# *MR – Random Forest Algorithm for Distributed Action Rules Discovery*

*Mythri S. Madhu, Prateek Mahendrakar, Vamsy Ram Kundula, Hanisha Marothu, Kaushal Kanakamedala, Navaneeth Reddy Matta*
*Department of Computer Science*
*University of North Carolina at Charlotte*
*Charlotte,NC,USA*
e-mail: mmadhu@uncc.edu, pmahend1@uncc.edu, vkundula@uncc.edu, hmarothu@uncc.edu, kkanakam@uncc.edu, nmatta1@uncc.edu

***Abstract*—Action rules are built from atomic expressions, which describe the possible transition of object from one state to another based on a decision attribute. An association action rule is like action rule, which involves change of several attributes in its decision part. Earlier algorithms to discover action rules required extraction of classification rules before constructing any action rule. Newest algorithms extract action rules directly from the given information system. Action rules assume that attributes are divided into two groups: *stable* and *flexible*. Action rules can be constructed from two rules extracted from database. These rules represent two different decision classes. Flexible attributes help in re-classifying objects from one class to another. The set of conditional attributes are partitioned into flexible, stable, semi-stable. The semi- stable attributes based on the semantics can sometimes be treated as flexible. A same new action rule can be constructed using the semi stable attributes, which are used to replace the existing action rule whose confidence is too low.**

***Keywords-Data-mining;Association Action rule; Map Reduce; Hadoop;***

## I. Introduction

Many business enterprises accumulate large quantities of data from their day-to-day operations. For example, huge amounts of customer purchase data are collected daily at the checkout counters of grocery stores.Retailers are interested in analyzing the data to learn about the purchasing behavior of their customers. Such valuable information can be used to support a variety of business-related applications such as marketing promotions, inventory management, and customer relationship management.Association analysis is useful for discovering interesting relationships hidden in large data sets. The uncovered relationships can be represented in the form of association rules or sets of frequent items.Besides market basket data, association analysis is also applicable to other application domains such as bioinformatics, medical diagnosis, Web mining, and scientific data analysis.There are two key issues that need to be addressed when applying association analysis to market basket data. First, discovering patterns from a large transaction data set can be computationally expensive. Second, some of the discovered patterns are potentially spurious because they may happen simply by chance.

Association rules are created by analyzing data for frequent if/then patterns.An association rule has two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent.

An association rule is an implication expression of the from X→Y, where X and Y are disjoint items, i.e; X∩Y = φ. The strength of association rule is measured in terms of **support** and **confidence.** Support is an important measure that defines how often a rule is applicable to a given data set. Support is used to eliminate un-interesting rules. A rule which has low support may occur by chance and it is likely to be of less importance. Confidence defines the number of times items in Y appear in transactions that contain X. Confidence provides an estimate of conditional probability of Y given X.

Consider a dataset of an information system of the form S={X ∪ Y ∪ Z}, where Y and Z are flexible attributes and X is the stable attribute. Following is an example of action rule if user desires the decision attribute value to change from z1 to z2; where {x1,x2,x3..xn} ⊆ X, {y1,y2,y3,…yn} ⊆ Y, {z1,z2,..zn}⊆ Z. Rule 1 means that if attribute x changes its value from 1 to 2, and attribute y value remains y2, then attribute z is expected to change from z1 to z2.

$$Rule1 = (X, x1 \rightarrow x2) \wedge (Y,y2) \rightarrow (Z, z1 \rightarrow z2)$$

As data in real-time is very large, to generate action rules on a single becomes expensive. Hence we use an algorithm which runs on a distributed system. To generate action rules in a distributed system we use MapReduce. To run MapReduce we used Apache's open source Hadoop framework, to process large datasets on multiple nodes in multiple clusters.

To discover action rules we used MapReduce- Random Forest Algorithm. We used algorithm known as *Action Rules Discovery Based on Grabbing Strategy* and *Learning from Example Based on Rough Sets* to the distributed MapReduce environment. Using *LERS* we extracted the action rules in Mappers. The output from Mapper is sent to reducers, where reducers user Random Forest to produce a set of action rules. These action rules are called Association Action Rules(AAR).

## II. Related Work

The action rules notion was initially proposed with the idea of splitting the attributes into stable attributes, flexible attributes and a decision attribute. In the earlier research, however the action rules were produced using some of the classification rules. Ras and Wyrzykowska [1] later gave a LERS [3] type of algorithm that considers only marked certain rules to construct the action rules. A new algorithm named ARoGS was proposed by Ras and Wyrzykowska [2] which combines each action rule generated from single classification rule with the remaining stable attributes to offer more action rules. This helps discover more action rules that were missed out in the previous algorithms.

## III. Methodology

We implemented the Random-Forest algorithm for action rules discovery using Hadoop framework and Google MapReduce. We take as an input a set of files: the data, the attribute names, user specified parameters such as: minimum support, and confidence thresholds, stable attribute names, flexible attribute names, decision attribute choice, decision attribute value to *change_from*, and decision attribute value to *change_to*, which is the desired value of decision attribute. We import these input files into the HDFS (Hadoop Distributed File System).

### A. MapReduce

MapReduce is a way to perform special-purpose computations that process large amounts of raw data to compute various kinds of derived data. MapReduce handle issues like how to parallelize the computation, distribute the data and handle failures. Though such computations are conceptually straightforward, the input data is generally large and the computations have to be distributed across hundreds of machines in order to finish the task in a reasonable amount of time. MapReduce is a new abstraction that allows expressing the simple computations and hides the messy details of parallelization, fault tolerance etc.

The abstraction is inspired by the *map* and *reduces* primitives present in Lisp and many other functional languages. Most of their computations involved applying a map operation to each logical record in input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately. The intermediate values are supplied to the user's reduce function via an *iterator* which allows to handle lists of values that are too large to fit in memory. The programming model can also be used to parallelize computations across multiple cores of the same machine.

The special features of MapReduce are:

*Fault Tolerance*: Worker failures are very well handled with the help of the master by periodically checking the responses from the workers. This mechanism helped in solving the issue of 80 unreachable machines.

*Locality:* The MapReduce master tries to schedule a map task on a machine that contains a replica of the corresponding input data so as to consume no network bandwidth - which is relatively scarce resource in our computing environment.

*Task Granularity:* The map and reduce phases should be considerably more than the worker threads so that having each worker perform many different tasks improves dynamic load balancing and also speeds up recovery when a worker fails.

*Backup Tasks:* There might be cases where a machine might take unusually more time due to some performance issues which delays the task assigned to that machine. But the mechanism of scheduling backup executions of remaining in-progress tasks by the master will significantly improve the execution time.

After including the locality optimization, dynamic load balancing of task execution across work machines, there has a broad use of MapReduce across a wide range of domains within Google. i.e. large-scale machine learning problems, clustering problems, extracting data to produce results for popular queries, processing of satellite imagery data, large scale graph computations and so on. The most significant use of MapReduce to date is a complete rewrite of the production indexing system that produces the data structures used for the Google Web search service.

### B. LERS

The LERS (Learning from Examples based on Rough Sets) method is used to find all certain and possible rules in a distributed scenario using MapReduce. The system induces rules from databases. Rules indication is based on four different algorithms. Two algorithms are used for indicating rules in minimal discriminant form, remaining two are used for discovering of all potential rules hidden in a database. Input data may be imperfect or data may contain errors like missing attributes, numerical attributes and inconsistencies. LERS classifies new unseen cases using principles similar to clustering. LERS is also equipped with a tool for multiple-fold cross validation. The system has been in medical area, nursing, global warming, environmental protection, natural languages and data transmission. LERS may process big datasets and frequently outperforms not only other rule induction system but also human experts. The decision table represents input data, gathered from any domain. In the decision table examples are described by values of attributes (e.g., tests) and characterized by a value of a decision, assigned by an expert. All examples with the same value of the decision belong to the same concept. The decision table is similar in format to a data base. The system LERS looks for regularities in the decision table. The output of LERS are rules, a knowledge about original data. These rules are more general than information contained in the original decision table, since in general more new examples may be correctly classified by rules than may be matched with examples from the original decision table. The system LERS belongs to the class of machine learning systems from examples. LERS, like other such systems, is finding a minimal description of a concept, described by 2 positive examples, excluding from the description of the concept the remaining, negative examples. Thus rules induced by LERS may be described as a minimal discriminant description of the concept. [6]

## *C. ARoGS*

ARoGS is Action Rules Discovery Based on Grabbing Strategy, which uses LERS. ARoGS are applicable for complete information systems. To extract the action rules, ARoGS use LERS to skip verifying few of the certain relations. The complexity of the algorithm decreases as the module is pre-processed in LERS and the classification rules are defined.

Let us assume $S = (U, A_1 \quad A_2 \quad \{d\})$ is a complete decision system, where

Elements of $A_1$ are stable conditions

Elements of $A_2$ are flexible conditions

d is the decision attribute and d $\quad A_1 \quad A_2$

Assume $d_1 \quad V_d$ and $x \quad U$. We say that $x$ is a $d_1$-object if $d(x) = d_1$

Also assume that $\{a_1, a_2, ..., a_p\} \quad A_1$, $\{a_{p+1}, a_{p+2}, ...a_n\}=A_1 - \{a_1, a_2, ..., a_p\}$, $\{b_1, b_2, ..., b_q\} \quad A_2$, $a_{[i,j]}$ denotes a value of attribute $a_i$, $b_{[i,j]}$ denotes a value of attribute $b_i$, for any $i, j$ and that

$$r = [[a_{[1,1]} \quad a_{[2,1]} \quad ... \quad a_{[p,1]} \quad b_{[1,1]} \quad b_{[2,1]} \quad ..... \quad b_{[q,1]}] \longrightarrow d_1]$$

is a classification rule extracted from $d_1$-objects in $S$. d1 is the preferable class and the expected result is to reclassify $d_2$ to $d_1$. Using the above classification rule we can reclassify $d_2$ as

$$r[d_2 \longrightarrow d_1] = [[a_{[1,1]} \quad a_{[2,1]} \quad ... \quad a_{[p,1]} \quad (b_1, \longrightarrow b_{[1,1]}) \quad (b_2, \longrightarrow b_{[2,1]}) \quad ..... \quad (b_q, \longrightarrow b_{[q,1]})] \longrightarrow (d, d_2 \longrightarrow d_1)]$$

Similarly the action rule schema is defined as

$$r[\longrightarrow d_1] = [[a_{[1,1]} \quad a_{[2,1]} \quad ... \quad a_{[p,1]} \quad (b_1, \longrightarrow b_{[1,1]}) \quad (b_2, \longrightarrow b_{[2,1]}) \quad ..... \quad (b_q, \longrightarrow b_{[q,1]})] \longrightarrow (d, \longrightarrow d_1)]$$

Now we generate the set of Action Rules from the available schema r[d2 →d1 ] .

```
Algorithm $AR(r, d_2)$
    i:=1
  while $i \leq n + q$ do begin
    j:=2;  m:=1
  while $j < J(i)$ do begin

if $[h[r[d_2 \longrightarrow d_1]] \quad c_{(i,j)}] \quad U_{[r,d_2]} \quad c_i \quad A_2$ then
    begin
    mark[$c_{(i,j)}$];

output Action Rule
 $[[h[r[d_2 \longrightarrow d_1]] \quad (c_i, c_{(i,j)} \longrightarrow c_{(i,1)})] \longrightarrow [d, d_2 \longrightarrow d_1]]$
    end

    if $[h[r[d_2 \longrightarrow d_1]] \quad c_{(i,j)}] \quad U_{[r,d_2]} \quad c_i \quad A_1$ then
    begin
    mark[$c_{(i,j)}$];

    output Action Rule
    $[[h[r[d_2 \longrightarrow d_1]] \quad (c_i, c_{(i,j)})] \longrightarrow [d, d_2 \longrightarrow d_1]]$
    end
    j:=j+1
  end end
   $I_k := \{i_k\}$;
   (where $i_k$ - index randomly chosen from $\{2, 3,..., q + n\}$).
   for all $j_k \leq J(i_k)$ do $[c_{(i_k,j_k)}]_{i_k \quad I_k} := c(i_k, j_k)$;
   for all $i, j$ such that both sets $[c_{(i_k,j_k)}]_{i_k \quad I_k}, c_{(i,j)}$
    are not marked and
   $i \quad I_k$
   do
   begin
    if $[[h[r[d_2 \longrightarrow d_1]] \quad [c_{(i_k,j_k)}]_{i_k \quad I_k} \quad c_{(i,j)}]] \quad U_{[r,d_2]} \quad c_i \quad A_2$ then
   begin
   mark $[[c_{(i_k,j_k)}]_{i_k \quad I_k} \quad c_{(i,j)}]$;

   output Action Rule
```

```
    $[[h[r_{[d_2 \longrightarrow d_1]}]\quad [c_{(i_k,j_k)}]_{i_k}\quad I_k\quad (c_i,\ c_{(i,j)} \longrightarrow c_{(i,1)})]$
    $\longrightarrow [d,\ d_2 \longrightarrow d_1]]$
    end

if $[[h[r_{[d_2 \longrightarrow d_1]}]\quad [c_{(i_k,j_k)}]_{i_k}\quad I_k\quad c_{(i,j)}]]\quad U_{[r,d_2]}$
  $c_i\quad A_1$
    then
    begin
    mark $[[c_{(i_k,j_k)}]_{i_k}\quad I_k\quad c_{(i,j)}]$;

output Action Rule
     $[[h[r_{[d_2 \longrightarrow d_1]}]\quad [c_{(i_k,j_k)}]_{i_k}\quad I_k\quad (c_i,\ c_{(i,j)})] \longrightarrow$
$[d,\ d_2 \longrightarrow d_1]]$
    end
    else
    begin
    $I_k := I_k\quad \{i\};\ [c_{(i_k,j_k)}]_{i_k}\quad I_k := [c_{(i_k,j_k)}]_{i_k}\quad I_k\quad c_{(i,j)}$
    end
```

In most of the classification techniques the classification rules are grouped into clusters of non-conflicting rules. Then all the possible pairs of classification rules in the clusters are used to build the action rules. In ARoGS algorithm the target decision is treated as a seed and then it grabs the remaining classification rules describing them as non-target values and builds decision rules from them. Rules grouped into a seed are only compared with that seed. Therefore, the number of pairs of rules which have to be checked gets reduced.

The confidence of generated action rules depends on the number of remaining objects supporting them.

If $Sup(r_{[d_2 \to d_1]}) =$ , then $r_{[d_2 \to d_1]}$ can not be used for reclassification of objects. Similarly, $r_{[\to d_1]}$ can not be used for reclassification, if $Sup(r_{[d_2 \to d_1]}) =$ , for each $d_2$ where $d_2 = d_1$.

## IV. EXPERIMENT AND RESULTS

For execution of our codes of Forest Algorithm to find association rules we choose two data sets: Mammographic Mass Data and Car Evaluation Database. Both are obtained from UCI Machine Learning Repository[5].

We chose UNC Charlotte DSBA cluster as our target server for execution. The Java application was exported as a JAR file and copied on to DSBA cluster. The data sets mentioned above are also copied to cluster server.

Command line interface on DSBA server for all the operations required for the experiment. The predominant commands used were Put, JAR.

University of North Carolina at Charlotte is configured to have 72 nodes. The default size of Hadoop blocks is 64Mb. We did not change this setting for our program execution.

We had two files for each type of data sets chosen. One file for attributes and one file with entire data set. Attributes file is space separated file.

The format of action and association rules output is as stated below:

(flexibleAttribute1,flexibleAttribute1_ValueX→flexibleAttribute1_ValueY)^(flexibleAttribute2,flexibleAttribute2_ValueA → flexibleAttribute1_ValueB)^(stableAttribute1,stableAttribute_ValueC→stableAttribute_ValueC) ⇒ (decisionAttribute,decisionAttribute_valueF→decisionAttribute_valueK) [Support : N1 , Confidence : N2%]

Meaning when flexibleAttribute1 changes its value from X to Y and flexibleAttribute2 changes value from A to B and stableAttribute1 retains value as C implies that decisionAttribute changes value from F to K with support N1 and confidence being N2%.

Let's see individual data set runs results:

### *A. Mammographic data set:*

Mammographic data set is donated by M. Elter.This data set can be used to predict the severity (benign or malignant) of a mammographic mass lesion from BI-RADS attributes and the patient's age. This dataset has 6 attributes namely : birads[4], age, shape, margin, density and severity.This data set has 961 instances.[5]

In one of the execution iteration, birads, age, shape were kept as stable attributes. margin, density were kept as flexible attributes. 'severity' was chosen to be our decision attributes. Minimum support was chosen to be 2 and minimum confidence to be 70%.

Our goal is to find results for action and association rules for the action severity changing from benign to malignant i.e 0 to 1. Which means we need to find the changes that required in flexible attributes if the severity is to change from 0 to 1 for minimum support 2 and confidence 70%.

After running MR Forest algorithm program on -mammographic data set we got 2 folders . One for action rules ARoGs output and one for association action rules output.

An example of result for ARoGS is as follows.

(margin,marginN → margin4)^(age,age63 → age63) ⇒ (severity,severity0 → severity1) 2 [Support: 2.0, Confidence: 100.0%]

This states if margin value changes from N to 4 and age remains same at 63 that implies severity would change from 0 to 1. This action would have support of 2 and confidence 100%.

An example of result for association action rule is as follows.

(shape,shape3→shape3)^(density,density3→density3)^(margin,margin1→margin5) ⇒ (severity,severity0→ severity1) [Support: 2.0, Confidence: 90.0%]

This states if shape and density values remain as they are and margin changes from 1 to 5 , severity can change from 0 to 1 with 90% confidence and support being 2.

Results from log:

| | **ARoGS** | **AAR** |
|---|---|---|
| Total no. of counters | 49 | 50 |
| FILE: Number of bytes read | 3055 | 2331 |
| FILE: Number of bytes written | 763850 | 762769 |
| FILE: Number of read operations | 0 | 0 |
| FILE: Number of large read operations | 0 | 0 |
| FILE: Number of write operations | 0 | 0 |
| HDFS: Number of bytes read | 21665 | 21665 |
| HDFS: Number of bytes written | 1648 | 2216 |
| HDFS: Number of read operations | 18 | 18 |
| HDFS: Number of large read operations | 0 | 0 |
| HDFS: Number of write operations | 2 | 2 |
| Launched map tasks | 5 | 5 |
| Launched reduce tasks | 1 | 1 |
| Data-local map tasks | 5 | 4 |
| Rack-local map tasks | - | 1 |
| Total time spent by all maps in occupied slots (ms) | 171822 | 112118 |
| Total time spent by all reduces in occupied slots (ms) | 19730 | 27956 |
| Total time spent by all map tasks (ms) | 85911 | 56059 |
| Total time spent by all reduce tasks (ms) | 9865 | 13978 |
| Total vcore-seconds taken by all map tasks | 85911 | 56059 |
| Total vcore-seconds taken by all reduce tasks | 9865 | 13978 |
| Total megabyte-seconds taken by all map tasks | 175945728 | 114808832 |
| Total megabyte-seconds taken by all reduce tasks | 20203520 | 28626944 |
| Map input records | 500 | 500 |
| Map output records | 180 | 115 |
| Map output bytes | 22611 | 14022 |
| Map output materialized bytes | 3607 | 3053 |
| Input split bytes | 660 | 660 |
| Combine input records | 0 | 0 |
| Combine output records | 0 | 0 |
| Reduce input groups | 169 | 96 |
| Reduce shuffle bytes | 3607 | 3053 |
| Reduce input records | 180 | 115 |

| | | |
|---|---|---|
| Reduce output records | 11 | 16 |
| Spilled Records | 360 | 230 |
| Shuffled Maps | 5 | 5 |
| Failed Shuffles | 0 | 0 |
| Merged Map outputs | 5 | 5 |
| GC time elapsed (ms) | 1870 | 1752 |
| CPU time spent (ms) | 31060 | 10500 |
| Physical memory (bytes) snapshot | 3922235392 | 3372052480 |
| Virtual memory (bytes) snapshot | 23549698048 | 23509757952 |
| Total committed heap usage (bytes) | 4859625472 | 4585422848 |
| File Input Format Counters Bytes Read | 21005 | 21005 |
| File Output Format Counters Bytes Written | 1648 | 2216 |

## A. *Car Evaluation Database*

Car Evaluation Database is donated by Marko Bohanec and Blaz Zupan. It was derived from a simple hierarchical decision model originally developed for the demonstration of DEX (M. Bohanec, V. Rajkovic: Expert system for decision making. Sistemica 1(1), pp. 145-157, 1990.)

Because of known underlying concept structure, this database may be particularly useful for testing constructive induction and structure discovery methods. It has 4 classes for Classification. Documentation on everything 1728 instances, 6 attributes ordered nominally. The data set doesn't have any missing values. [5] The 6 attributes in the dataset are : buying, maint, doors, persons, lug_boot, safety, label.

In one of the execution iteration we chose buying, maint, doors, persons as stable attributes; lug_boot and label were kept as flexible attributes . 'safety' was chosen to be the decision attribute.

Our goal was to check for action rule : (safety , safetyLow → safetyHigh) meaning what factor of change in attributes impact the car safety to change from low to high .

After running MR Random forest algorithm below are the results.

Example of a resulting action rule :

(maint,maintlow→maintlow)^(persons,persons4→persons4)^(label,labelunacc → labelgood)^(lug_boot,lug_bootsmall → lug_bootsmall) ⇒ (safety,safetylow → safetyhigh) 2 [Support: 4.0, Confidence: 100.0%]

Above statement means when maint changes from low to high and no. of persons remains at 4 as constant and label changes from unaccounted to good and lug_boot remains small then it implies safety measure has increased from low to high with Support 4 and confidence 100%.

Example of association action rule output :

(persons,personsmore→personsmore)^(label,labelunacc → labelvgood) ⇒ (safety,safetylow → safetyhigh) [Support: 17.5, Confidence: 81.23%]

The above-mentioned rule states if no. of persons remains more and label changes from unaccounted to very good then it implies that safety measure changes from low to high with support of 17.5 and confidence 81.23%

Results from log:

| | **ARoGS** | **AAR** |
|---|---|---|
| Total no. of counters | 50 | 49 |
| FILE: Number of bytes read | 3664 | 12871 |
| FILE: Number of bytes written | 765263 | 784243 |
| FILE: Number of read operations | 0 | 0 |
| FILE: Number of large read operations | 0 | 0 |
| FILE: Number of write operations | 0 | 0 |
| HDFS: Number of bytes read | 156140 | 156140 |
| HDFS: Number of bytes written | 3515 | 21281 |
| HDFS: Number of read operations | 18 | 18 |

| | | |
|---|---|---|
| HDFS: Number of large read operations | 0 | 0 |
| HDFS: Number of write operations | 2 | 2 |
| Launched map tasks | 5 | 5 |
| Launched reduce tasks | 1 | 1 |
| Data-local map tasks | 4 | 5 |
| Rack-local map tasks | 1 | |
| Total time spent by all maps in occupied slots (ms) | 509582 | 213854 |
| Total time spent by all reduces in occupied slots (ms) | 22436 | 22424 |
| Total time spent by all map tasks (ms) | 254791 | 106927 |
| Total time spent by all reduce tasks (ms) | 11218 | 11212 |
| Total vcore-seconds taken by all map tasks | 254791 | 106927 |
| Total vcore-seconds taken by all reduce tasks | 11218 | 11212 |
| Total megabyte-seconds taken by all map tasks | 521811968 | 218986496 |
| Total megabyte-seconds taken by all reduce tasks | 22974464 | 22962176 |
| Map input records | 1728 | 1728 |
| Map output records | 201 | 201 |
| Map output bytes | 38090 | 38090 |
| Map output materialized bytes | 4501 | 4501 |
| Input split bytes | 535 | 535 |
| Combine input records | 0 | 0 |
| Combine output records | 0 | 0 |
| Reduce input groups | 183 | 183 |
| Reduce shuffle bytes | 4501 | 4501 |
| Reduce input records | 201 | 201 |
| Reduce output records | 17 | 17 |
| Spilled Records | 402 | 402 |
| Shuffled Maps | 5 | 5 |
| Failed Shuffles | 0 | 0 |
| Merged Map outputs | 5 | 5 |
| GC time elapsed (ms) | 3983 | 3983 |
| CPU time spent (ms) | 191430 | 191430 |
| Physical memory (bytes) snapshot | 5157806080 | 5157806080 |
| Virtual memory (bytes) snapshot | 23554592768 | 23554592768 |
| Total committed heap usage (bytes) | 5950668800 | 5950668800 |
| File Input Format Counters Bytes Read | 155605 | 155605 |
| File Output Format Counters Bytes Written | 3515 | 3515 |

## V . CONCLUSION

In this activity, we implemented Random Forest algorithm from MapReduce framework of Hadoop big data technology to calculate Action Rules based on Grabbing Strategy and Association action rules. Comparing results, we observe that ARoGS job took lesser time than AAR job overall and less complex. The action rules calculation helped us to study how can we can extract novel knowledge by action rules. For car data we could train system and learn how safety measure would increase by other features and for mammographic data we could analyze what parameters reduce the possibility of having malignant tumor.

Since the data set used was between 900-1800 and the number of nodes were 72, it is clear from empirical result that more the nodes, better the performance of action rules jobs. The practical usage of this can be to analyze data of an e-commerce system or shopping site application database to predict possible behavior of customers and gain profit by giving customers recommendations or make them buy similar things grouping the things together. For larger data sets the data block size may further have to be increased to optimize performance. Not only these, there are many other fields with huge database can be benefited from action rules such as medical, financial , stock market , insurance and astrology.

## VI .REFERENCES

[1] Ras´ Z, Gupta S. Global action rules in distributed knowledge systems. Fundamenta Informaticae J 2002;51:175–184.

[2] M. Lichman (2013), UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[3] J. W. Grzymała-Busse, S. R. Marepally, Y. Yao (2013), An Empirical Comparison of Rule Sets Induced by LERS and Probabilistic Rough Classification , in Rough Sets and Intelligent Systems, Vol. 1, Springer, pp. 261-276.

[4] BI-RADS is an acronym for Breast Imaging-Reporting and Data System, a quality assurance tool originally designed for use with mammography. The system is a collaborative effort of many health groups but is published and trademarked by the American College of Radiology (ACR) : https://en.wikipedia.org/wiki/BI-RADS

[5] M. Lichman (2013), UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[6] R. S. Michalski: A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, T. M. Mitchell, eds. Machine Learning, Morgan Kaufmann 1983, 83–134.