An Algorithm to Preserve Infrequent Relations in Process Mining: An Application to Lab Tests Ordering Process

Qifan Chen[0000-0003-1068-6408], Yang Lu[0000-0002-9002-8650] and Simon Poon[0000-0003-2726-9109]

School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia
{qche8411, yalu8986}@uni.sydney.edu.au, simon.poon@sydney.edu.au

Abstract. Process mining connects data mining and process modelling. It is considered as one of the most efficient ways to provide insights into the healthcare domain with the rapid growth of electrical health records. Infrequent relations are defined as rare happened directly follow dependencies between activities. Those relations are often regarded as noises and filtered out during process discovery. However, we attempt to reveal important insights into the process through the infrequent relations, especially in the healthcare domain. Also, there may be necessary value-based conditions that can trigger these infrequent relations. This paper aims to discover value-based conditions and preserve good infrequent relations. This paper adopts the L* life-cycle methodology and Data-aware Heuristic Miner (DHM) as tools. In addition, we propose a novel data preprocessing approach. Our method is trialed using a publicly available EHR dataset - the MIMIC-III dataset. The results show that we can successfully find the value-based conditions and preserve useful infrequent relations of different lab tests for five different diseases. The preserved conditional infrequent relations achieve on average 87% accuracy.

Keywords: Process Mining, healthcare process data, conditional infrequent relations, data preprocessing.

1 Introduction

Process mining is a new technology that bridges the gap between process modelling and data mining. There are three different stages, which are process discovery, conformance checking, and process enhancement [1]. Among them, the most crucial part is process discovery, which involves generating representative models from the event log. Process mining has shown a wide usage in the healthcare domain, such as discovering the oncology pathways [2] and analyzing medical emergency processes [3].

However, many process discovery algorithms rely on frequency of occurrence as a measure of importance. Hence, they only pay attention to the main pathway and ignore lots of infrequent relations, which are usually treated as noise [4] and removed when number of occurrence is below a pre-set threshold. Nevertheless, some of the infrequent relations are critical in providing us with lots of useful insights into the process.

Infrequent relations are defined as directly follow relations between activities that rarely occur in the event log. Certain relations are dependent on the outcome of another activity. That is to say, the outcome of the previous activity will determine whether the later activity happens or not. In many algorithms, there is a pre-defined threshold. If the number of occurrences does not reach the threshold, the relations would likely to be discarded. Hence, it misses out the critical insight into the process.

Besides, with the widely-availability of electrical health records (EHR), we may derive some practical value-based conditions which trigger the infrequent relations—taking the process in the emergency room transfer in Fig. 1 as an example, the primary pathway is ED registration, admission, then transfer into different care units and finally discharge. Nevertheless, there are also few infrequent traces; after performing some lab tests, the patients are directly discharged. In a normal process mining algorithm, the $lab\ test \rightarrow discharge$ relation will be removed. However, this is due to the lab test results of the patient is normal, and this patient no longer serves as an emergency and then discharge from the emergency room. That is to say, the condition 'lab tests result normal' triggers the infrequent relation. This infrequent relation should not be removed, and it provides us with useful information about the process.

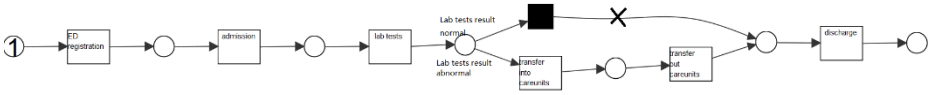

Fig. 1. A simplified process in Petri net from the emergency room transfer of a hospital

In this paper, we aim to discover value-based conditions and preserve legitimate infrequent relations using event log of lab tests. That is, we try to identify infrequent tests triggered by valued-based conditions from another lab test result. We aim to discover and preserve some unusual lab tests result may trigger infrequent relations between different lab tests.

The paper is organized as follows. Section 2 introduces the essential background and related work. Section 3 proposes the methodologies for this paper. Section 4 presents the results and discussion. Section 5 concludes the paper.

2 Background

Process discovery aims to find a process model that can represent the real process accurately, which means a balance between fitness, precision, generalization and simplicity [1]. As a result, most process mining techniques aim at discovering the main pathway of the process to avoid overfitting. However, there exist behaviors which are infrequent but important under specific context. In [4], researchers develop a tool to put the attributes of activities into consideration during the process discovery stage to discover such behaviors. Unfortunately, it has hardly been applied to the healthcare domain.

In recent years, there is an increasing number of applications of process mining in the field of healthcare. A big group of applications are based on fuzzy miner [5] and

Disco [6] since they are easy to use. These tools are also ideal for discovering casual relations between activities. For example, in [7], researchers use Disco to identify the process of emergency room and verify if certain clinical guidelines are satisfied. Since Disco and fuzzy miner use unified noise thresholds for all the behaviors [5, 6], such infrequent but important relations will be ignored. In [8], researchers combine the Disco tool with data mining techniques to estimate waiting time in the emergency department

Another group of researchers aim at extracting a model with semantics (i.e. BPMN, Petri-nets, casual-nets) from healthcare data. For example, [2] extracts data about patient transport from the MIMIC-III dataset to analyze how patients are transferred in or out of ICU by discovering process models using the inductive miner [9]. [9, 10] take both the order and resource of activities into consideration when discovering process models to study the allocation of resources of health services and medical facilities.

Although none of these studies focus on the critical infrequent relations among activities in healthcare, [9, 10] put the resource information into process discovery. [2] provides a guideline to extract event logs from the MIMIC-III database.

The dataset used for the application in this paper is Medical Information Mart for Intensive Care (MIMIC-III), which comprises EHR information relating to patients admitted to the critical care unit at a large tertiary care hospital [11]. By now, the latest version is MIMIC-III v1.4, which was released in September 2016 and is applied in the article. Meanwhile, besides activity names, MIMIC-III also contains other attributes for these activities, such as the results of the lab tests [11], which makes it possible to be applied in the tool proposed in [4] to discover those infrequent but important ordering conditional relations between activities.

3 Methodology

This study is exploratory to discover value-based conditions and preserve useful infrequent relations for standard lab tests in the MIMIC-III dataset. We modify and apply the L* life-cycle methodology [1], the data-driven heuristic miner (DHM) [4], and propose a novel data preprocessing approach to achieve the goal.

In this paper, we adopt standard lab tests obtained from [12]. There are several tables in the MIMIC-III dataset we used in data extraction. *d_lab_items* describes all the lab tests in the dataset, which severs as a dictionary. *labevents* contains all the lab measurements for patients and we adopt chart time as timestamp in the event log. *d_icd_diagnoses* is also a dictionary which has all the diseases along with their ICD-9 codes [13]. *diagnoses_icd* has all the diagnoses of the patients and coded using ICD system. *patients* contains the information of patients, such as gender.

The modified L* life-cycle methodology, shown in Fig. 2, contains four stages. Stage 0 is to plan and justify, which is the same as the first stage in the original L* life-cycle. This stage involves planning the research and proposing meaningful research tasks. Stage 1 is extracting and preprocessing, which consists of extracting the data needed from the MIMIC-III database and essential preprocessing to the data.

Because the data in the MIMIC-III dataset is not naturally designed for process mining task. So, some necessary preprocessing steps are also required. Stage 2 is to identify value-based conditional infrequent relations. This stage is designed explicitly for this study, which involves mining conditional infrequent relations using the DHM algorithm. Stage 3 is evaluation, which preserves useful and meaningful value-based conditions and infrequent relations through criteria.

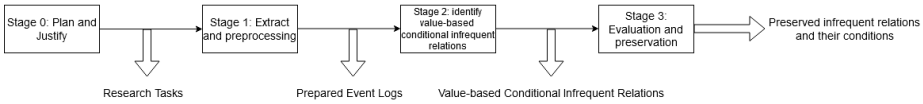

Fig. 2. Modified L* Life-Cycle Methodology

3.1 Stage 0: Plan and Justify

In this study, we present the following three research tasks to guide our research.

1. Identify infrequent relations between lab tests in different diseases.
2. Identify the value-based conditions that can trigger these infrequent relations.
3. Preserve useful and precise value-based conditions.

3.2 Stage 1: Extract and Preprocessing

We successfully spot 12 lab tests mentioned in [12], and their *item_id* are summarized in Table 1. Then, we extract the patients' lab test records, which contain one or more above tests. This operation returns a considerable dataset that has more than 53,132 traces and 1,781,458 events. We also select five various diseases using ICD-9 code [13], which are summarized in Table 2. Besides, the dataset of each illness is divided into two separate datasets, one contains female patients, and the other one contains male patients due to the need for evaluation.

Table 1. Summary of Common Lab Tests

#	*Item_id*	Description
1	50861	Alanine Aminotransferase (ALT)
2	50907	Total Cholesterol
3	51080	Creatinine Clearance
4	50889	C-Reactive Protein
5	51288	Sedimentation Rate
6	50924	Ferritin
7	50927	Gamma Glutamyltransferase
8	50809	Glucose
9	50811	Hemoglobin
10	50983	Sodium
11	50971	Potassium

12	50993	Thyroid Stimulating Hormone

Table 2. Summary of Diseases

#	ICD_9 code	Title
1	4019	Unspecified essential hypertension
2	4280	Congestive heart failure, unspecified
3	5849	Acute kidney failure, unspecified
4	41401	Coronary atherosclerosis of native coronary artery
5	42731	Atrial fibrillation

In this stage, three different preprocessing steps are applied to the data, which are changing data format, data cleaning and data aggregation. Since those conditions were likely to be discovered based on the attribute values (e.g. lab test results) from previous events, we have to consider all the lab test results and find a way to add these results to the event log as attributes. So, we manually add a column to store results for each test. Each column is named after the '$test\ name_{result}$' format to make it unique. So, during this stage, 12 columns are added, including ALT_{result}, $Sodium_{result}$.etc. Suppose the event is ALT test, then the result is stored at ALT_{result} column and all other columns of this event are left blank. Thus, each test has its attribute, and we can distinguish which test results can trigger the infrequent relations.

We also spot some traces may not be appropriate to use in this study. We conclude the common issues below. Those traces are all skipped to avoid producing unwanted and unrealistic relations.

- Traces contain only a single event.
- Traces contain only a single kind of test.
- Traces contain lab test results which have non-numerical or ambiguous value, such as 'greater than 2000', 'error,' 'smaller than 10', etc.

During our experiment, we find an interesting phenomenon of the *labevents* table in the MIMIC-III dataset is that many tests have the same timestamp, which means these tests usually happen together.

Since they are all happening at the same time, we cannot decide the actual order of these tests. If we still apply such event logs to the application, we may get unrealistic relations. So, we need to aggregate these tests while maintaining the result for each test. Another reason why we would like to aggregate these tests is that it is likely the doctor orders these tests in the same set. So, there are no value-based conditions among them.

The method takes the original log, pattern length, and pattern frequency as input. A pattern is a set of tests that happen at the same time. The length of the pattern determines the size of the pattern. For example, *l* is set to two, which means if two or more tests happening at the same time, we consider it is a pattern. Pattern frequency represents the times of occurrence for a specific pattern. The method first identifies possible patterns, which is larger than pattern length *I*, within each trace. Each time it finds a new pattern, adding it to the set E. All the patterns in E are unique. Next, it goes

through the set E and counts how many times each pattern appears in the log *L*. If the occurrence times of a pattern is greater than pattern frequency *f*, we merge each occurrence of that pattern by combining tests using '+' into a set of tests as a single event, while maintaining all the results and time for the tests as attributes. By this, we group tests that usually happen at the same time with high frequency. This helps to make the event logs transparent. Table 3 summarizes the datasets.

Table 3. Summary of datasets

#	Name	Total Traces	Total Events	Total Activities	Average Trace Length	Maximum Trace Length
1	Icd_42731_female	5381	111939	23	21	225
2	Icd_42731_male	7279	169773	25	23	348
3	Icd_4019_female	9170	147558	27	16	290
4	Icd_4019_male	11161	193266	26	17	571
5	Icd_41401_female	4250	80792	21	19	244
6	Icd_41401_male	7951	153178	23	19	305
7	Icd_5849_female	3894	83708	22	21	236
8	Icd_5849_male	5142	118856	24	23	571
9	Icd_4280_female	6007	120775	25	20	280
10	Icd_4280_male	6935	158164	26	23	478

3.3 Stage 2: Identify value-based conditional infrequent relations

In this stage, we apply the 'Interactive Data-aware Heuristic Miner (iDHM)' plug-in [4] in the ProM tool [14]. The plug-in is an implementation of the DHM algorithm. There are four primary parameters in the plug-in, which are:

- θ_obs controls the relative frequency of relations.
- θ_dep controls the frequency of relations.
- θ_bin controls the number of bindings.
- θ_con controls the quality of conditions.

DHM improves the traditional Heuristic Miner (HM) [15] and can reveal conditional infrequent behaviors from the event log. Fig. 3 illustrates the overall steps for DHM. First, the DHM tries to calculate the frequency of every relation by adopting the ideas from the traditional HM. Then, it identifies infrequent relations through θ_dep and θ_obs. The dependency of the relations is smaller than the dependence threshold, which is θ_dep, or the relative frequency is smaller than the observation threshold, which is θ_obs. Such relations would be considered as infrequent relations. We adopt the default value for θ_obs which is 0.1. We think the default value 0.9 for θ_dep is too limited for the lab tests in the MIMIC-III dataset. So, the dependency threshold is adjusted to 0.8 in our study. The next step is discovering the conditions which may trigger the infrequent relations we identified in the last step. DHM builds training instances for every infrequent relation and deploys the decision tree to train

the instances using attributes of these events. Unlike the attributes we usually see, the attributes act as specific identifiers for an activity. There are no unified attributes for all activities. Fig. 4 provides an example. Suppose the infrequent relation we find is $Test\ set\ A \rightarrow Test\ B$. Test set A consists of three different tests, and each test has a result. We store each result as an attribute for test set A, namely attribute A, B and C respectively. We also have a frequent relation, which is $Test\ set\ C \rightarrow Test\ A$. Test set C also contains two tests and each test result is stored as an attribute of test set C, called attribute C and D. When DHM tries to discover the conditions of the infrequent relation we just mentioned, it will take all the events' attributes before test B into consideration, which are attribute A, B, C and D. By doing this, the algorithm can consider all the attributes before the relation instead of only the one from the direct predecessor. However, it is notable that there are two attribute C in the diagram, the DHM will only consider the latest one, which is the one from test set A. Then the DHM provides two ways to evaluate the conditions. One is using Cohen's kappa [16] to assess the conditions directly. Those conditions score lower than θ_con will be discarded. We keep the default value for θ_con which is 0.5. The other way to evaluate the condition is that the frequency for the relations under the conditions must exceed the dependence threshold (θ_dep), which is 0.8 in this study. Since the iDHM plug-in returns a C-net, it is important to make sure the graph is connected. The DHM adopts accepted-task-connected heuristic to ensure the connected graph. Since in this paper the primary focus is not on the overall graph, we will not go into detail for this step. Last step is to rely on θ_bin to mine the suitable input and output binding for the graph. We also keep the default value for this, which is 0.1. In the plug-in, we choose to let the DHM consider all the lab test results when mining value-based conditions.

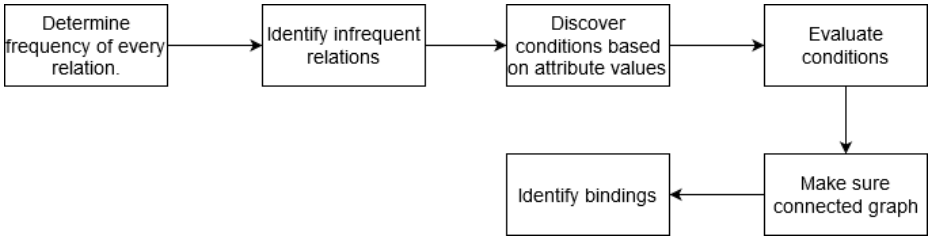

Fig. 3. Steps of DHM Algorithm

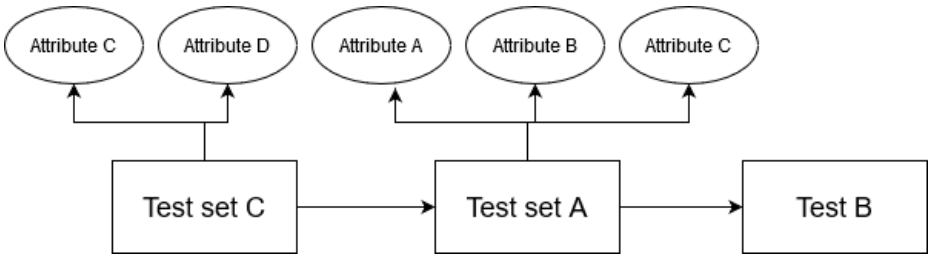

Fig. 4. An Example of How DHM Discover Conditions Based on Attributes

3.4 Stage 3: Evaluation and Preservation

We evaluate the results in stage 2 from several perspectives. First, we exclude the relations which contain the 'artificial start' or 'artificial end'. Because the iDHM plug-in adds these two activities to corresponding to the requirements of C-net. It makes no real sense. For example, $Sodium \longrightarrow Artificial\_end$ relation will be discarded, which is represented as 'Discard (1)' in Table 4. Since we are interested in the conditions derived from the previous test results, which may trigger another test, so all relations with no such conditions will not be included, written as 'Discard (2)'. For example, $Sodium \longrightarrow Glucose$ relation with the condition $Potassium_{result} > 12$ mEq/L will be ignored because the condition is irrelevant to the Sodium test. Besides, we also measure the accuracy of each condition. Those conditions with the accuracy below a certain threshold will not be included in the final result, represented as 'Discard (3)'.

Table 4. Summary of discard reasons

#	Reason
Discard (1)	Relations contain artificial start or end.
Discard (2)	Conditions are not fully derived from previous test results.
Discard (3)	Conditions accuracy below certain threshold.

We separate each disease dataset into male and female and randomly pick one as the training set, which is the input for the DHM algorithm, and the other one serves as the evaluation set. The concept of the confusion matrix is implemented in this stage for evaluation. $ALT \rightarrow Potassium$ relation with the condition $ALT_{result} < 21$ IU/L serves as an example to illustrate the confusion matrix. True-positive (TP) is the number of relations that happen under the condition we find, i.e. we can find a case when a patient does the potassium test directly after the ALT test with the $ALT_{result} < 21\ IU/L$. True-negative (TN) is the number of relations that do not occur if the condition is not met, i.e. a case when the patient with $ALT_{result} \geq 21\ IU/L$, we cannot find the directly follow relation mentioned above. False-positive (FP) is the number of relations that do not happen given the condition is met, i.e. there is a patient with $ALT_{result} < 21\ IU/L$, but he did not take Potassium test directly after the ALT test. False-negative (FN) is the number of relations that happen given the condition is not met, i.e. the patient toke the potassium test directly after the ALT test, but his $ALT_{result} \geq 21\ IU/L$. The equation for calculating accuracy is provided in equation (1). Then those infrequent relations, along with their value-based conditions, can be preserved if they pass a certain accuracy threshold.

$$Accuracy\ = (TP + TN)/(TP + TN + FP + FN\ ) \quad (1)$$

4 Results and Discussion

In total, we successfully identify 24 infrequent relations out of five diseases. Among them, 12 relations are discarded because of the evaluation standards and the rest are

preserved. Most of the infrequent relations we find occur less than 10 times in the event log. We regard the rest 12 as the useful relations with value-based conditions. Among the 12 removed relations, 5 are due to accuracy below 0.7 and 4 are because of containing artificial end. Each relation is evaluated using a separate dataset and methods proposed in stage 3. On average, the preserved relations achieve 87% accuracy. The highest accuracy is 97%, and the lowest is 70%.

Through applying the methodology described in section 3, the tasks proposed in stage 0 can be easily achieved. Regarding to the first task, we randomly pick a female or male dataset from each disease described in section 3 to mine the infrequent relations, and the results are summarized in Table 5. Taking atrial fibrillation (ICD-9: 42731) as an example. We pick the female dataset and find six infrequent relations. One infrequent relation can be after the patients going through a set of tests, which includes ALT, Ferritin, Potassium and Sodium. They may directly go for the Hemoglobin test under some circumstances. For the second task, the conditions for each infrequent relation are also presented in Table 5. For example, the condition triggers the first relation in atrial fibrillation is $Ferritin_{result} > 335\ \mathrm{ng/ml}$. We can interpret the condition in this way. Normally, this relation will not frequently happen in reality and are considered as noise. However, suppose the Ferritin result for a patient is greater than 355 ng/ml during the sets of tests, which includes ALT, Ferritin, Potassium and Sodium, then the patient is very likely to be asked to perform another Hemoglobin test afterwards. For task 3, we evaluate the conditions according to the criteria proposed in section 3. We remove the conditions that do not satisfy the requirements and preserve the rest, which is considered as useful. The accuracy threshold we adopt is 70%. These provide us with an insight into the infrequent lab test relations and their value-based trigger in different diseases.

However, we also notice that some conditions have low accuracy. This may result from several reasons. First, the infrequent conditional relations may not be the same in a variety of datasets because we separate the datasets of each disease according to gender. The natural difference may exist between them. Besides, various admission types exist in the MIMIC-III dataset, such as elective and emergency. Understandably, the ordering of lab tests for different admission types are varied, which leads to possible future research.

5 Conclusion

In this paper, we successfully implemented a value-based condition discovery pipeline on a publicly available hospital EHR dataset. We can preserve legitimate infrequent relations. Otherwise, it would have been removed from the original process mining algorithm. We modify and apply L* life-cycle methodology and Data-aware Heuristic (DHM) for the study. Besides, we propose a new way to automate aggregate events. The results and evaluation show that we can successfully preserve useful value-based conditions for infrequent relations discovered using event log of lab tests in the MMIMC-III dataset. Future work can be investing the differences between each disease and relying on domain knowledge to evaluate the conditions.

Table 5. Summary of Results

Infrequent Relation	Value-based Condition	Keep or Not	Accuracy
Training dataset: Icd_4280_female		**Evaluation dataset: Icd_4280_male**	
$ALT + Cholesterol + Potassium + Sodium \longrightarrow Hemoglobin$	$ALT_{result} <= 21$ IU/L	Discard (3)	0.69
$ALT + Cholesterol + Patassium + Sodium \longrightarrow Hemoglobin + Glucose$	$ALT_{result} <= 62$ IU/L	Discard (3)	0.69
$Hemoglobin \longrightarrow Ferritin + Potassium + Sodium + Thyroid\ Stimulating\ Hormone$	$Hemoglobin_{result} > 12.2$ g/dL	Keep	0.85
$Potassium + Sodium + Thyroid\ Stimulating\ Hormone \longrightarrow Sodium$	$Sodium_{result} <= 129$ mEq/L	Keep	0.97
Training dataset: Icd_5849_male		**Evaluation dataset: Icd_5849_female**	
$Cholesterol + Potassium + Sodium \longrightarrow Ferritin + Potassium + Sodium$	$(Sodium_{result} > 145$ mEq/ L)&&$(cholesterol_{result} < 169$mg/dL)	Keep	0.94
$Cholesterol + Potassium + Sodium \longrightarrow Cholesterol + Potassium + Sodium$	$cholesterol_{result} > 169$ mg/dL	Keep	0.70
$ALT + Potassium + Sodium + Thyroid\ Stimulating\ Hormone \longrightarrow ALT + Ferritin + Potassium + Sodium$	$Sodium_{result} <= 134$ mEq/L	Keep	0.81
$Ferritin + Potassium + Sodium \longrightarrow ARTIFICIAL\_END$		Discard (1)	
Training dataset: Icd_41401_male		**Evaluation dataset: Icd_41401_female**	
$ALT + Cholesterol + Potassium + Sodium \longrightarrow ALT + Cholesterol + Potassium + Sodium$	$(Potassium_{result} <= 3.4$ mEq/ L)&&$(cholesterol_{result} < 190$mg/dL)	Keep	0.91
$ALT + Ferritin + Potassium + Sodium \longrightarrow ALT + Ferritin + Potassium + Sodium$	$Sodium_{result} <= 131$ mEq/L	Keep	0.93
$Sodium \longrightarrow Potassium + Sodium +$	$(Potassium_{result} > 4.6$ mEq/	Discard (2)	

$Thyroid\ Stimulating\ Hormone$	L)&&($Sodium_{result}$ <= 129 mEq/L)		
Training dataset: Icd_4019_female		**Evaluation dataset: Icd_4019_male**	
$Potassium + Sedimentation\ Rate + Sodium \longrightarrow C - Reactive\ Protein$	ESR_{result} > 61.0 mm/hr	Discard (3)	0.55
$Potassium + Sedimentation\ Rate + Sodium \longrightarrow Potassium$	($Sodium_{result}$ > 140 mEq/ L)&&(ESR_{result} <= 36) mm/hr	Keep	0.90
$Thyroid\ Stimulating\ Hormone \longrightarrow ALT + Potassium + Sodium + Thyroid\ Stimulating\ Hormone$	$Sodium_{result}$ <= 135.0 mEq/L	Discard (2)	
$ALT + Cholesterol + Potassium + Sodium \longrightarrow Glucose + Hemoglobin$	$Sodium_{result}$ > 142 mEq/L	Keep	0.82
$Potassium + Sedimentation\ Rate + Sodium \longrightarrow Potassium + Sedimentation\ Rate + Sodium$	ESR_{result} > 65 mm/hr	Discard (3)	0.58
$Hemoglobin \longrightarrow Hemoglobin$	$Potassium_{result}$ <= 4.4 mEq/L	Discard (2)	
$ALT + Potassium \longrightarrow ARTIFICIAL\_END$		Discard (1)	
Training dataset: Icd_42731_female		**Evaluation dataset: Icd_42731_male**	
$ALT + Ferritin + Potassium + Sodium \longrightarrow Hemoglobin$	$Ferritin_{result}$ > 335 ng/ml	Discard (3)	0.43
$ALT + Ferritin + Potassium + Sodium \longrightarrow ARTIFICIAL\_END$		Discard (1)	
$Ferritin + Potassium + Sodium \longrightarrow Sodium$	$Sodium_{result}$ <= 128 mEq/L	Keep	0.98
$ALT + Ferritin + Potassium + Sodium \longrightarrow Cholesterol + Potassium + Sodium$	($Ferritin_{result}$ > 149ng/ ml)&&($Ferritin_{result}$ < 311ng/ml)	Keep	0.79
$Thyroid\ Stimulating\ Hormone \longrightarrow ARTIFICIAL\_END$		Discard (1)	
$ALT + Ferritin + Potassium + Sodium \longrightarrow Glucose + Hemoglobin$	$Ferritin_{result}$ <= 114 ng/ml	Keep	0.85

References

1. van der Aalst, W.: Data Science in Action. Process Mining: Data Science in Action, pp. 3-23. Springer Berlin Heidelberg, Berlin, Heidelberg (2016)
2. Kurniati, A.P., Hall, G., Hogg, D., Johnson, O.: Process mining in oncology using the MIMIC-III dataset. Journal of Physics: Conference Series 971, 012008 (2018)
3. Fernandez-Llatas, C., Ibanez-Sanchez, G., Celda, A., Mandingorra, J., Aparici-Tortajada, L., Martinez-Millana, A., Munoz-Gama, J., Sepúlveda, M., Rojas, E., Gálvez, V., Capurro, D., Traver, V.: Analyzing Medical Emergency Processes with Process Mining: The Stroke Case. In: Business Process Management Workshops, pp. 214-225. Springer International Publishing, (2019)
4. Mannhardt, F., de Leoni, M., Reijers, H.A., van der Aalst, W.M.P.: Data-Driven Process Discovery - Revealing Conditional Infrequent Behavior from Event Logs. In: Advanced Information Systems Engineering, pp. 545-560. Springer International Publishing, (2017)
5. Pelekis, N., Theodoulikis, B., Kopanakis, I., Theodoridis, Y.: Fuzzy miner: extracting fuzzy rules from numerical patterns. International Journal of Data Warehousing and Mining (IJDWM) 1, 57-81 (2005)
6. Günther, C.W., Rozinat, A.: Disco: Discover Your Processes. BPM (Demos) 940, 40-44 (2012)
7. Alvarez, C., Rojas, E., Arias, M., Munoz-Gama, J., Sepúlveda, M., Herskovic, V., Capurro, D.: Discovering role interaction models in the Emergency Room using Process Mining. Journal of Biomedical Informatics 78, 60-77 (2018)
8. Benevento, E., Aloini, D., Squicciarini, N., Dulmin, R., Mininno, V.: Queue-based features for dynamic waiting time prediction in emergency department. Measuring Business Excellence (2019)
9. Stefanini, A., Aloini, D., Benevento, E., Dulmin, R., Mininno, V.: A data-driven methodology for supporting resource planning of health services. Socio-Economic Planning Sciences 70, 100744 (2020)
10.Prokofyeva, E.S., Maltseva, S.V., Fomichev, N.Y., Kudryashov, A.G.: Data-Driven Approach To Patient Flow Management And Resource Utilization In Urban Medical Facilities. In: 2020 IEEE 22nd Conference on Business Informatics (CBI), pp. 71-77. (2020)
11.Johnson, A.E.W., Pollard, T.J., Shen, L., Lehman, L.-w.H., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Anthony Celi, L., Mark, R.G.: MIMIC-III, a freely accessible critical care database. Scientific Data 3, 160035 (2016)
12.Houben, P., Winkens, R., van der weijden, T., Vossen, R., Naus, A., Grol, R.: Reasons for ordering laboratory tests and relationship with frequency of abnormal results. Scandinavian journal of primary health care 28, 18-23 (2010)
13.https://www.cdc.gov/nchs/icd/icd9cm.htm
14.van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The ProM Framework: A New Era in Process Mining Tool Support. In: Applications and Theory of Petri Nets 2005, pp. 444-454. Springer Berlin Heidelberg, (2005)
15.Weijters, A.J.M.M., Ribeiro, J.T.S.: Flexible Heuristics Miner (FHM). In: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 310-317. (2011)
16.Cohen, J.: A coefficient of agreement for nominal scales. Educational and psychological measurement 20, 37-46 (1960)