RWTH AACHEN UNIVERSITY

Business Process Intelligence

Report Assignment Part 1

Business Process Intelligence
course

Summer semester 2022
semester

prof. dr. ir. Wil van der Aalst
lecturer

Group 77: Niklas Föcking (423932), Filiz Günal (431174), David Wenderdel (423885)
students

May 25, 2022
submission date

Q1 Extracting a situation table in Celonis

As a first step, we would like to have a situation table containing the case and activity tables merged. Luckily, this procedure was already explained in one of the instructions. Hence, we do not describe the procedure again within this report. However, the structure as well as some randomly selected rows of the resulting table are shown in the appendix in Table 4.

Q2 Decision Tree

(a) We would like to predict the outcome of applications and therefore train a decision tree since it is a fully explainable model. The process of training the decision tree in RapidMiner is shown in Figure 1.

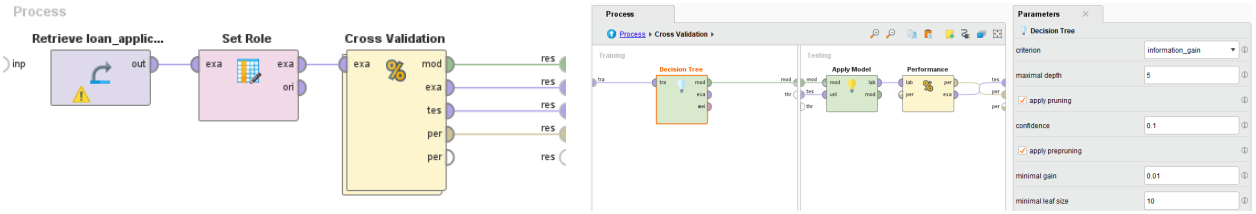


Figure 1: Screenshot of the process for training a Decision Tree in RapidMiner. The left sub-figure shows the overall process while the right sub-figure depicts the sub-process of cross validation.

We use the extracted situation table as our dataset. As we are only interested in predictor variables, we set roles for the variables *Case ID* and *Results*. We do not want to use *Case ID* as a predictor variable because it is an identifier and would overfit our model. So, we set role "id" for *Case ID*. Similarly, since we would like to predict *Results*, the role "label" is assigned to it. All other variables are used to train the decision tree. The decision tree was asked to be trained with the information gain criterion for splitting. Also, it has to have a depth of at least five, with a leaf size of at least ten. The last condition for the decision tree is to have at least 85% average accuracy. Hence, the Cross Validation operator with four folds is used to evaluate the performance of our model in terms of its accuracy. Inside the Cross Validation operator, we train the decision tree by setting the criterion to information gain, maximal depth to five, and minimal leaf size to ten. As a result of this process, we discover a decision tree as shown in Figure 2.

The trained decision tree contains seven leaves, three of which preserve high proportions of our dataset (about 90% in total) and four of which account for only minor parts.

The decision tree clearly demonstrates that the most important factor for a successful completion of an application is *Last Credit Score*. All applications with *Last Credit Score* higher than 297 are successfully completed. This accounts for roughly half of our cases (44%, 2,183 applications).

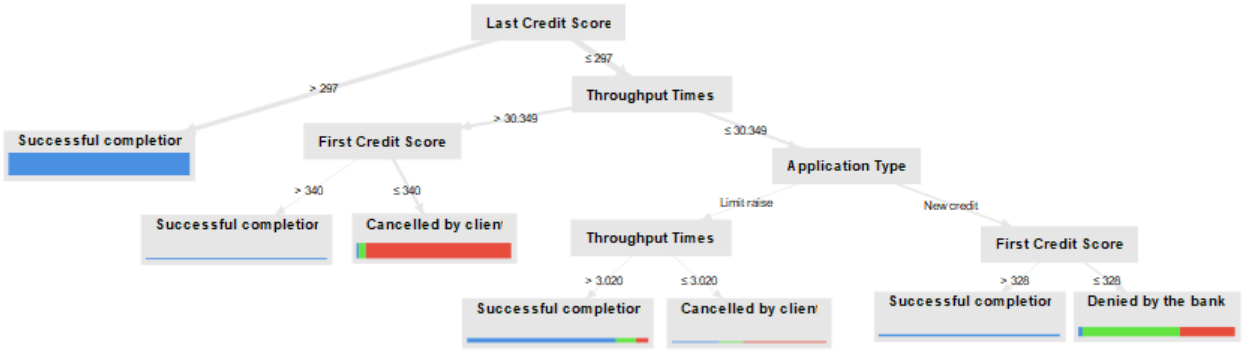
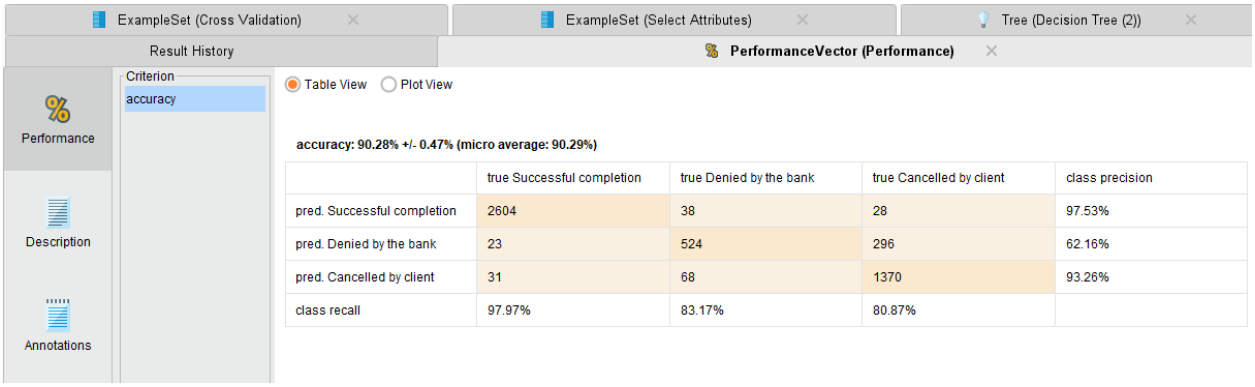


Figure 2: Trained decision tree in RapidMiner. The color of the box within each leaf indicates the result of the process (blue: successful completion, red: cancelled by client, green: denied by bank), while the height of the box indicates the proportion of the dataset.

Another relatively unambiguous leaf can be "reached" by having a last credit score of less than or equal to 297, a throughput time of more than about 30 days and a first credit score of less than or equal to 340. Cases having these characteristics are most likely to get cancelled by the client and make up 1,456 applications of our dataset (about 30%).



	true Successful completion	true Denied by the bank	true Cancelled by client	class precision
pred. Successful completion	2604	38	28	97.53%
pred. Denied by the bank	23	524	296	62.16%
pred. Cancelled by client	31	68	1370	93.26%
class recall	97.97%	83.17%	80.87%	

Figure 3: Screenshot of the confusion matrix in RapidMiner showing information about precision, recall and accuracy of the trained decision tree.

The cross validation operator allows us to evaluate our decision tree model as shown in Figure 3. We can conclude that the decision tree has an average accuracy of about 90%. So, the requirement of having at least 85% of average accuracy is fulfilled. The confusion matrix shows that the class "Successful completion" achieves the highest precision and recall (approx. 98% both). This means that not only the proportion of positive identifications that were actually correct is very high, but also the proportion of actual positive identifications that were correctly identified. While the class "Cancelled by client" also produces good results (recall of about 81%, precision of 93%), the class "Denied by the bank" is very inaccurate, having a precision of only 62%. However, recall is still at 83%. That means that the decision tree correctly classified a given denied application correctly in more than 4 out

of 5 cases correctly but at the same time predicted a denial way more often than it actually happened.

(b) As mentioned in exercise part a), the most decisive criterion is the last credit score. Even though this seems to be the best advice for a successful completion, we cannot derive any helpful advice for the bank manager because of two reasons: Firstly, the bank does not know the **last** credit score at the beginning of the process and secondly, the last credit score depends on the customer and is not "controllable" by the bank.

Another decisive criterion is the throughput time. For that, the bank might be able to accelerate the process in order to increase the number of successful completions. Since we do not yet have precise knowledge of the data set, we can only speculate here. In case that the long throughput times are caused by the bank, one of the possible improvements that the bank could make is preparing the first offer in a shorter period of time, especially for the clients who have a first credit score less than or equal to 340. Those cases are very likely to get cancelled. Alternatively – if the long throughput times are caused by the customer – making an effort to get faster responses from the client, e.g. by sending reminders, may increase the number of successful completions as it decreases the throughput time.

Q3 Clustering

(a) In this clustering step, we would like to group the applications concerning the variables *Throughput Time*, *Number Of Offers* and *Requested Amount*. To perform the clustering, we create a RapidMiner process as shown in Figure 4.

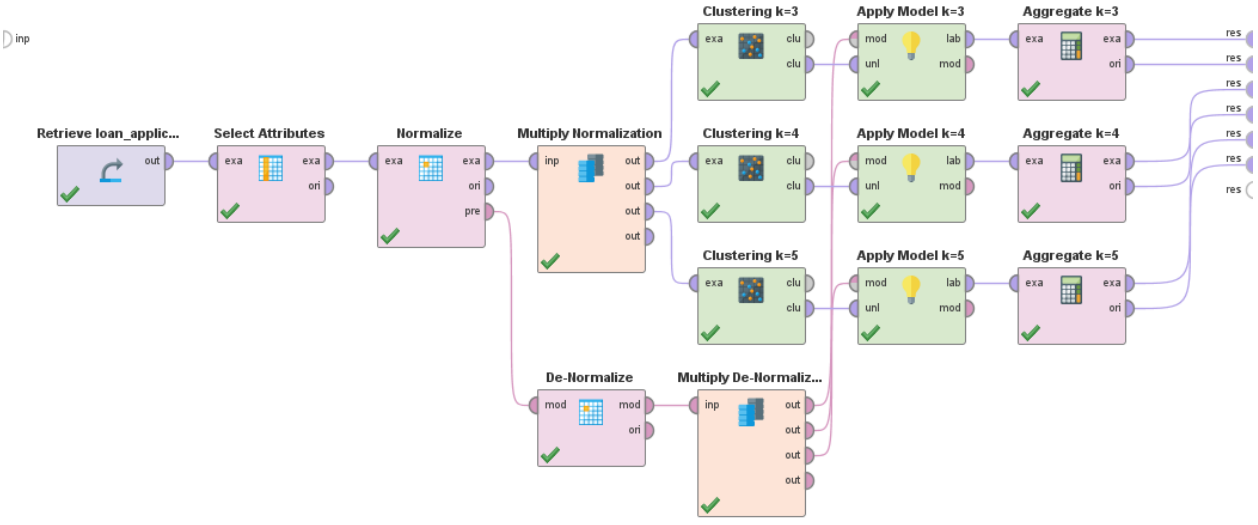
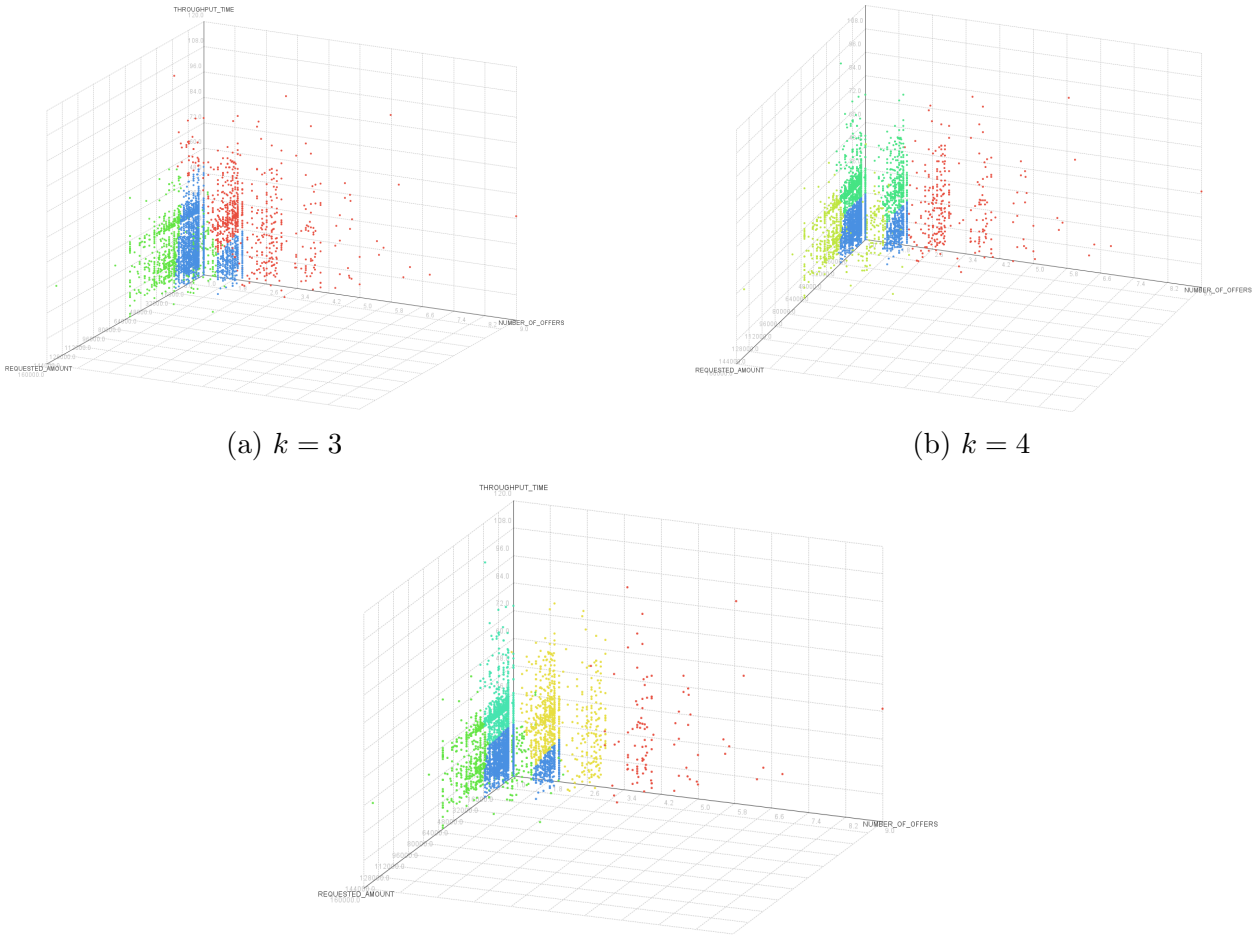


Figure 4: Screenshot of the process for clustering by k-means with three different ks in RapidMiner.

After loading our data, we remove those attributes that are not of interest to us by selecting the three previously mentioned attributes. Furthermore, we normalize our data set to im-

prove the clustering results. We are going to cluster our data using the k-means algorithm which has the disadvantage that we have to specify a k beforehand – even though we do not know yet, how many clusters we expect. Hence, we decide to cluster our data for three different k-values. Therefore, we use the Multiply operator in RapidMiner which allows us to use our data multiple times. Then, we insert one Clustering operator for each of our k-values (3, 4 and 5). For each clustering operator, we set the number of runs to 10, because k-means clustering is a non-deterministic algorithm, i.e. running the algorithm several times on the same data might return different results. By running the clustering 10 times, we want to make sure to find a good clustering. Finally, we de-normalize our data again by applying the De-Normalizer to the clustered data set and aggregate the data in order to get the size and centroids for each cluster. Once we designed our process in RapidMiner, we execute it and get the resulting plots shown in Figure 5.



(a) $k = 3$

(b) $k = 4$

(c) $k = 5$

Figure 5: Visualization for results of k-means clustering with $k = 3$, $k = 4$ and $k = 5$.

The scatter plot for $k = 3$ (Figure 5a) and the corresponding statistics in Table 1 indicate that the blue cluster is the closest to the origin, i.e. it has the smallest number of offers (approximately 1), the smallest requested amount (around 10,000) and the shortest throughput time (less than 19 days). It also is by far the largest cluster, containing more than two-thirds

of all applications. In contrast, there is the green cluster, which contains those applications that have more or less a similar number of offers, but a much higher requested amount (40,000) and also a longer throughput time. As a contrast in the other direction, namely with a higher number of offers (2.5) but approximately the same requested amount like the blue cluster, we have the red cluster which stands out due to high throughput times (more than 35 days on average).

Cluster	Size	Number Of Offers	Requested Amount	Throughput Time
• cluster_0	3340	1.126	10,394.030	18.607
• cluster_1	843	1.190	40,707.066	23.371
• cluster_2	799	2.512	15,143.607	35.185

Table 1: Statistics for the clusters found by k-means algorithm for $k = 3$ with the resulting cluster size and centroids. The colors refer to the visualization of the corresponding clusters shown in Figure 5.

The clusters for $k = 4$ are shown in Figure 5b as well as in Table 2. Again, the blue cluster is closest to the origin with the smallest number of offers (around 1), smallest requested amount (11,300) and shortest throughput time (12 days). However, this time it contains less data points than before (approximately 50%). The red cluster separates the data set almost perfectly into two subsets (when only looking at the number of offers) where the red one has three or more offers and the remaining one has less than three offers. The previously green cluster is now colored yellowish and a new green cluster is formed containing data points from all previous clusters. This new green cluster contains almost only applications where the through put time is higher than 24 days, resulting in the green cluster having the longest average throughput time (more than 34 days).

Cluster	Size	Number Of Offers	Requested Amount	Throughput Time
• cluster_0	2344	1.185	11,300.459	12.201
• cluster_1	296	3.497	17,404.375	30.679
• cluster_2	1703	1.267	12,162.848	34.008
• cluster_3	639	1.257	45,036.865	22.485

Table 2: Statistics for the clusters found by k-means algorithm for $k = 4$ with the resulting cluster size and centroids. The colors refer to the visualization of the corresponding clusters shown in Figure 5.

Lastly, we have our clusters for $k = 5$ as depicted in Figure 5c and Table 3. The blue cluster changed only slightly in comparison to before, still having the shortest throughput time (less than 12 days) and the smallest requested amount (11,000). Furthermore, it still contains most of the data points (44%). The previously red cluster, which had almost perfectly separated the data set by the number of offers, shifted a bit along its axis, i.e., it now divides the data set by the number of offers with a threshold of 3 offers. This has reduced the number of applications within this cluster to one third (100 data points). The dropped data points created a new cluster together with some data points of the previously green cluster resulting in a new yellow cluster. All applications within this cluster have two or three offers

and a requested amount of approximately 50,000. Another cluster that clearly differentiates data points is the turquoise one which contains only applications with 1 offer, a requested amount between 0 and 20,000 and a throughput time of 21 days. Lastly, the green cluster is the one containing most of the data points with higher requested amounts (minimum higher than 30,000; average higher than 45,000). The yellow, turquoise and red cluster have similar throughput times (around 32 days).

Cluster	Size	Number Of Offers	Requested Amount	Throughput Time
● cluster_0	613	1.228	45,620.974	21.674
● cluster_1	797	2.240	14,972.823	31.985
● cluster_2	1306	1	12,025.129	32.826
● cluster_3	2165	1.145	11,012.983	11.610
● cluster_4	101	4.455	16,682.525	31.475

Table 3: Statistics for the clusters found by k-means algorithm for $k = 5$ with the resulting cluster size and centroids. The colors refer to the visualization of the corresponding clusters shown in Figure 5.

Taking into consideration the above mentioned aspects, the clustering with $k = 4$ seems to be the most suitable one. In comparison to the clusters from $k = 3$, there is at least one cluster (red) that clearly contains the data points where the number of offers is greater than two. Clustering with $k = 5$ does not bring any advantages but rather nullifies the previously clear(er) clusters. However, none of the clusters is really clear and meets the "expectations".

(b) After analyzing the various clustering results, we did not get many insights – at least not many meaningful ones. We know that almost 50% of all applications are in one cluster (blue cluster) which is also the "minimal" one regarding all three dimensions, i.e. the mean requested amount is about 11,000, the mean throughput time is 12 days and only one offer is necessary. A second group (green cluster) has similar mean values like the first group, but differs in the average throughput time, which is almost three times longer on average. Then there is a minority (red cluster) which stands out because of its significantly higher number of offerings (i.e. 3.5). Another small group (approximately 13%) can be characterized by its requested amount which is about four times as much as for the first mentioned group.

Q4 Frequent Itemsets and Association Rules

(a) Figure 6 depicts the RapidMiner process that finds association rules with a minimum support count of 250 and a minimum confidence of 0.8. First, we remove not interesting attributes based on the exercise description. In the next steps, we discretize the numerical attributes based on the given rules in the exercise. The upper limits in "Discretize"-blocks in RapidMiner are inclusive. In contrast to that, the exercise uses exclusive upper bounds ($<$). However, as said in the final Q&A session, we can use exactly the constraints given in the exercise as upper limits in the "Discretize"-blocks. After discretizing the data, we find the frequent itemsets that have at least a support count of 250. Therefore, we use "frequency"

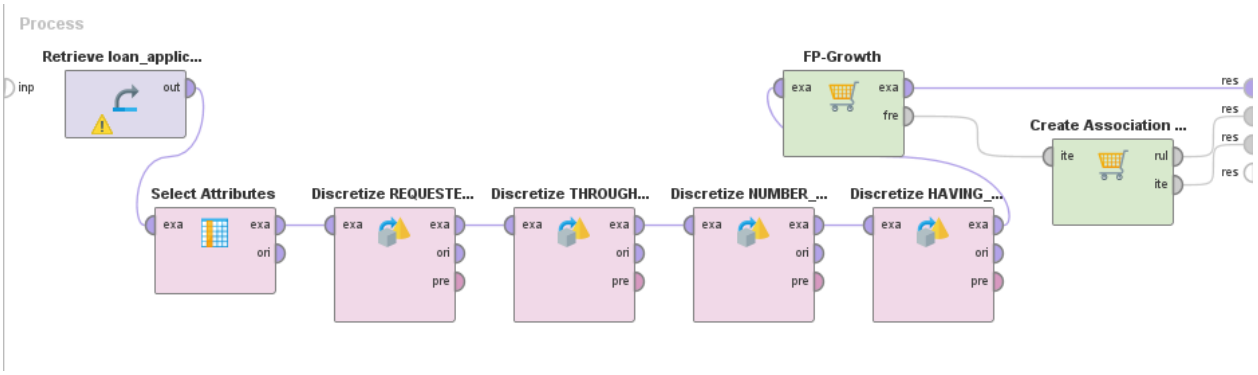


Figure 6: Screenshot of the process for finding association rules in RapidMiner.

as the "min requirement" in the FP-growth block of RapidMiner. As a frequency, we use the value 250 as given in the exercise description. Finally, we search for the association rules with a minimum confidence of 0.8. We find 158 different association rules; note that one has to decrease the slider in the bottom left of RapidMiner to see all association rules. The ten association rules with the highest confidence are shown in Figure 7.

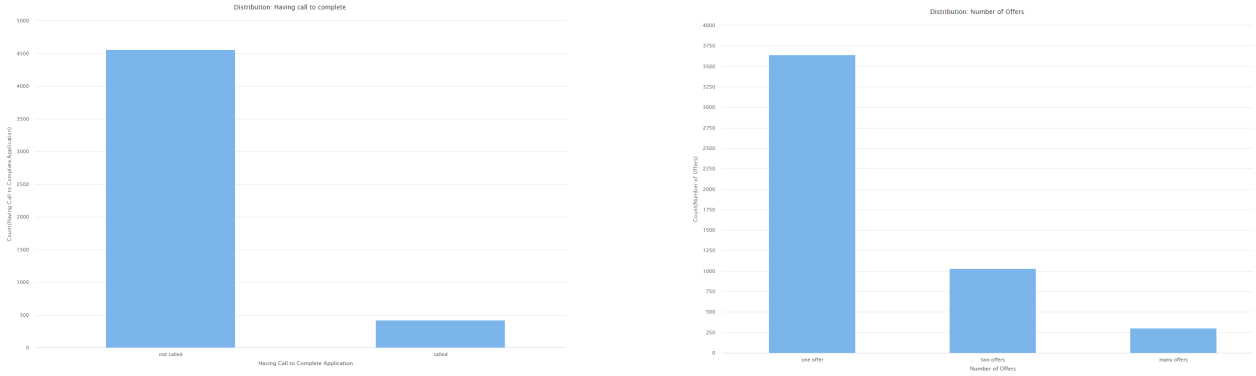
No.	Premises	Conclusion	Support	Confiden... ↓	LaPlace	Gain	p-s	Lift	Convicti...
158	one offer, med duration, Cancelled by client, Car	not called	0.063	0.991	0.999	-0.064	0.005	1.082	8.979
157	one offer, med duration, med request, Cancelled b...	not called	0.071	0.983	0.999	-0.074	0.005	1.074	5.096
156	med duration, Cancelled by client, Home improve...	not called	0.058	0.983	0.999	-0.060	0.004	1.074	5.015
155	one offer, med duration, low request, Cancelled by...	not called	0.079	0.983	0.999	-0.082	0.005	1.074	4.877
154	med duration, med request, Cancelled by client	not called	0.089	0.982	0.999	-0.093	0.006	1.073	4.796
153	med duration, Cancelled by client, Car	not called	0.078	0.982	0.999	-0.081	0.005	1.073	4.780
152	one offer, med duration, Cancelled by client	not called	0.200	0.981	0.997	-0.208	0.013	1.072	4.529
151	med duration, Cancelled by client	not called	0.249	0.976	0.995	-0.261	0.015	1.066	3.470
150	med duration, Cancelled by client, high request	not called	0.063	0.972	0.998	-0.067	0.004	1.062	3.040
149	med duration, low request, Cancelled by client	not called	0.096	0.972	0.997	-0.102	0.006	1.062	2.989

Figure 7: The ten association rules with the highest confidence

(b) The association rule with the highest confidence is the following:

$$\{\text{one offer, med duration, Cancelled by client, Car}\} \rightarrow \{\text{not called}\}.$$

It has a confidence of 0.991, a support of 0.063 and a lift of 1.082. The support of 0.063 indicates that there are 314 cases ($0.063 \cdot 4982 \approx 314$) in the dataset that have one offer, a medium duration, a car as the loan goal, no call of bank, and that were cancelled by the client. The confidence value of 0.987 indicates that nearly every time when we have a application with one offer, a medium duration, a car as the loan goal, which was cancelled by the client, we also did not make a call. So, this seems to show that the given rule is a very strong one. However, the lift value is very close to one. Hence, the premises and the conclusion of the rule are independent. Figure 8a illustrates a reason for the independence: in a large majority of cases, the bank does not make a call. Because of the independence, the rule does not provide many insights.



(a) Having call to complete (b) Number of Offers

Figure 8: Histograms for different features

The association rule with the lowest confidence is the following:

$$\{\text{med duration, Cancelled by client}\} \rightarrow \{\text{one offer}\}.$$

It has a confidence of 0.800, a support of 0.204 and a lift of 1.094. So, approximately 20 percent of the cases in the dataset have a medium duration, one offer of the bank, and were cancelled by the client. Obviously, the lift is in the interval $[0.8, 1]$ because we used a threshold of 0.8. Hence, if a case has a medium duration and was cancelled by the client, in 80 percent of the cases, there was exactly one offer. The lift is very close to zero, indicating that the premises and the conclusion are independent. The reason is that the number of offers is equal to one in about three quarters of the cases (see Figure 8b). In conclusion, the found association rule does not lead to further insights. However, one interesting information for the bank's manager can be derived from the high support of the rule: there are a lot of cases that have a medium duration, one offer, and were cancelled by the client.

Q5 Process Analysis

(a) First, we would like to get an overview of the process. Therefore, we create a new sheet in Celonis using the "Process Overview" template.

We find out that we have 4,982 applications (cases) in our event log covering a time period from January 2016 to February 2017.

In general, the dataset contains 1,559 different variants. About 6% of the cases (320 applications) flow via the happy i.e. the most frequent path which looks like the following: *A_Create Application* → *A_Submitted* → *A_Concept* → *A_Accepted* → *O_Create Offer* → *O_Created* → *A_Complete* → *O_Sent (mail and online)* → *W_Complete application* → *A_Cancelled* → *O_Cancelled*. Furthermore, the six most common variants cover about a quarter of cases in the given dataset. However, there exist approximately 1200 variants that are related to a single case.

If one looks at the distribution of cases per day or events per day, one recognizes an increase in cases respective days throughout the year reaching the peak in September. After that peak, the number of cases respective events seems to decrease again. Without having more information on this topic, there might be a seasonal impact. A brief research shows that there **may** be seasonal behavior in lending [1]. The small valley in the months of April and May could result from the high number of public holidays in those months (e.g. long weekends due to Easter and Christian holidays in Germany). We also plotted the frequency of loan goals against time, but could not find any particular patterns.

While the average throughput time is 21 days, we see a shift in the development of the throughput time: Until December 2016, the average is around 20 days. Between December 2017 and January 2017, this average rises to 27 days and even to 48 days in February 2017. Besides we realize that the throughput time of our happy path is around 9 days (30 days in total) longer than the average throughput time.

When looking at the distribution of the throughput times, we realize that there is a peek of cases running about 30 days. Hence, we visualize the data in RapidMiner and investigate the distribution depending on the result of the application (see Figure 9). It is easily recognizable that most of the cases lasting longer than 30 days are cancelled by the client. Since the result "Cancelled by client" is described as an "activity [that] describes a situation where the applicant does not get back to the bank after the bank has sent out an offer", we assume that these applications are closed automatically. Since most of the cases are not closed exactly after 30 days, but mainly between 30 and 35 days, we furthermore assume that this process is not entirely automated but semi-automated, i.e. maybe the back employee gets a reminder to close the application.

Another observation we make is that the data contain an increasing number of cases respectively events per day at the beginning and a decreasing number of cases respectively events per day towards the end of the time horizon. For the beginning of the time horizon, our assumption is that the recording of events started in the beginning of 2016. So, there are

[1] `https://fred.stlouisfed.org/series/WSB`

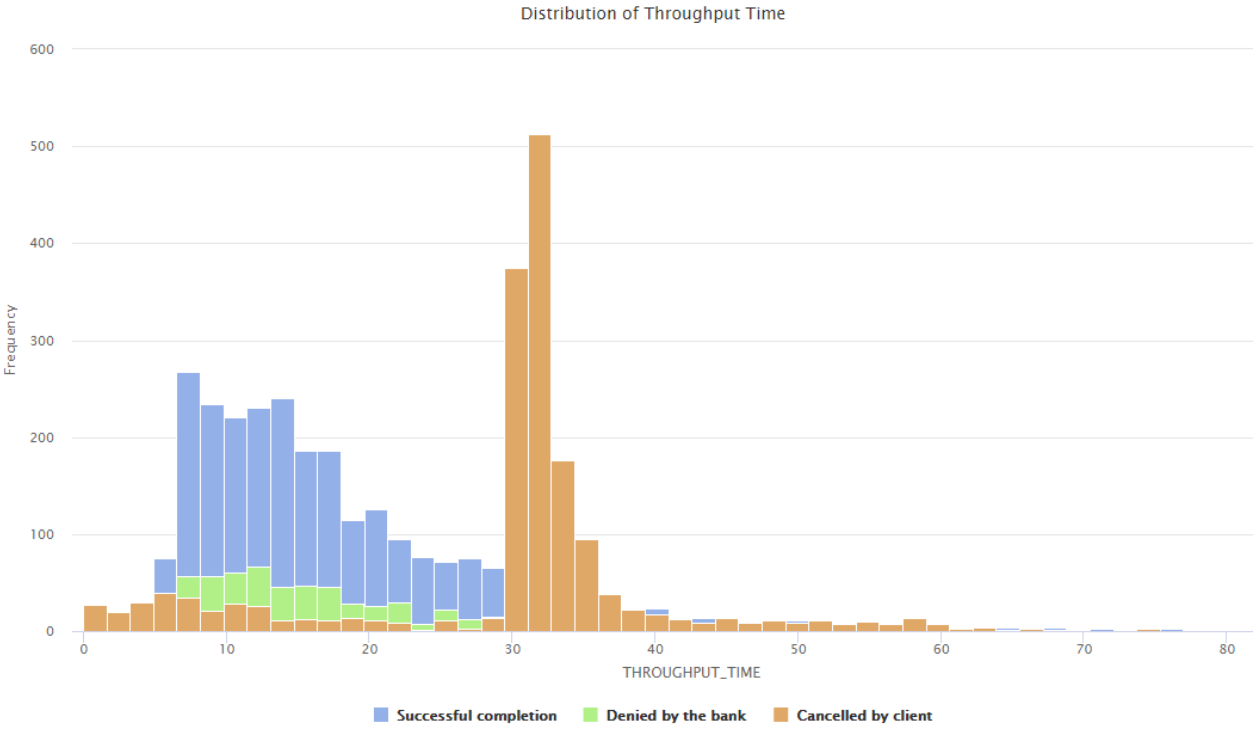


Figure 9: Distribution of throughput times shown in RapidMiner. The color indicates the result of the application. Note that the x-axis is cut off after 80 days.

no running cases from the previous year, which leads to a lower amount of running cases compared to time periods later in the year 2016. For the end of the time horizon, we analyze the timestamp of the first activity of each case. We observe that the last case started on December 31, 2016. This is an explanation for the longer average throughput time in February 2017 mentioned earlier, because no new cases start in January or February. Hence, all cases that are still not completed in February 2017 already started in 2016. This also explains the decreasing amount of cases in the beginning of 2017.

(b) Now that we have a basic overview of the process, we want to go a little deeper into it. We are still interested in the long-lasting cases, i.e. cases that take longer than 30 days. Hence, we first create a BPMN model for the cases that last less than 30 days. To do so, we set a filter to consider only those cases having a throughput time of 30 days or less. Then, we add the "Conformance" template as a new sheet to our Celonis project and select the option "Mine the target process". In the following window, we get the option to choose those variants we want to consider for our target model. We choose all of the variants and launch the analysis. An excerpt of the resulting process model is shown in Figure 10 (for the full model, please refer to Figure 19 in the appendix). Since we used all variants for mining the target model, the conformance is obviously 100%. However, the model was mined using only those cases lasting 30 days or less. Hence, we now remove the filter to check the conformance of the whole log to our model. A bit unexpectedly, we still have a conformance of 100%. Thus, no behavior exists in the remainder of the log that is not already allowed in our model.

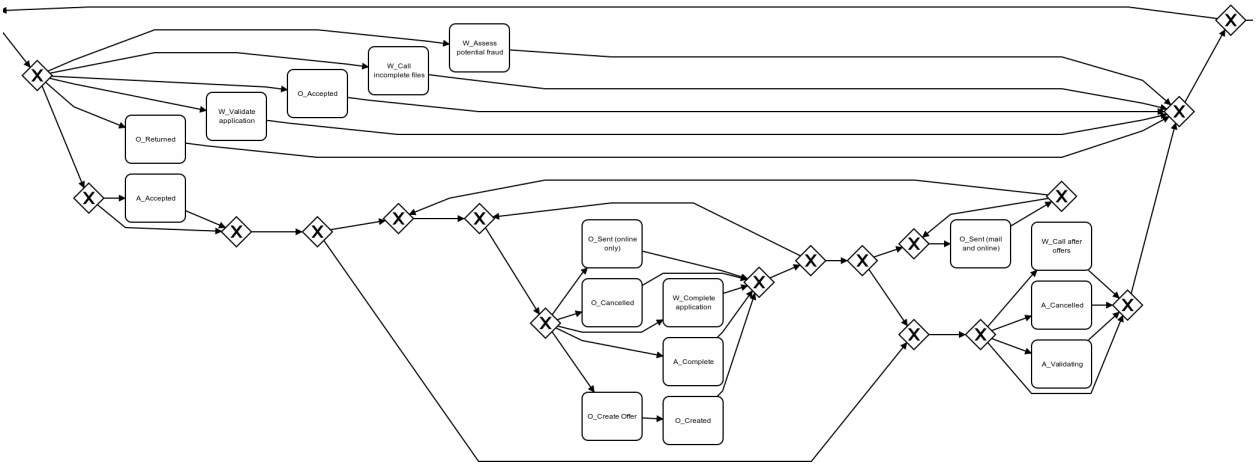


Figure 10: Excerpt of the BPMN model mined by the Celonis for the cases that last 30 days or less.

We therefore take a closer look at the model and realize that the model is very general. It is possible to skip almost every activity, and usually, there are options to go back a few steps using the choice constructs (see Figure 10). This explains that the possibly different behaviour of long-lasting cases is already captured in the model and does not lead to any violations. Thus, we assume the model to have good fitness but poor precision.

Another possibility to investigate the long-lasting cases is to use scatter plots. With them, we can detect correlations between different attributes. For this purpose, we create a new blank sheet in Celonis and gradually add six scatter plots to this new sheet. For each scatter plot, we use the predefined formula "Total throughput time in days" as y-value and define the x-values with the help of PQL queries:

1. `"Case_Table_csv"."REQUESTEDAMOUNT"`
2. `PU_FIRST("Case_Table_csv", "Activity_Table_csv"."FIRSTWITHDRAWALAMOUNT"))`
3. `PU_LAST("Case_Table_csv", "Activity_Table_csv"."MONTHLYCOST")`
4. `PU_FIRST("Case_Table_csv","Activity_Table_csv"."NUMBEROFTERMS")`
5. `PU_LAST("Case_Table_csv ", "Activity_Table_csv"."CREDITSCORE")`
6. `PU_LAST("Case_Table_csv", "Activity_Table_csv"."OFFEREDAMOUNT")`

In addition, we remove the previously used filter because we are interested in the long lasting cases (i.e., cases lasting longer than 30 days), and therefore, we would like to see the correlations with these cases in particular. Furthermore, we set the number of data points to be displayed in the scatter plots to 5,000 in order to see all our cases. The resulting scatter plots are shown as a screenshot in Figure 11. Even though we find some anomalies, like the

fact that the last credit score is either 0 or greater than 600 (see scatter plot 5), there seems to be no clear correlation between any of the six attributes and the throughput time.

Figure 11: Scatter plots for selected attributes in Celonis

Now, we would like to explore the DFG of our process. Therefore, we add a new sheet to our project and select "Process Explorer". Initially, Celonis considers all activities for the DFG. However, we restrict ourselves to the following activities: *A_Create Application, O_Created, W_Complete Application, O_Accepted, O_Refused, O_Cancelled, and A_Cancelled*. We do so by using the button with the crossed eye on the left hand side of the view and deselect all other activities. Afterwards, we set the two filters for the percentage of activities and the percentage of connections to the maximum in order to explore the full DFG. Since we removed all our filters already for the scatter plots, there are no active filters. The resulting DFG is shown in Figure 12. As we can easily see, the two longest lasting edges are from *W_Complete Application* to *A_Cancelled* as well as from *O_Created* to *A_Cancelled*, both last a median of 31 days. There are some more traces leading to *O_Refused* and *O_Accepted* that take around 10 days.

Taking into consideration the above mentioned aspects, it seems that the long-lasting cases are almost exclusively related to the fact that they are cancelled cases. Figure 9 shows that the distinctive spike of cases running over 30 days relates to cancelled cases that are likely to result from (semi-)automated processes. The DFG (see Figure 12) supports the thesis, because the only edges that last longer than 30 days (or even longer than 13 days which

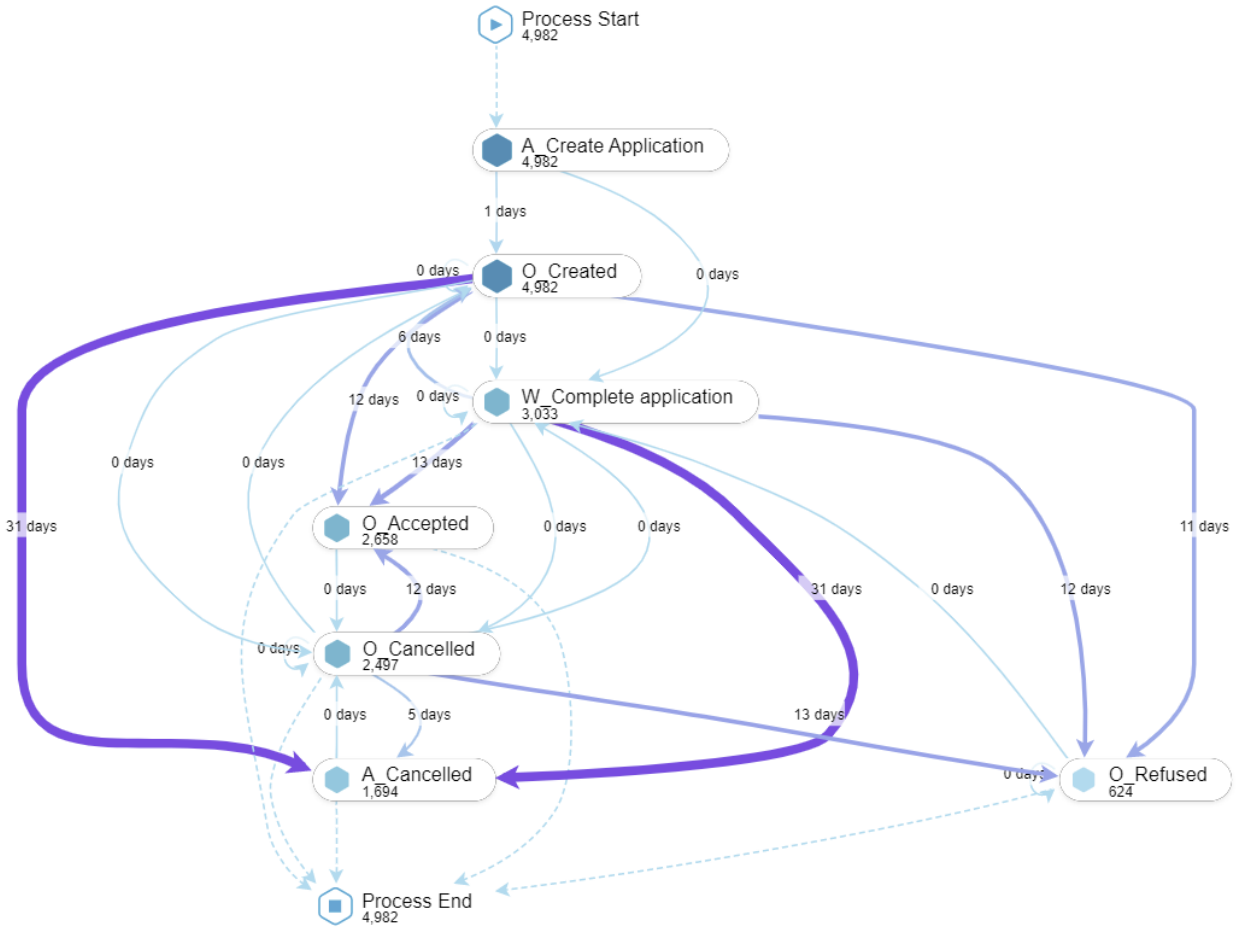


Figure 12: Full directly follows graph shown in Celonis for the following activities: *A_Create Application, O_Created, W_Complete Application, O_Accepted, O_Refused, O_Cancelled, and A_Cancelled*. The edges are annotated with their regarding performance in days.

is the highest median for all other edges) are those of cancelled cases. In order to prove our assumption, we filter the log to contain only traces lasting longer than 30 days and not containing *A_Cancelled*. Only 419 (8%) of all cases remain in the log. This means that although our assumption is not absolutely correct, such non-cancelled cases that last longer than 30 days are an exception. We are going to investigate that further in the next part of this exercise.

(c) With our knowledge from the previous exercise and the statement of the manager, we know that cancelled cases are no longer of interest to us. Hence, we apply a filter to our event log so that we only consider cases that do not flow through *A_Cancelled*. After applying the filter, 66% of the cases remain (3,288 cases).

The algorithmic happy path starts similar than before, but after executing *O_Sent (mail and online)*, the path differs. While the application was cancelled in the happy trace from

before, we now continue with *A_Validating* → *O_Returned* → *A_Pending* → *O_Accepted* after *O_Sent (mail and online)*. However, this happy path represents only 3.22% of the cases. We have 1,266 variants for 3,288 cases. This implies a very individual behavior.

Now, we look a bit closer into the throughput times again. The average throughput is depicted with 17 days in the process overview and with 18 days in the throughput times tab in the Celonis process overview template. The reason is that the process overview automatically filters out extreme outliers – as described by a small hint in the software. The distribution of throughput times is visualized in Figure 13. Because the cancelled cases are filtered out, we do not observe the distinctive spike in cases running over 30 days that we have seen previously. In contrast to that, we observe that most cases now have much smaller throughput times.

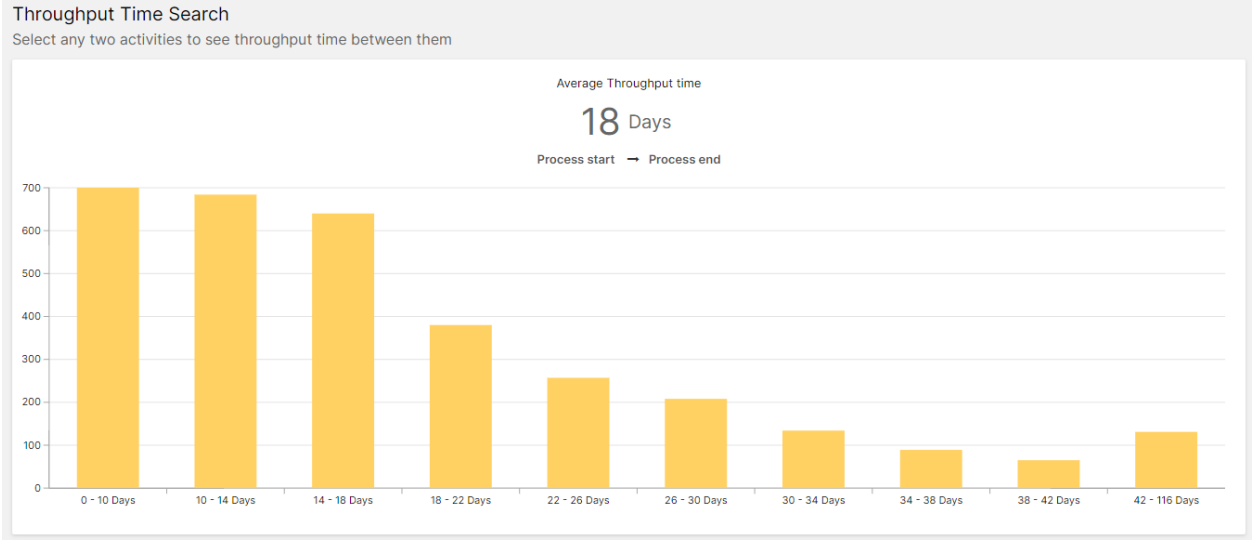


Figure 13: Distribution of throughput times for not cancelled cases

For a deeper analysis, Celonis offers an option to investigate the bottlenecks of our process very easy by listing those connections that increase the process throughput time considerably. We see that event *A_Validating* is target for the two most time consuming connections, i.e. *O_Sent (mail and online)* → *A_Validating* (throughput time of 8 days, 45% affected cases) and *W_Complete application* → *A_Validating* (throughput time of 9 days, 49% affected cases).

Next, we add an additional filter for long lasting cases, i.e. cases that take more than 30 days. Using this filter, the previously found bottlenecks have an even higher throughput time of 15 respective 13 days (see Figure 14). Furthermore, both bottlenecks are very frequent for the sublog resulting from the application of the previously mentioned filters as they occur in 37 respective 56 percent of the long lasting cases.

Since we only have single timestamps per activity and not separated timestamps for start and end, we can assume that the target activity of each connection causes the delay, i.e. in this case, *A_Validating* seems to be the main bottleneck in our process. We investigate the activity *A_Validating* for the filtered log with the help of the process explorer and see that

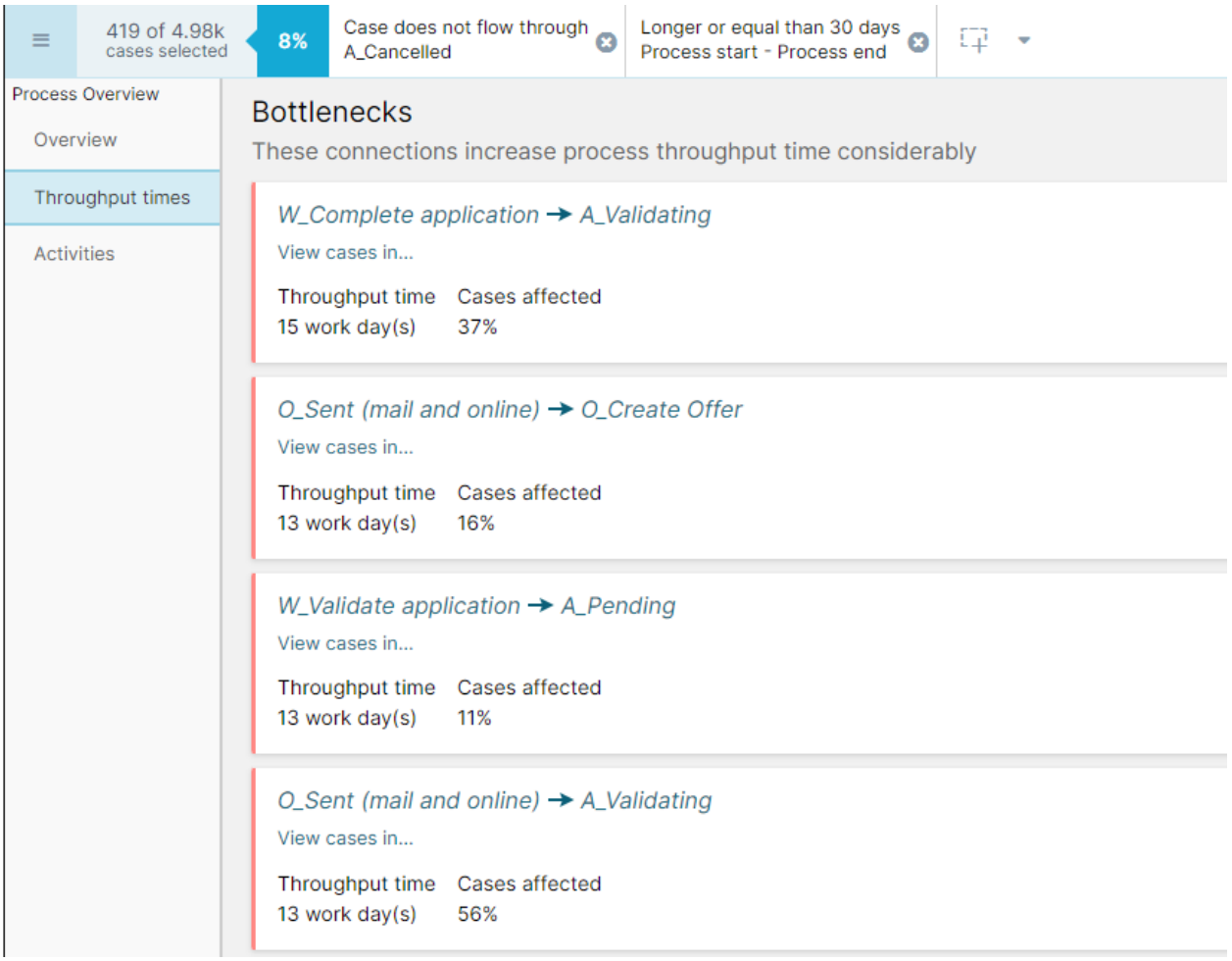


Figure 14: Screenshot of "Process Overview" in Celonis showing the bottlenecks for cases lasting longer (or equal) than 30 days and not flowing through *A_Cancelled*.

it occurs in all cases (see Figure 15). Furthermore, we realize that this activity even occurs more than twice per case on average and most of the incoming connections are lasting at least 6 days and even 16 days in average if the successor activity is *A_Complete*.

Looking into the subprocess of workflow activities in Figure 16, we find various bottlenecks. The first observation is that *W_Validate application* has two ingoing arcs with high throughput times of 18 days (from *W_Handle leads*) and 12 days (from *W_Complete application*) while being a very frequent activity. This strengths our previous finding of a long lasting validation process. The second activity that is frequent and has ingoing arcs with high throughput times is *W_call incomplete files* with throughput times of 16 and 14 days and the same predecessor activities as the previously considered activity. So, there are long waiting times until the bank can call for incomplete files.

Finally, we investigate a possible relationship between the throughput time and the number of offers. Our first assumption is that every additional offer delays the throughput time since an additional cycle must be run through. To check our assumption, we export the filtered

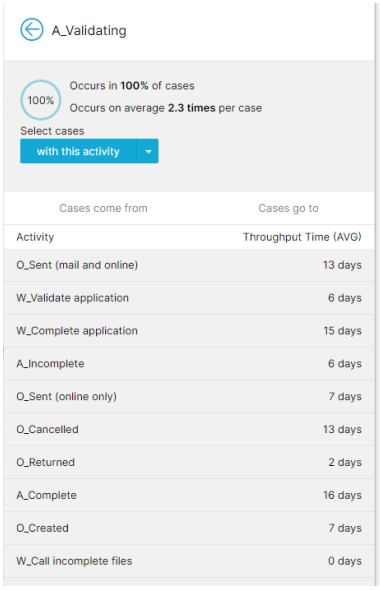


Figure 15: Screenshot of activity information in the "Process Explorer" in Celonis focusing on activity *A_Validating*.

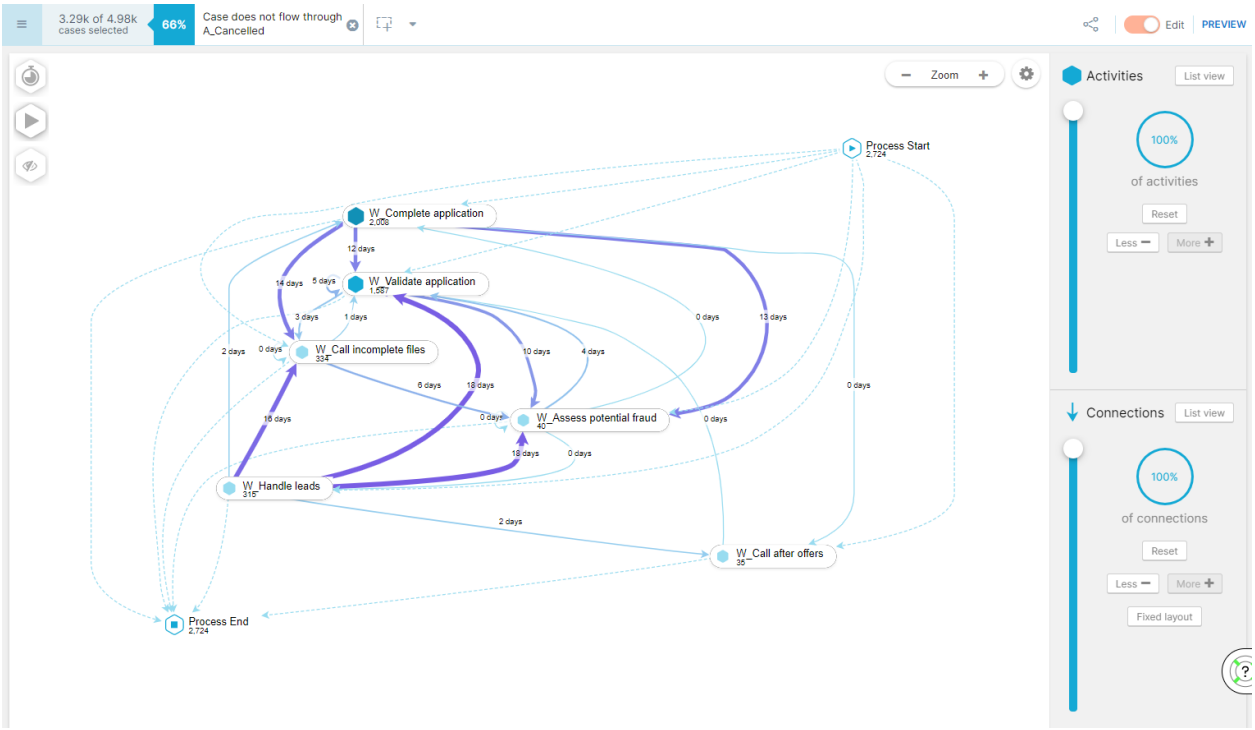


Figure 16: Screenshot of "Process Explorer" in Celonis containing only workflow activities, i.e. activities starting with *W_*.

dataset, which contains only not cancelled cases, using an OLAP table. Afterwards, we create a scatter plot using RapidMiner to visualize the relationship between both attributes (see Figure 17). In the scatter plot, we add a linear regression line and color the data

points based on their completion state. When only looking at the data points, we cannot clearly observe a relationship. However, looking at the regression line, there is a moderate positive linear correlation: the higher the number of offers, the higher the throughput time. However, it does not seem to be a very strong correlation. We clearly see that there are also a lot of long lasting cases that contain only a single offer. Furthermore, a majority of cases contains less than three offers, so the regression line might be very much influenced by some outliers for cases with more than two offers. We come to the conclusion that there is no clear correlation between the number of offers and the throughput time. Even if we consider the result of the regarding case (color attribute) we cannot spot a clear correlation. Although we did not expect this observation, it makes sense when we consider our earlier findings where we realized that the main reason for long lasting cases is the fact that they are cancelled. Apart from those cases, which we do not consider in this part of the exercise, the main reason for delays is the activity *A_Validating*.

Figure 17: Scatterplot showing the relationship between the number of offers and the throughput time. The color of the data points indicates the result of the regarding case. Note that some jitter is applied for the number of offers.

(d) Taking into account the previous insights, the main reason for a long throughput time seems to be the behaviour of cancelled cases (see Figure 9). It is clearly visible that almost only cancelled cases are lasting longer than 30 days, apart from a few exceptions. Applications that get cancelled have two main bottlenecks each causing a delay of 28 days (see Figure 18). As mentioned above, we assume that this is a semi-automatic procedure, where applications are closed if the customer does not respond within four weeks. That could be improved in two ways. The bank could either automate this process to become

fully automated. The cases would then not be processed faster (or at least only by a few days), but the manual effort would decrease considerably. A second way to improve this process that could impact the throughput time is to send weekly reminders to customers, for example. This would allow them to proactively cancel the application and thus end the process early.

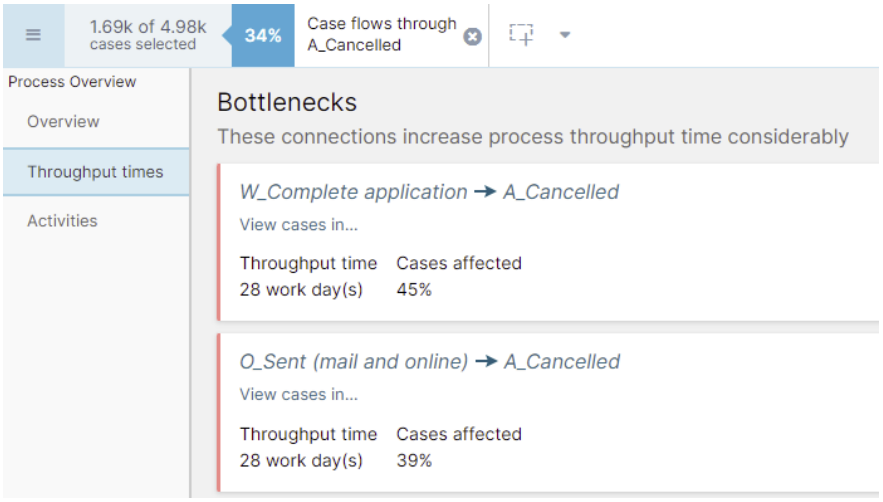


Figure 18: Screenshot of "Process Overview" in Celonis showing the bottlenecks for cancelled cases.

However, since the manager mentioned that cancelled cases are not of interest for the company, we also looked at bottlenecks for non-cancelled cases. An activity that has heavily impact on the process duration is *A_Validating* which has average throughput times of eight to nine days. Looking at long-lasting cases only, the throughput time even increases up to 15 days (see Figure 14). Since we do not know how this activity looks like in reality, we can only make assumptions about possible improvements. If the validation is a complex process that is research intensive for certain applications, one could increase the resources (e.g. number of staff working on it). Otherwise, if the validation is simply pending, one could introduce escalation systems to force faster processing.

Somewhat surprising is the fact that the throughput time does not seem to be related to the number of offers, contrary to our expectation. There might be a small impact of a few days, but no delay to a long-lasting case.

Appendix

Excerpt from Situation Table (Exercise 1)

CASE ID	APPLICATION TYPE	LOANGOAL	REQUESTED AMOUNT	FIRST CREDIT SCORE	LAST CREDIT SCORE	...
Application_1000386745	New credit	Car	5000	1080	1080	...
Application_1001177986	New credit	Existing loan takeover	26000	972	972	...
Application_1001866944	New credit	Car	10000	896	896	...
Application_1002013470	New credit	Home improvement	17000	0	0	...
Application_1002348901	New credit	Car	15000	0	0	...
Application_1003264681	Limit raise	Home improvement	20000	0	0	...
...	...	...	...	...	...	...

HAVING CALL TO COMPLETE	NUMBER OF OFFERS	FIRST OFFER	LAST OFFER	FIRST OFFER MONTHLY COST	LAST OFFER MONTHLY COST	THROUGH-PUT TIME	RESULTS
0	1	5000	5000	100.25	100.25	11	Successful completion
0	1	26000	26000	264.74	264.74	29	Successful completion
0	1	25000	25000	535.22	535.22	36	Successful completion
0	1	17000	17000	319.2	319.2	22	Denied by the bank
0	1	15000	15000	157.59	157.59	32	Cancelled by client
0	1	20000	20000	350	350	17	Successful completion
...	...	...	...	...	...	...	...

Table 4: Extract from the situation table created in Celonis (split into two parts).

BPMN Model (Exercise 5b)

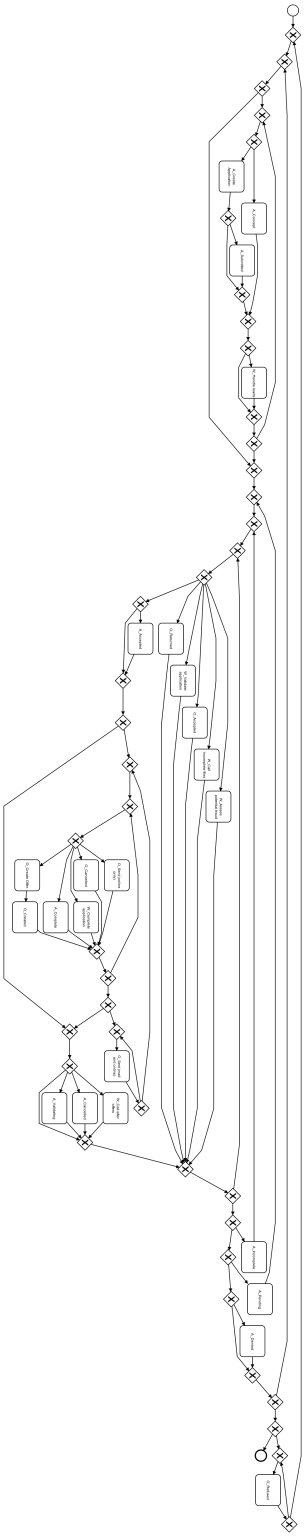


Figure 19: BPMN model mined by Celonis for the cases that last 30 days or less.