International Journal for Multidisciplinary Research (IJFMR)
E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

# Scaling Up Data Mining Algorithms for Big Data

**Pankras K. Kandengukila[1], Daudi Mashauri[2]**

[1]Assistant Lecturer, Department of Management Studies, Tanzania Institute of Accountancy (TIA), Singida, Tanzania.
[2]Assistant Lecturer, Department of Management Studies, Tanzania Institute of Accountancy (TIA), Dar es Salaam, 15108 Tanzania.

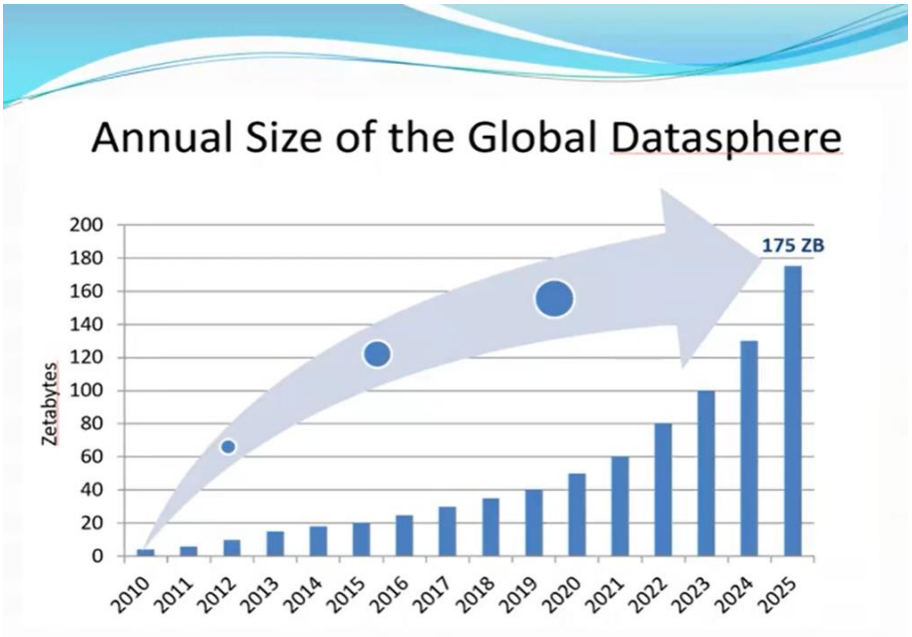
**Abstract**
The rapid development of science and technology and replacement of digital equipment have presided over today's era of big data. Automatically discovering and extracting hidden knowledge in the forms of patterns from these big data is known as data mining. However, the emergence of big data era has brought a series of challenges to data mining techniques including too long processing time, insufficient memory capacity and excessive power consumption. Aim of this paper is to study scaling up data mining algorithms for big data by Random Forest and Naïve Bayes. The background and applications of data mining, big data and cloud computing are briefly introduced together with the basic principles of Random Forest and Naive Bayes as well as MapReduce model in cloud computing. Then, the feasibility of parallelism of Random Forest and Naive Bayes is studied. Two parallel Random Forest and Naive Bayes algorithms based on MapReduce are developed and realized in Hadoop platform. Finally, the parallelism of Random Forest and Naive Bayes is validated by experiments. Their execution efficiency is analyzed through the experimental results on the different sizes of data sets and different numbers of clusters. It is shown that the proposed methods have a good performance and can be applied in process of big data.

**Keywords**: Data Mining, Big data, Cloud Computing, Random Forest, Naïve Bayes

## 1. Introduction

In recent decades, there has been a tremendous progress in information and communication technology, which has resulted in the creation of vast volumes of data that are larger than what can be handled by traditional means. The result of this growing reliance on data will be an infinite growth in the Global Datasphere's size. The Global Datasphere is expected to reach 175 ZB (1 billion terabyte) by 2025, from an estimated 33 ZB in 2018, according to International Data Corporation (IDC) [1].

**Figure 1: Annual Size of the Global Datasphere in Zettabyte**




Undoubtedly, we are living in time when data is abundant. These copious amounts of data originated from variety of source, including sensors, smart devices, photos, videos web pages, social networking interactions, private browsing histories, online transactions [1]. This data is commonly referred to as "Big Data" because of its volume, the velocity with which it arrives and the variety of forms it takes. In 2001, Gartner proposed a three-dimensional or 3Vs (volume, variety and velocity) view of the challenges and opportunities associated with data growth [3].

Most of the widely used algorithms in machine learning were developed when the typical dataset sizes were much smaller. Some of those learning algorithms can work with huge data sets, but many are not able to deal with such large amounts of information. Their applications may be hindered by memory demands, impracticable running times or both. In all the fields of applications of machine learning, with the growing size of the datasets, the need is also growing to scale up Data Mining Algorithms (DMA) [8]. The two most significant challenges driving changes in Data Mining (DM) are scalability and performance. Organizations want data mining to be more powerful so that they can analyze and compare multiple datasets, not just individual large datasets, as is traditionally the case. Along with the large amount of data available, there is also a compelling need for producing results accurately and fast. Efficiency and scalability are indeed. These are keys issues when designing data mining systems for big dataset [10].

However, the study on Scaling up Data Mining Algorithms can be summarised as follows:

- This paper bridged the gap between users and programmers by providing generic abstraction for large scale Data Mining, enabling the users to run their own algorithms with minimum effort.
- As a result, scaling allows companies to process larger amounts of data without having to deal with prohibitive delays in algorithm execution.

The primary goal of this study is to develop a methodology for scaling up DMA and their practical applications within the field of DM. More specifically, this overarching goal can be subdivided into several distinct objectives.

- Designing a powerful scaling methods focusing on the objective of achieving a good balance between

efficiency and performance.

- To apply scaling methodology specifically to Random Forest (RF) and Naïve Bayes (NB) algorithms, through Cloud Computing Technology (CCT) in parallel processing.
- To analyze computing power required in processing big data in CCT.

The remainder of this study is organized as follows: Section 2 present a conceptual framework of research techniques with a detailed discussion of the algorithms used. Datasets are downloaded from the University of California Irvine (UCI), Canter for Machine Learning and Intelligent Systems. For this study, we use three types of datasets: Wine datasets, Senses income datasets, and KDD Cup datasets. Each algorithm requires some parameter optimization, as explained in Section 2, and their effects on computation time and performance are demonstrated through experiments. The results section, described in Section 3 shows graph and histograms obtained as a result of running time and performance. Section 4 discusses the experimental results in detail and possible problems. One of the major result obtained in this research is that RF and NB Algorithms performs consistently well on all types of datasets despite requiring the most computation time. Finally, Section 5 concludes this study.

**a) Data deluge**

The amount of data produced is growing as a result of increased Internet usage, leading to gigantic data known as big data [9]. This volume of data is produced by a multitude of sources, including clickstream data, financial transaction data, log files from online or mobile apps, sensor data from the Internet of Things (IoT), player behavior during games, telemetry from linked devices, and several more sources [4].

**b) Big Data Processing**

Turning big data into actionable information requires using computational techniques to unveil trends and patterns within and between these extremely large socioeconomic datasets. New insights gleaned from such data mining should complement official statistics, survey data, and information generated by Early Warning Systems, adding depth and nuances on human behaviors and experiences and doing so in real time, thereby narrowing both information and gaps [3].

**Cloud Computing Technology**

To date, several attempts have been made to scale Data Mining Algorithms for large data. Furthermore, the majority of their efforts used conventional algorithms rather than the more recent cloud computing technologies. Examined the issue of scaling big data algorithms for popular association rule mining, namely the creation of the Apriori Algorithm in the Map Reduce model on cloud computing. The findings may lead to a number of non-naïve MapReduce options based on big data [7].

Through cloud computing, we can use and access a variety of Internet-based services, including servers, computers, databases, and storage. By using parallel computing, a large task is divided into smaller chunks, each chunk being processed simultaneously by multiple processors on a single processor or computer cluster, so that make everything work faster and more efficiently to deal with big data [11].

**c) Random Forest and Naïve Bayes Algorithms.**

In previous research it was demonstrated that Naïve Bayes induction algorithms exhibit surprising accuracy in numerous classification tasks, even when the conditional independence assumption they rely on is not upheld [11]. NB Tree, a hybrid classifier combining decision tree (DT) classifiers and Naïve Bayes NB. The nodes in the DT contain univariate splits like a regular DT, but the leaves contain NB

classifiers. This approach maintains the interpretability of both NB and DT, while producing classifiers that often outperform both components, particularly in the large database that was tested [9].

## 2. Materials and methods

In this study, we describe our proposed model for dealing with scalability problem. To achieve this objective, we use RF and NB Algorithms on the advantage of CCT in parallel processing. .

**Experimental setup**

The experiment environment is composed of 6 machines: 1 master and 5 slaves. Type of machines is Dell Power Edge R710 (Xeon E556). Each machine contains 1TB of hard disk and 12 GB memory running on LTS 24.04 LTS operating system. Java 1.8.0_01 and Hadoop is installed on these machines to form a MapReduce running time environment.

The objective of this evaluation is to determine if our approaches could give better results than traditional algorithms and to confirm whether previous steps are adequately handled, and highlight some measures taken to handle some of the challenges.

### a) Random Forest Algorithms

RF is a data mining algorithms in the field of data mining, the massive number of data processing and mass calculation is a common problem. The RF traditional algorithms can only suitable for small-scale data sets, when the input data set increases, often lead can't even run the program slows down.

RF is a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for trees in the forest. The generalization error of a forest converges as to a limit as the number of the trees in forest becomes large. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them.

It is a classifier consisting of a collection of tree structured classifiers $\{h(x\cup), k=1,..\}$ where $\{\cup k\}$ independent identically distributed random vectors is and each tree casts a unit vote for the most popular class input $x$

**1. Build Tree (Data Set, Output)**

*If all output values are the same in Dataset, return a leaf node that says "predict this unique output"*
*If all input values are the same, return a leaf node that says "predict the majority output"*
*Else find attribute* $X$ *with highest Info Gain*
*Suppose* $X$ *has* $nX$ *distinct values.*
*Create and return a non-leaf node with* $nX$ *children.*
*The* $i^{th}$ *child should be built by calling Build Tree (*$DSi$*, Output)*
*Where* $DSi$ *built consists of all those records in Dataset*
For which $X = i^{th}$ distinct value of $X$ .Return a decision tree node, splitting on $j^{'th}$ attribute. It has two children corresponding to whether the $j^{'th}$ attribute.

**2. Attribute Selection**

To decide which attribute should be tested first, simply find the one with the highest information gain.

**Information gain (IG)**

In general terms, the expected information gain is the change in information entropy from a prior state to a state that takes some information as given:

$$IG(T \mid C) = H(T) - H(T \mid C) \quad (1)$$

While $H$ is the information entropy.

From Shannon theory

For any random variable x, its entropy is defined as

$$H(X) = -\sum_{x} P(x)\log_2[P(x)] \quad (2)$$

For the set of independent random samples $C$ categories. And $C$ it stands for $C_1, C_2, C_3 \ldots\ldots\ldots C_n$. Meanwhile probability of occurrence of each category is $P(C_1), P(C_2), P(C_3) \ldots\ldots\ldots\ldots P(C_n)$ .Whereby $n$ is the total number of categories.

Then the classification system entropy can be expressed by,

$$H(C) = -\sum_{i=1}^{n} P(C_i).\log_2 P(C_i) \quad (3)$$

Condition for entropy,

$$X = x_i \quad (4)$$

Then,

$$H(C \mid X = x_i) \quad (5)$$

For conditional entropy $X$ is fixed.

$$H(C \mid X) \quad (6)$$

That is to say,

$$H(C \mid X = P_1 H(C \mid X = x_1) + P_2 H(C \mid X = x_2) + P_3 H(C \mid X = x_3) + \ldots\ldots + P_n H(C \mid X = x_n) \quad (7)$$

$$= \sum_{i=1}^{n} P_i \; H(C \mid X = x_i) \quad (8)$$

Information gain is given by

$$IG(T) = H(C) - H(C \mid T) \quad (9)$$

$$IG = -\sum_{i=1}^{n} P(c_i)\log_2 P(c_i) + P(t)\sum_{i=1}^{n} P(c_i \mid t)\log_2 P(c_i \mid t) + P(t')\sum_{i=1}^{n} P(c_i \mid t')\log_2 P(c_i \mid t') \quad (10)$$

Calculate information gain of each attribute which selected random from the numbers of the attribute which given in the training datasets. Then the attribute which has the highest information gin will be tested first.

### 3. Growing Random Forest

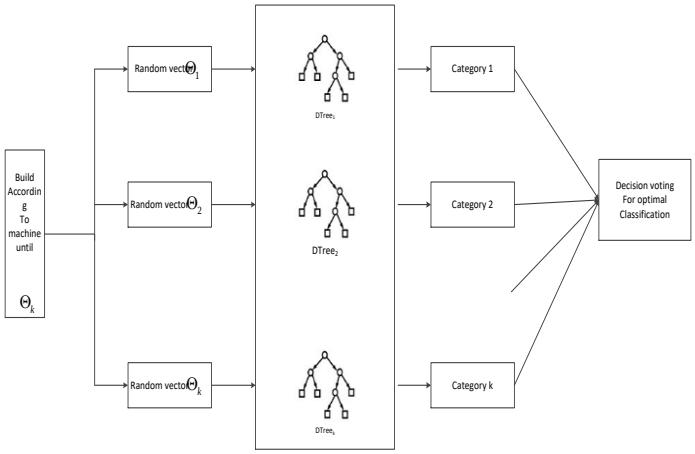
Each tree is grown as follows:

In the field of classification, parallelization has been extensively used for scaling up

- If the number of cases in the training set is $N$ , samples $N$ cases at random-but with replacement, from the original data. This sample will be the training set for the growing tree.
- If there are $M$ input variable, a number $m << M$ is specified such that at each node, $m$ variables are selected at random of $M$ and the best split on these $m$ is used to split the node. The value $m$ is held constant during the forest growing.
- Each tree is growing to its large extent possible. There is no pruning.

In RF only two parameters should be determined, number of trees to be used and the number of variables $(m)$ to be randomly selected from the available set of variables as shown in Figure 2.

**Figure 2: Random Forest growing**




### b. Naïve Bayes algorithms

The NB algorithm is based on conditional probabilities. It uses Bayes' Theorem, a formula that calculates a probability by counting the frequency of values and combinations of values in historical data. Bayes' Theorem finds the probability of an event occurring given the probability of another event that has occurred. In simple terms a NB classifier assumes that the presence or absence of particular feature is unrelated to the presence or absence of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round and about 4" in diameter. A NB classifier considers each of these features to contribute independently to the probability that this fruit is an apple regardless of the presence of the other features.

For the sample C is typically high-dimension, and it needs to estimate probability of $P(C \mid E)$ from limited data. For the category, $\{E_1, E_2, .....E_i\}$ and
Feature/attribute vector $x = [C_1, C_2, ....., C_d]^t$

If event C and E are independent each other then, conditional

Probability of an event to occur given by,

$$P(C \bigcap E) = P(C) * P(E \mid C) = P(E) * (C \mid E) \quad (11)$$

$$P(C \mid E) = \frac{P(EC)}{P(E)} \quad (12)$$

From Bayes' Theorem $$P(C_i \mid E) = \frac{P(EC_i)}{P(E)} = \frac{P(E \mid C_i)P(C_i)}{P(E)} \quad (13)$$

Naïve Bayes can solve problem for big data, for the data set which having thousands of attributes the following formula is involved:

$$P(C_i \mid E_1, E_2, ..., E_n) = \frac{P(E_1, E_2, ..., E_n \mid C_i) \times P(C_i)}{P(E_1, E_2, ..., E_n)} \quad (14)$$

$$= \frac{P(E_1 \mid C_i) \times P(E_2 \mid C_i) \times ... P(E_n \mid C_i) \times P(C_i)}{P(E_1, E_2, ..., E_n)} \quad (15)$$

Classifier formula is given by

$$c(X) = \arg\max(P(C_i \mid E)) = \arg\max(\frac{P(E \mid C_i)P(C_i)}{P(E)}) \quad (16)$$

When the posterior probabilities of different categories are constant then the denominator $P(E)$ is always constant, and therefore $P(E)$ can be ignored. And classification formula becomes,

$$c(X) = \arg\max_{C_i \in C}(P(E \mid C_i)P(C_i)) \quad (17)$$

## 3. Implementation

### a) Random Forest Implementation

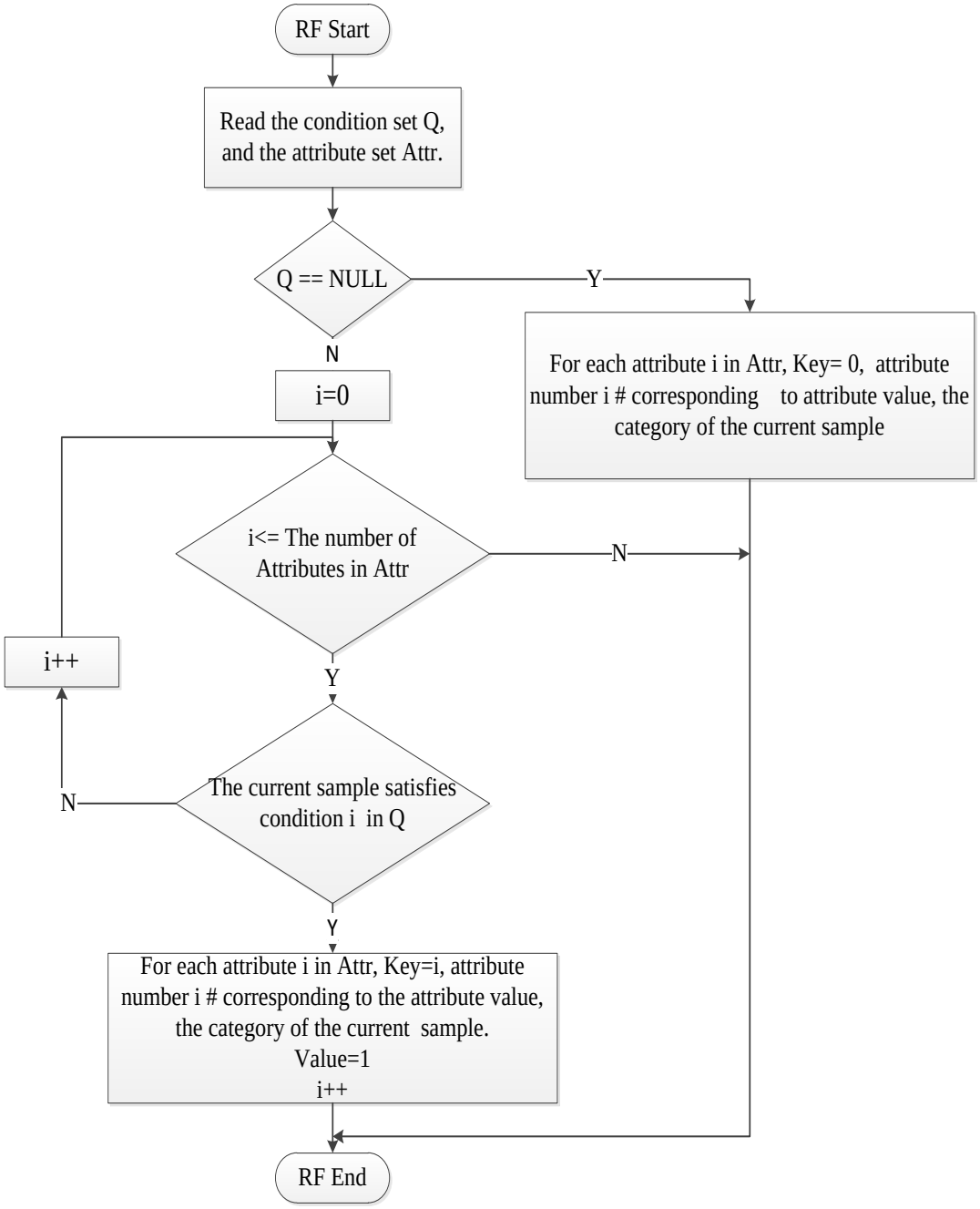
In RF, each tree is grown to the fullest extent possible without pruning. A random number of attributes are chosen for each tree. These attributes form the nodes and leafs using standard tree building algorithms. Training stage and test stage are considered to implement RF in Mapreduce.

The training set of data is selected as the statistical data in parallel sections. To generate decision tree in MapReduce, the information gain and classification of various properties are required in the program constructions. Each time the Information gain is calculated. The process of calculating information gain is repeated in the terms of iterations. Therefore, the attribute which has the highest information gain will be selected as a candidate for the best classification attribute. Basically the idea design of RF in MapReduce has Map stage, Reduce stage, Tree construction stage and Main processing stage.

Map Stage: On map phase forests are growing during training. After data has been validated, then randomly $n$ numbers of rows and in each row $N$ number of columns are generated and selected by BootStrap replacement sampling for training phase and stored in the temporary file.

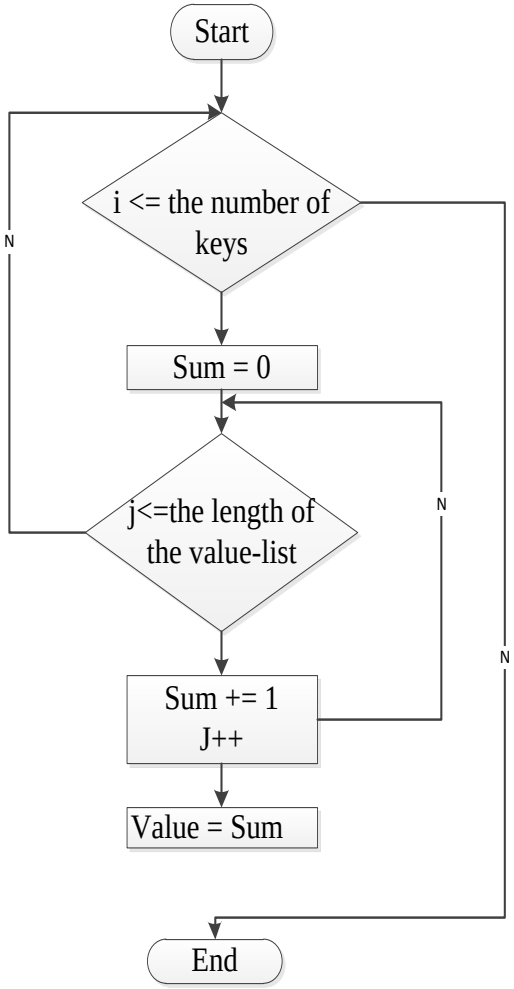
Under the descriptions in Figure 3-4, the mapper class reads the input file (training data) supplied by the user and it is stored in a temporary data structure. At the end of the map phase, the mapper sends all local histograms in a set of (key, value set) pairs to reducer. The key is the id of a decision tree and the value is local histograms of the nodes in that tree. In this way, we ensure all local histograms for the same decision tree are sent to one reducer.

**Figure 3: RF map flow chart.**



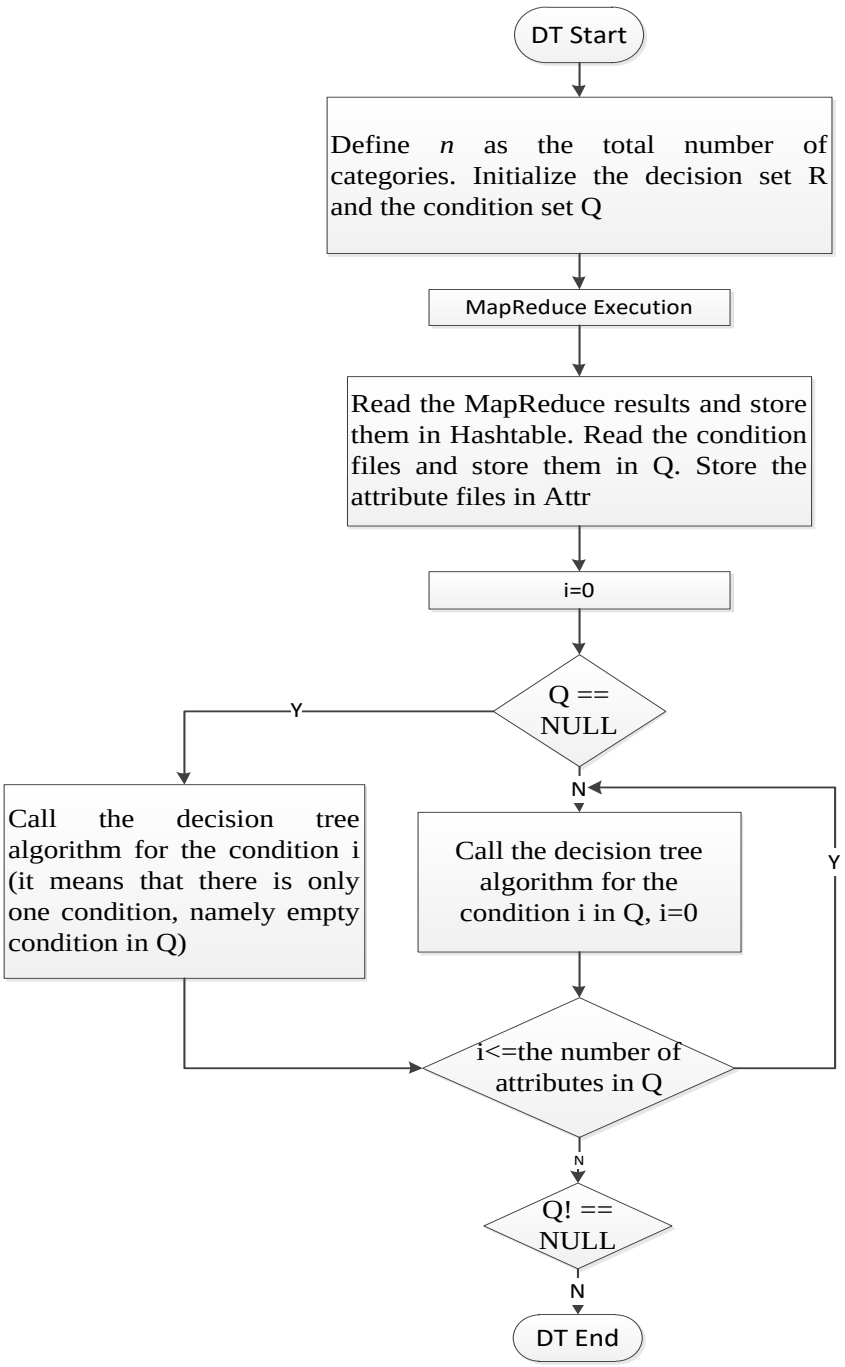
**Reduce stage:** The descriptions in Figure 4. Shows that at the reduce stage, the output of the Map stage, e.g., key-value pairs, is calculated. The pairs with the same key-value merge together. Value is counted as a sum of values with the same key-value, namely, the number of key-value pairs with the same key-value.

**Figure 4: RF Reduce flow chart**



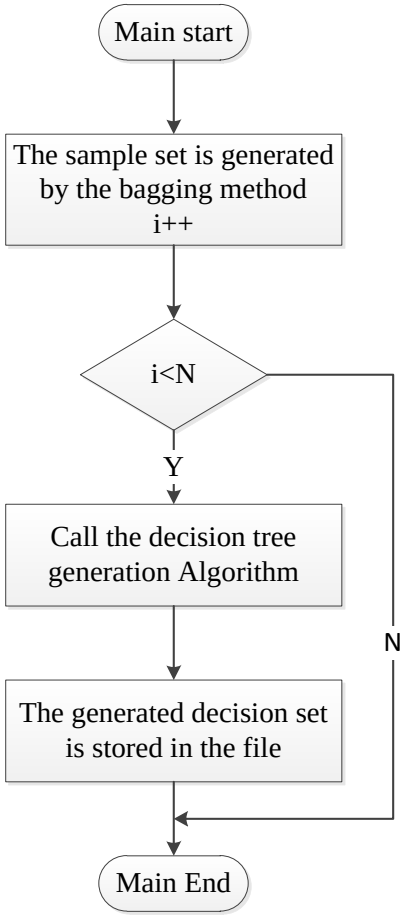

**Decision Tree:** 5 describe the construction of the decision tree on Mapreduce. Decision tree construction is an iterative process. The splitting attribute or leaf nodes on the same level is generated by each iteration. The iteration condition is the current condition set Q is not empty.

**Figure 5: DT flow chart**




**Main program:** Figure 6. Describes the main program design of RF. The main program is a loop which generates $N$ random forest trees in the construction process of RF.

**Figure 6. Main program flow charts**




**b. Naïve Bayes implementation**

1. The implementation of parallel NB in MapReduce model is divided into training and testing stages. The distributed computing of Hadoop is divided into two phases which are *map* and *reduce*. The input format which belongs to the Hadoop framework loads the input data into small data blocks known as data fragmentation. The *map* function figures out the categories and properties of the input data, including the values of categories and properties. The attributes and categories of the input records are separated by a comma, and the final attribute is the property of classifications.
2. **Map stage:** Each map processes a single split, and the map task passes the split to the get RecordReader() method on InputFormat to gain a RecordReader for that split. The RecordReader is an iterator of the records. Then the map task uses a RecordReader to generate key-value pairs, which send to the map function. Map task processes all input data in terms of keys and values $< key, value >$ in training data before sending to the Reduce stage as described in Figure 7.
3. **Reduce stage:** The reduce stage counts key and value from the output by the map stage. The reduce function aggregates the number of each attribute and category value, which results in the form of output as the training model as described in Figures 7, 8, 9 and 10.

**Figure 7: NB Job1 map flow chart**

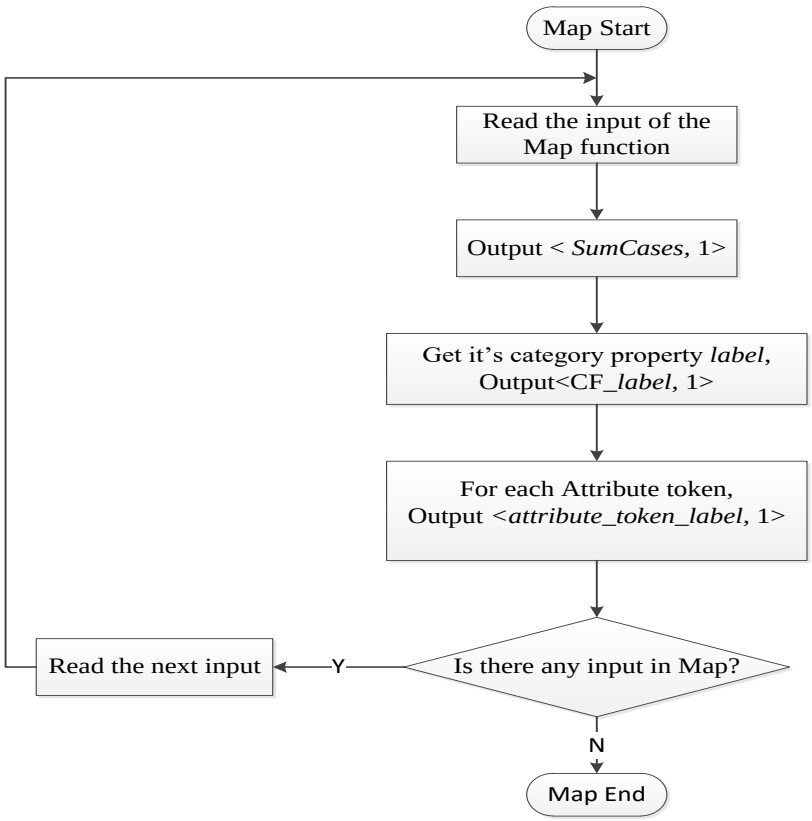



**Figure 8: NB Job1 reduce flow chart**

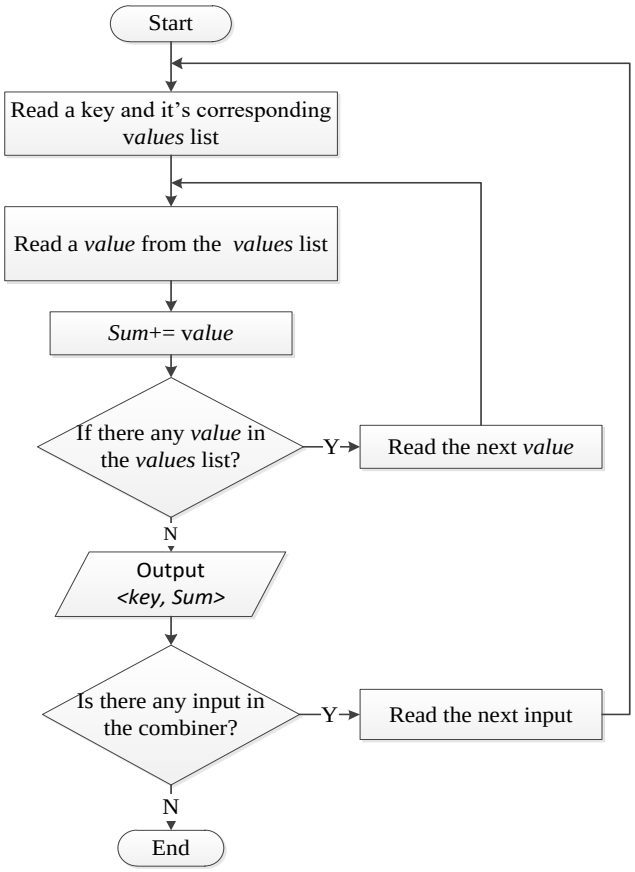


**Figure 9: NB Job 2 map flow chart**

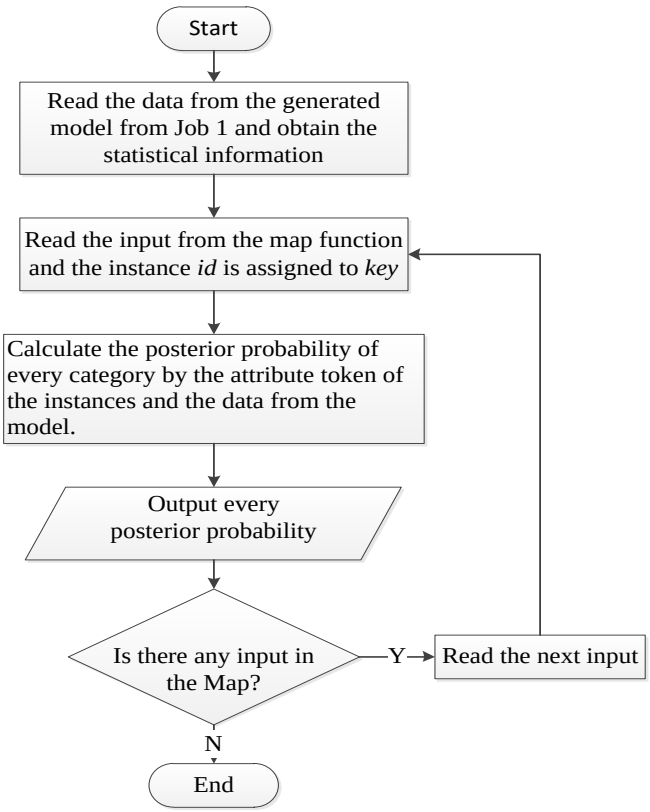
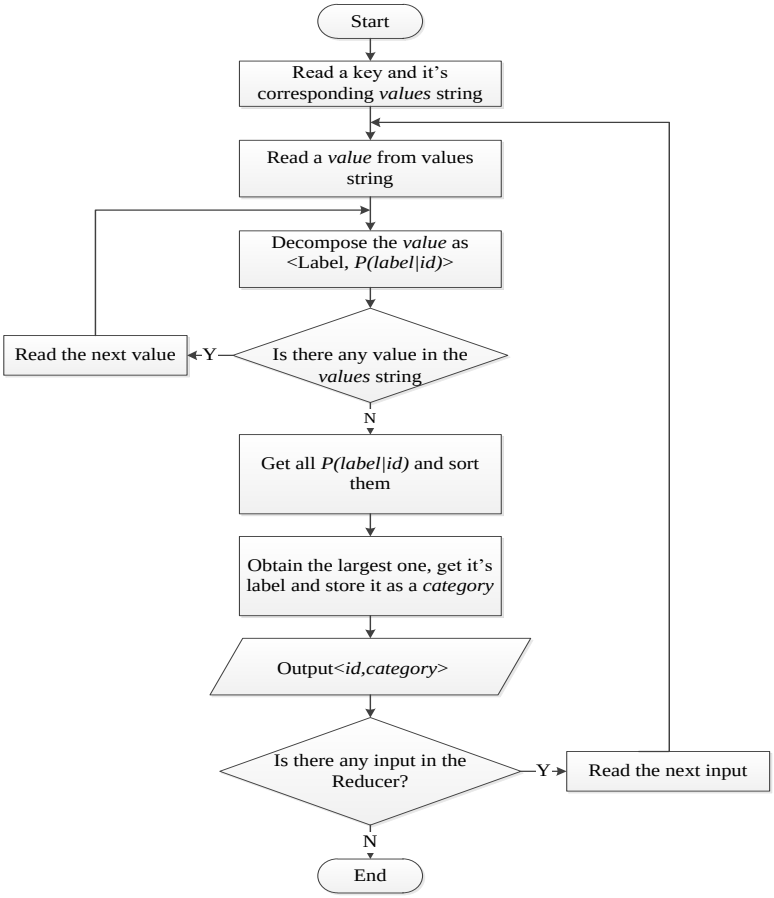



**Figure 10: Job2 reducer flow chart**

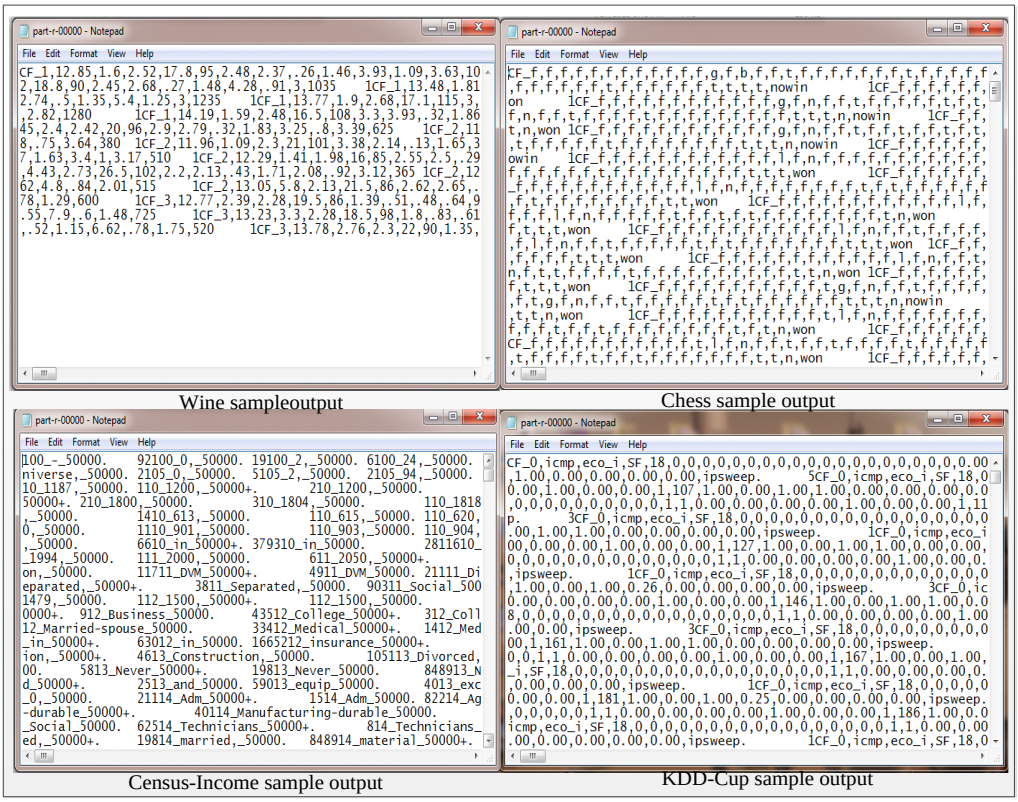


## 4. Results

In previous, we describe how to implement our methods in CCT. In this chapter, we are going to study output and results obtained after execution. We use different type, and size of dataset in order to experience different behavior of the system.

Figure 11 shows sample classifications results of our experiments according to the input data shown on early chapter. The output sample of the data sets shows there is data identification, category and classified data of the datasets. We perform large number of experiments across, algorithms and different size of dataset as described before. In this section, we summarize the results and provide analyses and insight based on our findings with respect to trade-offs discussed earlier.

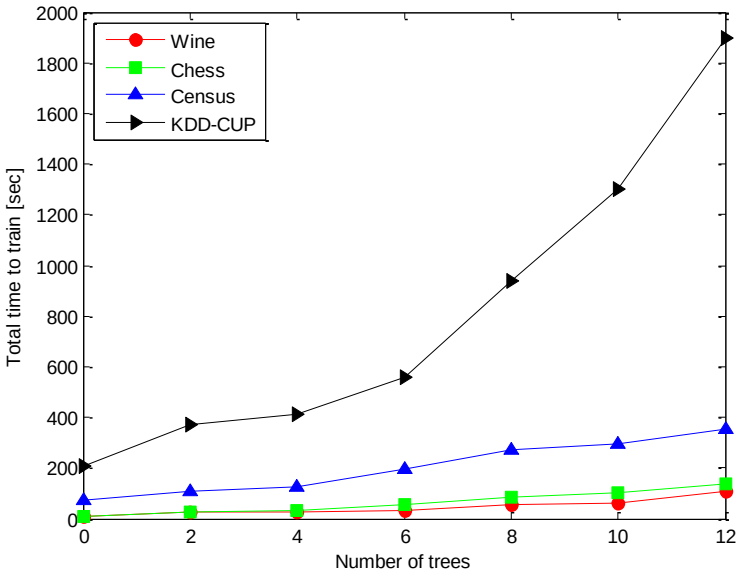
**Table 1: Sample of classifications results**




### Results 1: Random Forest experiment

Figure 12. Shows the run time with respect to the number of trees. The figure shows the scalability on four data sets with respect to the number of trees in RF, the size of data block and the number of machines used. The figure shows that the time used to build a random forest model increases linearly as the number of trees increases in the model. The run time increases slowly as more trees are added to the model. The larger the data sets, the more time it takes when more trees are added. However, the speed of time increases very slowly. These results demonstrate that RF is scalable to the big data.

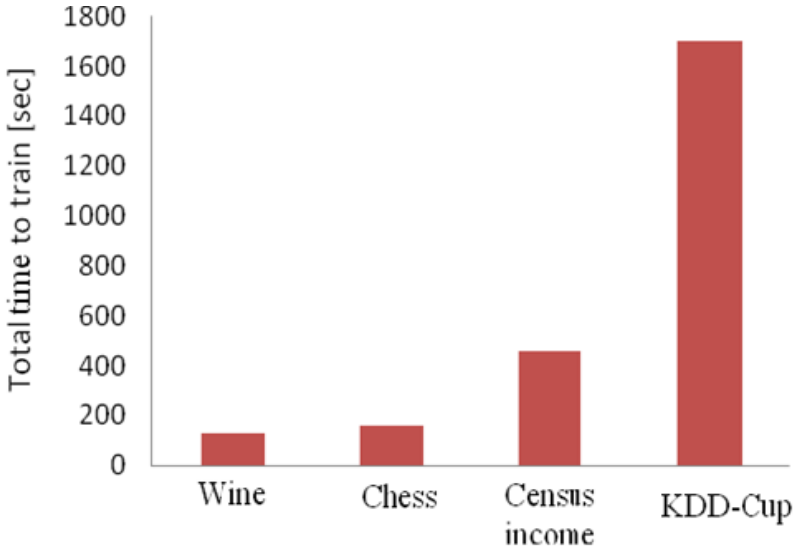
**Figure 12: Run time with respect to number of trees for RF**




**Results 2: Random Forest**

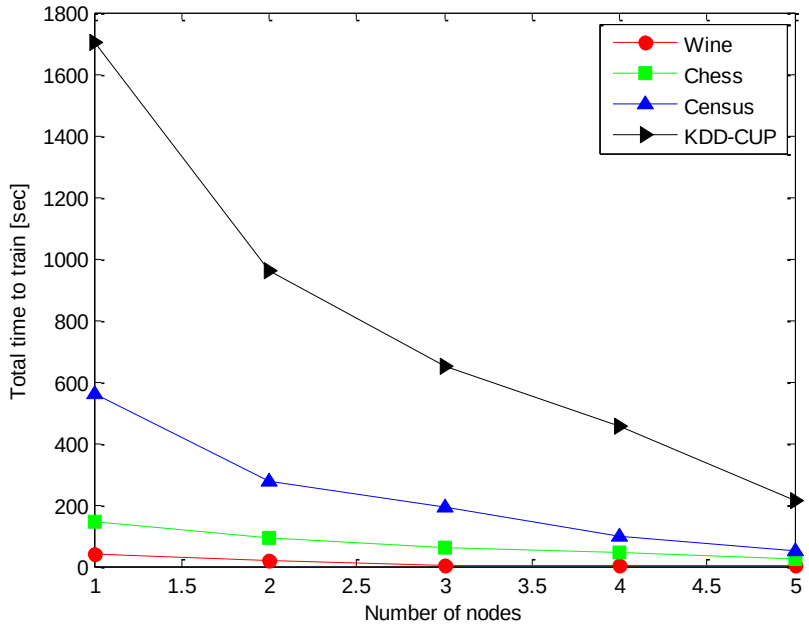
Figure 13. Shows the change of run time over the change of the size of data block. The size of data block has impact on the run time of RF. On the one hand, the smaller data block the more mappers generate. However, if the number of mappers exceeds the mapper capacity of the system, the run time will increase rapidly. On the other hand, the larger data block the heavier load mapper generates.

**Figure 13: Run time with respect to data size for RF**




In Figure 14. Shows the scalability of RF with respect to the number of machines involved. We can see that the run time drops rapidly as more machines are added. This demonstrates that RF is able to handle big data by adding more computing resources.

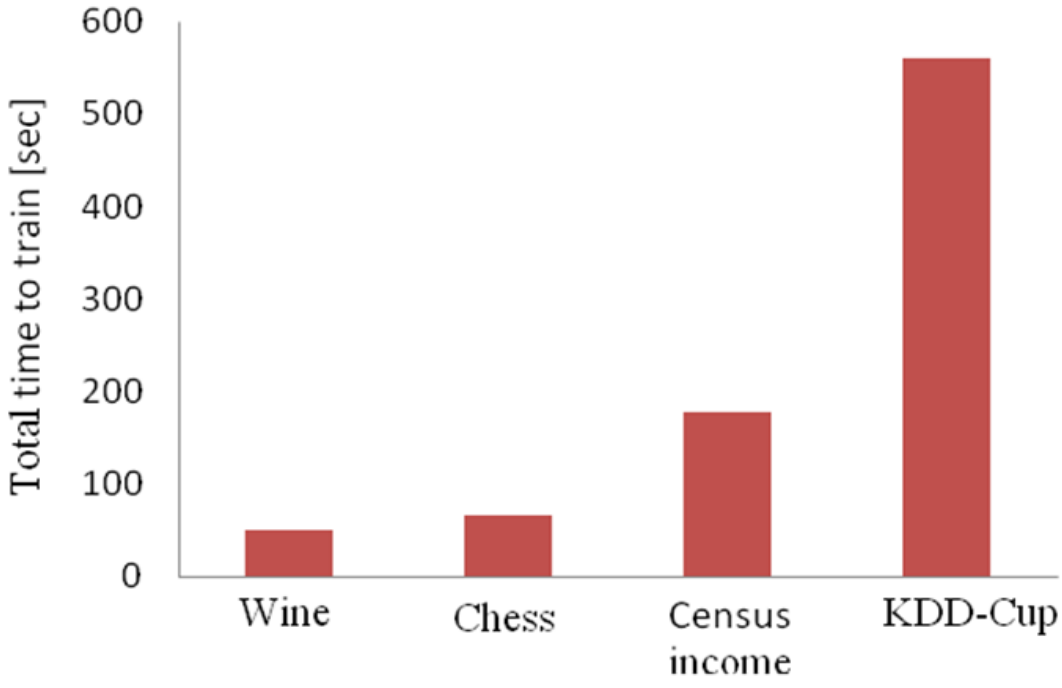
**Figure 14: Run time with respect to the number of nodes for RF**




**Results 3: Naïve Bayes experiment.**

The results in Figures 15. Shows the run time with respect to the data size through NB method. We observe the consistent improvement of the execution time in NB approach. In Figure 15, it shows that once you increase the size of data sets, the execution time increases.

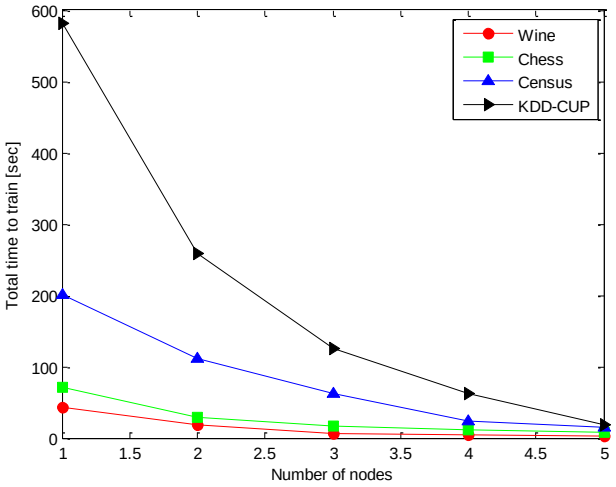
**Figure 15: Run time with respect to data size for NB**



**Results 4: Naïve Bayes experiment**

And Figure 16. Shows the execution time with respect to the number of nodes. As shown in the figure once the number of computers increases, the running time decreases quickly.

**Figure 16: Run time with respect to number of nodes for NB**




**General Discussion**

Generally, since our primary interest lies in the scalability analysis as shown in the previous figures. We examine the execution time (speed up) and the changing size of data sets (size up). We have seen that both of our methods have good results despite the fact that RF is executed slower than NB. This is because some of our datasets have large number of instances as we have seen in Table 1.

And also we experience that when we increase the number of machines in the system, the running time decreases. As data set sizes increase, more computing capacity is required for processing. In general, we can say that, both of the two methods based on MapReduce have gained high scalability.

Then experiments shows advantages of CCT as per figure 14 and figure 16. Once we increase the number of machines in the system, running time decreases. As data set sizes increases more compute capacity is required for processing.

## 5. Conclusion

Advances in digital and computer technology have led to massive amounts of information and collections of data to be processed, containing many terabytes and exabytes of data. KDD and DM can help dealing with this problem because they aim to turn raw data into nuggets and create special edges. KDD practitioners would like to be able to apply DM learning algorithms to these large data sets in order to discovery useful knowledge.

However, the increase of the database's size is staple problem which is known as scaling up problem. The question of scalability asks whether the algorithm can process large data sets efficiently while building best possible models from them. As raw data are rarely used directly and manual analysis simple cannot keep up with the fast growth of data, with the growing size of the datasets the need is also growing to scale up DMA in all fields of applications of machine learning. Scientific research, ranging from astronomy to human natural genome, text mining or security among others faces the same problem of how to deal with vast amounts of information.

Moreover, just selecting a representative subset of the original dataset is not an option, due to several reasons: if the dataset is huge, sub sampling a percentage can itself pose a challenge and, above all, increasing the size of the training set often increases the accuracy classifiers avoiding over fitting.
In this study we have presented two algorithms, RF and NB and their implementation in MapReduce. Through two methods we study that scalability can be obtained when CCT had been applied. Either we showed that RF can produce more trees which implies that it can process big data. Either NB through Bayes theorem gained high scalability as shown in experiments.

## 6. References

1. Ireland, G. Brown, K. Wright, A, "International Data Cooperation" https://www.idc.com/search/singleproduct/perform_.do?page=1&hitsPerPage=25&sortBy=RELEVANCY&srchIn=ALLRESEARCH&src=&athrT=10&cmpT=10&pgT=10&prodid=268871403&trid=145587129&siteContext=IDC, 2021.
2. Alotaibi, N.M. and Abdullah, M.A, "Big data mining: a classification perspective", in Communication, Management and Information Technology, Proceedings of the International Conference on Communication, Management and Information Technology, pp. 687-696, 2016.
3. Bumblauskas, D., Nold, H., Bumblauskas, P., & Igou, A. Big data analytics: transforming data to action. *Business Process Management Journal*, *23*(3), 703–720. https://doi.org/10.1108/BPMJ-03-2016-0056, 2017.
4. Djafri, L. Dynamic Distributed and Parallel Machine Learning algorithms for big data mining processing. *Data Technologies and Applications*, *56*(4), 558–601. https://doi.org/10.1108/DTA-06-2021-0153, 2022.
5. Günther, W. A., Rezazade Mehrizi, M. H., Huysman, M., & Feldberg, F. Debating big data: A literature review on realizing value from big data. *Journal of Strategic Information Systems*, *26*(3), 191–209. https://doi.org/10.1016/j.jsis.2017.07.003, 2017
6. Kijsanayothin, P., Chalumporn, G., & Hewett, R. On using MapReduce to scale algorithms for Big Data analytics : a case study. *Journal of Big Data*, 1–20. https://doi.org/10.1186/s40537-019-0269-1, 2019.
7. Rawal, A., & Lal, B. Predictive model for admission uncertainty in high education using Naïve Bayes classifier. *Journal of Indian Business Research*, *15*(2), 262–277. https://doi.org/10.1108/JIBR-08-2022-0209, 2023.
8. Sarker, I. H. Machine Learning: Algorithms, Real-World Applications and Research Directions. *SN Computer Science*, *2*(3), 1–21. https://doi.org/10.1007/s42979-021-00592-x, 2021.
9. Sathyaraj, R., Ramanathan, L., Lavanya, K., Balasubramanian, V., & Saira Banu, J. Chicken swarm foraging algorithm for big data classification using the deep belief network classifier. *Data Technologies and Applications*, *55*(3), 332–352. https://doi.org/10.1108/DTA-08-2019-0146, 2020.
10. Sternic, N., Pavlovic, A., Miljic, P., Bajcetic, M., Lackovic, M., & Lackovic, V. Cerebral autosomal dominant arteriopathy with subcortical infarcts and leukoencephalopathy with severe factor XII deficiency. *Neurology India*, *57*(5), 657–659. https://doi.org/10.4103/0028-3886.57806, 2019.
11. Syam, N., & Kaul, R. Random Forest, Bagging, and Boosting of Decision Trees. In *Machine Learning and Artificial Intelligence in Marketing and Sales* (pp. 139–182). Emerald Publishing Limited. https://doi.org/10.1108/978-1-80043-880-420211006, 2021.