International Journal of Data Mining & Knowledge Management Process (IJDKP), Vol.13, No. 1/2, March 2024

DISCOVERY OF ACTIONABLE PATTERNS THROUGH SCALABLE VERTICAL DATA SPLIT METHOD WITH META ACTIONS AND INFORMATION GRANULES

Angelina A. Tzacheva[1] and Sanchari Chatterjee[2] and Zbigniew W. Ras[2]

[1] Computer Science and Information Technology
College of Computing and Engineering,
WestCliff University
Irvine, CA 92614,USA

[2]Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC, 28223, USA

ABSTRACT

Action Rules are rule based systems that extract actionable patterns which are hidden in big volumes of data. Users need recommendations on actions they can undertake to increase their profit or accomplish their goals, this recommendations are provided by Actionable patterns. In the technological world of big data, massive amounts of data are collected by organizations, including in major domains like financial, medical, social media and Internet of Things(IoT). To analyze and store such a massive amount of data, distributed computing frameworks like Hadoop and Spark are introduced to store the big data in a distributed fashion which manage and analyze them in parallel. The traditional Action Rules extraction models, which analyze the data in a non-distributed fashion, do not perform well when dealing larger datasets. Serious complications of discovering Action Rules with such distributed environments are - data distribution among computing nodes and calculation of major parameters including : support, confidence, utility, and coverage, that represent the whole data. Information granules form basic entities in the world of Granular Computing(GrC), which represents meaningful smaller units derived from a larger complex information system. In this research, we focus on the data distribution phase of the distributed Actionable Pattern Mining problem. To handle the data distribution task by splitting the big data in both horizontal and vertical fashions - we propose partition threshold rho. In this work, we concentrate on using information granules to implement a vertical data splitting strategy with Meta Actions. Hence our results discover valuable Actionable Knowledge with application in Business and Education domains.

KEYWORDS

Action Rules, Meta Action, Information granules, Data distribution, Spark

1. INTRODUCTION

The modern world is full of data where extracting these data creates new opportunities in the field of business, this allows the business to generate a valuable data which can be used to analyse the customer patterns. The paper contains the study about data mining and different business intelligence that will benefit out of data mining implementation [2]. Nowadays it is highly recommended to use the data mining for business, to achieve more desirable state and higher profits [13]. Granular Computing (GrC) is a domain that makes use of information granules for solving complex humancentric problems [32]. Although, the topic of Granular Computing is being used since its birth by Zadeh [33] in 1997, its ideas are widely used in multiple domains like machine learning, attribute selection, rough set theory, decision trees, artificial intelligence, etc. With the key idea of information granules, Granular Computing can also be used in Knowledge representation and data mining.

DOI:10.5121/ijdkp.2023.14202

Information granules are a collection of granules, where each *granule* is a set of data objects are stacked together based on their similarity, functionality or indistinguishability [33].
Thus, a granule can be seen as a subset of a larger problem, that can be used effectively to solve a complex task. Information granulation is the process of breaking a complex object into smaller pieces called information granules. Information granulation thus can solve more complex problems by considering meaningful levels of granularity of the problem [31].
Discovering surprising, unknown, and useful knowledge from a massive data in the crucial task of data mining. Most of the data mining or machine learning algorithms manifest the relationship of data objects with other objects (Clustering) or classes (Classification). The Rule based learning tasks intend to circumscribe methods that identifies, learns, or evolves 'rules' to store and manipulate knowledge. In the field of data mining, many algorithms are available to generate rules which are used for association - to find frequently associated patterns in the data and classification - to classify patterns to one or more classes. Rules takes the format as given in Equation 1, where the *antecedent* (left side of the rule) is a conjunction of conditions, and the *consequent* (right side of the rule) is a resulting pattern for the conditions in antecedent.

$$condition(s) \rightarrow result(s) \qquad (1)$$

Action Rule is a rule based data mining technique that recommend possible transitions of data from one state to another, which the user can use to their advantage. For example, one would want to find actionable patterns in the data to improve his/her salary. Some of the applications for Action Rules are: improving customer satisfaction in business [15] and reducing hospital readmission in the medical field [4]. Action Rules are extracted from Decision table [25]. The data can be viewed as decision tables when the attribute space of the data can be split into *Stable Attributes* and *Flexible Attributes* along with the *Decision attribute* which is the final decision that the user need to achieve [25]. Stable attributes in any Action Rule *AR* remain constant or cannot form action in *AR*. While flexible attributes can change their value from a_i to a_j. Action Rules can take the representation as given in Equation 2, where Ψ represents a conjunction of stable features, $(\alpha \rightarrow \beta)$ represents a conjunction of changes in values of flexible features and $(\theta \rightarrow \varphi)$ represents desired decision action.

$$[(\Psi) \wedge (\alpha \rightarrow \beta)] \rightarrow (\theta \rightarrow \varphi) \qquad (2)$$

More than a decade, lot of research have been conducted over Action Rules extraction giving rise to several algorithms like DEAR [29], ARAS [26] and Association Action Rules [23]. These algorithms extract Action Rules in an expected time frame when the dataset size is limited, which is not the case these days. Limited research has been done on extracting Action Rules in a distributed scenario. Some methods like MR-Random Forest [30] and SARGS [5] have been proposed to introduce the concept of distributed Action Rule discovery. The ultimate challenges in extracting Action Rules in a distributed fashion is that distribution of data among the computation nodes has to be done in such a way that there is minimum loss of actionable knowledge extracted from the distributed data. In this paper, we propose a granularity-based method to handle the data distribution task and extract a intermediate stage of meta actions to generate the Action Rules efficiently using a popular distributed computing called Spark [34]. We will apply our algorithm to datasets in the domains of transportation, medical and business.

2. RELATED WORK

In modern days most of the companies are concern about Customer churn, which is the major factors for the companies to leads higher customer acquisition cost, lower volume of service consumption and reduce product purchase. Therefore, companies need to take effective and critical strategies to reduce customer outflow. Authors of paper [8] introduces high quality action rules that can provide valid and trustworthy recommendations to improve customer churn rate. They proposed

a Semantic-aided Customer Attrition Management System (SaCAMS), in which they use reducts for feature engineering, then applied hierarchical clustering to build the semantic similarity relationship among clients, also, depicted action rule mining to discover the actionable patterns, and extracted metaactions to get the final recommendations. The experimental results showed that not only SaCAMS can discover high quality action rules, but also can extract effective metaactions to generate recommendations,based on the improved action rules. Their proposed method SaCAMS utilizes meta-node to provide decision-makers with efficient,valid and trustworthy strategies, which are quantified by effectiveness scores. The authors for this paper gathered customer feedback data and transaction data on their satisfaction in heavy equipment repair and service sector from a consulting company in Charlotte. The dataset contains over 400,000 survey records collected from 34 heavy equipment repair companies from 2011 to 2017.

In paper [9] authors Duan and Ras explain about customer churn which is a major issue to most companies. They propose a recommender system that is utilizing action rule mining technology, and they show its great value in reducing customer churn. In action rule mining, confidence, support, and coverage are used to measure the quality of the discovered action rules. Action rules with higher confidence and support are more useful to users. To improve the quality of action rules extracted from a given client, their research paper proposes a guided (by threshold) agglomerative clustering algorithm by utilizing the knowledge extracted from semantically similar clients. The main idea is to pick up only such clients that are doing better in business than another comparable client that is semantically similar. By doing that, the given client can follow business recommendations from the better-performing clients. The algorithm is guided by the threshold value. It checks how large is the confidence improvement of the discovered action rules. The algorithm stops if the improvement is lower than this threshold. Authors of the paper [1], focus on identifying the customers with high chance of attrition and provide valid and trustworthy recommendations to improve their customer churn rate. To this end, authors design and implement a recommender system that can actually provide actionable recommendations to improve customer churn rate. Authors use both transaction and survey data from heavy equipment repair and service sector from 2011 to 2017. This data is collected by a consulting company based in Charlotte, North Carolina. In the survey data, customers give their thoughts, feelings, expectations and complaint by free-form text. They apply aspect-based sentiment analysis on the review text data to gain insightful knowledge on customers' attitudes toward the service. Action rule mining and meta-action triggering mechanism are used to recognize the actionable strategies to help with reducing customer churn.

Although Granular computing was proposed by Zadeh [33] purely to represent human cognition, the idea of the topic has been adopted in multiple problems like decision trees [19], divide and conquer [28], set theory, data reduction or compression, discretization [14], etc. One of the applications of information granules are finding optimal solution that satisfies certain human assigned conditions, which can be imprecise [28]. Granular computing has been used in various applications like in image processing, processing large amount of high pixel images takes a lot of computation power. Thus instead of analyzing all pixels in an image, we can group some pixels into semantically meaningful clusters or granules. Li, et.al [17] proposed such a technique for cost effective face recognition. Recently, Liu, et.al [22] used information granules for time series models.
A rule based learning is a strategy used in data mining process to make decision making. Rule based systems are used popularly in multiple machine learning methods like classification, regression and association [21]. There are decent amount of work on relating rule based systems with information granules. Rule based systems used for searching, extraction and representation of knowledge find great use of information granules to do their tasks efficiently [19]. Liu, et.al [20] considers each rule from the learning algorithm as a granule and use such granules for rule generation, rule simplification and rule representation. They also give a tree representation for their granular structure. Drawbacks of rule based systems include recommending many rules that user cannot evaluate easily. Ahmad and Pedrycz [3] used information granules to reduce the rule set size, which can be helpful in evaluating and representing them.

The perception of Action Rules is first introduced in 2000, when Ras and Alicia proposed an idea of Action Rules help many businesses to gain profit by finding interesting actionable patterns in the

data [25]. In the literature, Action Rules are extracted using two methods. First is a rule based approach, in which intermediate classification rules are extracted first using efficient rule generation algorithms such as LERS or ERID. From these extracted rules, action rules are generated with systems like DEAR [29], which extracts Action Rules from two classification rules, or ARAS [26], which extracts Action Rules using a single classification rule. Second method is object-based approaches, in which the Action Rules are extracted directly from the decision table without any intermediary steps. Systems ARED [12] and Association Action Rules [23] works in the object-based approach. Out of all algorithms, only Association Action Rules [23] extracts all possible Action Rules from the given decision table but it is inefficient in terms of time to extract rules. Other algorithms runs much faster with the aim of extracting rules that are benefits the user to the maximum and extracts only limited recommendations.

Considering the growth of big data, some research [30] [5] has been done to extract Action Rules using popular distributed computing frameworks such as MapReduce [7] and Spark[34]. The main challenge involved in distributed processing of rule based data mining is load balancing and obtaining global optimum results. [30] proposed a method to distribute the data in random to multiple sites, gather results from all sites and take an average on parameters like Support and Confidence. [5] handle the load balancing by clustering the data into d partitions, where d is the number of unique values of a decision attribute. Data rows that contains d_i belongs the d_i cluster. Equal proportion of data is taken from each cluster to create n final partitions of the data. Although some Action Rules extraction methods are using some form of information granules, no interesting technique were proposed on distributing the data to multiple nodes.

In this paper, we propose a novel approach for partitioning the given data using information granules. We also give a new algorithm to extract meta action as the intermediate state before generation of all Action Rules, based on the algorithm proposed in [6] and [23]. We test how fast our new method works compared to our previous distributed Action Rule extraction algorithms and also we check validity of the new data distribution method by comparing number of Action Rules generated and rule coverage of Action Rules from system with classical Association Action Rules [23] on a single machine and SARGS [5] systems.

3. BACKGROUND KNOWLEDGE

In this section, we give basic knowledge about Decision system, Action Rules, Spark and GraphX frameworks to understand our methodology.

3.1. Decision System

Consider an information system given in Table 1

Information System can be represented as $S = (X,A,V)$ where, X is a nonempty, finite set of objects: $X=\{x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8\}$

A is a nonempty, finite set of attributes: $A = a,b,c,d$ and V_i is the *domain* of attribute a which represents a set of values for attribute $i : i \in A$. For example, $V_b = b_0,b_2$.
An information system becomes Decision system, if $A = A_{St} \cup A_{Fl} \cup d$, where d is a *decision attribute*. The user chooses the attribute d if they wants to extract desired action from $d_i : i \in V_d$. A_{St} is a set of *Stable Attributes* and A_{Fl} is a set of *Flexible Attributes*. For example, *ZIPCODE* is a Stable Attribute and *User Ratings* can be a Flexible Attribute.

Lets assume from Table 1 that $c \in A_{St}$. a, b $\in A_{Fl}$ and $d \in d$. and the decision maker desires Action Rules that triggers the decision attribute change from d_1 to d_2 throughout this paper for examples.

Table 1. Sample Decision System *S*

X	a	b	c	d
*x*1	*a*1	*b*1	*c*1	*d*1
*x*2	*a*3	*b*1	*c*1	*d*1
*x*3	*a*2	*b*2	*c*1	*d*2
*x*4	*a*2	*b*2	*c*2	*d*2
*x*5	*a*2	*b*1	*c*1	*d*1
*x*6	*a*2	*b*2	*c*1	*d*2
*x*7	*a*2	*b*1	*c*2	*d*2
*x*8	*a*1	*b*2	*c*2	*d*1

Table 2. Meta-actions Influence Matrix for S

	a	b	d
$M1, M2, M3$		$(b_1 \rightarrow b_2)$	$(d_1 \rightarrow d_2)$
$M1, M3, M4$	(a_2)	$(b_2 \rightarrow b_3)$	
M_5	(a_1)	$(b_2 \rightarrow b_1)$	$(d_2 \rightarrow d_1)$
$M2, M4$		$(b_2 \rightarrow b_3)$	$(d_1 \rightarrow d_2)$
$M1, M5, M6$		$(b_1 \rightarrow b_3)$	$(d_1 \rightarrow d_2)$

3.2. Action Rules

In this subsection, we give definitions of action terms, action rules and properties of action rules [25]

Let $S = (X, A \cup d, V)$ be a decision system, where d is a decision attribute and $V = \cup V_i : i \in A$. Action terms can be given by the expression of $(m, m_1 \rightarrow m_2)$, where $m \in A$ and $m_1, m_2 \in V_m$. $m_1 = m_2$ if $m \in A_{St}$.

In that case, we can simplify the expression as (m, m_1) or $(m = m_1)$. Whereas, $m_1 \neq m_2$ if $m \in A_{Fl}$ Action Rules can take a form of $t_1 \cap t_2 \cap .... \cap t_n$, where t_i is an atomic action or action term and the Action Rule is a conjunction of action terms to achieve the desired action based on attribute d. Example Action Rule for the Decision System in Table 1 is given below: $(a, a_1 \rightarrow a_2).(b, b_1 \rightarrow b_2) ==> (d, d_1 \rightarrow d_2)$

3.2.1. Properties of Action Rules

Action Rules are considered interesting based on the metrics such as Support, Confidence, Utility and Coverage. Higher these values, more interesting they are to the end user. Consider an action rule R of form:

$(Y_1 \rightarrow Y_2) ==> (Z_1 \rightarrow Z_2)$ where,

Y is the condition part of R

Y is the decision part of R

Y_1 is a set of all left side action terms in the condition part of R

Y_2 is a set of all right side action terms in the condition part of R

Z_1 is the decision attribute value on left side

Z_2 is the decision attribute value on right side

In [25], the support and confidence of an action rule R is given as

$Support(R) = min\{card(Y_1 \cap Z_1), card(Y_2 \cap Z_2)\}$

Confidence(R) = [card(Y1∩Z1)/ card(Y1)]. [card(Y2∩Z2)/card(Y2)]

Later, Tzacheva et.al [2] proposed a new set of formula for calculating Support and Confidence of Action Rules. Their idea is to reduce complexities in searching the data several times for Support and Confidence of an Action Rule. The new formula are given below.

$$Support(R) = \left\{ card(Y_2 \cap Z_2) \right\}$$

$$Confidence(R) = [\frac{card(Y_2 \cap Z_2)}{card(Y_2)}]$$

Tzacheva et. al [2] also introduced a notion of utility for Action Rules. Utility of Action Rules takes a following form. For most of cases Utility of Action Rules equals the Old Confidence of the same Action Rule.

$$Utility(R) = [\frac{card(Y_1 \cap Z_1)}{card(Y_1)}]$$

Coverage of an Action Rule means that how many decisions from values, from the entire decision system S, are being covered by all extracted Action Rules. In other words, using the extracted Action Rules, *Coverage* defines how many data records in the decision system can successfully transfers from Z_1 to Z_2 .

3.3. Meta Action

As an action rule can be seen as a set of atomic actions that need to be made happen for achieving the expected result, meta-actions are the actual solutions that should be executed to trigger the corresponding atomic actions, Table 2 shows an example of influence matrix which describes the relationships between the meta-actions and atomic actions influenced by them.
In Table 2, a, b and d are same attributes in decision system S as mentioned in previous section, and {M1, M2, M3, M4, M5, M6} is a set of meta-actions which hypothetically trigger action rules generated from S. Each cell in a row shows an atomic action that can be invoked by the set of meta-actions listed in the first column of that row. For instance, the first row shows that the atomic actions (b1 → b2) and (d1 → d2) can be activated by executing meta-actions {M1}, {M2} and {M3} together. Unlike the traditional influence matrix in [30] and [10] which involves only one single meta-action in each row, here the transaction of atomic actions in our domain relies on one or more meta-actions.
Clearly, an action rule can be triggered with the set of meta-actions listed in one single row of an influence matrix as long as all of its atomic actions are listed in that row. Otherwise, selecting a proper set of meta-actions combined from multiple rows becomes necessary. If we would like to activate action rule r given in previous section, it is quite obvious that the combination of meta-actions listed in the first and second row of Table 2 could trigger $\{r\}$, as meta-actions $\{M_1,M_2,M_3,M_4\}$ cover all required atomic actions $((a, a_1), ((b, b_1 \rightarrow b_2))$ and $(d, d_1 \rightarrow d_2)$ in $\{r\}$. On the other hand, one set of meta-actions could possibly trigger multiple action rules, like the meta action set $\{M_1,M_2,M_3,M_4\}$ triggers not only $\{r\}$ as mentioned but also action rule $[a, a_2, b, b_1 \rightarrow b_2 ==> (d, d_1 \rightarrow d_2)]$ by following the second and forth row in Table 2, if such action rule exists in $\{S\}$.

Suppose a set of meta-actions $M = (M_1, M_2,\ldots,M_n : n > 0)$ can trigger a set of action rules $(r_1, r_2,\ldots, r_m : m > 0)$ that covers certain objects in $\{S\}$. The coverage or support of $\{M\}$ is the summation of support of all covered action rules as shown below, which is the total number of objects in the initial state that are going to be affected by $\{M\}$. The confidence of $\{M\}$ is calculated by averaging the confidence of all covered action rules.
Furthermore, the effect of applying $\{M\}$ is defined as the product of its support and confidence ($\sup(\{M\}) \bullet \text{conf}(\{M\})$). It represents the number of objects in the system that are going to be improved by applying $\{M\}$ which also is used for calculating the increment of NPS rating caused by it. Therefore, greater is the effect of $\{M\}$, larger increment of NPS rating will be produced.

$$\sup(\{M\}) = \sum_{i=1}^{m} sup(\{r_i\})$$

$$\text{conf}(\{M\}) = \overline{M}\ \text{conf}(\{r_i\} : i = 1,\ldots, m)$$

$$(\sup(\{M\}) \bullet \text{conf}(\{M\}))$$

$$\text{frac sup}(\{r_i\} \div (\sum_{i=1}^{m} sup(\{r_i\})$$

3.4. Spark

Spark [34] is a framework that is similar to MapReduce [7] to process large quantity of data efficiently in a parallel fashion and in a short span of time. Spark introduces a distributed memory abstraction strategy named Resilient Distributed Datasets(RDD). The RDDs works by splitting the data into multiple partitions, do in-memory computations of the split partitions on several nodes in parallel and store the results in memory itself for future analysis. These results can be accessed for future processes and analyses, which in-turn create another RDD. Thus, Spark cuts-off large number of disk accesses for storing intermediate outputs like in Hadoop MapReduce. Spark functions in two stages: 1. *Transformation*, 2. *Action* During *Transformation* stage, computations are made on the data split and results are stored in-memory of the worker node. Results of all worker nodes together form another RDD. While the *Action* stage on an RDD collect results from all workers and send it to the driver node or save the results to a storage system.

Spark helps machine learning algorithms, by following multiple iterations over a single learning function, on the given data with the help of RDD's in-memory computation. Spark handles node failures at certain point in a long running process by constructing a lineage graph of RDDs. The lineage graph is a Directed Acyclic Graph(DAG) where each node represents a transformation stage. Figure 1 shows a sample lineage graph of combining RDDs from two inputs. When a failure occurs at a certain stage, Spark uses the last visited or working node (RDD) in the lineage graph and restart all computations from that node rather than repeating the entire process from the beginning or saving the intermediate results and replicating them across multiple nodes. Spark, with these strategies of data management, fault tolerance and in-memory processing makes Spark to do computations faster than MapReduce.

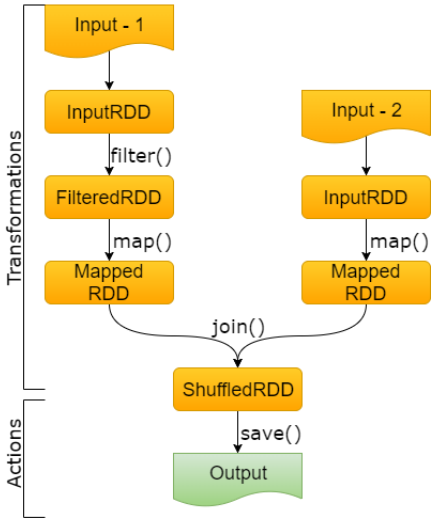


Figure.1. Spark Lineage Graph Example

4. METHODOLOGY

In this section we focus on discussing different methods adapted for extracting action rules.

4.1 Data distribution strategy based on granules

The basic advantage of information granularity is that we can break bigger problems into fine grained granules. Since our problem is with distribution of data, we incorporate information granules to our method. Algorithm 1 gives a brief description about the process we use to measure overlaps between 2 granules and sub granules in each granules. By granules, we mean a finite set of attributes from the attribute set A from the information system S. For our initial experiments, we consider only 2 granules from the information system S and measure overlaps of subgranules from each granule. We choose a granule combinations that has minimum overlap.

Algorithm 1 Granule Based Data Distribution

Require: partitions, dataSplit1, dataSplit2 *split1Sum* ← 0.0
for data1 in dataSplit1 **do**
subpartitions ←[ ]
subpartitionsCount ← 0 **for** data2 in dataSplit2 **do**
commonLines ← *data*1.*lines* ∩ *data*2.*lines* **if** *commonLines* 6= ∅ **then**
subpartitions.addAll(commonLines) *subpartitionsCount*+ = 1
if |*subpartitions*| == |*data*1.*lines*| **then**
*split*1*Sum*+ = 1*/subpartitionsCount*
break *split*2*Sum* ← 0.0
for data2 in dataSplit2 **do**
subpartitions ←[ ]
subpartitionsCount ← 0 **for** data1 in dataSplit1 **do**
commonLines ← *data*1.*lines* ∩ *data*2.*lines* **if** *commonLines* 6= ∅ **then**
subpartitions.addAll(commonLines) *subpartitionsCount*+ = 1
if |*subpartitions*| == |*data*2.*lines*| **then**
*split*2*Sum*+ = 1*/subpartitionsCount*
break
*split*1*Avg* = *split*1*Sum/*|*dataSplit*1|
*split*2*Avg* = *split*2*Sum/*|*dataSplit*2| **return** *split*1*Avg* − *split*2*Avg*

A brief description about our data distribution process has been given in Figure 2.

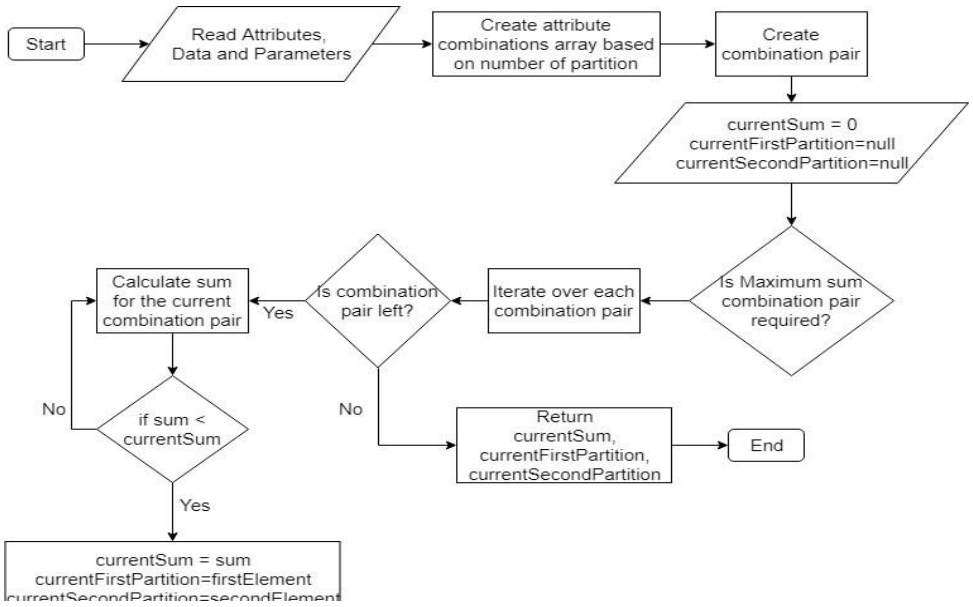


Figure.2. Data Distribution Strategy based on information granules

4.2. *Scalable association action rules extraction*

In this work, we follow a parallelized version of Association Action Rules [23] extraction technique in contrast to the parallel Action Rules extraction technique followed by horizontal data splitting approach. Association Action Rules extraction is an exhaustive A-Priori like method, which extracts all possible action rules by taking combinations of Action Terms through an iterative procedure. Due to this combinatorial process, Association Action Rules is considered to be a computationally expensive algorithm out of all Action Rules extraction techniques. The method does not scale very well with high dimensional data, and suffers from in the prospective of performance time. Therefore, this algorithm requires attention, and needs to be adapted for scalable and parallel computation. In this work, we create partitions by splitting the data by attributes in a high dimension data. We perform Association Action Rule extraction algorithm on each of those partitions in parallel, which allows for much faster computational time for Association Action Rules extraction in Cloud platforms.

On the positive side, this Association Action Rules algorithm extracts all possible Action Rules, while other algorithms, have a chance to lose large volume of valuable actionable rules. But this advantage comes with a cost of longer execution time than other Action Rules extraction algorithms. Association Action Rules algorithm is similar to Association Rules [11] extraction algorithm with A-priori method. Association Rules find patterns that occur most frequently together in the given data. The most popular algorithm for extracting Association Rules is the Apriori algorithm [27]. Apriori algorithm starts with 2 element pattern and continues n iterations until it finds n element patterns, where n is the number of attribute in the given data. Sample Association rule, which means that when a pattern a1 ∩ b2 occur together in the data, pattern (c1 ∩ d2) also occurs in the same data, are given below.
(a,a1) ∩ (b,b2) −⟶ (c,c1) ∩ (d,d2)
Action Rules also have relation to the Association Rules. When an actionable pattern (a,a1− > a2) ∩ (b,b1− > b2) ∩ .... ∩ (l,l1− > l2) occurs, where each term similar to (a,a1− > a2) is an atomic action, the actionable pattern based on the decision attribute d1 also occurs with minimum support(supp) and confidence(conf). Association Action Rules starts extracting patterns with just an atomic action term on the left side of the rule and decision action on the right side. The algorithm continues for maximum of n-1 iterations, where n is number of attributes in the data and gives all actionable patterns in the data until there are no more possible combinations to extract from data. We follow a strategy that provides very broad recommendations and that suits data that has higher attribute space. But the method works good with small data also. In this method, we split the data vertically into 2 or more partitions and with each partition the data can be split horizontally by the default settings of the Spark framework. Figure 3 presents an example vertical data partitioning with the sample Decision system in Table 1. Our algorithm runs separately on each partition, does transformations like map(), flatmap() functions and combine results with join() and groupBy() operations.

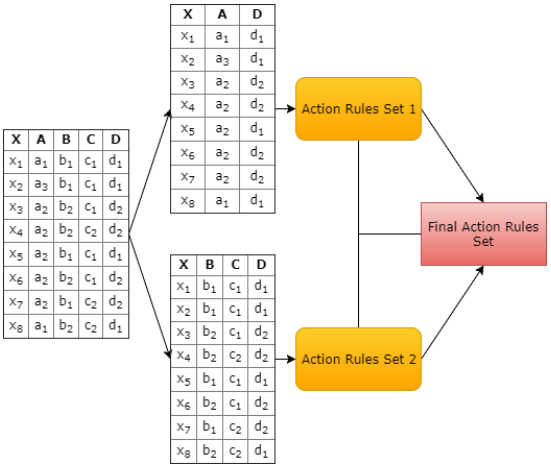


Figure.3. Example Vertical Data Distribution for Table 1

Algorithm 2 gives our new algorithm to extract all possible Action Rules from the data in a parallel fashion.

Algorithm 2 ActionRulesExtract

Require: data of type (r_{id},r_{values})) and d_{FROM},d_{TO} values **procedure** MyProcedure

2: d_A := ($s \in r | r \in data | (s,r_{id})$).*groupByKey*() *cOLD* ← *dA*

4: i ←*parallel :*

6: Generate META ACTION

while i 6= n **do**:

8: *c* ← *data.flatmap*(*r* => (*comb*(r_{values},*i*)),r_{id}) c_{NEW} ← *c.groupByKey*()

10: *cVALID* ← *cNEW.filter*()

cFROM ← *cVALID.filter*(*dFROM*)

12: *cTO* ← *cVALID.filter*(*dTO*) **if** c_{FROM} = ∅ or c_{TO} = ∅ **then break**

14: *atomic* ← c_{FROM}.*jon*(c_{TO}).*filter*() $action_{supp}$ ← (*r* ∈ *atomic* — *findSupp*(*r*)).*filter*()

16: **if** $action_{supp}$ = ∅ **then break**

$atomic_{FROM}$ ← *atomic.filter*()

18: $atomic_{TO}$ ← *atomic.filter*() *aFROM* ← *atomicFROM.join*(*cOLD*)

20: a_{TO} ← $atomic_{TO}$.*join*(c_{OLD})

actionconf ← *aFROM.join*(*aTO*)

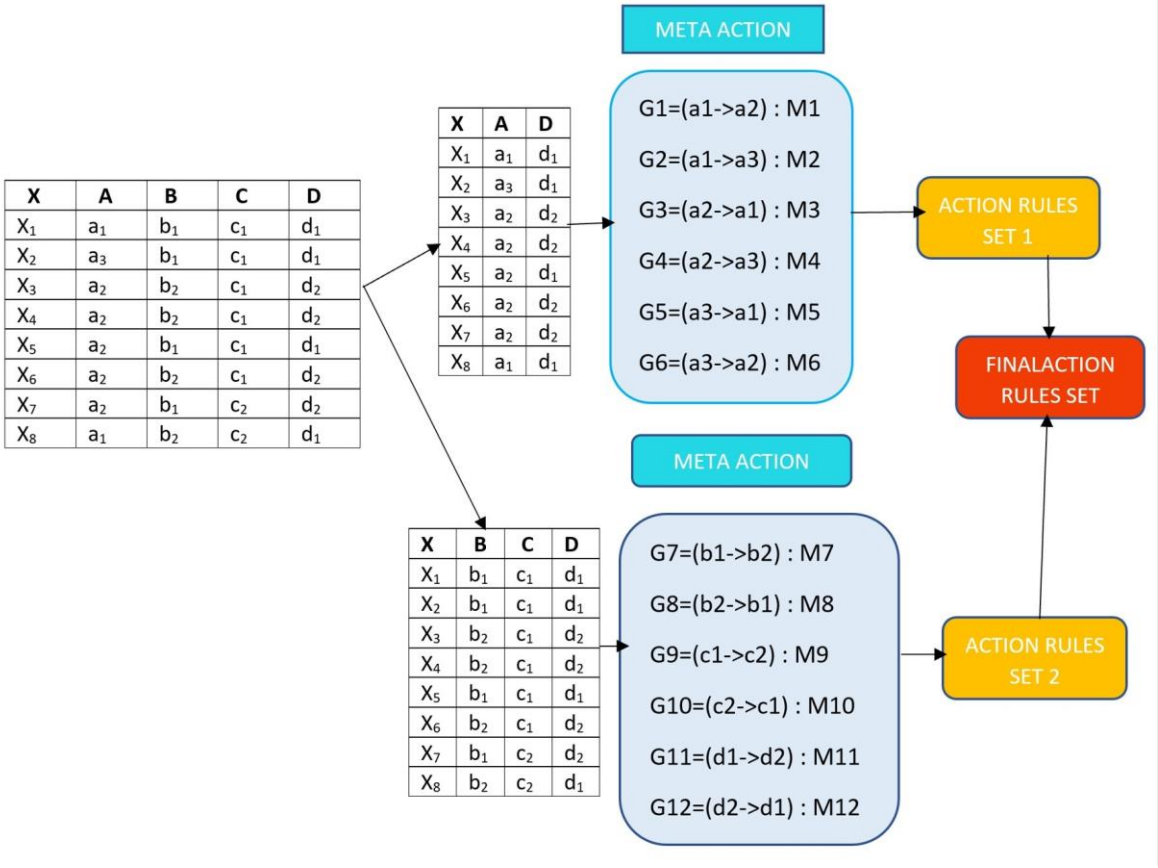
22: *actions* ← $action_{supp}$.*join*($action_{conf}$) collect *actions*

24: *cOLD* := *cNEW*

i := *i* + 1

Our algorithm gets the pre-processed data (rid,rvalues) as input, where rid is the row id and rvalues is a list of values for each record in the data. The algorithm also takes decision from (dFROM) and decision to (dTO) values as parameters. Step 2 of the algorithm gets all distinct attribute values and their corresponding data row indexes. This step involves a Map phase and a groupByKey phase of the Spark frameworks. We collect the data row indexes to find the Support and Confidence of Action Rules.

Finding Support and Confidence is an iterative procedure. It takes O(**nd**) times to collect Support and Confidence of all Action Rules, where **n** is the number of Action Rules and **d** is the number of data records. To reduce this time complexity, we store a set of data row indexes of each attribute value in a Spark RDD. In Step 3, we assign the distinct attribute values to old combinations (cOLD) RDD to start the iterative procedure. Thus the dA RDD acts as a seed for all following transformations. The algorithm runs maximum of (n) iterations, where n is the number of attributes in the data. During the ith iteration, the algorithm extracts Action Rules with i-1 action set pairs on the left hand side of the rule. Step 8 uses flatMap() transformation on the data to collect all possible i combinations from a data record. We sort the combination of attribute values since they act as key for upcoming join() and groupBy() operations. We also attach a row id rid to all combinations to get the support (which data records contains a particular pattern) of each combination with the use of Spark's groupByKey() method in Step 9. We do sequential filtering in following steps.



X	A	B	C	D
X_1	a_1	b_1	c_1	d_1
X_2	a_3	b_1	c_1	d_1
X_3	a_2	b_2	c_1	d_2
X_4	a_2	b_2	c_1	d_2
X_5	a_2	b_1	c_1	d_1
X_6	a_2	b_2	c_1	d_2
X_7	a_2	b_1	c_2	d_2
X_8	a_1	b_2	c_2	d_1

X	A	D
X_1	a_1	d_1
X_2	a_3	d_1
X_3	a_2	d_2
X_4	a_2	d_2
X_5	a_2	d_1
X_6	a_2	d_2
X_7	a_2	d_2
X_8	a_1	d_1

X	B	C	D
X_1	b_1	c_1	d_1
X_2	b_1	c_1	d_1
X_3	b_2	c_1	d_2
X_4	b_2	c_1	d_2
X_5	b_1	c_1	d_1
X_6	b_2	c_1	d_2
X_7	b_1	c_2	d_2
X_8	b_2	c_2	d_1

Figure.4. Vertical Data Split with Meta Action for Table 1

In Step 10, we filter out combinations for which the indexes count is less than the given support threshold supp. From the filtered combinations, we get combinations (cFROM and cTO) that has decision from dFROM value and decision to dTO values in Step 11 and Step 12 respectively. In Step 14, we join cFROM and cTO based on attribute names, filter outdFROM and dTO values since we know the decision action, which is not required in finding confidence of Action Rules. This results in Action Rules of the form (attributes, (fromValues, fromIndexes), (toValues, toIndexes)). We then calculate actual support of the resultant Action Rules and filter out rules that has support atleast the given support threshold supp in Step 15 and reformat it to the form given in Equation 2. From Step 14, we have |Y1. Z1| and |Y2. Z2| or in other words numerator of the Confidence formula. To find the denominator |Y1| and |Y2|, we again from values indexes and to values indexes in Step 17 and Step 18 respectively. We perform join() operation with Old combinations and assign values to aFROM and aTO in Step 19 and Step 20 respectively. Subsequently we perform division operation of the Confidence formula in the same steps. In Step 21, we join aFROM and aTO and perform multiplication operation on the Confidence formula and reformat it the form given in Equation 2. We now join Action Rules with Support from Step 15 and Action Rules with Confidence from Step 21 to get final set of Action Rules. In Step 23, we assign new combination to the old combinations and pass the same to the next iteration.

4.3. Vertical Data distribution method with Meta Action

Meta-actions, are a tabular format to trigger action rules discovered from user data. Meta-actions are the actions that need to be executed in order to trigger corresponding [16] and it can be one or more than one set to invoke action rules in our method a set of meta actions triggered the generation of action rules. In our paper, we present a approach for partitioning the given data using information granules. We give a new algorithm to generate meta action as the intermediate state before extraction of all Action Rules, based on the algorithm proposed in [6] and [23]. We test how fast our new method works with two different data sets one with the NPS dataset and the other is Student Survey dataset and compared to our previous distributed Action Rule extraction algorithms. A brief description about our vertical data distribution process with meta- actions has been given in Figure 4. We check validity of the new data distribution method by comparing number of Action Rules generated by our method and rule coverage of Action Rules from system with classical Association Action Rules [23] on a single machine and SARGS [5] systems.

5.EXPERIMENTS

To test our methods, we use three datasets: Car Evaluation data [18], Mammographic Mass data [18], and the Net Promoter Score dataset data [24]. The Car Evaluation and Mammography are obtained from the Machine Learning repository of the Department of Information and Computer Science of the University of California, Irvine [18]. The Car Evaluation Data consists of records describing a car's goodness and acceptability based on features such as buying frequency, maintenance cost, safety measure, seating capacity and luggage boot size. Mammographic is the most effective method for screening breast cancer. The Mammographic Mass data contains records that measure severity of the cancer based on patient's age, cancer shape, cancer density and BI-RADS (a test score to denote how severe the cancer is). Since these datasets are relatively small in size, in order to test them for scalability with the proposed distributed processing algorithms, we replicate their data rows 1024 and 2056 times respectively for CarEvaluation and MammographicMass datasets, in order to increase data size. We test our algorithm on both replicated and non-replicated data for CarEvaluation and Mammographic datasets. We also used a sample of Net Promoter Score dataset [24] for our experiments. The NPS (Net Promoter Score) dataset is collection of customer feedback data related to heavy equipment repair. The entire dataset consists of 38 companies, located in multiple sites across the whole United States as well as several parts of Canada. Overall, there are about 340,000 customers surveyed in the database over time span of 2011-2015. Customers were randomly selected to answer a questionnaire which was specifically designed to collect information relevant to NPS (structured into so-called "benchmarks"). All the responses from customers were saved into database with each question (benchmark) as one feature in the dataset. Benchmarks include numerical scores (0-10) on certain aspects of service: e.g. if job done correctly, are you satisfied with the job, likelihood to refer, etc. The dataset also contains customer details (name, contact, etc.) and service details (company, invoice, type of equipment repaired, etc.).

The decision attribute in the dataset is PromoterStatus which labels each customer as either promoter, passive or detractor. The decision problem here is to improve customer satisfaction / loyalty as measured by Net Promoter Score. The goal of applying action rules to solve the problem is to find minimal sets of actions so that to "reclassify" customer from "Detractor" to "Promoter" and the same improve NPS. For our experiments, we used survey given by customers for 4 companies over the year of 2015.

Table 3. Dataset Properties

	Car Evaluation Data	**Mamm. Mass Data**	**NPS Data: Company 16**	**NPS Data: Company 16 31**	**NPS Data: Company 17**	**NPS Data: Company 30**
Attributes	7 attributes -Buying - Maintenance -Doors -Persons -Luggage Boot -Safety -Class	6 attributes -BI-RADS - Patient' s age -Shape -Margin - Density - Severity	20 attributes including - Client Name - Division -Benchmark: Dealer communication Benchmark: Overall satisfaction -Benchmark: Technician communication	23 attributes including - Survey Type - Survey Name - Benchmark: Order accuracy - Benchmark: Personnel knowledge - Benchmark: Time taken to place order	22 attributes including - Channel Type - Division - Benchmark: Ease of ordering - Benchmark: Parts avail- Ability -Benchmark: Referral behavior	23 attributes including -Channel type -Survey name - Benchmark: Ease of using online store - Benchmark: Likelihood to be a repeat customer -Benchmark: How orders are placed
Decision attribute values	Class (unacc, acc, good, vgood)	Severity (0 - benign, 1malignant)	Estimated Sales (Detractor Passive Promoter)			
# of instances /decision value	unacc - 1210 acc - 384 good - 69 vgood - 65	0 - 516 1 - 445	Detractor - 61 Passive - 180 Promoter - 939	Detractor - 65 Passive - 190 Promoter - 1806	Detractor - 22 Passive - 61 Promoter - 459	Detractor - 94 Passive - 391 Promoter - 2821
Replication Factor	1024	2048	N/A			
# of instances	1,769,472	1,968,128	1180	2078	547	3335

Table 4. Parameters used in all Action Rule discovery algorithms

Property	**Car Evaluation Data**	**Mamm. Mass Data**	**NPS Data**
Stable attributes	Maintenance, Buying Price, Doors	Age	Survey name Survey type Division Channel type Client name
Required decision action	(Class) *unacc* → *acc*	(Severity) 1 → 0	Promoter Status *Detractor* → *Promoter*
Minimum Support α and Confidence β	2048, 70%	4096, 70%	2, 80%

Each of NPS data consists of around 1500 unique surveys from multiple customers with around 25 unique questions. Table 3 gives an overview of some properties of all datasets that we used to test our methods. Table 4 show parameters that we set for each dataset to collect Action Rules. For the Car Evaluation data, we choose Class attribute as a decision attribute and we collect Action Rules to help the car company to change the car from Unacceptable state to Acceptable state. For the Mammographic Mass data, we choose Cancer Severity as a decision attribute and we collect Action Rules to reduce the severity from Malignant to Benign. For all Net Promoter Score data, we choose Promoter Status as a decision attribute and we collect Action Rules to convert Detractors to Promoters of a company from a list of surveys that the customers have participated in.

Due to space limitation, we show the Action Rules extracted using our methods in different tables: Table 5, Table 6, Table 7, Table 8, Table 9 and Table 10. In Table 5, we give Action Rules extracted from Car Evaluation and Mammographic Mass datasets and in Table 6, we give Action Rules extracted from 4 NPS datasets. These Action Rules provide actionable recommendations to users who wants to achieve the desired decision action. For example, from Table 5, when a user wants to achieve the action Class, unacc → acc, our system give actionable recommendations to achieve that goal. Action Rules ARC1, ARC2 and ARC3 in Table 5 are example recommendations given by our system for the appropriate parameters given in Table 4. For example, Action Rule ARC1 recommends if the car company maintains Buying cost to medium and Maintenance Cost to Very high and decrease the Seating capacity from More than 4 to 4 and increase Safety measures from medium to high with support of 1107 and minimum confidence of 74%.

Similarly, we give example Action Rules for all NPS datasets: 16, 16 31, 17 and 30 in Table 6, Table 7, Table 8, Table 9 and Table 10. In all cases, we consider that the user wants to convert a user's Promoter Status from Detractor to Promoter with parameters given in Table 4. For example, consider ARN2 which recommends that if the company can improve user's ratings on Completion of repair correctly from 5 points to 10 pints and improve user's ratings on Technician communication from 3 points to 9 points, the company can convert some Detractors to Promoters with support of 2.0 and confidence of 90.0%.

In Figure 5, we give run time analysis of Association Action rules extraction methods implemented in non-parallel method and our technique of splitting the data by attributes and extracting Association Action rules in parallel using Apache Spark framework. As mentioned earlier, since Association Action rules methodology is a complex algorithm, we can see from the Figure 5 that non-parallel method takes a while to give away actionable recommendations. On the other hand, our method fraction of minutes to give results. The speed increases when the data set size increases. For example, for NPS data: company 30, which is the largest in our datasets in accordance with number of attributes and rows, the non-parallel version takes around an hour to complete execution, while our method just takes 3 minutes.

In Table 11, we compare number of Action Rules extracted by our parallel and non-parallel algorithms. Due to the data partitioning step involved in our method, we lose some Action Rules. In Table 12, we give Coverage measure of Action Rules extracted for each dataset and compare them with Coverage of Action Rules extracted using non-parallel methods. In 50% of our experiments we are able to achieve the same coverage as Action Rules extracted using non-parallel approach. From this table, we can assure that our method of extracting Action Rules using the distributed computing framework like Spark [34] can produce results in a faster runtime by losing only a limited knowledge. Meta Actions are provided by experts, but we have some probable predictions of Meta Actions. In Figure 6, 7, 8, 9, 10 we are showing the Meta Action for a particular Action Rule. Each Meta Action corresponds to a particular Action Rule. Each figure represents the Action Rule table it belongs to. For example figure 6 belongs to Table 7.

Table 5. Action Rules of Car Evaluation and Mammographic Mass datasets

Car Evaluation Data
1. AR_{C1} : (*buying* = *med*) ∧ (*maint* = *vhigh*) ∧ (*persons*,*more* → 4) ∧ (*safety*,*med* → *high*) =⇒ (*class*,*unacc* → *acc*)[*Support* : 1107,*OldConfidence* : 74%,*NewConfidence* : 100%,*Utility* : 74%]
2. AR_{C2} : (*buying* = *med*) ∧ (*maint* = *vhigh*) ∧ (*persons*,*more* → 4) ∧ (*safety*,*med* → *high*) =⇒ (*class*,*unacc* → *acc*)[*Support* : 1353,*OldConfidence* : 85%,*NewConfidence* : 100%,*Utility* : 85%]
3. AR_{C3} : (*buying* = *med*) ∧ (*lugboot*,*small* → *big*) ∧ (*maint* = *med*) ∧ (*persons*,2 → *more*) ∧ (*safety*,*low* → *med*) =⇒ (*class*,*unacc* → *acc*)[*Support* : 4096,*OldConfidence* : 100%,*NewConfidence* : 100%,*Utility* : 100%]
Mammographic Mass Data
1. AR_{M1} : (*BIRADS*,6 → 2) ∧ (*Density*,4 → 3) =⇒ (*Severity*,1→0)[*Support* : 12288, *OldConfidence* : 100%,*NewConfidence* : 100%,*Utility* : 100%]
2. AR_{M2} : (*Age* = 42) ∧ (*Density*,1 → 3) ∧ (*Shape* = 1) =⇒ (*Severity*,1 → 0)[*Support* : 10240, *OldConfidence* : 100%,*NewConfidence* : 100%,*Utility* : 100%]
3. AR_{M3} : (*Age* = 62) ∧ (*Margin*,5 → 1) =⇒ (*Severity*,1 → 0)[*Support* : 12288, *OldConfidence* : 100%,*NewConfidence* : 100%,*Utility* : 100%]

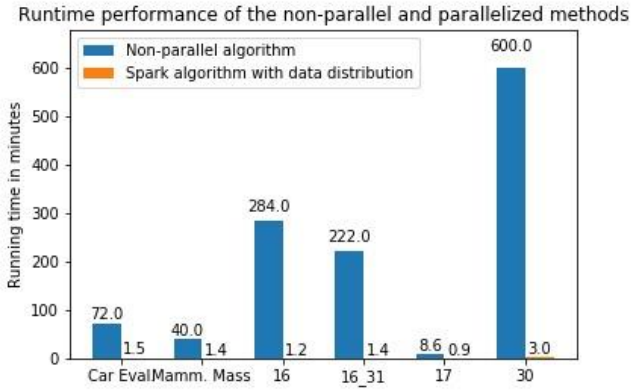


Figure.5. Speed Performance of Non-parallel and Parallel algorithms of Association Action Rules extraction

6. CONCLUSIONS

The proposed approach divides the data in more intelligent way, rather than the Random Distribution, which is the classical method for cloud platforms, such as MapReduce and Spark. This method reduces the computational complexity and speeds up the process. It provides more coherent data partitions in comparison than existing cloud platforms. We take intersections of sets with high support between partitions, and we generate Meta Actions. The existing Association Action Rules follows iterative method to extract all possible Action Rules and it has a complexity disadvantage. In this research, we improve the distributed Actionable Pattern Mining method. We handle data distribution task by splitting the data both horizontally and vertically by using partition threshold Rho. We produce information granules with Meta Actions. Hence our results discover valuable Actionable Knowledge with application in Business and Education domains.

Table 6. Action Rules of NPS datasets: 16, 16_31, 17, 30

Company 16
1. AR_{N1} : (*ChannelType=Ag*) ∧ (*Benchmark*:*Service* − *TechnicianCommunication*,1 → 10) → (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 87.5%,*NewConfidence* : 100.0%,*Utility* : 87.5%] 2. AR_{N2} : (*Benchmark* : *Service* − *RepairCompletedCorrectly*,5 → 10) ∧ (*Benchmark* : *Service* − *TechnicianCommunication*,3 → 9) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 90.0%,*NewConfidence* : 90.0%,*Utility* : 100.0%]
16_31
1. AR_{N3} : (*BenchmarkPartsOrderAccuracy*,3 → 10) ∧ (*ClientName* = *HOLTCAT*) → (*Division* = *Parts*) ∧ (*BenchmarkPartsTimeitTooktoPlaceOrder*,4 → 9) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 80.0%,*NewConfidence* : 100.0%,*Utility* : 80.0%] 2. AR_{N4} : (*BenchmarkPartsHowOrdersArePlaced*,2 → 4) ∧ (*BenchmarkPartsOrderAccuracy*,6 → 10) ∧ (*ClientName* = *HOLTCAT*) ∧ (*Division* = *Parts*) ∧ (*BenchmarkPartsKnowledgeofPersonnel*,5 → 10) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 100.0%,*NewConfidence* : 100.0%,*Utility* : 100.0%]
17
1. AR_{N5} : (*BenchmarkAllDealerCommunication*,6 → 10) ∧ (*BenchmarkAllLikelihoodtobeRepeatCustomer*,6 → 9) ∧ (*Division* = *Parts*) ∧ (*SurveyType* = *Parts*) ∧ (*BenchmarkPartsTimeitTooktoPlaceOrder*,6 → 10) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 100.0%,*NewConfidence* : 100.0%,*Utility* : 100.0%] 2. AR_{N6} : (*BenchmarkAllDealerCommunication*,6 → 10) ∧ (*BenchmarkAllOverallSatisfaction*,6 → 9) ∧ (*ClientName* = *HoltofCalifornia*) ∧ (*Division* = *Parts*) ∧ (*BenchmarkPartsTimeitTooktoPlaceOrder*,6 → 10) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 91.66%,*NewConfidence* : 100.0%,*Utility* : 91.66%]
30
1. AR_{N7} : (*BenchmarkPartsOrderAccuracy*,5 → 9) ∧ (*ChannelType* = Construction All) ∧(*SurveyType* = Parts) ∧(*BenchmarkPartsTimeitTooktoPlaceOrder*,5 → 7) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 80.0%,*NewConfidence* : 100.0%,*Utility* : 80.0%] 2. AR_{N8} : (*BenchmarkPartsOrderAccuracy*,7 → 10)∧ (*ChannelType* = *ConstructionAll*)∧ (*ClientName* = Empire Cat) ∧ (*SurveyType* = Parts) ∧ (*BenchmarkAllOverallSatisfaction*,4 → 7) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : 2.0,*OldConfidence* : 100.0%,*NewConfidence* : 100.0%,*Utility* : 100.0%]

Table 7. Action Rules of NPS datasets: NPS action rules California part1ActionRules

17 California part1ActionRules
AR_{N1} : (*BenchmarkPartsEaseofCompletingPartsOrder*,5 → 9) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : −4.0,*Confidence* : −52.72%]
AR_{N2} : (*BenchmarkPartsEaseofCompletingPartsOrder*,5 → 8) =⇒ (*PromoterStatus*,*Detractor* → *Promoter*)[*Support* : −2.0,*Confidence* : −61.53%]
AR_{N3} :(*BenchmarkPartsHowOrdersArePlaced*,2→3) ∧ (*ChannelType*, *ConstructionAll*→*ConstructionAll*)∧(*SurveyType*,*Parts* → *Parts*)∧(*BenchmarkPartsPromptNotificationofBackOrders*,*7*→ 9) ∧ (*BenchmarkPartsTimeitTooktoPlaceOrder*,8→10)⇒(*PromoterStatus*,*Detractor*→*Promoter*)[*Support* : −2.0,*Confidence* : −95.65%]
AR_{N4} :(*BenchmarkAllOverallSatisfaction*,7→7)⇒(*PromoterStatus*,*Detractor*→*Promoter*)[*Support* : −3.0,*Confidence* : −83.33%]
AR_{N5} : (*ClientName*,*HoltofCalifornia* → *HoltofCalifornia*) =⇒ (*PromoterStatus*,*Detractor* →*Promoter*)[*Support* : −2.0,*Confidence* : − 53.43%]

Table 8. Action Rules of NPS datasets: NPS action rules California part2ActionRules

17 California part2ActionRules
ARN6 : (ChannelType;NotDefined → NotDefined) ^ (Division; Parts→Parts) ^ (SurveyName; PartsPartsDivision → PartsPartsDivision) ^ (BenchmarkAllOverallSatisfaction; 7 →10) ^ (BenchmarkPartsKnowledgeofPersonnel; 9→10) ⇒ (PromoterStatus;Detractor ! Promoter)[Support :-2:0;Confidence :- 66:66%]
ARN7 : (BenchmarkPartsP artsAvailability; 3 → 10) ^ (ChannelType;ConstructionAll ! ConstructionAll) ^ (SurveyType; Parts → Parts) ^ (BenchmarkAllOverallSatisfaction; 3 →7) ⇒ (PromoterStatus;Detractor → Promoter)[Support : - 2:0;Confidence : - 72:72%]
ARN8 : (ChannelType; PowerSystemsAll →PowerSystemsAll) ^ (Division;EPS --Parts →EPS--Parts) ^ (SurveyType; Parts → Parts) ^ (BenchmarkAllLikelihoodtobeRepeatCustomer; 5 →10) ^ (BenchmarkPartsEaseofCompletingPartsOrder; 5 → 10) ⇒ (PromoterStatus;Detractor → Promoter)[Support : - 3:0;Confidence :- 91:66%]
ARN9 : (BenchmarkPartsPromptNotificationofBackOrders; 7 → 10) ⇒ (PromoterStatus;Detractor →Promoter)[Support : - 3:0;Confidence :- 92:30%]
ARN10 : (BenchmarkPartsExplanationofDeliveryOptionsCosts; 7 → 8) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 2:0;Confidence :- 100:0%]

Table 9. Action Rules of NPS datasets: NPS action rules 30-35 part1ActionRules

30-35 part1ActionRules
ARN11 : (BenchmarkPartsPartsAvailability; 4 → 9) ^ (Division;WagnerHeavyEquipment - Parts → WagnerHeavyEquipment-Parts) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 2:0;Confidence :- 84:90%]
ARN12 : (BenchmarkPartsPartsAvailability; 3 → 10) ^ (SurveyType; Parts → Parts) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 1:0;Confidence :- 93:17%]
ARN13 : (BenchmarkAllOverallSatisfaction; 3 → 10) ^ (SurveyType; Parts → Parts) ⇒ (PromoterStatus;Detractor →Promoter)[Support : - 2:0;Confidence : - 88:87%]
ARN14 : (BenchmarkPartsP artsAvailability; 2 → 10) ^ (ChannelType; PowerSystemsEngine → PowerSystemsEngine) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 1:0;Confidence :- 89:36%]
ARN15 : (BenchmarkPartsP artsAvailability; 1 → 8) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 1:0;Confidence :- 36:45%]

Table 10. Action Rules of NPS datasets: NPS action rules 30-35 part2ActionRules

30-35 part2ActionRules
ARN16 : (BenchmarkAllOverallSatisfaction; 1 → 10) ^ (Division;WagnerHeavyEquipment - Parts → WagnerHeavyEquipment - Parts) ⇒ (PromoterStatus;Detractor ! Promoter)[Support :- 5:0;Confidence :- 63:74%]
ARN17 : (BenchmarkAllOverallSatisfaction; 5 → 9) ^ (BenchmarkPartsT imeitTooktoPlaceOrder; 9 → 10) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 4:0;Confidence :- 73:17%]
ARN18 : (BenchmarkAllLikelihoodtobeRepeatCustomer; 5 → 7) ⇒ (PromoterStatus;Detractor → Promoter)[Support :- 4:0;Confidence : - 85:35%]
ARN19 : (BenchmarkAllDealerCommunication; 1 → 9) ^ (BenchmarkAllLikelihoodtobeRepeatCustomer; 3 →8) ⇒ (PromoterStatus;Detractor → Promoter)[Support : - 4:0;Confidence :- 96:34%]
ARN20 : (BenchmarkPartsHowOrdersArePlaced; 2 → 3) ^ (BenchmarkPartsPartsAvailability; 8→ 9) ⇒ (PromoterStatus;Detractor → Promoter)[Support : - 4:0;Confidence : - 73:17%]

Table 11. Performance, in terms of number of resulting Action Rules, using parallel and nonparallel versions of Association Action Rules extraction

Dataset	Non-parallel algorithm	Parallel algorithm
Car Evaluation data	3500	3496
Mammographic Mass data	5790	5756
NPS data: 16	900	798
NPS data: 16 31	83643	83000
NPS data: 17	37256	37000
NPS data: 30	184000	182000

Table 12. Performance of algorithms for all datasets in terms of a Coverage measure

Dataset	Non-parallel algorithm	Parallel algorithm - Random distribution	Parallel algorithm - Granule distribution
Car Evaluation data	99.5%	99%	99%
Mammographic Mass data	94%	92.53%	92.53%
NPS data: 16	89.2%	86.9%	86.9%
NPS data: 16 31	73%	70.77%	70.77%
NPS data: 17	72.7%	72.7%	72.7%
NPS data: 30	75.5%	75.5%	75.5%

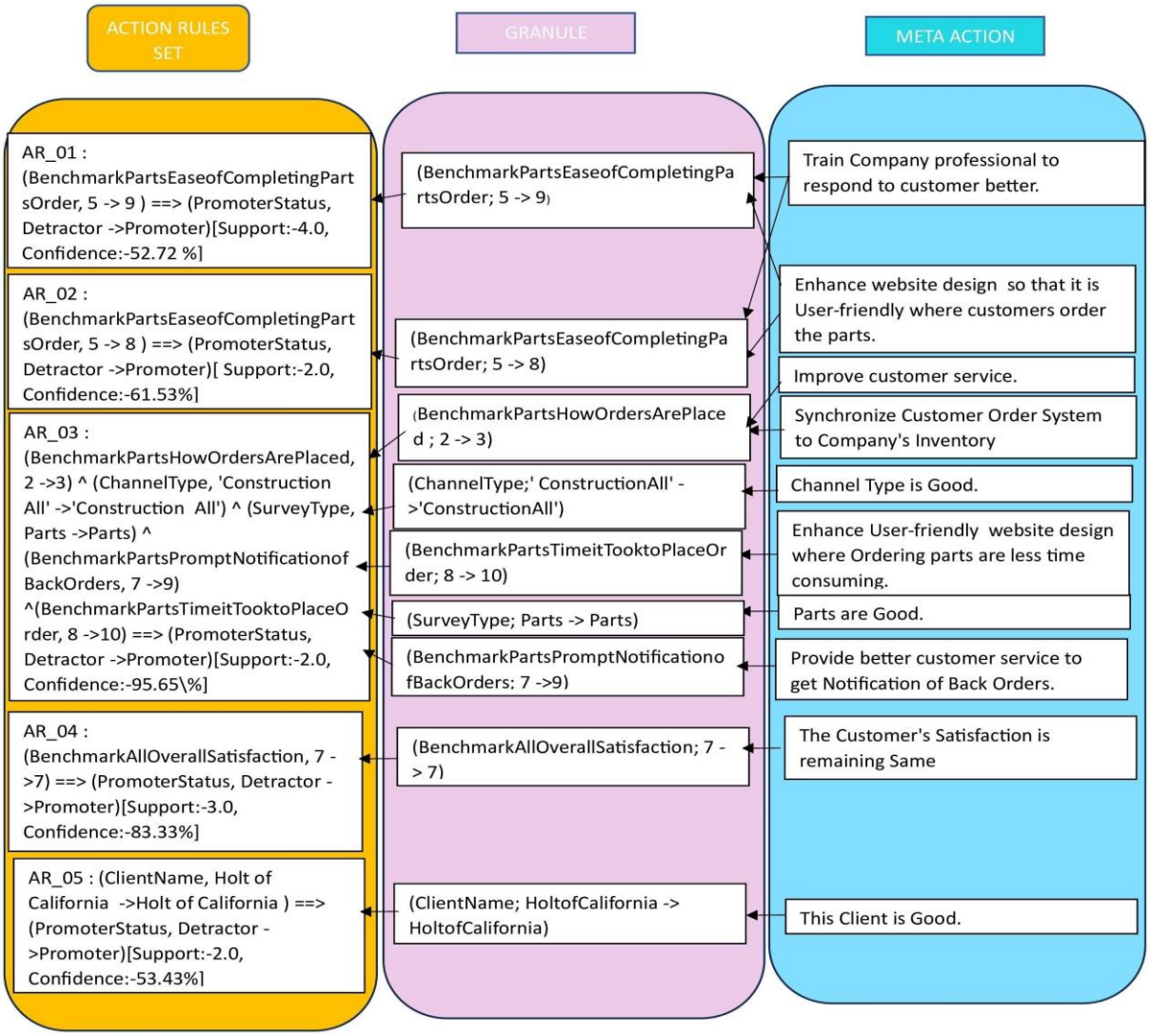


Figure.6. Example Meta Action for Table 7 From Action Rule 1 To Action Rule 5

Figure.7. Example Meta Action for Table 8 From Action Rule 6 To Action Rule 8

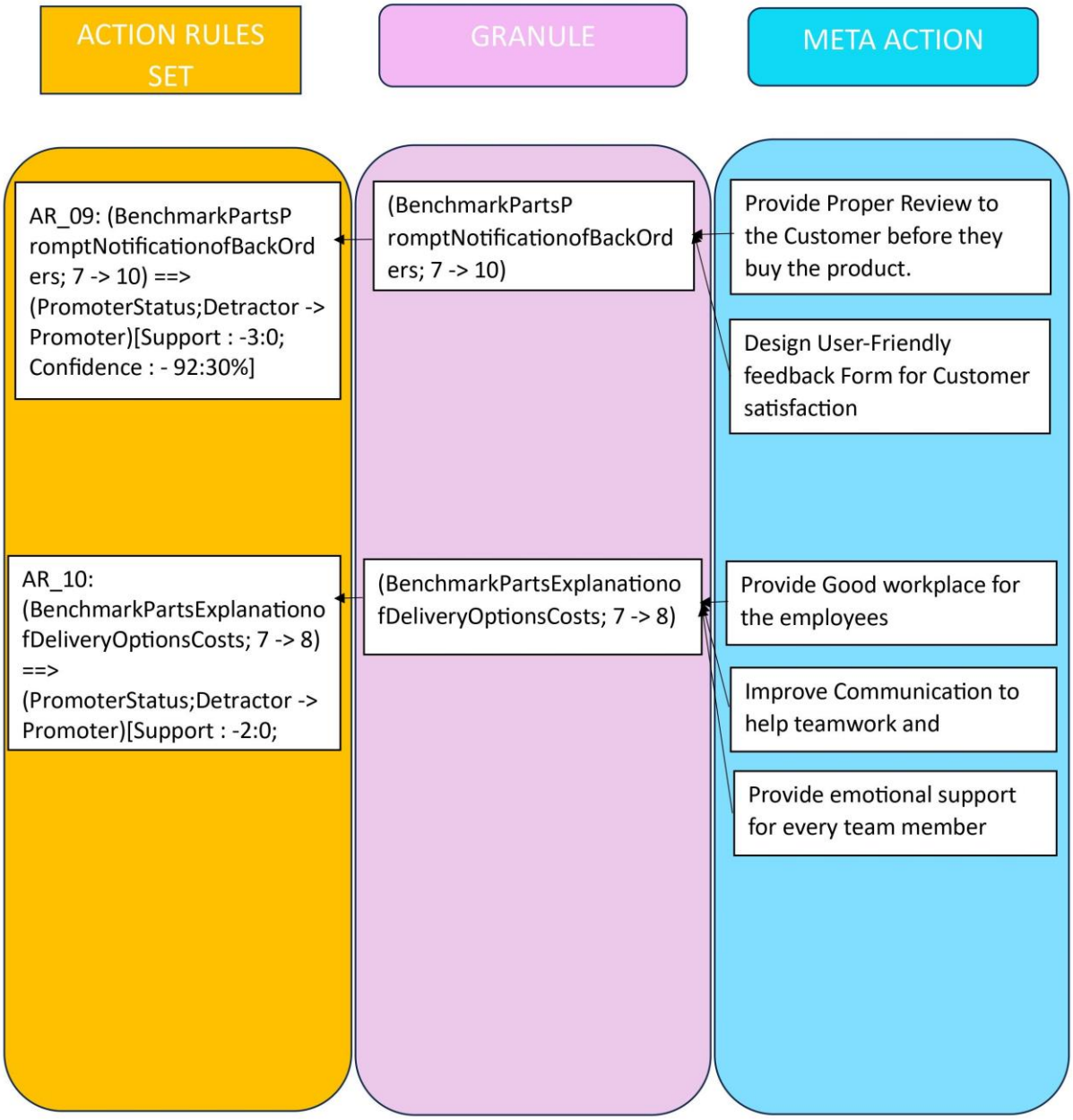


Figure.8. Example Meta Action for Table 8 From Action Rule 9 To Action Rule 10

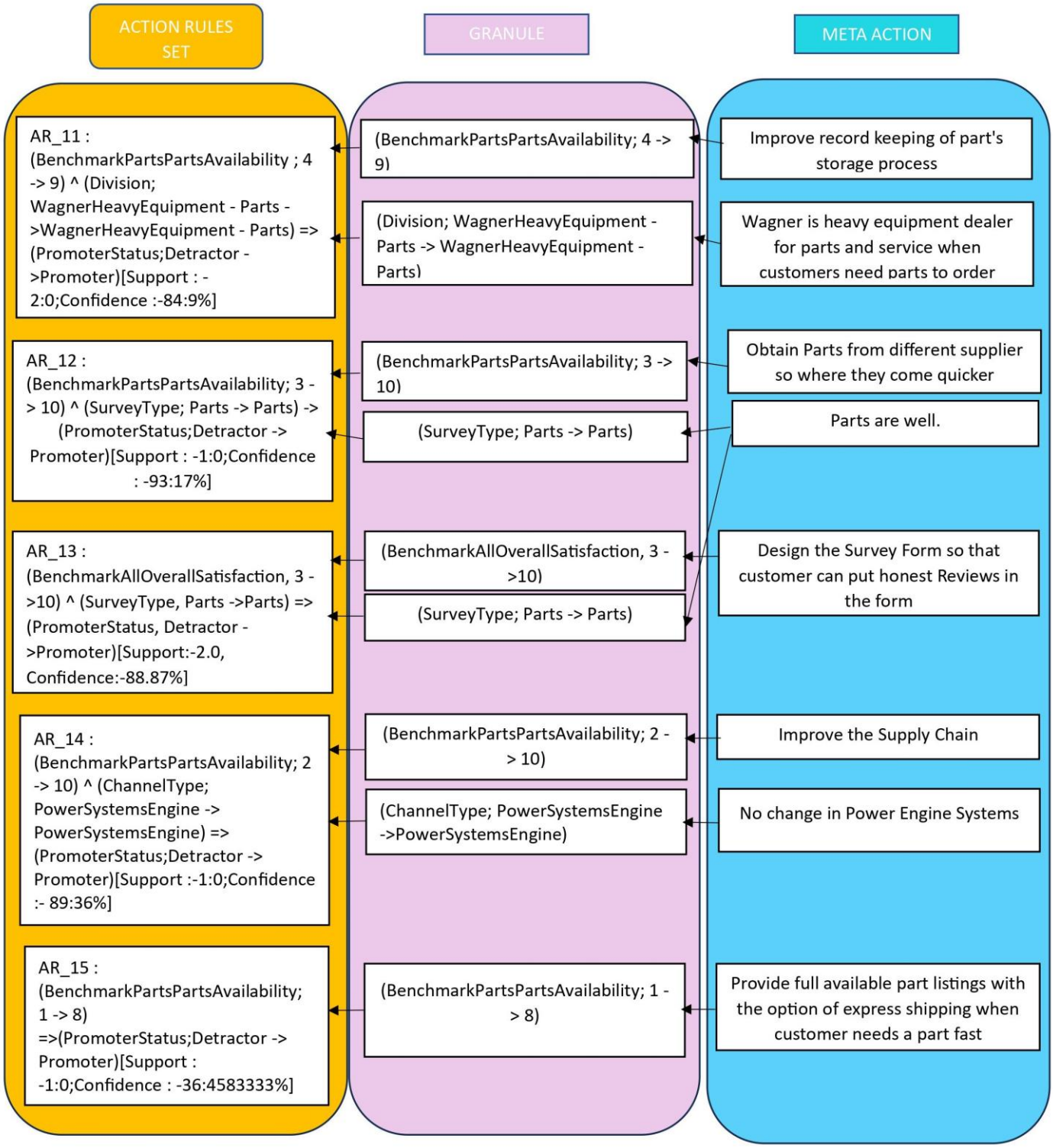


Figure.9. Example Meta Action for Table 9 From Action Rule 11 – To Action Rule 15

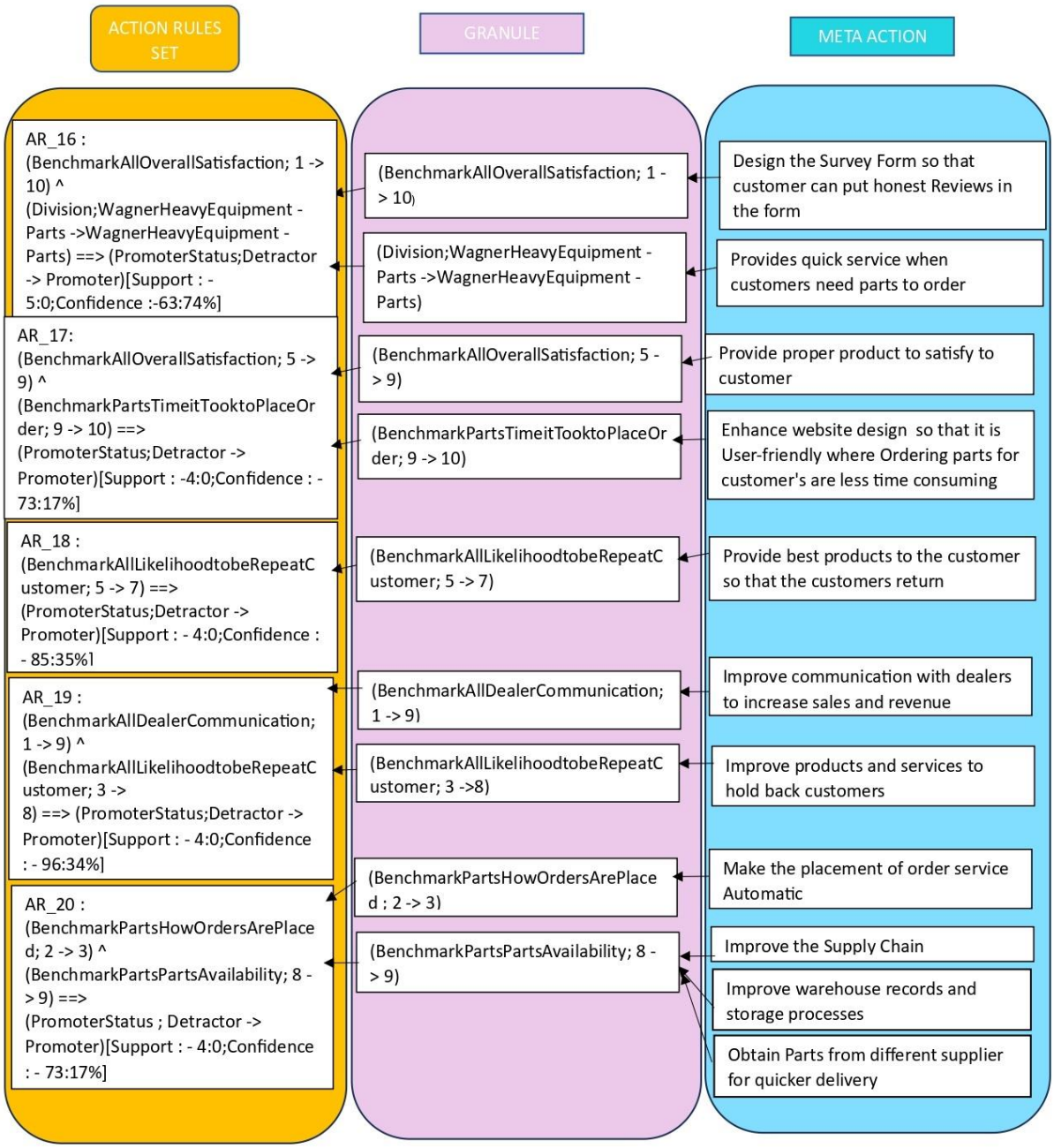


Figure.10. Example Meta Action for Table 10 From Action Rule 16 -To Action Rule 20

7. FUTURE WORK

Our proposed method produces higher quality rules, with good processing time for big and intricate datasets. We plan to use our method for more complex and larger data like HCUP datasets in future.

ACKNOWLEDGEMENTS

The authors would like to thank just everyone!

REFERENCES

[1] Duan, Yuehua & Ras, Zbigniew , (2022) "Recommendation System for Improving Churn Rate based on Action Rules and Sentiment Mining", International Journal of Data Mining, Modelling and Management. Vol .14. 10.1504/IJDMMM.2022.10041468

[2] A.A. Tzacheva, C.C. Sankar and R.A. Shankar, (2016)" Support Confidence and Utility of Action Rules Triggered by Meta-Actions", *In proceedings of 2016 IEEE International Conference on Knowledge Engineering and Applications* (Singapore) *(ICKEA 2016)*. IEEE Computer Society.

[3] Sharifah Sakinah Syed Ahmad and Witold Pedrycz, (2017). "The development of granular rule-based systems: a study in structural model compression", *Granular Computing, Vol.* 2, pp. 1–12.

[4] Mamoun Al-Mardini, Ayman Hajja, Lina Clover, David Olaleye, Youngjin Park, Jay Paulson, and Yang Xiao, (2016) "Reduction of Hospital Readmissions through Clustering Based Actionable Knowledge Mining", *In proceedings of Web Intelligence (WI)*, IEEE/WIC/ACM International Conference on. IEEE, 444–448.

[5] A. Bagavathi, P. Mummoju, K. Tarnowska, A. A. Tzacheva, and Z. W. Ras, (2017) "SARGS method for distributed actionable pattern mining using spark", *In proceedings of 2017 IEEE International Conference on Big Data (Big Data),* Pp. 4272–4281. https://doi.org/10.1109/BigData.2017.8258454

[6] Arunkumar Bagavathi, Abhishek Tripathi, Angelina A. Tzacheva, and Zbigniew W. Ras, (2018) "Actionable Pattern Mining - A Scalable Data Distribution Method Based on Information Granules", *In proceedings of 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 32–39. https://doi.org/10.1109/ICMLA. 2018.00013

[7] Jeffrey Dean and Sanjay Ghemawat, (2008) " MapReduce: simplified data processing on large clusters". Commun. ACM, Vol. 51 No. 1, pp. 107–113.

[8] Yuehua Duan and Zbigniew Ras, (2022) "Developing customer attrition management system: discovering action rules for making recommendations to retain customers". Applied Intelligence. Vol - 53. https://doi.org/10.1007/ s10489-022-03614-0

[9] Yuehua Duan and Zbigniew W. Ras, (2023) "Customer Churn Reduction Based on Action Rules and Collaboration". In Encyclopedia of Data Science and Machine Learning, John Wang (Ed.), IGI Global, Vol-11. https://doi.org/10.4018/978-1-7998-9220-5.ch035

[10] A. Hajja H. Touati, J. Kuang and Z. Ras, (2013) "Personalized Action Rules for Side Effects Object Grouping," International Journal of Intelligence Science, Vol- 3. Pp. 24-33. https://doi.org/10.4236/ijis. 2013.31A004

[11] Michael Hahsler and Radoslaw Karpienko, (2017)" Visualizing association rules in hierarchical groups", Journal of Business Economics, vol- 87, 3, pp. 317– 335.

[12] Seunghyun Im and Zbigniew W Ras, (2008) "Action´ rule extraction from a decision table: ARED", In International Symposium on Methodologies for Intelligent Systems, Springer, pp. 160–168.

[13] Moaiad Khder, Ibrahim Abu-AlSondos, and Yousif Bahar, (2021) "The impact of Implementing Data Mining in business Intelligence", International Journal of Entrepreneurship 25, pp. 1–9.

[14] Vladik Kreinovich, (2008) Interval computations as an important part of granular computing: an introduction. Handbook of Granular Computing, pp. 1–31.

[15] Jieyan Kuang, Albert Daniel, Jill Johnston, and Zbigniew W Ras, (2014)" Hierarchically structured recommender system for improving NPS of a company", *In proceedings of International Conference on Rough Sets and Current Trends in Computing*, Springer, pp. 347– 357.

[16] Jieyan Kuang, Zbigniew W. Ras, and Albert Daniel, (2015) "Personalized Meta-Action Mining for NPS Improvement", In Foundations of Intelligent Systems, Floriana Esposito, Olivier Pivert, MohandSaid Hacid, Zbigniew W. Ras, and Stefano Ferilli (Eds.), Springer International Publishing, Cham, pp. 79–87.

[17] Huaxiong Li, Libo Zhang, Bing Huang, and Xianzhong Zhou, (2016) "Sequential three-way decision and granulation for cost-sensitive face recognition", Knowledge-Based Systems 91, pp. 241–251.

[18] M. Lichman, (2013) UCI Machine Learning Repository, Technical Report, Irvine, CA, USA.

[19] Han Liu and Alexander Gegov, (2015) Collaborative decision making by ensemble rule based classification systems, In Granular Computing and Decision Making, Springer, pp. 245–264.

[20] Han Liu, Alexander Gegov, and Mihaela Cocea, (2016) Rule-based systems: a granular computing perspective. Granular Computing , Vol. 1 , pp. 1– 16.

[21] Han Liu, Alexander Gegov, and Frederic Stahl, (2014) Categorization and construction of rule based systems, *In proceedings of International Conference on Engineering Applications of Neural Networks*, Springer, pp. 183– 194.

[22] Shuai Liu, Witold Pedrycz, Adam Gacek, and Yaping Dai, (2018) "Development of information granules of higher type and their applications to granular models of time series", *Engineering Applications of Artificial Intelligence* Vol. 71, pp. 60–72.

[23] Zbigniew W Ras, Agnieszka Dardzinska, Li-Shiang Tsay, and Hanna Wasyluk, (2008) Association action rules, *In proceedings of Data Mining Workshops, ICDMW'08, IEEE International Conference on. IEEE*, pp. 283–290.

[24] Zbigniew W Ras, Katarzyna A Tarnowska, Jieyan Kuang, Lynn Daniel, and Doug Fowler, (2017) "User Friendly NPS-Based Recommender System for Driving Business Revenue", *In proceedings of International Joint Conference on Rough Sets*, Springer, pp. 34–48.

[25] Zbigniew W Ras and Alicja Wieczorkowska, (2000) "Action-Rules: How to increase profit of a company", *In proceedings of European Conference on Principles of Data Mining and Knowledge Discovery*, Springer, pp. 587–592.

[26] Zbigniew W Ras, Elzbieta Wyrzykowska, and Hanna Wasyluk, (2007) "ARAS: Action rules discovery based on agglomerative strategy", *In proceedings of International Workshop on Mining Complex Data*, Springer, pp. 196–208.

[27] Sanjay Rathee, Manohar Kaul, and Arti Kashyap, (2015) "R-Apriori: an efficient apriori based algorithm on spark", *In Proceedings of the 8th Workshop on Ph. D. Workshop in Information and Knowledge Management*. ACM, pp. 27–34.

[28] Andrzej Skowron and Jaroslaw Stepaniuk, (2007) Modeling of high quality granules, *In proceedings of International Conference on Rough Sets and Intelligent Systems Paradigms*, Springer, pp. 300–309.

[29] Li-Shiang Tsay and Zbigniew W Ras, (2005) " Action´ rules discovery: system DEAR2, method and experiments", *Journal of Experimental & Theoretical Artificial Intelligence*, Vol- 17 No.1-2 , pp. 119–128.

[30] Angelina A Tzacheva and Zbigniew W Ras, (2010) "Association action rules and action paths triggered by meta-actions". *In proceedings of Granular Computing (GrC), IEEE International Conference on IEEE*, pp. 772–776.

[31] Wei-Zhi Wu, Yee Leung, and Ju-Sheng Mi, (2009)" Granular computing and knowledge reduction in formal contexts", *IEEE transactions on knowledge and data engineering* ,Vol .21, pp. 1461–1474.

[32] Jing Tao Yao, Athanasios V Vasilakos, and Witold Pedrycz, (2013) Granular computing: perspectives and challenges, *IEEE Transactions on Cybernetics*, Vol- 43, pp. 1977–1989.

[33] Lotfi A Zadeh, (1997)" Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic", *Fuzzy sets and systems*, Vol . 90, pp. 111–127.

[34] Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, and Ion Stoica (2012) "Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing", *In Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation (San Jose, CA) (NSDI'12)*, USENIX Association, Berkeley, CA, USA, 2–2, https://dl.acm.org/doi/10.5555/2228298.222830

[35] Chatterjee S., Tzacheva, A.A., Ras, Z.W., "Scalable Action Mining Hybrid Method for Enhanced User Emotions in Education and Business Domain", *In proceedings of 3rd International Conference on NLP, Data Mining and Machine Learning (NLDML 2024)*, Vol. 13 No. 1, January 13 ~ 14, 2024, Virtual Conference, ISBN : 978-1-923107-15-1, https://ijcionline.com/volume/v13n1 , https://ijcionline.com/paper/13/13124ijci04.pdf

AUTHORS

Angelina Tzacheva is a Professor and Program Director of Computer Science and Information Technology, College of Computing and Engineering at WestCliff University, Irvine, CA, 92614. Her research interests include data mining, knowledge discovery in databases, distributed knowledge systems, medical imaging, multimedia databases, bioinformatics.

Sanchari Chatterjee is a Ph.D. student in Computer Science department at The University of North Carolina at Charlotte. She received her M.S. in computer science in 2020. Her research interests include data mining, text mining, knowledge discovery in databases, natural language processing, machine learning, social media mining.

Zbigniew Ras is a professor of Computer Science at the University of North Carolina, Charlotte. He is also the Lab director of Knowledge Discovery in Database Laboratory. His research interests Data Mining Algorithms and Methods, Actionable Knowledge Mining, Business Analytics, Data Semantics, Decision Support Systems for Fine Art, Flexible Query Answering, Folksonomy, Health Informatics, Multimedia Databases, Music Information Retrieval, Recommender Systems, Sentiment Analysis, Social Good, Text Mining.