Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital

Sjoerd van Der Spoel, Maurice Van Keulen, Chintan Amrit

Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital

Sjoerd van der Spoel, Maurice van Keulen, Chintan Amrit

University of Twente, The Netherlands,
{s.j.vanderspoel;m.vankeulen;c.amrit}@utwente.nl

Abstract. Predicting the amount of money that can be claimed is critical to the effective running of an Hospital. In this paper we describe a case study of a Dutch Hospital where we use process mining to predict the cash flow of the Hospital. In order to predict the cost of a treatment, we use different data mining techniques to predict the sequence of treatments administered, the duration and the final "care product" or diagnosis of the patient. While performing the data analysis we encountered three specific kinds of noise that we call *sequence noise*, *human noise* and *duration noise*. Studies in the past have discussed ways to reduce the noise in process data. However, it is not very clear what effect the noise has to different kinds of process analysis. In this paper we describe the combined effect of *sequence noise*, *human noise* and *duration noise* on the analysis of process data, by comparing the performance of several mining techniques on the data.

Key words: process prediction, process mining, classification, cash flow prediction, data noise, case study

1 Introduction

In the Netherlands, insurance companies play an important role in settling the finances of medical care. Hospitals and other care providers claim their costs for treating a patient with the patient's insurance company, who then bill their customer if there are costs not covered by the insurance.

Starting January 1st, 2012, Dutch hospitals are required to use a new system, called "DOT", for claiming their costs for treating patients. The central principle of DOT is that the amount of the claim is based on the actual care provided, which is only known after the treatment has finished. Previously, the amount of the claim was based on the diagnosis for which average costs would be determined by negotiation between the government, hospitals and insurance companies.

This change confronted the hospitals with a finance management problem. Whereas they previously could claim costs already after diagnosis, they now have to wait until treatment has finished. Since it is unknown ahead of time when treatments will finish, it also unknown when the hospital can expect cash flow for their treatment processes and how large those cash flows will be.

Cash flow prediction for treatment processes seems like a problem that a combination of data and process mining could well help to solve. A treatment

process of one patient, called a *care path*, can be seen as a sequence of activities which determines the so-called *care product* with an associated cost. Based on a data set with all details about completed care paths and associated care products, one could develop a predictor that, given only the start of such a path, could predict the rest of the path and its associated duration and cost.

This paper describes our experiences and results with this case study in data and process mining. The paper specifically addresses a problem with three kinds of noise in our data that significantly affected our results. It is of course common knowledge that noise in the underlying data may affect the result of any kind of data mining including process mining. We encountered, however, three different kinds of noise that were rather specific to our process mining and that were so prevalent that known solutions from literature did not apply. We call these kinds of noise (a) *sequence noise* meaning errors in or uncertainty about the order of events in an event trace, in our case study the order of activities in a care path, (b) *duration noise* meaning noise arising from missing or wrong time stamps for activities and variable duration between activities. (c) *human noise* meaning noise from human errors such as activities in a care path which were the result of a wrong or faulty diagnosis or from a faulty execution of a treatment or procedure.

The reason why sequence uncertainty was so prevalent in our case study is as follows. During the day, a medical specialist typically sees many patients. During a consult or treatment, however, it is often too disruptive for the specialist to update the patient's electronic dossier. It is common practice that (s)he updates the dossiers at the end of day or even later. As a consequence, the modification time of a patient's dossier (which is what is recorded) does not reflect the actual moment of the activity, and it often does not even reflect the order in which the activities took place during that day. Furthermore, it also common that patients undergo several activities and see several specialists on the same day. For example, a hospitalized patient may receive a visit from a specialist doing his/her rounds, receive medication, undergo surgery, results from a blood analysis may be finished, all on the same day. In the case study data sets, we have averages of about 2.5 to 10 activities per day. This inherent noisy timestamp problem and its magnitude causes major *sequence noise* in the underlying data.

Furthermore, the noisy timestamp issue also causes the *duration noise*, as it makes the calculation of the duration between activities very noisy and error prone. Also, much of the *duration noise* comes from the fact that in our case study data two activities are considered by the client to be consecutive even if they are separated by many days or weeks, as long as they are part of the same treatment and no other activities have occurred in between them.

On the other hand, *human noise* arises from human errors which include (i) Noise due to a wrong or faulty diagnosis - this leads to a faulty extra series of process steps at the beginning of the treatment process, (ii) Noise due to faulty execution of the treatment and/or erroneous procedures that need to be repeated. So, our data contains large amounts of *sequence noise* + *human noise*+ *duration noise* and when combined, these make the prospect of noise removal

very complicated and difficult to perform. Hence, in our paper we proceed by analysing the noisy data to demonstrate what kind of results one can expect with different data mining analysis techniques.

1.1 Contribution

The contributions of this paper are:
- A case study of applying data and process mining for cash flow prediction in a Dutch hospital.
- Accuracy results for different prediction tasks: given a diagnosis and start of a care path, predict the rest of the path, the duration of the path, the final care product, and the associated cash flow.
- Experimental comparison of the performance of several mining techniques on these tasks.
- Analysis and discussion on the effects of sequence and human noise, both are kinds of data noise specific for process mining which have received little attention in literature.

2 Case study

Starting January 1st, 2012, Dutch hospitals will use a new system for claiming the costs they make for treating patients. Hospitals claim these costs at patient's insurance companies, who then bill their customers.

2.1 DOT registration system

The new registration and claiming system addresses the problem that actual costs are unequal to what is charged, in an effort to increase the transparency of cost calculation for provided care. The system is referred to as DOT, which stands for "DBCs towards transparency". A DBC is a combination of diagnosis (D) and treatment (Dutch: behandeling, B).

The central principle of DOT is to decide the care product based on the care provided. Whereas, in the previous registration system the care product was based solely on the diagnosis.

A treatment path in the DOT system consists of the following:

- Care product and associated care product code: This is the name and code given to the treatment performed, which is made up one or more sub paths (Care types).
- Sub path: Is a sequence of activities performed (with one or more Activity types) in a treatment sequence
- Care type and associated care type code: Is the specific name and code associated with the sub path
- Activity type and associated activity type code: Are the activities performed for a particular care type in a sub path

The hospital data is structured into several tables, of which two are important in this case: *Activities* and *Sub paths*. An activity represents an item of work, like surgery or physical therapy, but also, days spent in hospital are considered activities. Groups of activities that are performed to treat the individual patient's diagnosis are called sub paths. They are called sub paths because multiple more-or-less independent sub paths make up the total care path for a patient: the path from the patient registering some complaint to being fully treated. Care paths do not have care products, but sub paths do. Table 1 shows the structure of the activity data, table 2 shows the same for the sub path data.

A care product has an associated cost, which is claimed at a patient's insurance company. To derive which care product a patient has received, the DOT system uses a system called the *grouper*. This grouper consists of rules that specify how care products are derived from performed activities. In the DOT methodology, it is the activities performed that decide the care product. This does not mean that every activity influences the care product, in fact, laboratory work and medicines have no influence on the grouper result. DOTs are processed by the grouper after they have been closed. When a DOT registration is closed depends on the amount of time that has passed since the last activity. If more than the specified number of days has passed, the DOT is marked closed. Different types of activities have different durations after which the DOT is to be marked as closed. The grouper is maintained and operated by the independent DBC Maintenance authority. The rules that make up the grouper are the results of negotiations between the academic Dutch hospitals and insurance companies.

The DOT system should lead to a better matching between actual provided care and associated care product. In turn, this leads to what the insurance companies (and therefore patients) pay.

While a patient is still undergoing treatment, the DOT system poses two problems:

- The hospital does not know how much they will receive for the care they have provided, as they don't know what care product will be associated with the open DOTs.
- The hospital does not know when a DOT is likely to be closed, because a DOT closes only some time after the final treatment. If the patient turns out to require another treatment before the closing date of the DOT (based on the previous treatment), the closing date moves further into the future. Because the hospital does not know when a DOT closes, they also don't know when they can claim the cost of the associated care product.

These problems are in essence process prediction problems. The first involves predicting the process steps, because these process steps dictate the care product. The second problem involves predicting the duration of a process. To see how well different approaches work for this practical case, we test predicting care product, product cost and care duration. For this purpose we use anonymized patient data from a Dutch hospital. The data we have available is based on heart and lung specialties.

Dataset construction To produce data sets for our experiment, the activity and sub path tables where joined to get a new table with the sub path id, care product, activity code and registration date. A separate table was made for every diagnosis, as the diagnosis is not part of prediction, but is known at the start of a sub path. The produced tables where converted to a *Sub path id* − *Care product* − *Activity 1* − . . . − *Activity n* format. The data format is explained in detail in section 5.

Because activities are recorded every day and not at the moment they are performed, we do not know the sequence of activities within a day. This leads to *sequence noise* and *duration noise* in the data sets, as the inferred sequence is not necessarily the right one, as we approximate the sequencing by ordering activity codes alphabetically. The *sequence noise*, *duration noise* and the *human noise* increases the complexity of the task of finding the right set of future activities, and then determining its duration.

Note that we did not artificially add noise: the noise is the result of the lack of explicit time-based sequence of activities in the original data, so we had to recover the exact sequence where it was implicit. The approaches we present in this paper for predicting "care products" will have to deal with this fact, as it is a consequence of the way data is stored at the hospital.

3 Problem formalization

We start with providing notation for and defining the most important concepts in our case study in order to be able to more precisely define the prediction tasks that we distinguish.

3.1 Basic concepts

An *activity* Act is defined as a label taken from the set of possible DBC codes. A *care path* P is a sequence of activities $P = Act_1, \ldots, Act_n$. We focus our prediction only on subpaths which have a unique care product, so a care path should be interpreted as a subpath. We sometimes denote a care path with $\hat{P}$ to emphasize that it is *closed*, i.e., it belongs to a finished treatment or to a DOT that was closed by the grouper for some reason. The last activity in a closed care path is denoted with Act_{end}; the last activity of an 'incomplete' care path is often denoted with Act_{cur} to emphasize its role as 'current' activity. $P_1 \oplus P_2$ denotes the *concatenation* of P_1 and P_2. $d = d_P$ denotes the *duration* of care path P in terms of time (measured in days in our case study).

The *grouper* Grp is a function that determines the *care product* $C = Grp(\hat{P})$ for a given path $\hat{P}$. The associated *cost* of a care product C is denoted by $cost(C)$.

From a subset of our data, we construct a directed weighted *process graph* $G = (N, E)$, where the nodes N represent activities and the edges e the possibility that one activity can follow another. The *weight of an edge* $w(e)$ is defined as

the number of occurrences that these two nodes follow each other in that order. We added a node "Start" to each G to obtain a single starting point for all care paths. We furthermore added the care products as separate terminal nodes, such that for each $\hat{P}$, $Grp(\hat{P}) = C = Act_{\text{end}}$.

We assume that the available underlying data is in the form of a set of complete care paths. We chose to work with separate sets of care paths each belonging to one specialty and diagnosis code.

3.2 Prediction tasks

With the notation above, we can precisely define the prediction tasks we consider in this paper. Let $P = Act_2, \ldots, Act_{\text{cur}}$[1], $P' = Act_{\text{cur}+1}, \ldots, Act_{\text{end}}$, and $\hat{P} = P \oplus P'$. The prediction tasks are:

1a) Given an incomplete care path P, predict the care product C directly (i.e., without considering or predicting P'). We view this prediction task as a *classification* task. We use a subset of the closed care paths belonging to one diagnosis with their associated care product as training data for supervised learning of the classifier.
1b) A predicted care product determines its cost $cost(C)$.
2a) Given a process graph G constructed from a subset of the care paths belonging to one diagnosis, an activity Act_{cur}, and a care product $C = Act_{\text{end}}$, predict the path P' in between. We view this prediction task as a *process mining* task. Note that we attempt our prediction given only the current activity Act_{cur} and not the path leading to this activity P. The latter is left to future research.
2b) A predicted path determines its duration, i.e., $d_{P'}$, from which a prediction can be derived for the full duration $d_{\hat{P}}$ given G, Act_{cur}, and C.

In the end the financial department of the hospital is interested in an amount of money (i.e., $cost(Grp(\hat{P}))$) and a moment in time (which can be derived from $d_{\hat{P}}$). It may happen, however, that an entirely wrong care product is predicted, but that it has a similar cost, or that an entirely wrong path P' is predicted with a similar length. In those cases, the prediction of what we are ultimately interested in is close, but rather unjustifiably so. Therefore, we target not only the b-tasks, but also the a-ones. Moreover, examining the results of predicting the rest of the path also provides more insight into the effects of sequence and human noise.

4 Literature Review

4.1 Classifier Algorithms

Prediction tasks 1a and 1b are about assigning a class - the careproduct - to a combination of independent variables - the activities. That is why we consider

[1] We start with activity 2, because activity 1 is always "Start".

this to be a classification problem. The field of classifier algorithms divides into four categories: decision tree, clustering, bayesian and neural network classifiers [1]. Besides these classifiers, there are some classifier aggregation techniques that serve to augment the accuracy of individual classifiers.

Decision tree classifiers are based on Hunt's algorithm for growing a tree by selecting attributes from a training set. The attributes are are converted into rules to split the data. Although this can be an accurate technique, the decision tree family of classifiers is sensitive to overfitting or overtraining: training the classifier on a data set with noisy data will include bad (too training-set specific) rules in the tree, reducing accuracy. Algorithms like C4.5 [2] use pruning to prevent overtraining, but the fact remains that decision trees are not effective in very noisy data.

Clustering algorithms for classification use distance (or equivalently: proximity) between instances to group them into clusters or classes. New instances are classified based on their proximity to existing clusters [3]. Examples are nearest-neighbour classifiers and support vector machines.

Bayesian classifiers use Bayes' rule to select the class that has the highest probability given a set of attribute values. Bayes' rule assumes independency of the attributes in a data set, if this is not the case, classifier accuracy could be hindered.

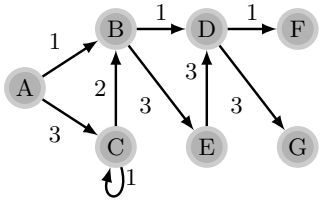
Neural networks are composed of multiple layers of elements that mimic biological neurons, called perceptrons. The perceptrons are trained to give a certain output signal based on some input signal, which is propagated to the next layer of the neural network. The network consists of one input layer, one output layer and multiple hidden layers in between. Neural networks are computationally intensive, but are accurate classifiers. However, neural networks are sensitive to overtraining [3].

Besides techniques consisting of a single (trained) classifier instance, there are techniques that aggregate results from multiple classifiers. Examples are bootstrap aggregating [4], boosting [5] and the Random Forests algorithm [6]. The techniques have the same paradigm: let every individual classifier vote on the class of the data instance, the class with most votes is selected. Breiman found that this principle is especially accurate when the underlying classifier has mixed performance on the dataset [4], something that occurs most often with the less complex algorithms. That is also the case for the Random Forest classifier, which is comprised of multiple randomly grown decision trees. All these aggregation techniques generally display better accuracy than the individual classifiers.

Curram *et al.* [7] show that decision tree classifiers (such as CART and C4.5 [2]) are outperformed by neural networks. In turn, neural networks are less accurate than the boosting type of classifier aggregation, as shown by Alfaro *et al.* [8]. Furthermore, the Random Forest classifier [6] has been shown to have good accuracy compared to other classifiers [9] [10]. Therefore, we use boosting (specifically, Freund and Schapire's Adaboost algorithm [5]) and Random Forests for care product classification. For some comparison, we also investigate

#	*Trace*
1.	$\{A, B, E, D, G\}$
2.	$\{A, C, C\}$
3.	$\{B, E, D, F\}$
4.	$\{A, C, B, D, G\}$
5.	$\{B, D, G\}$
6.	$\{A, C, B, E, D, G\}$

(a) Example set of traces



(b) Elicitated graph

Fig. 1: Converting traces to a graph

the performance of the CART decision tree algorithm and the naive Bayesian classifier.

4.2 Process prediction algorithms

To predict the care activities (path through the process graph), we need for a complete and accurate graph. A possible approach is to take the event log and create a graph that contains every edge in every trace of the log. Consider the set of traces in figure 1a, converted to the graph in figure 1b. Every combination of two subsequent activities is an edge in the graph, every activity is a node. There are no duplicate edges or nodes, but the number of occurrences of two nodes subsequently is noted as the weight of the edge.

The algorithm is naive, as it also includes possibly noisy edges that only occur a few times in the data set. In figure 1b, the $C - C$ edge could be noise, it occurs only once in the data. This makes the naive graph elicitation algorithm inaccurate. To remedy this flaw inherent to the naive approach, the weights of the nodes can be considered as the likelihood of an edge *not* being noise. The more times an edge occurs, the less likely it is to be noise.

Not only individual edges are noisy, also larger sets of edges might be noise. This would result in several nodes connected by low-weight edges. This points to a possible solution: use a path algorithm to determine paths that have the highest weight, or to exclude the paths with the lowest weight. This type of algorithm is known as a shortest-path-algorithm: find a path through the graph that has the lowest possible weight [11].

The most well-known shortest-path-algorithm is Dijkstra's algorithm, which calculates the single shortest path through a series of nodes. This algorithm can be used to determine the distances from a start node to every other node. A distance is the compound weight of the edges towards a node. A modification of Dijkstra's algorithm determines the path that has the highest weight for all its edges combined. To remove (infinite) loops duplicate activities within a trace are

relabelled so that they are unique. In figure 1b, this means that C is replaced by C and C', $C - C$ is replaced by $C - C'$.

There are some problems when using Dijkstra: the algorithm has to be run for every node in the graph to give the shortest or longest path from the start node to some other node. Second: Dijkstra cannot give the shortest path between two specific nodes, it only gives the node that has the shortest/highest distance from "start". A different shortest-path algorithm, known as Floyd-Warshall does have this capability: it gives every shortest path between two nodes. This makes it possible to find the most likely path or most unlikely path from one node to another.

Besides shortest path approaches that determine path likelihood by taking the sum of edge weights, another approach is to take the product of the probabilities of the edges. The probability of an edge E from node A to B is determined by the number of times it is traversed divided by the total outgoing weight of A's edges. Given the edge probabilities, the most likely path (based on Bayes' rule, assuming independency of the next edge on previously traversed edges) is the path with the highest product of edge probabilities. This is similar to a Markov chain.

Floyd-Warshall and Dijkstra's algorithm produce the path (set of edges) that has the lowest sum of edge weights. Since both produce the shortest path, both algorithms can be used, but we choose Floyd-Warshall, as it is easier to implement Floyd-Warshall. The adaptation we made on Floyd-Warshall's algorithm does not take the sum of edge weight as the measure of length, but the product of edge weights. We then look for the path (set of edges) that has the maximum product of edge weights: the most likely path.

Finally, we have considered techniques from the process mining field, such as the alpha-algorithm proposed by Van der Aalst *et al.*, the Little Thumb algorithm, InWolve and the suite of algorithms provided by the ProM framework[12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. We have found these algorithms to be unsuitable for our goals, as they do not deal with cycles in the data [23]. Since cycles are a central aspect of the noise in the data available, we do not consider the process mining family of techniques/algorithms useful for our purposes.

4.3 Noise in Process Mining

Datta (1998) [24] defines noise in a business process as an unrelated activity that is not part of any activity of the business process. An example of such an activity would be a phone call or a lunch break [24]. They suggest three techniques for the removal of such noise. The techniques being a stochastic strategy and two algorithmic strategies based on a finite state machine [24].

Weijters and van der Aalst [25] regard noise as (i) a missing activity and/or (ii) randomly swapped activities in the workflow log. To extract the work flow process from the log, Weijters and van der Aalst [25] construct a dependency-frequency (D/F) table (which holds the frequency and order of task co-occurrence or dependencies), from which they construct a D/F graph (the

graph of the task dependencies)and finally a work-flow net (or WF net, that represents the D/F graph along with the splits and joins) using heuristics [25].

Both Agrawal et al. [26] as well as Huang and Yang's [27] define noise as instances when executed activities were not collocated, timestamps of the activities are mistakenly recorded or exceptions that deviate from the normal processing order [27]. They furthermore use similar probabilistic reasoning and estimate how much error their process discovery algorithm would give due to the noise [27].

As we have explained earlier, the noise that we describe in our paper is however in a slightly different league from the noise described in process literature. Hence instead of cleaning the noise, we take a different approach and show what results one can expect by analysing very noisy data with different data mining techniques.

5 Design of Case Analysis

Table 4 shows the main attributes of the data in the three data sets we use. Each set contains of rows of activities that are each represented by a code. Each specific activity, such as "Perform surgery X" has its own unique code in the DOT method. Only when creating a graph are these activities relabelled to remove loops, so table 4 does not include relabelled activities.

An idea of the extent of noise in the data is given by the *Average number of activities per day* and *Average path length* in table 4: large parts of the sequence of activities are uncertain and may well be wrong - although we have no way of knowing. Hence, in order to reduce the *sequence noise* a subset of the data was used for the classification tasks. To create the subset, we removed the activity codes for lab-work and medications from the raw data, because these were known to have no effect on the grouper result of a sub path. The characteristics of the data set for the classification task are shown in table 5. The effect of removing these activities is best shown by the shorter average path length and the fewer average number of activities per day.

We created three data sets from the hospital's cardiology and lung care department data. Each set contained the data from patients with a specific diagnosis. The three diagnoses were pericarditis (an inflammation of the heart), represented by code 320.701; angina pectoris (chest pain), diagnosis code 320.202; and malignant lung cancer, diagnosis code 302.701. We chose these data sets as they were clearly quite different, not only in size, but also in the amount of sequence noise, path length and duration. This made the sets well suited to test the prediction approaches presented in this paper on a broad sample of actual process data. Also, only data from the heart and lung specialties was available to us for this research.

Table 3 shows the columns that make up each dataset, as well as some example content. The number of columns varies between 286 + 2 for the dataset 302.66 to 703+2 for 302.701.

Activity id	*Care path id*	*Sub path id*	*Care activity code*
The unique id of the performed activity	Unique identifier of the care path containing the sub path the activity is part of	The identifier of the sub path the activity is part of	A (non-unique) code describing the activity

(cont.d)	*Date*	*Performing specialty*	*Amount*
	The date on which the activity was registered	The specialty performing the activity	The quantity of the activity, for example used for medication

Table 1: The performed activity table

Sub path id	*Care path id*	*Start date*	*End date*	*Medical specialty*
The unique identifier of the sub path	The care path the sub path is part of	The date on which the sub path was opened	The date on which the sub path was closed	The specialty responsible for this sub path
(cont.d)	*Care type*	*Diagnosis code*	*Care product code*	*Closure reason*
	The care type of the sub path.	A code for the diagnosis that led to this sub path	The care product for the closed sub path	The reason the care path was closed, for example, enough time had expired.

Table 2: The sub path table

Columns:

Sub path	Care product	Activity 0	Activity 1	Activity 2	...	Activity n

Example contents:

1700750	979001104	33229	299999	33231	190205	33229
40176052	99499015	33285	10320			

Table 3: The columns in each dataset and example dataset contents

	320.202	*320.701*	*302.66*
Number of sub paths	9950	274	1985
Number of unique activity codes	315	190	239
Number of unique care products	22	6	11
Largest path length $\|\hat{P}\|$	559	703	286
Average path length $\|\hat{P}\|$	15.99	41.36	15.07
Average number of activities per day	6.79	10.13	2.49
Average duration (in days)	35.81	31.23	126.22

Table 4: Main characteristics of the data sets

	320.202	*320.701*	*302.66*
Number of sub paths	9950	274	1985
Number of unique activity codes	160	70	149
Number of unique care products	22	6	11
Largest path length $\|\hat{P}\|$	152	174	104
Average path length $\|\hat{P}\|$	6.91	11.35	11.39
Average number of activities per day	3.18	3.03	2.15

Table 5: Main characteristics of the classification data set. Lab-activities are excluded from this set.

Using these data sets we performed the prediction tasks in section 3.2. The setup per task is:

Task 1a) *Predict care product* C from P. We wanted to know the prediction accuracy for different lengths of incomplete care path C. So, for path length $1 \ldots 30$, we took a sample of the closed care paths of at least that length for training - the remainder was the test set. We took the first n activities per path P in the training and test sets. The training set of paths of length n in combination with the care product C were then used to train a classifier. The test set of paths was used to assess the performance of the classifier.

Task 1b) *Predict care product cost:* $cost(Grp(\hat{P}))$. This task builds on task 1a: The same procedure was used to determine a care product C, but now we looked up the cost $cost(C)$. The predicted cost for a partial path P was then compared to the cost of the actual care product $C = Grp(\hat{P})$. We measured the difference in predicted cost and actual cost E_{total} as well as the average absolute error $E_{average}$:

$$E_{total} = \frac{|\sum cost(Grp(\hat{P})), \sum cost(Grp(\hat{P}'))|}{\sum cost(Grp(\hat{P}))}$$

$$E_{average} = \frac{|cost(Grp(\hat{P})), cost(Grp(\hat{P}'))|}{|\hat{P}|}$$

Task 2a) *Predict* P'.. Here, we determined the precision and recall of predicting the path P' between Act_{cur} and Act_{end}. So, we created the process graph G from a training sample of the complete paths $\hat{P}$. The remainder of the data set was then used for testing. Next, for activity $Act_n \in \hat{P}, n \in \{1 \dots 15\}$ for each complete path in the test set we predicted P'. We compared P' to the sub path $Act_n \dots Act_{\text{cur}} \in \hat{P}$ and determined the precision and recall of the prediction.
In order to make sense of the results of both algorithms, we introduce two performance metrics: precision and recall. These metrics are based on finding the number of true positives, false positives and false negatives. True positives are the number of activities that occur in both predicted P' and actual P'. False positives are the activities that are predicted but do not occur in the actual path. False negatives are the number of nodes that are not in the predicted P', but are in the actual P'.
Given these three numbers, precision and recall are calculated by:

$$Precision = \frac{tp}{tp + fp} \qquad Recall = \frac{tp}{tp + fn}$$

Task 2b) *Predict* $d_{\hat{P}}$. Given the predicted path P' we created a predicted complete path $\hat{P}' = P \oplus P'$. We looked up the duration for each edge $e \in \hat{P}'$, where duration is the average time difference between $Act_a, Act_b \in e$ for all occurrences of e in the training set. These durations were combined into a duration prediction $d_{\hat{P}'}$ and were compared to the actual time difference between $Act_1, Act_{\text{end}} \in \hat{P}$.

6 Results

6.1 Task 1a: Predicting $Grp(C)$

Table 6 shows the results for the care product predicting task. As expected from the literature review, we found the Random Forest algorithm to be the best performing classifier in terms of accuracy. This table shows the accuracy for care product prediction for path lengths 1 to 30. Table 7 shows the Random Forest results when only taking paths that have at least length 10. Figures 3 and 4 show the accompanying Random Forest accuracy plots for these two tables. The plot in figure 2 shows the number of paths of length n for each of the data sets, which also explains the cut off point of 10 activities: the number of complete paths $\hat{P}$ with more than 10 activities is low.

To put the Random Forest accuracy in perspective, we compare the results to a more naive approach: take a random guess of care product. This would result a chance of predicting the care product of 16 percent maximum, given that the data set with the fewest number of care products has six products. The Random Forest classifier seems to outperform this method. The comparison with

a somewhat less naive approach, taking the most occurring care product is shown in figure 3 and 4. This shows that the Random Forest classifier is always better than a *random classifier* for two of the three datasets. The dashed lines in both plots represent the expected accuracy when always selecting the most occurring care product.

The accuracy of the classifier does not improve with a larger path length $|P|$. This is the result of noise: as the number of activities in sequence increases, the likelihood of that sequence containing noise also increases. This noise has a negative effect on the classifier's performance.

Training set: 50%.

320.202				
	Min	Max	Mean	Median
Random Forest	0.30	0.69	0.47	0.44
CART Tree	0.24	0.49	0.37	0.35
Adaboost.M1	0.24	0.59	0.41	0.40
Naive Bayes	0.21	0.45	0.30	0.28
k Nearest Neighbour	0.20	0.53	0.31	0.28
320.701				
	Min	Max	Mean	Median
Random Forest	0.44	0.71	0.55	0.53
CART Tree	0.00	0.75	0.47	0.50
Adaboost.M1	0.11	0.75	0.52	0.56
Naive Bayes	0.00	0.80	0.45	0.45
k Nearest Neighbour	0.11	0.63	0.42	0.43
302.66				
	Min	Max	Mean	Median
Random Forest	0.28	0.55	0.40	0.38
CART Tree	0.16	0.55	0.35	0.33
Adaboost.M1	0.24	0.56	0.38	0.35
Naive Bayes	0.19	0.46	0.31	0.29
k Nearest Neighbour	0.20	0.45	0.28	0.25

Table 6: Accuracy (fraction correct of total) of prediction of care product for path lengths $|P| = 1 \ldots n$

6.2 Task 1b: Predicting $cost(Grp(C))$

The goal of predicting a care product C is predicting the cost, so the hospital knows what it will receive for provided care. Table 8 shows the results for a test of predicting $cost(C)$ in terms of the total error fraction E_{total}. Figure 5 shows the average absolute error $E_{average}$ in predicting the cost. These errors cancel each other out in some cases, which explains the lower best performance

	Min	Max	Mean	Median
320.202	0.43	0.58	0.49	0.50
320.701	0.36	0.54	0.43	0.40
302.66	0.43	0.48	0.45	0.45

Table 7: Prediction of care product from path length $|P| = 1 \dots 10$

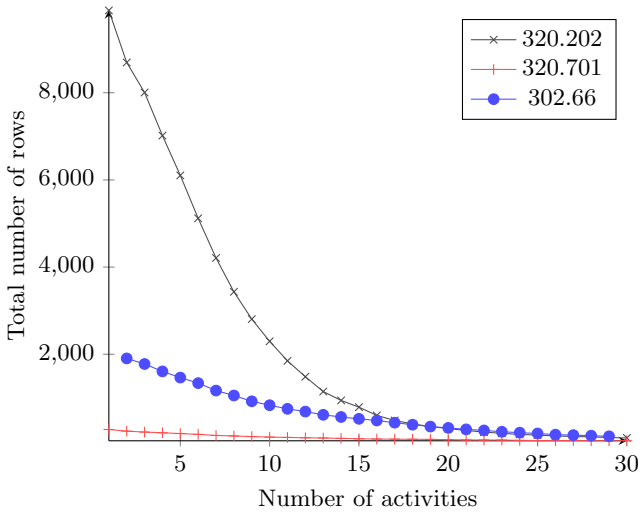


Fig. 2: Number of sub paths of length $|P| \geq 1 \dots n$

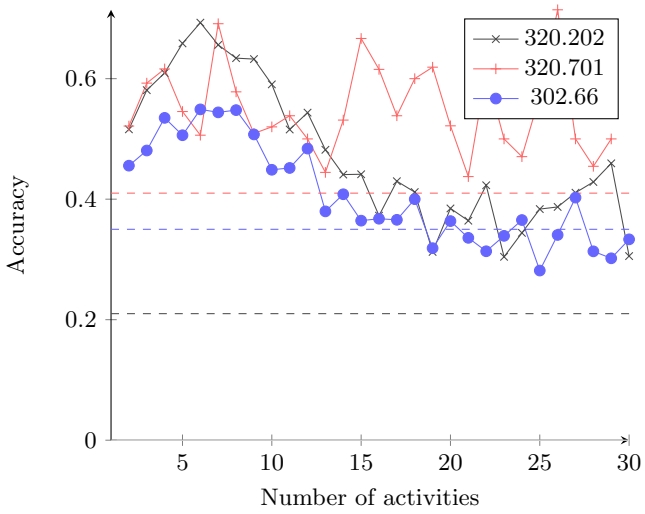


Fig. 3: Random Forest accuracy for path length $|P| = 1 \dots n$. Dashed lines represent the expected accuracy when randomly selecting care product

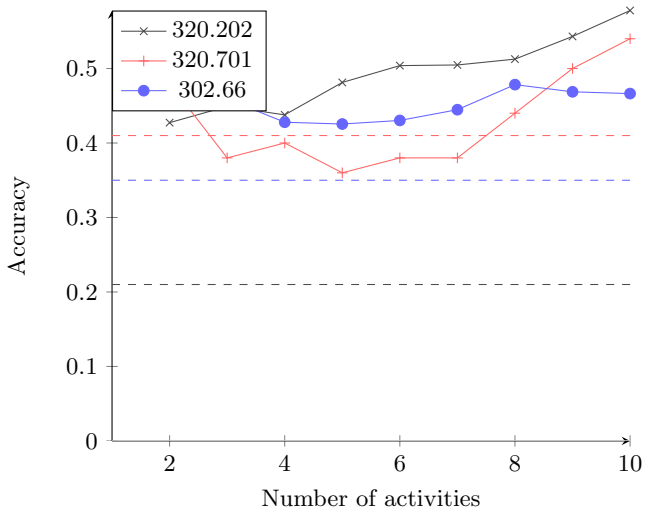


Fig. 4: Random Forest accuracy for path length $|P| = 1 \ldots 10$ and $|\hat{P}| \geq 10$. Dashed lines represent the expected accuracy when randomly selecting care product

results shown in table 8 than in 5. The test shows again that Random Forests provide decent predictions of the care product: Even though it averages around 40 percent correctly predicted care products, the 60 percent it predicts wrong apparently has a similar cost.

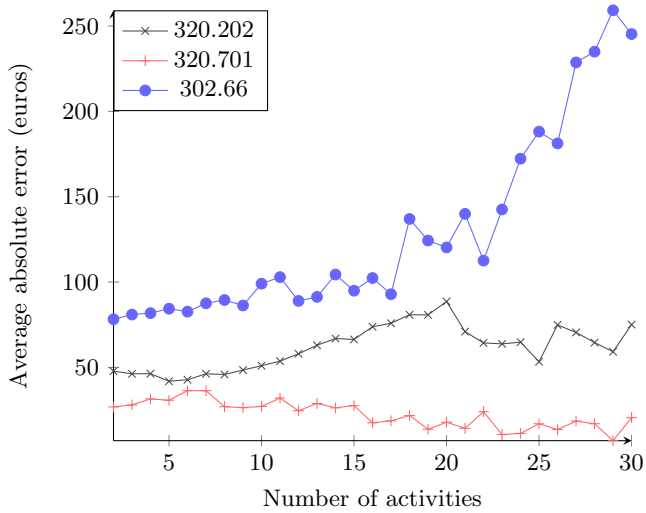


Fig. 5: Random Forest cost error for path length $|P| = 1 \ldots n$

6.3 Task 2: Predicting P' and $d_{\hat{P}}$

Path search algorithms are the tools we have used to predict the path P'. The algorithms we use are variations on Floyd-Warshalls shortest path algorithm, as discussed above. We have experimented with two algorithms, that both take an activity and a care product and attempt to find the activities in between.

The first variation, called Longest Path, returns P' such that the sum of the weight of the edges is maximized. Here, the edge weight is the number of times that the start and end node of that edge occurred consecutively. The second variation, called Most Likely Path, returns P' such that the product of edge weights is maximized. In this scenario, the weight of an edge shows the likeliness of that edge being traversed.

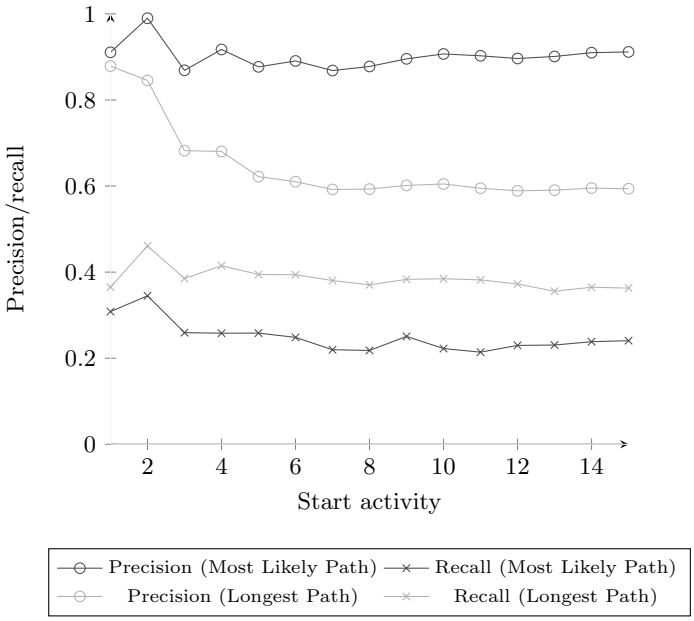


Fig. 6: Precision and recall for dataset 320.202

Figures 6, 7 and 8 show the precision and recall results for the three data sets. In all these figures, the lines marked "o" are the precision results and lines marked "x" are recall results. Darker lines are the Most Likely Path results, lighter lines are the Longest Path result. The plots show that precision of the Longest Path algorithm is almost always lower than Most Likely Path, but on the other hand, its recall is higher. This means that Longest Path predicts more of the actual activities, but also predicts activities that do not actually occur.

Tables 9a and 9b show the effect of the path predictions on the duration prediction for Most Likely Path and Longest Path, respectively. Both algorithms

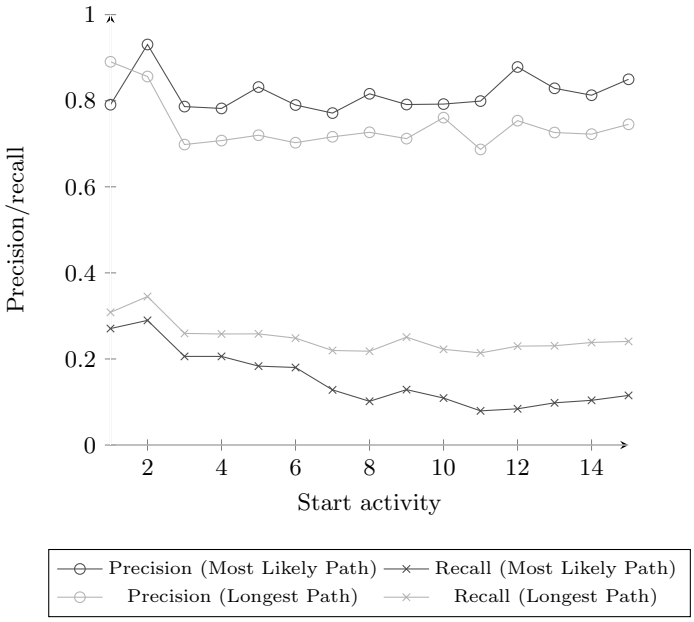


Fig. 7: Precision and recall for dataset 320.701

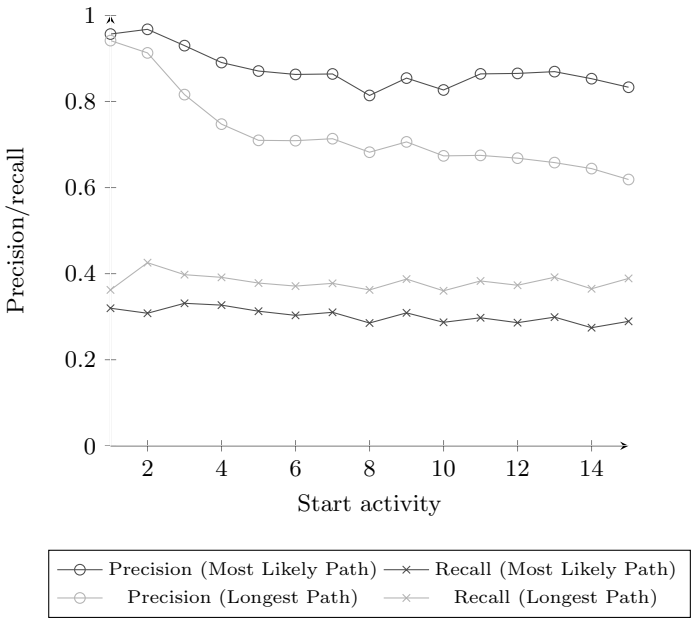


Fig. 8: Precision and recall for dataset 302.66

show lower predictions than the actual durations. Again, the Longest Path shows the effect of having greater recall: the predicted path lengths are longer and closer to the actual.

The average duration prediction error in table 9 seems reasonable at up to around two weeks. This changes when taking the predicted durations into account: they are close to zero for the Most Likely Path. Of course, this problem is related to the lack of recall using MLP. Longest Path is performing a bit better, but still gives short predicted durations. If the predicted path is much shorter than the actual the predicted duration will also be much lower: duration is calculated as the sum of average edge durations in the predicted path.

7 Discussion

We can analyse and discuss the results obtained above in the light of the three different kinds of noise in the data: namely, the *sequence noise*, *duration noise* and *human noise*.

Let us consider the results from each of the goals of this paper. Beginning with (1a), predicting $Grp(C)$. The accuracy of the prediction does not go up when we consider sequences containing a greater number of activities 3, the reason for this is that we assumed a sequence for the activities conducted every day - *sequence noise*. Hence, when we have a greater number of activities on a certain day, we would expect a larger *sequence noise*, and as a result the accuracy of predicting the care product would decrease. For this measurement we used a subset of the actual dataset, the main characteristics of which can be seen in Table 5. Comparing the average number of activities in Table 5, with those in Table 4, we can clearly see that there has been a large decrease in *sequence noise*, especially for data sets *320.202* and *320.701*. Hence, we can say that for this task the noise in the data sets is nearly of the same magnitude. This is the case as the *duration noise* and the *human noise* (that does not influence the *sequence noise*) do not affect the classification task. So, it comes as no surprise that the results of the classification (taking the mean Random Forest values) shown in Table 6, are nearly the same for the three sets of data. Added to this is the fact that if we use the best performing Random Forest classifier, it helps in taking care of some of the error in the data. Furthermore, if we consider only subpaths whose path lengths are at least 10, we see from Fig.4 that the accuracy for predicting the care product has the same trend for all the three data sets. However, we do notice from Figure 3 (and even from Figure 4) that the accuracy of predicting the path for data set *302.66* drops after about 15 activities and gets even worse than randomly selecting the care product. One could attribute this to a certain amount of *human noise* present in the data set *302.66*, which could be greater than the *human noise* present in the other two data sets.

Task (1b), predicting $cost(Grp(C))$, again we see the Random Forest algorithm performing the best. Though the mean values of E_{total} is similar for *320.202* and *320.701*, it is higher for the data set *302.66*. This maybe explained partly due to the greater amount of *human noise* present in the data set *302.66*,

Training set: 50%.

320.202				
	Best	Worst	Mean	Median
Random Forest	0.00	0.15	0.07	0.06
CART Tree	0.00	0.21	0.07	0.07
Adaboost.M1	0.00	0.20	0.08	0.07
Naive Bayes	0.16	0.62	0.33	0.25
k Nearest Neighbour	0.00	0.21	0.06	0.03
320.701				
	Best	Worst	Mean	Median
Random Forest	0.00	0.18	0.06	0.04
CART Tree	0.00	0.23	0.07	0.05
Adaboost.M1	0.01	0.20	0.07	0.05
Naive Bayes	0.01	0.43	0.15	0.11
k Nearest Neighbour	0.01	0.16	0.07	0.07
302.66				
	Best	Worst	Mean	Median
Random Forest	0.00	0.51	0.17	0.14
CART Tree	0.00	0.47	0.15	0.14
Adaboost.M1	0.00	0.51	0.18	0.18
Naive Bayes	0.01	0.72	0.39	0.52
k Nearest Neighbour	0.00	0.24	0.11	0.10

Table 8: E_{total} in prediction of $cost(C)$ for path lengths $|P| = 1 \ldots 30$

	202	701	66
Average predicted duration	0.4	0.7	0.7
Average prediction error(days)	11.7	3.6	14.3
Average % of prediction error	33%	11.5%	11.3%

(a) Most Likely Path

	202	701	66
Average predicted duration	3.7	0.5	9.5
Average prediction error	8.4	3.7	5.5
Average % of prediction error	24%	11.8%	4.1%

(b) Longest Path

Table 9: Average predicted duration and prediction error, in days

which also explains the trend of E_{total} in Figure 5, where similar to Figure 3 the average absolute error of *302.66* increases as compared to the other two data sets.

Task (2a), predicting P'. For tasks (2a) and (2b) we need to keep in mind that we use the actual raw data set (Table 4) and not a subset(Table 5) as in the case of Tasks (1a) and (1b), the reason being that the laboratory tests do affect the result of the tasks (2a) and (2b). From Figure 6, Figure 7 and Figure 8 we see that recall is always lower than the precision. The reason behind this is that the methods we used to predict the path, returned paths that were shorter than the actual paths the data had and hence the recall is low. While on the other hand, the precision is higher as the predicted activities are mostly in the actual path. We also see a consistently high trend of precision for the Most Likely Path algorithm for all three data sets, which, as expected, has a decreasing trend for *302.66*. This could be due to the higher presence of *human noise* (as we noted earlier). However, when we observe the trend for the Recall values (in Figure 6, Figure 7 and Figure 8) we see that the performance is in the order: Recall(*302.66*) > Recall(*320.202*) > Recall(*320.701*). We think this is an indicator of the *sequence noise* present in the data sets which also follows the same trend, going by the number of average activities per day in Table 4. We think that *sequence noise* affects the recall more than the precision as the greater the *sequence noise* the greater is the number of actual activities not in a particular path, i.e. the false negatives are higher.

Task (2b), predicting $d_{\hat{P}}$. From Table 9a and Table 9a, we see that the Longest Path gives less error than the Most Likely Path algorithm. This could be as we explained earlier a consequence of the Longest Path algorithm having a higher recall (Figure 6, Figure 7 and Figure 8) and hence having fewer false negatives and being closer to the original path.We also notice that the prediction error is higher for *320.202* compared to *302.66* and *320.701*. This result seemingly contradicts the trend of *sequence noise* between the data sets *320.202* and *302.66*, as discussed in the earlier paragraph. We can however reason this result, by arguing that *sequence noise* does not affect the duration prediction as much as *duration noise* does. For example, in our Hospital data two identical subpaths (with the same activities and in the same order) can have entirely different durations, because of the different timestamps and the different intervals between activities (that can vary from a couple of minutes to weeks. A practical example of such a case is when an young and a relatively old patient is admitted in a Hospital for a similar fracture or injury. Both the individuals would get a similar treatment(Care product - and would have the same subpaths, however, the older patient could take a much longer time to recover. Hence, we can reason the data in Table 9a and Table 9a by concluding that the data set *320.202* has greater *duration noise* than the data sets *302.66* and *320.701*.

An issue prevalent in the data sets and which could be one of the major causes of the *duration noise* and *human noise*, is the fact that all patients, irrespective of their particular demographic background, are subjected to the same treatment procedures. This would cause a lot of *human noise*, apart from

the *duration noise* (as explained in the earlier paragraph), as we can expect that all patients would not respond to the same treatments in an identical manner. Hence, the likelihood of faulty treatment procedures (*human noise*) increases.

8 Conclusions

In our case study we have shown how one can combine data and process mining techniques for forecasting cash flow in Dutch hospitals. The techniques and processes we have demonstrated, we think can be used to analyse the cash flow for other hospitals with a few modifications.

For patients still undergoing treatment, we managed to, given a diagnosis and past activities, predict the care product with an accuracy of a little less than 50% (for all the 3 sets of data), which was still (on an average) better than randomly predicting the care product. On the other hand, for predicting the associated cost of the Care, we obtained an error of of less than 10% for 2 data sets and 17% for one. These results were obtained with a Random Forest classifier which was shown to perform the best. This shows that our method of predicting is a viable solution to the cash flow problem we raised in the introduction.

We furthermore managed to, given a diagnosis, the most recent activity, and the (predicted) care product, predict the remaining activities using two algorithms (Longest Path and Most Likely Path). We achieved a precision of about 80% for the Most Likely Path and about 60-70% for the Longest Path algorithm and a recall of 35-40% (for the three data sets). We algorithms predicted the associated duration with an error of about 30% for 202 and around 11% for the data sets 701 and 66.

The case study is of particular interest because of the causes for the modest to weak prediction results for the process mining tasks. In our opinion, three process mining-specific types of noise significantly affected our results: sequence noise, human noise, and time noise. In this paper, we thoroughly discuss the causes for these types of noise and their effects.

From the predicted cost and duration, it is possible to derive a prediction (albeit with some error) for the future cash flow for all patients currently undergoing treatment. This we think is a major contribution to research and practice, as given the noise in the data, we were still able to make reasonable predictions which could cheaper and more practical than asking a few expert practitioners.

Future work can include attempts at removing or decreasing the different kinds of noise in the data and then carrying out the prediction analysis to compare the results.

References

[1] Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Morgan Kaufmann Pub (2011)

[2] Quinlan, J.: Improved use of continuous attributes. Journal of Artificial Intelligence Research **4** (1996)
[3] Alpaydın, E.: Introduction to Machine Learning. The MIT Press (2004)
[4] Breiman, L.: Bagging predictors. Machine Learning **24** (1996) 123–140
[5] Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Machine Learning: Proceedings of the Thirteenth International Conference. (1996) 148–156
[6] Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32
[7] Curram, S., Mingers, J.: Neural networks, decision tree induction and discriminant analysis: an empirical comparison. Journal of the Operational Research Society (1994) 440–450
[8] Alfaro, E., García, N., Gámez, M., Elizondo, D.: Bankruptcy forecasting: An empirical comparison of adaboost and neural networks. Decision Support Systems **45**(1) (2008) 110–122
[9] Banfield, R., Hall, L., Bowyer, K., Kegelmeyer, W.: A comparison of decision tree ensemble creation techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(1) (2007) 173–180
[10] Gislason, P., Benediktsson, J., Sveinsson, J.: Random forests for land cover classification. Pattern Recognition Letters **27**(4) (2006) 294–300
[11] Baase, S., Gelder, A.: Computer algorithms: introduction to design and analysis. Addison-Wesley (2000)
[12] van der Aalst, W., van Dongen, B., Herbst, J., Maruster, L., Schimm, G., Weijters, A.: Workflow mining: a survey of issues and approaches. Data & Knowledge Engineering **47**(2) (2003) 237–267
[13] van der Aalst, W., Weijters, A., Maruster, L.: Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering **16**(9) (2004) 1128–1143
[14] van der Aalst, W., Günther, C.: Finding structure in unstructured processes: The case for process mining. In: Application of Concurrency to System Design, 2007. ACSD 2007. Seventh International Conference on, IEEE (2007) 3–12
[15] van der Aalst, W., Reijers, H., Weijters, A., van Dongen, B., Alves de Medeiros, A., Song, M., Verbeek, H.: Business process mining: An industrial application. Information Systems **32** (2007) 713–732
[16] van der Aalst, W., Rubin, V., Verbeek, H., van Dongen, B., Kindler, E., Günther, C.: Process mining: a two-step approach to balance between underfitting and overfitting. Software and Systems Modeling **9**(1) (2010) 87–111
[17] van der Aalst, W., Weijters, A.: Process mining. Process-Aware Information Systems (2011) 235–255
[18] Mans, R., Schonenberg, M., Song, M., van der Aalst, W., Bakker, P.: Application of process mining in healthcare–a case study in a dutch hospital. Biomedical Engineering Systems and Technologies (2009) 425–438
[19] Gunther, C., Rinderle-Ma, S., Reichert, M., Van Der Aalst, W.: Using process mining to learn from process changes in evolutionary systems. In-

ternational Journal of Business Process Integration and Management **3**(1) (2008) 61–78
[20] Weijters, A., van der Aalst, W.: Rediscovering workflow models from event-based data using little thumb. Integrated Computer Aided Engineering **10**(2) (2003) 151–162
[21] Herbst, J., Karagiannis, D.: Workflow mining with InWoLve. Computers in Industry **53** (2004) 245–264
[22] van Dongen, B., de Medeiros, A., Verbeek, H., Weijters, A., van der Aalst, W.: The prom framework: A new era in process mining tool support. Applications and Theory of Petri Nets 2005 (2005) 444–454
[23] Kiepuszewski, B., ter Hofstede, A., van der Aalst, W.: Fundamentals of control flow in workflows. Acta Informatica **39** (2003) 143–209
[24] Datta, A.: Automating the discovery of as-is business process models: Probabilistic and algorithmic approaches. Information Systems Research **9**(3) (1998) 275–301
[25] Weijters, A., van der Aalst, W.: Rediscovering workflow models from event-based data using little thumb. Integrated Computer Aided Engineering **10**(2) (2003) 151–162
[26] Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. Advances in Database TechnologyEDBT'98 (1998) 467–483
[27] Hwang, S., Yang, W.: On the discovery of process models from their instances. Decision Support Systems **34**(1) (2002) 41–57