Analysis. *International Journal of Interactive Multimedia and Artificial Intelligence, 5*(2), 26-31. doi:10.9781/ijimai.2018.02.004

Han, J., & Kamber, M. (2006). *Data Mining: Concepts and Techniques* (2nd ed.). San Francisco, CA, USA: Morgan Kaufmann.

Han, J., Kamber, M., & Pei, J. (2012). *Data Mining Concepts and Techniques* (3rd ed.). Waltham, MA, USA: Morgan Kaufmann.

Hand, D., Mannila, H., & Smyth, P. (2001). *Principles of Data Mining.* Cambridge, MA: The MIT Press.

Hiremath, R., & Patil, P. (2016). A Study - Knowledge Discovery Approaches and it's Impact with reference to Cognitive Internet of Things (CIOT). *International Journal of Information Sciences and Techniques, 6*(1), 247-256. doi:10.5121/ijist.2016.6227

Jaffar, J., Santosa, A. E., & Voicu, R. (2008). Efficient Memoization for Dynamic Programming with Ad-Hoc Constraints. *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, (pp. 297-303). Chicago, USA.

Jaseena , K. U., & David, J. M. (2014). Issues, Challenges and Solutions : Big Data Mining. *Sixth International Conference on Networks & Communications*, (pp. 131–140). doi:10.5121/csit.2014.41311

Jiawei , H., & Micheline , K. (2006). *Data Mining: Concepts and Techniques.* San Francisco: Elsevier Inc.

Kahn, M. (n.d.). Diabetes Data Set. *UCL Machine Learning Repository*. St. Louis: Center for Machine Learning and Intelligent Systems. Retrieved August 13, 2019, from https://archive.ics.uci.edu/ml/datasets/Diabetes

Kamthania, D., Pahwa, A., & Madhavan, S. S. (2018). Market Segmentation Analysis and Visualization Using K-Mode Clustering Algorithm for E-Commerce Business. *Journal of Computing and Information Technology, 26*(1), 57-68. doi:10.20532/cit.2018.1003863

Khoshgoftaar, T. M., Allen, E. B., & Deng, J. (2002). Using regression trees to classify fault-prone software modules. *IEEE Transactions on Reliability, 51*(4), 455 - 462. doi:10.1109/TR.2002.804488

Kumbhare, T. A., & Chobe , S. V. (2014). An Overview of Association Rule Mining Algorithms. *International Journal of Computer Science and Information Technologies, 5*(1), 927-930.

Lavrenco, V., & Sutton, C. (2009). IAML: Decision Trees. *Lecture notes of School of Informatics*. University of Edinburg, UK.

Lin, F., Wu, N., & Tsay, T. (2017). Applications of Cluster Analysis and Pattern Recognition for Typhoon Hourly Rainfall Forecast. *Advances in Meteorology, 2017*(2), 1-17. doi:10.1155/2017/5019646

Lippman, R. P. (1988). Neural Network Classifiers for Speech Recognition. *The Lincoln Laboratory Journal, 1*(1), 107-124.

Ilter, N., & Guvenir, H. A. (1998, January 1). Dermatology Data Set. *UCI Machine Learning Repository*. Ankara, Turkey: Center for Machine Learning and Intelligent Systems. Retrieved August 13, 2019, from https://archive.ics.uci.edu/ml/datasets/Dermatology

Mahmoud, H., Hegazy, A., & Khafagy, M. H. (2018). An approach for big data security based on Hadoop distributed file system. *International Conference on Innovative Trends in Computer Engineering* (pp. 109-114). Aswan, Egypt: IEEE. doi:10.1109/ITCE.2018.8316608

Majumdar, J., Naraseeyappa, S., & Ankalaki, S. (2017). Analysis of agriculture data using data mining techniques: application of big data. *Journal of Big Data, 4*(1), 1-15. doi:10.1186/s40537-017-0077-4

Mansour, A. M. (2018). Texture Classification using Naïve Bayes Classifier. *International Journal of Computer Science and Network Security, 18*(1), 112-120.

Michie, D. (1968). "Memo" Functions and Machine Learning. *Nature Publishing Group, 218*, 19-22. doi:10.1038/218306c0

Mu, Y., Liu, X., Yang, Z., & Liu, X. (2017). A parallel C4.5 decision tree algorithm based on MapReduce. Concurrency and Computation. *Concurrency and Computation: Practice and Experience, 29*(8), 1-12. doi:10.1002/cpe.4015

Muslim, M. A., Rukmana, S. H., Sugiharti, E., Prasetiyo, B., & Alimah, S. (2018). Optimization of C4.5 algorithm-based particle swarm optimization for breast cancer diagnosis. *Journal of Physics: Conf. Series, 983*(2018), 1-8. doi:10.1088/1742- 6596/983/1/012063.

Neelamegam, S., & Ramaraj, E. (2013). Classification algorithm in Data mining: An Overview. *International Journal of P2P Network Trends and Technology, 3*(5), 1-5.

Norvig, P. (1991). Techniques for automatic memoization with applications to context-free parsing. *Computational Linguistics, 17*(1), 91-98.

Piatetsky-Shapiro, G. (1991). Knowledge Discovery in Real Databases. *AI Magazine, 11*(5), 68-70.

Podgorelec, V., KoKol, P., Stiglie, B., & Rozman, I. (2002). Decision trees: an overview and their use in medicine. *Journal of Medical Systems, 26*(5), 445-463.

Quinlan, J. (1993). *C4.5: Programs for machine learning* (Vol. 16). San-Mateo, CA: Morgan Kaufman.

Rajesh, K., & Sangeetha. (2012). Application of Data mining Methods and Techniques for Diabetes Diagnosis. *International Journal of Engineering and Innovative Technology, 2*(3), 224-229.

Rajeshinigo, D., & Jebamalar, J. P. (2017). Accuracy Improvement of C4.5 using K Means Clustering. *International Journal of Science and Research, 6*(6), 2755-2758.

Rokach, L., & Maimon, O. (2014). *Data mining with decision trees: Theory and applications* (2nd ed., Vol. 81). Singapor: World Scientific Publishing Company.

Rui , X., & Donald , C. W. (2005). Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks, 16*(3), 645 - 678. doi:10.1109/TNN.2005.845141

Saleh, I., & Knieper, T. (2017). *The Visual Politics of Wars.* Newcastle, UK: Cambridge Scholars Publishing.

Sathiyapriya, S., & Kanagaraj, A. (2018). Basics of Data Mining Techniques and it's Application. *International Journal of Computing and Technology, 5*(4), 44-47.

Seema, S., Agrawal, J., & Sharma, S. (2013). Classification through Machine Learning. *International Journal of Computer Applications, 82*(16), 20-27.

Shahid, N., Rappon, T., & Berta, W. (2019). Applications of artificial neural networks in health care organizational decision-making: A scoping review. *PLOS ONE, 14*(2), 1-22. doi:10.1371/journal.pone.0212356.

Sharma, H., & Kumar, S. (2016). A Survey on Decision Tree Algorithms of Classification in Data Mining. *International Journal of Sciences and Research, 5*(4), 2094-2097.

Siddesh, G. M., Suresh, K., Madhuri, K. Y., Nijagal, M., Rakshitha, B. R., & Srinivasa, K. G. (2016). Optimizing Crawler4j using MapReduce Programming Model. *Journal of the Institution of Engineers, 98*(3), 329–336. doi:10.1007/s40031-016-0267-z

Sniedovich, M., & Lew, A. (2006). Dynamic Programming : an overview. *Control and Cybernatics, 35*(3), 513-533.

Song, J., Kim, K., & Lee, M. (2018). A Big Data Analysis and Mining Approach for IoT Big Data. *International Journal of Advances in Computer Science and Technology, 7*(1), 1-3. doi:10.30534/ijacst/2018/01712018

Subeesh, V., Maheswari, E., Saraswathy, G. R., Swaroop, A. M., & Minnikanti, S. S. (2028). A comparative Study of Data Mining Algorithms used for Signal Detection in FDA AERS Databases. *Journal of Young Pharmacists, 10*(4), 444-449.

Suresh, A., Swamy, B. N., Rohou, E., & Seznec, A. (2015). Intercepting Functions for Memoization: A Case Study Using Transcendental Functions. *ACM Transsactions on Architecture and Code Optimization, 12*(2), 18:1-18:23.

Turban, E., Aronson, J. E., Liang, T., & Sharda, R. (2007). *Decision Support and Business Intelligence Systems* (8th ed.). NJ: Prentice-Hall, Inc.

Utgoff, P., & Brodley, C. (1990). An Incremental Method for finding Multivariate Splits for Decision Trees. *Proceedings of the seventh International Conference on Mchine Learning* (pp. 58-65). Austin, Texas: Morgan Kaufmann.

Vladan, D. (2001). *Knowledge Discovery and Data Mining in Databases. Handbook of Software Engineering and Knowledge Engineering.* China: World Scientific.

Wang, B., Huang, S., Qiu, J., Liu, Y., & Wang, G. (2014). Parallel online sequential extreme learning machine based on MapReduce. *Neurocomputing, 149*, 224-232.

Wang, J., Chen, X., Zhou, K., Wang, W., & Zhang, D. (2008). Application of Spatial Data Mining in Accident Analysis System. *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing* (pp. 472-475). Shanghai, China: IEEE. doi:10.1109/ETTandGRS.2008.253

Wang, J., Liu, J., Li, T., Yin, S., & He, X. (2018). The Research of Regression Method for Forecasting Monthly Electricity Sales Considering Coupled Multi-factor. *IOP Conference Series: Earth and Environmental Science.108*, pp. 1-5. IOP Publishing Ltd. doi:10.1088/1755-1315/108/5/052105

Wang, Y., Makedon, F. S., Ford, J. C., & Pearlman, J. (2005). HykGene: A hybrid approach for selecting marker genes for phenotype classification using microarray gene expression data. *Bioinformatics, 21*(8), 5030-5037. doi:10.1093/bioinformatics/bti192

Xu, R., & Donald. (2005). Survey of Clustering algorithms. *IEEE transactions on Neural Networks, 16*, 645-678.

Yue, D., Wu, X., Wang, Y., Li , Y., & Chu, C. (2007). A Review of Data Mining-Based Financial Fraud Detection Research. *International Conference on Wireless Communications, Networking and Mobile Computing* (pp. 5519-5522). Shanghai, China: IEEE. doi:10.1109/WICOM.2007.1352

Zhao, Q., & Fränti, P. (2014). WB-index: A sum-of-squares based index for cluster validity. *Data & Knowledge Engineering, 92*(2014), 77–89.

Zhu, F., Tang, M., Xie, L., & Zhu, H. (2018). A Classification Algorithm of CART Decision Tree based on MapReduce Attribute Weights. *International Journal of Performability Engineering, 14*(1), 17-25. doi:10.23940/ijpe.18.01.p3.1725

Zwitter, M., & Soklic, M. (1988, July 11). Breast Cancer Data Set. *UCI Machine Learning Repository*. Ljubljana, Yugoslavia: Center for Machine Learning and Intelligent Systems. Retrieved August 13, 2019, from http://mlr.cs.umass.edu/ml/datasets/Breast+Cancer

APPENDICES

Appendix A: Memoised MapReduce SourceCodes

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
//import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapreduce.Job; //.JobConf;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.mapreduce.Job;

import java.io.IOException;
import java.util.Arrays;
import java.util.StringTokenizer;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import java.util.function.*;
import java.util.*;
//import mapreduceframework.Memoizer;


public class C45 extends Configured implements Tool {


    public static Split currentsplit=new Split();

    public static List <Split> splitted=new ArrayList<Split>();;

    public static int current_index=0;

  public static void main(String[] args) throws Exception {

        MapClass mp=new MapClass();
        splitted.add(currentsplit);


        int res=0;
        double bestGain=0;
```

```
        boolean stop=true;
        boolean outerStop=true;
        int split_index=0;
        double gainratio=0;
        double best_gainratio=0;
        double entropy=0;
        String classLabel=null;
        int total_attributes=mp.no_Attr;
        total_attributes=4;
        int split_size=splitted.size();
        GainRatio gainObj;
        Split newnode;
       //
        while(split_size>current_index)
         {
        currentsplit=(Split) splitted.get(current_index);
        gainObj=new GainRatio();


          res = ToolRunner.run(new Configuration(), new C45(),
args);
          System.out.println("Current  NODE INDEX .
::"+current_index);
          int j=0;
          int temp_size;
          gainObj.getcount();
          entropy=gainObj.currNodeEntophy();
          classLabel=gainObj.majorityLabel();
          currentsplit.classLabel=classLabel;
         if(entropy!=0.0 &&
currentsplit.attr_index.size()!=total_attributes){
           System.out.println("");
           System.out.println("Entropy  NOTT zero   SPLIT
INDEX::    "+entropy);
           best_gainratio=0;
          for(j=0;j<total_attributes;j++){       //Finding the
gain of each attribute
             if(currentsplit.attr_index.contains(j)){  //
Splitting all ready done with this attribute
                  // System.out.println("Splitting all ready
done with  index  "+j);
                }
                else{
                     gainratio=gainObj.gainratio(j,entropy);

                  if(gainratio>=best_gainratio)
                  {
                      split_index=j;

                      best_gainratio=gainratio;
                   }

           }
          }
```

```
          String
attr_values_split=gainObj.getvalues(split_index);
          StringTokenizer attrs = new
StringTokenizer(attr_values_split);
          int number_splits=attrs.countTokens(); //number of
splits possible with  attribute selected
          String red="";
          int tred=-1;
          System.out.println(" INDEX ::  "+split_index);
          System.out.println(" SPLITTING VALUES
"+attr_values_split);
          for(int
splitnumber=1;splitnumber<=number_splits;splitnumber++)
          {
            temp_size=currentsplit.attr_index.size();
            newnode=new Split();
            for(int y=0;y<temp_size;y++)   // CLONING OBJECT
CURRENT NODE
            {

      newnode.attr_index.add(currentsplit.attr_index.get(y));

      newnode.attr_value.add(currentsplit.attr_value.get(y));
            }
            red=attrs.nextToken();
            newnode.attr_index.add(split_index);
            newnode.attr_value.add(red);
            splitted.add(newnode);
          }
          }
          else
          {
            System.out.println("");
            String rule="";
            temp_size=currentsplit.attr_index.size();
            for(int val=0;val<temp_size;val++)
            {
            rule=rule+" "+currentsplit.attr_index.get(val)+"
"+currentsplit.attr_value.get(val);
            }
            rule=rule+" "+currentsplit.classLabel;
            writeRuleToFile(rule);
            if(entropy!=0.0)
                  System.out.println("Enter rule in file::
"+rule);
            else
            System.out.println("Enter rule in file Entropy zero
::   "+rule);


          }

          split_size=splitted.size();
          System.out.println("TOTAL NODES::    "+split_size);
```

```
          current_index++;

        }

       System.out.println("COMPLEEEEEEEETEEEEEEEEE");
           System.exit(res);

  }
  public static void writeRuleToFile(String text) {
          try {
           BufferedWriter bw = new BufferedWriter(new
FileWriter(new File("/home/hduser1/c45/rule.txt/"), true));
           bw.write(text);
                 bw.newLine();
                 bw.close();
          } catch (Exception e) {
          }
   }

  public int run(String[] args) throws Exception {

          Configuration conf = new Configuration();
        //conf.set("yarn.resourcemanager.address",
"192.168.33.75:50001"); // see step 3
          //conf.set("mapreduce.framework.name", "yarn");
          //conf.set("fs.defaultFS", "hdfs://0.0.0.0:9000"); //
see step 2
          //conf.set("mapreduce.app-submission.cross-platform",
"true");
          //conf.set("mapred.remote.os", "Linux");

          conf.set("yarn.application.classpath",

"{{HADOOP_CONF_DIR}},{{HADOOP_COMMON_HOME}}/share/hadoop/common/
*,{{HADOOP_COMMON_HOME}}/share/hadoop/common/lib/*,"
                  + "
{{HADOOP_HDFS_HOME}}/share/hadoop/hdfs/*,{{HADOOP_HDFS_HOME}}/sh
are/hadoop/hdfs/lib/*,"
                  +
"{{HADOOP_MAPRED_HOME}}/share/hadoop/mapreduce/*,{{HADOOP_MAPRED
_HOME}}/share/hadoop/mapreduce/lib/*,"
                  +
"{{HADOOP_YARN_HOME}}/share/hadoop/yarn/*,{{HADOOP_YARN_HOME}}/s
hare/hadoop/yarn/lib/*");

          Job job = new Job(conf, "c45");
          job.setJarByClass(C45.class);


          // Set the outputs for the Map
          job.setMapOutputKeyClass(Text.class);
          job.setMapOutputValueClass(IntWritable.class);
          job.setOutputKeyClass(Text.class);
          job.setOutputValueClass(Text.class);
```

```
        job.setMapperClass(MapClass.class);
        job.setCombinerClass(Reduce.class);
        job.setReducerClass(Reduce.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setNumReduceTasks(1);

      FileInputFormat .setInputPaths(job, new Path(args[0]));
      FileOutputFormat.setOutputPath(job, new Path(args[1]));

      boolean success = job.waitForCompletion(true);
      System.out.println(success);

      return 0;

  }


}
```

Appendix B: C4.5 Algorithm Description (Mu *et al.*, 2017)

Input: A root Node X

$= (x_i)_{i=1}^{N}$, where x_i is the $i-$ th instance with n attributes $(A_k)_{k=1}^{N}$

Output: A $C4.5-$ Tree.

1: Ω is initialised as an empty set

2: Add X to Ω

3: **while** Ω is not empty **do**

4: **for** each attribute $A_k, k = 1,2, \ldots\ldots, n$ **do**

5: **if** A_k is numerical then

6: Sort its values $x_{1k}, \ldots\ldots, x_{Nk}$ and record as $x_{1k}^*, \ldots\ldots, x_{Nk}^*$

7: Find all cut points $cp_{ik} = \frac{x_{ik}^* + x_{i+1,k}^*}{2}$, $i = 1, \ldots\ldots, N-1$

8: **for** each cut points cp_{ik} **do**

9: Calculate the Information Gain:

10: $$Gain(cp_{ik}) = Info(X) - \left[\frac{|x_i^1|}{|x|} Info(X_i^1) + \frac{|x_i^2|}{|x|} Info(X_i^2)\right]$$

where $X_i^1 = \{x_i \in X | x_{ik} \leq cp_{ik}\}, X_i^2 = \{x_i \in X | x_{ik} > cp_{ik}\}$ and symbol $|x|$ is size of x

11: **end**

12: Select the optimal cut point $cp_k = cp_{i^*k}$ of A_k , where $i^* = \arg\max\{Gain(cp_{ik})\}_{i=1}^{N-1}$

13: Calculate the split information of cp_k

14: $$Split(cp_k) = -\left[\frac{|x_{i^*k}^1|}{|x|}\log_2\frac{|x_{i^*k}^1|}{|x|} + \frac{|x_{i^*k}^2|}{|x|}\log_2\frac{|x_{i^*k}^2|}{|x|}\right]$$

15: **else**

16: Cut points: $cp_k = \bigcup_{i=1}^{C}\{x_{ik}^{'}\}$, where $x_{ik}^{'} \in \{x_{1k}, \ldots, x_{Nk}\}$, C is the number of attribute values

17: Calculate the Information Gain:

18: $$Gain(cp_k) = Info(X) - \sum_{i=1}^{C}\frac{|x^i|}{|x|} Info(X^i) \quad \text{where } X^i = \{x_i \in X | x_{ik} = x_{ik}^{'}\}$$

19: Calculate the split information of cp_k

20: $$Split(cp_k) = -\sum_{i=1}^{C}\frac{|x^i|}{|x|}\log_2\frac{|x^i|}{|x|}$$

21: **end**

22: Calculate the gain ratio of A_k: $Ratio(A_k) = \frac{Gain(cp_k)}{Split(cp_k)}$

23: **end**

24: Get the best attribute A_{k^*} and cut points cp_{k^*}, where $k^* = \arg\max\{Gain(A_k)\}_{k=1}^{n}$

25: Split X into m subsets $\{X_i\}_{i=1}^{m}$ based on attribute A_{k^*} and cut points cp_{k^*}

26: Remove X and add $\{X_i\}_{i=1}^{m}$ to Ω

27: **end**

In equation 2.1, the symbol $Info(\mathrm{X})$ is the information entropy which reflects the class impurity and the amount of information in Node X.

$$Info(\mathrm{X}) = -\sum_{i=1}^{m} p_i(\mathrm{X}) \times \log_2(p_i(\mathrm{X})) \tag{2.1}$$

where *m* represent the class number of X and $p_i(\mathrm{X})$ is the i^{th} - class probability in X.

In line 5, when considering a continuous attribute $A_k = (x_{1k}, \ldots\ldots, x_{Nk})$, the values should be sorted in ascending order first, and only distinct values are retained to select the candidate cut points. From the algorithm, the sorted values are recorded as $x^*_{1k}, \ldots\ldots, x^*_{Nk}$. The C4.5 algorithm selects the optimal cut point cp_{ik} that maximizes information gain to divide samples in X into two subsets:

$$\text{Find all cut points } cp_{ik} = \frac{x^*_{ik} + x^*_{i+1,k}}{2}, i = 1, \ldots., N-1 \tag{2.2}$$

For each cp_{ik}, samples in X are split into two intervals $X_i^1 = \{x_i \in X | x_{ik} \leq cp_{ik}\}$ and $X_i^2 = \{x_i \in X | x_{ik} > cp_{ik}\}$ to compute the information gain for all candidate cut points shown Equation (2.3).

Information Gain (IG) measures the information that is gained by partitioning X in accordance with the test based on attribute A_k. This represents the difference between the information needed to identify an element of X and the information needed to identify an element of X after the value of attribute A_k has been obtained.

$$\mathrm{Gain}(cp_{ik}) = \mathrm{Info}(\mathrm{X}) - \left[\frac{|x_i^1|}{|x|}\ \mathrm{Info}(X_i^1) + \frac{|x_i^2|}{|x|}\ \mathrm{Info}(X_i^2)\right] \tag{2.3}$$

where $X_i^1 = \{x_i \in X | x_{ik} \leq cp_{ik}\}, X_i^2 = \{x_i \in X | x_{ik} > cp_{ik}\}$ and symbol $|x|$ is size of x

Where the information entropy $\mathrm{Info}(X)$ measures the class impurity and the amount of information as mentioned in Eqn. 2.1.

Then, the optimal cut point is selected as a threshold value for attribute A_k in Equation 2.4

$$\text{Select the optimal cut point } cp_k = cp_{i^*k} \text{ of } A_k\,, \tag{2.4}$$

$$\text{where } i^* = \arg\max\{\mathrm{Gain}(cp_{ik})\}_{i=1}^{N-1}$$

Once threshold value is selected, the algorithm calculates the gain ratio to compare the distinctive ability of the candidate attributes and select the optimal one in order to split the current node.$\mathrm{GainRatio}(A_k)$ is the proportion of information generated by the split that is useful for classification.

$$\text{Gain ratio of } A_k\text{: } \mathrm{Ratio}(A_k) = \frac{\mathrm{Gain}(cp_k)}{\mathrm{Split}(cp_k)} \tag{2.5}$$

Where

$$\mathrm{Split}(cp_k) = -\sum_{i=1}^{c} \frac{|x^i|}{|x|} \log_2 \frac{|x^i|}{|x|} \tag{2.6}$$

The split information value represents the potential information generated by splitting the training data set X into c partitions, corresponding to c outcomes on attributeA_k.

Appendix C: Attribute Selection Algorithm (MR-A-S) (Mu *et al.*, 2017)

Input: A training data X：
Output: A selected attribute A_k *and the cut points* $cp_{k^*}(X)$.
1: Initialise a Hadoop Job SELECTJOB:
2: Set SelectTaskMapper as the Mapper class
3: Set SelectTaskReducer as the Reducer class
4: Adjust the block size of HDFS until the dataset X can be split into m subsets $\{X_j\}_{j=1}^{m}$
5: **In the j − th SelectTaskMapper**:
Input: $X_j = (x_i)_{i=1}^{N}$, where x_i is the i − th instance with n attributes $(A_k)_{k=1}^{n}$
Output: ⟨key, value⟩ = $\langle A_k, [Ratio_k(X_j, A_k), cp_k(X_j)]\rangle$
6: **for** each attribute A_k, k = 1,2, … . . , n **do**
7: **if** A_k is numerical then
8: Sort its values $x_{1k}, \dots \dots, x_{Nk}$ and record as $x_{1k}^*, \dots . . , x_{Nk}^*$
9: Find all cut points $cp_{ik} = \frac{x_{ik}^* + x_{i+1,k}^*}{2}$, i = 1, … . , N − 1
10: **for** each cut points cp_{ik} **do**
11: Calculate the Information Gain:
12: $$Gain(X_j, cp_{ik}) = Info(X_j) - \left[\frac{|x_{ij}^1|}{|x_j|} Info(X_{ij}^1) + \frac{|x_{ij}^2|}{|x_j|} Info(X_{ij}^2)\right] \text{ where}$$
$X_{ij}^1 = \{x_i \in X_j | x_{ik} \le cp_{ik}\}, X_{ij}^2 = \{x_i \in X_j | x_{ik} > cp_{ik}\}$ and symbol |x| is size of x
13: **end**
14: Select the optimal cut point $cp_k(X_j) = cp_{i^*k}$ of A_k:
15: $i^* = \arg\max\{Gain(X_j, cp_{ik})\}_{i=1}^{N-1}$
16: Calculate the split information of $cp_k(X_j)$:
17: $$Split\left(X_j, cp_k(X_j)\right) = -\left[\frac{|x_{i^*k}^1|}{|x_j|}\log_2\frac{|x_{i^*k}^1|}{|x_j|} + \frac{|x_{i^*k}^2|}{|x_j|}\log_2\frac{|x_{i^*k}^2|}{|x_j|}\right]$$
18: **else**
19: Cut points: $cp_k = \bigcup_{i=1}^{C}\{x_{ik}^{'}\}$,
where $x_{ik}^{'} \in \{x_{1k}, \dots, x_{Nk}\}$, C is the number of attribute values
20: Calculate the Information Gain:
21: $$Gain(X_j) = Info(X_j) - \sum_{i=1}^{C}\frac{|x_j^i|}{|x_j|}Info(X_j^i) \text{ where } X_j^i = \{x_i \in X | x_{ik} = x_{ik}^{'}\}$$
22: Calculate the split information of cp_k
23: $$Split\left(X_j, cp_k(X_j)\right) = -\sum_{i=1}^{C}\frac{|x_j^i|}{|x_j|}\log_2\frac{|x_j^i|}{|x_j|}$$
24: **end**
25: Calculate the gain ratio of A_k: $Ratio(X_j, A_k) = \frac{Gain\left(X_j, cp_k(X_j)\right)}{Split\left(X_j, cp_k(X_j)\right)}$
26: Mapper Output: ⟨key, value⟩ = $\langle A_k, [Ratio_k(X_j, A_k), cp_k(X_j)]\rangle$
27: **end**
28: **In the SelectTaskReducer**:
Input: ⟨key, value⟩ = $\langle A_k, List[Ratio_k(X_j, A_k), cp_k(X_j)]\rangle$, j = 1,2, … , m
Output: ⟨key, value⟩ = $\langle A_{k^*}, [Ratio_{k^*}(X), cp_{k^*}(X)]\rangle$
29: **for** each attribute A_k **do**
30: $Ratio_k(X) = \sum_{j=1}^{m} Ratio_k(X_j, A_k)$
31: $cp_k(X) = \frac{\sum_{j=1}^{m} cp_k(X_j)}{m}$ (A_k is Numerical) or $cp_k(X) = U_{j=1}^{j=m} cp_k(X_j)$ (A_k is not Numerical)
32: **end**
33: Select the optimal index, where $k^* = \arg\max\{Ratio_k(X)\}_{k=1}^{n}$
34: **Reducer Output**: ⟨key, value⟩ = $\langle A_{k^*}, [Ratio_{k^*}(X_j), cp_{k^*}(X)]\rangle$
35: **return** A_{k^*} and $cp_{k^*}(X)$.

Appendix D: Dataset Splitting Algorithm (MR-D-S) (Mu *et al*., 2017)

Input: A training data X; an attribute A_{k^*}; the cut point cp_{k^*}; an empty set Ω
Output: Ω

1: Initialise a Hadoop Job SPLITJOB:
2: Set Split TaskMapper as the Mapper class
3: Set Split TaskReducer as the Reducer class
4: Adjust the block size of HDFS until the dataset X can be split into m subsets $\{X_j\}_{j=1}^{m}$
5: **In the j − th Split TaskMapper**:
Input: $X_j = (x_i)_{i=1}^{N}$, where x_i is the i − th instance with n attributes $(A_k)_{k=1}^{n}$
Output: ⟨key, value⟩ = ⟨id, x_i⟩, where the id is the label of output subset.
6: **for** each attribute x_i, i = 1,2, … . . , n **do**
7: **if** A_k is numerical **then**
8: **if** $x_{ik^*} > cp_{k^*}$ **then**
9: id = 1
10: **else**
11: id = 0
12: **end**
13: **else**
14: $id = x_{ik^*}$
15: **end**
16: Mapper Output: ⟨key, value⟩ = ⟨id, x_i⟩
17: **end**
18: **In the Split TaskReducer**:
19: Input: ⟨key, value⟩ = ⟨id, X_{id}⟩, where X_{id} is the set of List(x)
20: Output: ⟨key, value⟩ = ⟨id, X_{id}⟩.
21: **for** each X_{id} **do**
22: Define k is the class number; $\{p_i\}_{i=1}^{k}$ is the class probability and N is the size of X_{id}
23 **if** K>1and $\max\{p_i\}_{i=1}^{k} \leq \theta$ and $N \geq N^*$
24: Build no leaf Node
25: Add X_{id} *to* Ω
26: **else**
27: Build a leaf Nodes
28: **end**
29: **Reducer output**: ⟨key, value⟩ = ⟨id, X_{id}⟩.
30: **end**
31: **return** Ω

Appendix E:MR-C4.5-Tree construction Algorithm (Mu *et al*., 2017)

Input: A training dataset Ω_0; the maximum H^*
Output: An $\mathrm{MR}-\mathrm{C4.5}-\mathrm{Tree}$

1: **if** $Tree's\ depth\ > H^*$ **then**
2: **return**
3: **end**
4: Ω is initialized as an empty set
5: Select one node from $\Omega_0, denoted\ by\ x$
6: **for** $\mathrm{X} : \Omega_0$ **do**
7: Get the best attribute $\mathrm{A_{k^*}}$ and cut points $\mathrm{cp_{k^*}}$ from x by Algorithm 2
8: Split x into n child nodes $(\mathrm{X_i})_{i=1}^{n}$ based on $\mathrm{A_{k^*}}$and $\mathrm{cp_{k^*}}$ by Algoirthm 3
9: Add$(\mathrm{X_i})_{i=1}^{n}$ to Ω
10: **end**
11: Tree's depth $=$ Tree's $+ 1$
12: **if** Ω is not empty **then**
13: $\Omega_0 = \Omega$
14: Recursive search the new node from $\Omega_0$by algorithm 4
15: **end**