Scalable Action Mining for Recommendations to Reduce Hospital Readmission

Abstract

Hospital re-admission problem is one of the long-time issues of healthcares in USA. Unplanned re-admissions to hospitals not only increase cost for patients, but also for hospitals and taxpayers. Action mining is one of the data mining approaches to recommend actions to undertake for an organization or individual to achieve required condition or status. In this work, we propose a scalable action mining method to recommend hospitals and taxpayers on what actions would potentially reduce patient readmission to hospitals. We use the Healthcare Cost and Utilization Project(HCUP) databases to evaluate our approach. All our proposed scalable approaches are cloud based and use Apache Spark to handle data processing and to make recommendations.

1 Introduction

US hospital expenditures are proven increasing for the past 2 decades, constituting total spending of $3.3 trillion and 17.9% of US economy as of 2016, given by US health records [11]. These expenditures are attributed due to various factors like inpatient care price, ambulatory prices, pharmaceutical prices, unnecessary visits, and emergency services apart from other social factors like population growth and aging [8]. Out of these, research show that 31% of the expenditures are due to inpatient care [14] and 20%-30% of patients gets readmission within 30-90 days of discharge [12].

Data mining offers several techniques to extract surprising, unknown, and interesting knowledge patterns from a massive data. The rule based learning is one of the simple data mining methods that intends to identify, learn, and recommend knowledge as rules. Many rule based methods like association rules and decision rules exist to generate rules to associate patterns and classify data respectively. In general, we represent rules as given in Equation 1, where the *antecedent* is a conjunction of conditions and the *consequent* is the resulting pattern in the given data for the given conditions in antecedent.

$$condition(s) \rightarrow result(s) \tag{1}$$

Action rule is also a knowledge extraction technique developed in the context to recommend possible transitions for a person to move from one state to another. For example, recommending the business to improve customer satisfaction [17] and sentiment analysis on Twitter [19]. Action rules follow the representation, similar to Equation 1, 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 $(\theta \rightarrow \phi)$ represents desired decision action.

$$[(\Psi) \wedge (\alpha \rightarrow \beta)] \rightarrow (\theta \rightarrow \phi) \tag{2}$$

The existing actionable pattern extraction algorithms [24, 22] extract knowledge efficiently when the data is small, which is not the case in this era of big data. Limited research like MR-Random Forest[26], SARGS [3], and distributed Association Action Rules mining [5] has been done on extracting Action Rules in a distributed scenario. 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 an extension to our previous work on distributed actionable pattern mining [5]. We propose additional parameters, robust data distribution strategies to the existing methodology for extracting actionable patterns in cloud setup and compare its efficiency across all previous methods. We use the *Healthcare Cost and Utilization Project* (HCUP) databases [9, 10] as use case for the proposed pattern extraction method. Particularly, we aim to extract actionable recommendations from the HCUP datasets, that help hospitals to give better care to its patients and reduce their overall costs for patients in the future.

2 Related Work

Unplanned hospital readmissions are expensive for both patients and healthcare, and they create unfortunate outcomes to everyone(patients, physicians, tax payers, and healthcare systems) [14]. Many research studies have focused on using the voluminous real world datasets for healthcare applications and decision making using such data mining and knowledge extraction techniques [15]. For example, in particular to hospital readmission, researchers created a machine learning model to predict patient readmissions using just billing codes and basic patient admission characteristics [12]. Some focus on predicting the liklihood of patient readmitting to the hospital, modelled as risk prediction, using Support Vector Machines [6], Logistic Regression [23], and Neural Networks [28]. Recently, there is an interesting study on designing a personalized procedure graphs, which gives a probability on patient's future procedure and recommend hospitals in making decisions for a patient [1, 2].

In the literature, action rules are extracted using two methods. First method 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 [24], which extracts Action Rules from two classification rules, or ARAS [22], 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 [13] and Association Action Rules [20] works in the object-based approach. Algorithms, except association action rules, runs much faster with the aim of extracting rules that are benefits the user to the maximum and extracts only limited recommendations.

Recently, due to the advent of big data, some research [26, 3, 5, 4] started applying distributed computing frameworks like MapReduce [7] and Spark [27] have been done to extract actionable recommendation completely in a clustered setup. All these methods aim to extract complete/approximated results, showing efficiency over non-parallel methods for big datasets.

3 Background

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

3.1 Decision System

Information System can be represented as $\mathbb{T} = (\mathbb{X}, \mathbb{A}, \mathbb{V})$ where,

X is a nonempty, finite set of data objects or rows

A is a nonempty, finite set of attributes

V_i is the *domain* of attribute a which represents a set of values for attribute $i | i \in \mathbb{A}$

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.

3.2 Action Rules

In this subsection, we give definitions of action terms, action rules and properties of action rules [21]

Let $\mathbb{T} = (\mathbb{X}, \mathbb{A} \cup d, \mathbb{V})$ be a decision system, where d is a decision attribute and $\mathbb{V} = \cup V_i : i \in \mathbb{A}$. 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 is given below: $(a, a_1 \rightarrow a_2).(b, b_1 \rightarrow b_2) \longrightarrow (D, D_1 \rightarrow D_2)$

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 $\mathcal{R}$ of form:

$(Y_1 \rightarrow Y_2) \longrightarrow (Z_1 \rightarrow Z_2)$ where,

Y is the condition part of $\mathcal{R}$, which comprise of sequence of action terms

Z is the decision part of $\mathcal{R}$

In [21], the support and confidence of an action rule $\mathcal{R}$ is given as

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

$Confidence(\mathcal{R}) = [\frac{card(Y_1 \cap Z_1)}{card(Y_1)}] \cdot [\frac{card(Y_2 \cap Z_2)}{card(Y_2)}]$

Coverage of an Action Rule means that how many decision 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

Table 1: Interesting attributes in the datasets

Attribute Name	Attribute Description
DX, DXCCS	Diagnosis codes representing all diagnosis that a patient follows during their hospital visit
PR, PRCCS	Procedure codes representing all procedures that are followed on a patient during their hospital stay
LOS	Length of hospital stay
VisitLink	Identifier of a patient
DaysToEvent	Number of days before next admission

4 Dataset Description

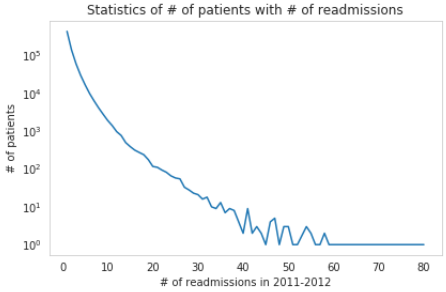


Figure 1: Representation of number of patients in Florida with their corresponding readmission frequency in 2011-2012

The use case that we used for this research is the medical domain data: Healthcare Cost and Utilization Project(HCUP) data. For our analysis we use the HCUP *State Inpatient Data* (SID) of the state Florida of years 2011-2012 [10]. All our data are organized as each data record representing a patient's hospital visit and each patient visit has 298 attributes. Table 1 give a brief description about interesting attributes that are available in our datasets and that we use in all our methods. In total, our SID dataset has 4,008,182 patient visits of 2,625,083 unique patients. Although, we do not have a separate boolean attribute called *Readmitted*, we measure this attribute for each patient visit using attributes *LOS* and *DaysToEvent*. Based on this measured *Readmitted*, attribute, we give a plot on number of patients with their corresponding number of hospital readmission in Figure 1. From this figure, we can note that the number of patients who have been readmitted to the hospitals at very low frequency decreases constantly and have sudden spikes between readmission frequencies 40 and 60, and becomes constant for higher frequencies. We also note that almost 72% patients in the data has atleast been readmitted to the hospital atleast once.

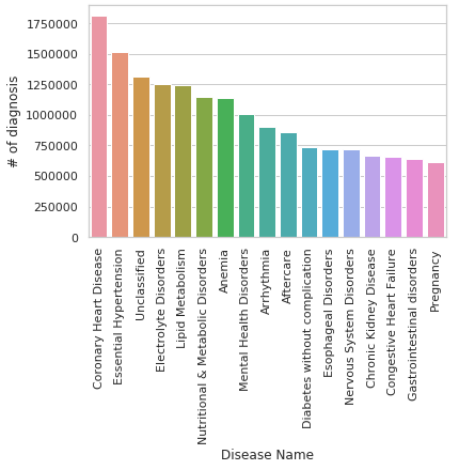


Figure 2: Top 15 diagnosed diseases for patients in Florida based on their frequency in 2011-2012

Out of 298 attributes, 70% of the attributes are allocated to mark diagnoses and procedures that a patient follows during their hospital stay. Most importantly, these diagnoses and procedures are represented as *ICD-9-CM*(International Classification of Diseases, Ninth Revision, Clinical Modification) codes. The data reports two varieties of diagnoses and procedures for covering these ICD-9-CM codes. One is simple ICD-9-CM code with around 8,900 unique codes in all diagnoses and procedures and the other is an *Clinical Classification Software*(CCS) with around 520 unique codes, which are aggregated versions of ICD-9-CM codes. In all our experiments, we use the later versions of codes.

In Figure 2, we represent diseases that have been most diagnosed in Florida in years 2011 and 2012. Interestingly in the data, most diagnosed disease(*Choronary Heart Disease*) is not present in the top procedures list and similarly, the top procedure(*Benign Neoplasm*) is not present in top diagnosis. According to our data, almost 12% of patient visit are resulted in death after following certain procedures to cure diseases.

5 Methodology

5.1 Data Partitioning strategies

We emphasize the concept of semantic data partitioning module in our actionable recommendation extraction system. It is the special case of the technique proposed in [2], where they define *Personalization* as all possible combinations of diseases that a patient have

been followed in all their hospital vists. Since such partitioning creates sparse data clusters and results in complex result aggregation, we create partition for all possible diagnosis and bucket all patient visits that follow diagnosis for these diseases. This significantly reduces the number of data partitions and all partitions have fair number of patient visits to extract recommendations from them. With this data distribution strategy, we perform horizontal data partitioning, similar to [3], but more semantically instead of random partitioning and later divide the number of attributes in the data by vertical data distribution [5]. Since one big data is broken into multiple chunks of small resilient distributed datasets, we can extract actionable patterns from all such small partitions in parallel. However, the small chunks may sometimes increase the complexity of our algorithms proposed in previous sections. For that reason, we propose the following load balancing approach to handle the situations, where the data has large number of attributes.

5.2 Privacy settings and Load balancing modules

In addition to parameters that we assign for extracting action rules, we define two parameters for privacy settings and load balancing in our algorithms to manage privacy and efficiency respectively.

5.2.1 Privacy parameters

The data that we use for extracting actionable recommendations may comprise of sensitive information like user identifiers, places, and practices. Some of these information when placed together reveals greater privacy details to others. In most of the cloud computing techniques, the data is distributed to multiple servers and the corresponding algorithms use such distributed data to mine or extract patterns. Since the data gets distributed to multiple servers, privacy of the data may get compromised and it is important to address such issues. Although, we do not handle such problems algorithmically, we give options to users to set input parameters that gives intuition to our data partitioning module on which attributes to be given more importance.

5.2.2 Load balancing parameter

Load balancing is another issue in cloud computing that we address in this work for extracting actionable recommendations. Our methods handles two levels of load balancing. Attracted from dynamic load balancing algorithms like Ant Colonization Optimization [18] and Honey Bee foraging [16], we propose load balancing that follows binary tree structure as given in Figure 3. We define a function that process this parameter and split the data accordingly. This method is an extension of the vertical data partitioning methodology given in Section 5.3. Instead of terminating the data partitioning at depth 1, the data partitioning continues to depth **n** in a binary fashion, based on the assigned load balancing parameter. From the binary tree, each node is a given as a load to the action rule extraction algorithm and thus reducing the total load into tiny chunks. Since assigning very high and very low value to this parameter increases complexity of extracting patterns, we recommend to set a mid-range value to this parameter. For all our experiments, we set this parameter as 2. The next level of load balancing is in-built load balancing functions in Apache Spark framework.

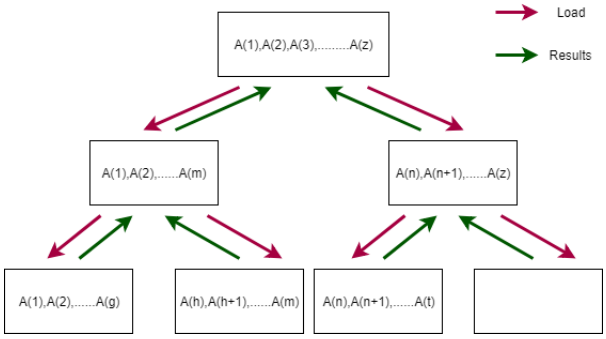


Figure 3: Binary Tree Load Balancing Strategy for Data Distribution

5.3 Distributed Action rules extraction algorithm

Given the load balancing parameter, we require a model to split the data at each level in the binary tree. We use the information granules methods proposed by Bagavathi et.al [5] to distribute the data. Information granularity solutions 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.

We represent a finite set of attributes from the attribute set A from the information system as granules. We minimize the correlations of granules given by Equation 3, where $C(G)$ represents correlation of a sub-granule with sub-granules of the other granule and m,n represents number of combinations of values of granules 1 and 2 respectively. The idea is that with minimum correlation between partitions, there is minimum combinations of results from multiple data partitions.

Algorithm 1 Granule Based Data Distribution

Require: dataSplit1, dataSplit2
function GETSCORE(split1, split2)
  $splitSum \leftarrow 0.0$
  for data1 in dataSplit1 **do**
    $p \leftarrow [\,]$
    $s_c \leftarrow 0$
    for data2 in dataSplit2 **do**
      $L \leftarrow data1.lines \cap data2.lines$
      if $L \neq \emptyset$ **then**
        p.addAll(L)
        $s_c = s_c + 1$
        if $|p| == |data1.lines|$ **then**
          $splitSum \mathrel{+}= 1/s_c$
          break
  return $splitSum$
$split1Avg = getScore(split1, split2)/|dataSplit1|$
$split2Avg = getScore(split2, split1)/|dataSplit2|$
return $split1Avg - split2Avg$

Table 2: Sub-granules of granules: A,B and C,D

A,B	C,D
Y, N - $\{x_1\}$	N, D_1 - $\{x_1, x_5, x_8\}$
Y, H - $\{x_2, x_3\}$	Y, D_2 - $\{x_2, x_6, x_7\}$
N, N - $\{x_4, x_6\}$	Y, D_1 - $\{x_3\}$
N, H - $\{x_5, x_7, x_8\}$	N, D_2 - $\{x_4\}$

$$\frac{\sum_{i=1}^{m} C(G_i) + \sum_{j=1}^{n} C(G_j)}{2} \tag{3}$$

Given an information system $\mathbb{T}$, we run our optimization(*minimizing Equation 3*) on all granules: ($\{A,B\}$ and $\{C,D\}$,$\{A,C\}$ and $\{B,D\}$ ...). Example of such combinations and sub-granules are given in Table 2. In the given example, the number of sub-granules, $m, n = 4$. We measure the correlation of each sub-granule($C(G_i)$,$C(G_j)$) by checking the overlap count of the sub-granule with sub-granules of other granule. For example, $C(G_{Y,H}) = \frac{1}{2}$, since Y,H from A,B overlaps with Y, D_2 and Y, D_1 of the granule C,D.

To extract actionable patterns, we follow a distributed version of Association Action Rules [20]. We in particular choose this method because of its disposition over datasets with large number of attributes and its ability to extract the complete knowledge in an efficient execution time. Another advantage of this method is to calculate properties of actionable patterns such as support and confidence on the fly unlike other algorithms, which waits until extracting all patterns.

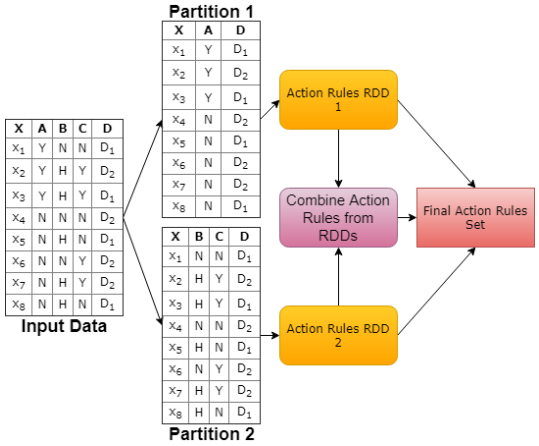


Figure 4: Example Vertical Data Distribution for Table 1

Table 3: HCUP data attributes and Algorithm parameters

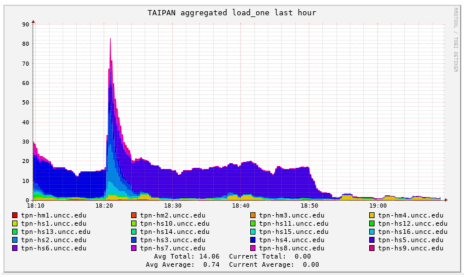
Property	Description
Attributes	67 attributes with DX(1-31) ; PR(1-31) ; Gender ; Race ; IsHomeless
Stable attributes	Gender ; Race ; IsHomeless ; PR(1-31)
Decision attribute	IsReadmitted
Required decision action	IsReadmitted($1 \rightarrow 0$)
Minimum support	30
Minimum confidence	40%
No. of diseases	262

6 Experiments and Results

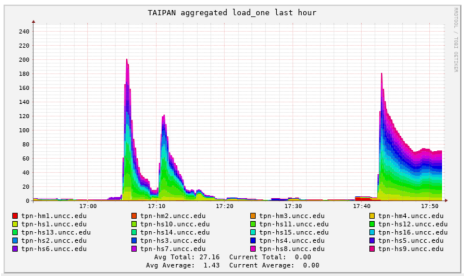
To evaluate our methods(load balancing and vertical data distribution), we use the existing cloud based methodologies to extract actionable patterns from large datasets. For all experiments, we use HCUP data (described in Section 4) and we use all methods to extract actionable patterns that recommends to reduce hospital readmission. Following are the methods that we are evaluating this data:

1. *Non-parallel approach* [26]: A basic approach to extract actionable patterns that does not incorporate any cloud based techniques to retrieve knowledge

2. *MR-Random Forest* [25]: A Hadoop MapReduce based approach that partitions the data in random by rows and find patterns in them

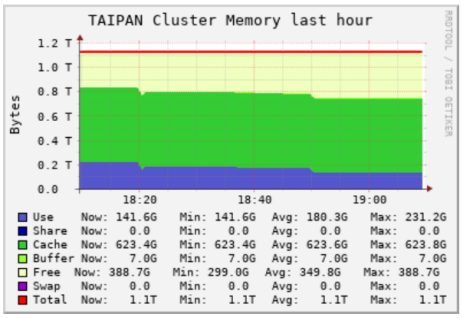


(a) Vertical Data Distribution cluster usage

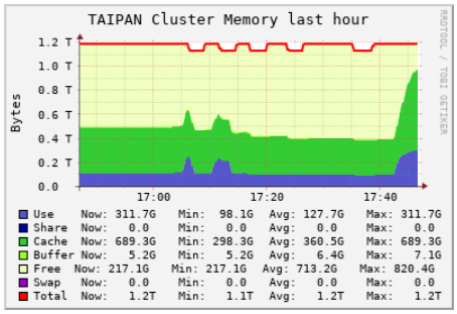


(b) Semantic Data Distribution cluster usage

Figure 5: Total cluster usage by simple vertical data distribution and semantic data distribution



(a) Vertical Data Distribution cluster memory



(b) Semantic Data Distribution cluster memory

Figure 6: Total cluster memory occupied by simple vertical data distribution and semantic data distribution

3. *SARGS* [3]: A Spark based approach that follows stratified sampling to distribute the data

4. *Simple vertical data partitioning* [5]: The basic approach of the proposed methodology without load balancing

Since all above methods cannot handle personalization, we run these methods sequentially on all diseases. In Table 3, we give a very short description about the data and also, we give parameters that we set for all the above algorithms. For semantic data distribution, we split the data by their diagnosis codes first. We set these partitioned data as a source of input for algorithms.

In Table 4, we give the execution time of all our proposed algorithms. We can note that for big datasets like the HCUP data, the non-parallel version of the action rule extraction method takes very long time to extract action rules. Whereas, the proposed cloud based performs much faster than the non-parallel methods. Particularly, our latest paralellized vertical data distribution algorithm achieves better execution time compared to other cloud based counterparts.

In Figure 5, we give total node usage in the cluster by simple vertical data distribution and semantic data distribution algorithms for the diagnosis code *250*. It is notable from this figure that the simple vertical data distribution takes more than *1 hour* to complete extracting all actionable patterns, whereas the semantic data distribution taken only around *17 minutes* for extracting the recommendations. Also we can note that in the vertical data distribution method from Figure 5a, one or two nodes execute most of the time and in our semantic data distribution method from Figure 5b utilizes much parallalization in the cluster.

We also give the cluster memory usage in Figure 6. We can note that semantic data distribution in Figure 6b occupies only limited quantity of data in the memory while the algorithm progress in extracting actionable patterns compared to the vertical data distribution method in Figure 6a.

In Table 5, we give action rules of diagnoses given in Table 4. We consider these actionable patterns as recommendations to hospitals in such a way that for a given disease, if the hospital provides treatment or care for recommended diseases, hospitals can potentially reduce the number of hospital readmissions. For example with disease code *250*, if hospitals give treatment for disease codes *21*(*Bone cancer*) and *251*(*Abdomen pain*), hospitals can reduce readmission by 50%. The support of *143* shows that the framework identifies 143 entries in the data to acquire this change.

Table 4: Execution time of algorithms for the HCUP data

Dataset	**Non-parallel algorithm**	**MR-Random Forest**	**SARGS**	**Vertical Data Distribution**	**Semantic Data Distribution**
670(Mental Health Disorders)	>2 days	3.87 hours	2.14 hours	49 mins	13.7 mins
250(Nausea & Vomiting)	>2.5 days	4.53 hours	2.62 hours	1.14 hours	17 mins
654(Developmental disorders)	>2days	3.2 hours	1.8 hours	35 mins	11.5 mins
233(Intracranial injury)	>2days	4.8 hours	2.3 hours	57 mins	13.8 mins
236(Open wounds)	>2days	3.5 hours	2 hours	46 mins	8.6 mins

Table 5: Sample Action Rules from the HCUP data for selected diagnosis

670(Mental Health Disorders)

250(Nausea & Vomiting)

1. $H250_{AR1}$: $(DX, 21$ (Bone Cancer) $\rightarrow$ 251 (Abdominal pain)) $\Rightarrow$ $(Readmission, 1- > 0)[Support : 143.0, Confidence : 58.68\%]$
2. $H250_{AR2}$: $(DX, 21$ (Bone Cancer) $\rightarrow$ 234 (Internal injury)) $\Rightarrow$ $(Readmission, 1 \rightarrow 0)[Support : 38.0, Confidence : 60.52\%]$

654(Developmental disorders)

1. $H654_{AR1}$: $(DX, 238$ (Surgical procedure complication) $\rightarrow$ 11 (Neck/Head Cancer)) $\Rightarrow$ $(Readmission, 1 \rightarrow 0)[Support : 49.0, Confidence : 51.43\%]$
2. $H654_{AR2}$: $(DX, 45$(Radiotherapy) $\rightarrow 100$ (Myocardial infarction)) $\Rightarrow$ $(Readmission, 1 \rightarrow 0)[Support : 44.0, Confidence : 63.72\%]$

233(Intracranial injury)

1. $H233_{AR1}$: $(DX, 228$ (Skull fracture) $\rightarrow 52$ (Nutritional deficiency)) $\Rightarrow$ $(Readmission, 1 \rightarrow 0)[Support : 65.0, Confidence : 51.5\%]$

7 Conclusion

In this work, we have provided an extension to our existing work [5] by adding additional parameters to set privacy settings and load balancing to the actionable pattern extraction techniques. More importantly we proposed a binary tree based load balancing module that split the data by attributes upto certain depth. Our results showed that this method improved the algorithm performance in execution time for very large data like H-CUP. Our analysis is the first effort upto our knowledge to extract actionable recommendations for reducing hospital readmission in much efficient time and improved personalization.

Although our methods proved efficient in execution time, it is not very optimal in memory usage in the distributed setup. Also, our methods lack subject matter experts input to evaluate actionable recommendations. In future, we plan to address these problems by providing more optimal load balancing modules that are both memory and time optimal. We also plan to use experts input to evaluate our results.

References

[1] M. Al-Mardini, A. Hajja, L. Clover, D. Olaleye, Y. Park, J. Paulson, and Y. Xiao. Reduction of hospital readmissions through clustering based actionable knowledge mining. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*, pages 444–448. IEEE, 2016.

[2] M. Almardini, A. Hajja, Z. W. Raś, L. Clover, D. Olaleye, Y. Park, J. Paulson, and Y. Xiao. Reduction of readmissions to hospitals based on actionable knowledge discovery and personalization. In *Beyond Databases, Architectures and Structures. Ad-*

vanced Technologies for Data Mining and Knowledge Discovery, pages 39–55. Springer, 2015.

[3] A. Bagavathi, P. Mummoju, K. Tarnowska, A. A. Tzacheva, and Z. W. Ras. Sargs method for distributed actionable pattern mining using spark. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4272–4281, Dec 2017.

[4] A. Bagavathi, V. Rao, and A. A. Tzacheva. Data distribution method for scalable actionable pattern mining. In *Proceedings of the First International Conference on Data Science, E-learning and Information Systems*, page 3. ACM, 2018.

[5] A. Bagavathi, A. Tripathi, A. A. Tzacheva, and Z. W. Ras. Actionable pattern mining-a scalable data distribution method based on information granules. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 32–39. IEEE, 2018.

[6] P. Braga, F. Portela, M. F. Santos, and F. Rua. Data mining models to predict patient's readmission in intensive care units. In *ICAART 2014-Proceedings of the 6th International Conference on Agents and Artificial Intelligence*, volume 1, pages 604–610. SCITEPRESS, 2014.

[7] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[8] J. L. Dieleman, E. Squires, A. L. Bui, M. Campbell, A. Chapin, H. Hamavid, C. Horst, Z. Li, T. Matyasz, A. Reynolds, et al. Factors associated with increases in us health care spending, 1996-2013. *Jama*, 318(17):1668–1678, 2017.

[9] A. for Healthcare Research and M. Quality, Rockville. Hcup national readmission databases (nrd). healthcare cost and utilization project (hcup). 2011, 2011.

[10] A. for Healthcare Research and M. Quality, Rockville. Hcup state inpatient databases (nrd). healthcare cost and utilization project (hcup). 2011-2012, 2011-2012.

[11] C. for Medicare and M. Services. National health expendire data.

[12] D. He, S. C. Mathews, A. N. Kalloo, and S. Hutfless. Mining high-dimensional administrative claims data to predict early hospital readmissions. *Journal of the American Medical Informatics Association*, 21(2):272–279, 2014.

[13] S. Im and Z. W. Raś. Action rule extraction from a decision table: Ared. In *International Symposium on Methodologies for Intelligent Systems*, pages 160–168. Springer, 2008.

[14] S. F. Jencks, M. V. Williams, and E. A. Coleman. Rehospitalizations among patients in the medicare fee-for-service program. *New England Journal of Medicine*, 360(14):1418–1428, 2009.

[15] H. C. Koh, G. Tan, et al. Data mining applications in healthcare. *Journal of healthcare information management*, 19(2):65, 2011.

[16] P. V. Krishna. Honey bee behavior inspired load balancing of tasks in cloud computing environments. *Applied Soft Computing*, 13(5):2292–2303, 2013.

[17] J. Kuang, A. Daniel, J. Johnston, and Z. W. Raś. Hierarchically structured recommender system for improving nps of a company. In *International Conference on Rough Sets and Current Trends in Computing*, pages 347–357. Springer, 2014.

[18] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi, et al. Load balancing of nodes in cloud using ant colony optimization. In *2012 14th International Conference on Modelling and Simulation*, pages 3–8. IEEE, 2012.

[19] J. Ranganathan, A. S. Irudayaraj, A. Bagavathi, and A. A. Tzacheva. Actionable pattern discovery for sentiment analysis on twitter data in clustered environment. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–15, 2018.

[20] Z. W. Ras, A. Dardzinska, L.-S. Tsay, and H. Wasyluk. Association action rules. In *Data Mining Workshops, 2008. ICDMW'08. IEEE International Conference on*, pages 283–290. IEEE, 2008.

[21] Z. W. Ras and A. Wieczorkowska. Action-rules: How to increase profit of a company. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 587–592. Springer, 2000.

[22] Z. W. Raś, E. Wyrzykowska, and H. Wasyluk. Aras: Action rules discovery based on agglomerative strategy. In *International Workshop on Mining Complex Data*, pages 196–208. Springer, 2007.

[23] B. Strack, J. P. DeShazo, C. Gennings, J. L. Olmo, S. Ventura, K. J. Cios, and J. N. Clore. Impact of hba1c measurement on hospital readmission rates: analysis of 70,000 clinical database patient records. *BioMed research international*, 2014, 2014.

[24] L.-S. Tsay* and Z. W. Raś. Action rules discovery: system dear2, method and experiments. *Journal of Experimental & Theoretical Artificial Intelligence*, 17(1-2):119–128, 2005.

[25] A. A. Tzacheva, A. Bagavathi, and P. D. Ganesan. Mr-random forest algorithm for distributed action rules discovery. *International Journal of Data Mining & Knowledge Management Process (IJDKP)*, 6(5):15–30, 2016.

[26] A. A. Tzacheva and Z. W. Ras. Association action rules and action paths triggered by meta-actions. In *Granular Computing (GrC), 2010 IEEE International Conference on*, pages 772–776. IEEE, 2010.

[27] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica. 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*, NSDI'12, pages 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[28] B. Zheng, J. Zhang, S. W. Yoon, S. S. Lam, M. Khasawneh, and S. Poranki. Predictive modeling of hospital readmissions using metaheuristics and data mining. *Expert Systems with Applications*, 42(20):7110–7120, 2015.