Advanced Process Mining SS24

Report Assignment Part 2

Group 99*:*
Lennart Hüsing (422486)
Gabrijel Sokcevic (422606)

Statement on the usage of LLMs

We used GPT for transferring tables generated inside the notebook to this TEX file. We also used it at some places for RegEx and for placing figures in the way we want them to. When using GPT inside the Code we put a comment there.
This is allowed since we solved the tasks on our own.

Introduction

In this report we analyze your loan application process and give you various insights into the process. We provide you with numbers regarding the process, but also different models and deviations discovered during the analysis. We distinguish between two branches, whereas the first one allows for online and in bank applications for a loan and uses a different information system than the second branch. Branch two does not allow for online applications and also has fewer event attributes. Nevertheless, the branches have many similarities but also differences, which are expressed in this report.

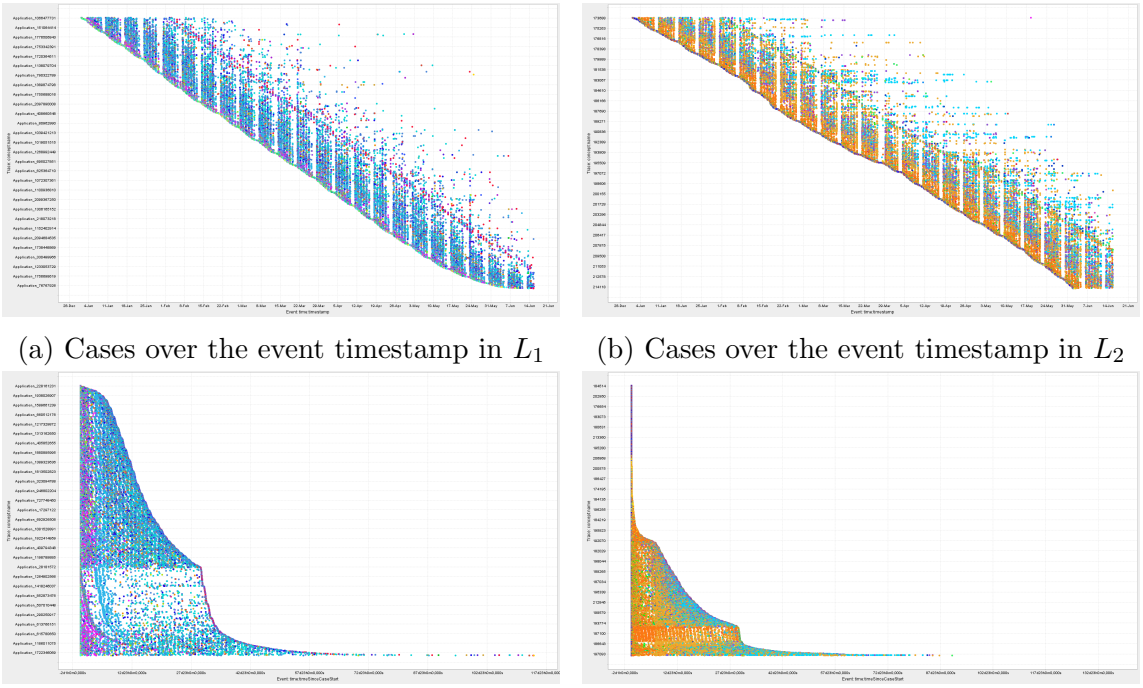
Preliminary Analysis

In this section we are going to analyze two different branches used by a bank for their loan application process and state the key differences one can obtain by looking at statistics. The process usually starts and should start with an application by the customer and end with a response from the bank. Most often, we see that applications get send in, accepted and then the offer is prepared and the customer is notified about the success. The process ends with activities regarding the workflow, but also with possible cancellations and calls after the offer.

a) Both branches seem to have similar amounts of applications, which can be seen as the amount of traces in the first log is **14159** and **13087** in the second log. There are **7697** trace variants in the first log and **4366** in the second log. This might be explained due to the first log having **527123** events and the second one only **262200**. Therefore, looking at the amount of events, the second log has less possibilities to be diverse. Further, more events happening might explain why the average length of a trace in the first log is **37.229** and therefore higher than the average of **20.035** observed in the second log. Being more active, the first branch has **124** resources working while the second branch has only **68** resources, excluding "nan" (at this point it is not clear what "nan" could mean and since it is not asked for the task at hand we do not elaborate it further). Looking at the longest work both branches had for a process we observe a duration of approx. **115** days for the first log and **137** for the second log. Contrary, examining the shortest duration of a trace we see that the first branch took **282** seconds for its shortest traces and only **2** seconds lasts the shortest trace in the second log. The average case duration is **21.51** days together with a median of **7.28** days for the first branch and an average of **8.62** days and a median of **0.81** days for the second branch. Both logs start on 01/01/2022 and end on 06/15/2022. Looking at the activities that happen only on one branch and not on the other we have that *A_Validating, O_Sent*

(online only), A_Pending, W_Shortened completion, A_Concept, A_Create Application, O_Sent (mail and online), and A_Incomplete occur only in branch one and *A_Approved, A_Preaccepted, O_Sent (mail only), A_Activated, A_Partially Submitted, A_Registered, and W_Modify contract details* occur only in the second branch. However, without knowing how the process on each branch works exactly, the names might not have the same meaning if the sound similar (and vice versa).

b) In the following figure we can see the 4 dotted charts.



(a) Cases over the event timestamp in L_1

(b) Cases over the event timestamp in L_2

(c) Cases over time since case start in L_1

(d) Cases over time since case start in L_2

Figure 1: Visualization of the dotted charts (Qualitative picture in the images folder)

Cases over the event timestamp In both charts we can see a structured layout, where we observe column-wise behavior over time representing each week. Also, this structured layout might be explained due to strict times lots reserved for certain activities. On Sundays there are usually only few activities (acceptances, completions, validations, rejections etc.) which *could* be system generated. Both branches allow for a continuous start of an application, which is seen by the bottom linear slope of the chart where the first branch has a higher case arrival rate than branch two (but the case arrival rate for the first branch becomes less towards the end). Moreover, the first branch has a much "denser" slope of events, which might be explained due

to it having more possibilities to apply for a loan than the second branch. Further, we can observe daily batching in the first branch every day at 9PM for the first half of the timeline and then shifting to 10AM for the second half on the activities for canceling and calling after offers. In the second log we observe only a little bit of batching, happening at 10AM in the first half and then at 11:15AM in the second half and batching the same events as branch one. Looking at the weekdays everything, especially regarding the workflow, can happen in any order in both branches. Lastly, we discover a second slope (red in branch one and slightly orange in branch two) "above" most activities and parallel to the first slope mentioned above. This slope represent the end of most of the processes and indicates that most traces usually have the same duration on their branch, except some outliers, which can be seen as the seemingly randomized dots in the right upper part of each chart (mostly W_Call incomplete files).

Cases over time since case start Changing the time interpretation yields the two bottom subfigures of *Figure 1.* The first observation is the spike in the chart for the second branch, which describes that on this branch roughly half of the traces are stopped almost immediately (might be a result of the automated assessment, which is either more strict or simply shorter on branch two). This is supported by the low median found in sub task a. This constitutes a big difference between the two branches and could be considered noise. Therefore, filtering it out and scaling the chart up we end up with a quite similar chart as for the first branch. One additional observation is the white space towards the bottom of both charts, where the applications starts and then not much happens and then the process ends after roughly 30 days. This could imply a cluster of customers, where many customers apply, and then do not interact with the process anymore and then get reminded by the system (blue vertical line in chart c) and then after no response the process gets canceled by the system automatically. This can be also seen in the chart for L_2, however here we observe more W_Complete application and more offers after hours, which could imply that this is a cluster of people negotiating more and that this might be more frequent on branch two. Hence, we observe a visual similarity with two different meanings. The described cluster of customers for chart c can be seen in chart d towards the bottom.

c) We have three sublogs, which is why we look at all there abstract processes and try to find necessary activities. Looking at the application part of a log we see that, by the description of the process, *A_Concept* represents the start of an application and *A_Pending*, *A_Denied* and *A_Canceled* represent the end of an application process, hence these are mandatory for any trace. The last two also allow for termination of a trace.

Looking at the offer DFG we see that offers usually end with *O_Refuse*, *O_Accept* or *O_Cancel*. This however is tied to the condition that an offer has been created before. Therefore, our filtering strategy regarding the offer part is that creating an offer appears if and only if the offer is either accepted, canceled or refused.
Examining the workflow DFG (after filtering out some noise) we see that *W_Complete application* is part of every workflow. The workflow is terminated if and only if *W_Complete application* appears in the trace.

d)

1. The fraction of traces with *A_Pending* in the first branch is 56% while the fraction of successful traces in the second branch is 17.7%. This means that the applications happening on the second branch are far less successful. Maybe they are stricter with their loan offers or just have worse customers in terms of credit scores.
2. On the first branch a total of 43.995% of the traces are not successful, whereas the number for the second branch is lies at 82.3%. This result is complementary to the the previous one.
3. The fraction of traces with more than one offer is 100% for both branches, which could be the result of negotiations over the sum of the credit.

Preprocessing

Before discovering the process one might filter the event data and filter out everything that does not appear too often, since this would be regarded noise and might defer the impression of the actual process. One example for this might be the short traces on the second branch (see *Figure 1*). In addition, we could shorten processes that need much longer to complete than the average, since they could be considered outliers. In this log in particular three types of events exist (W, O, A). Therefore we split the log into three parts as well. We apply these preprocessing strategies.

Process Discovery

In the following we discover models for the three stages of the process and the two logs. We start by looking at the application process.

Process of Application States We first mine the log of the first branch. Starting with the Alpha Miner we do not end up with a workflow net and thus this miner is not applicable for us. The heuristics miner together with a frequency threshold of 0.1

yields a quite large and not simple model, which is why we omit it. Further, using a threshold of 0.3 gives us a too overfitting model. The inductive miner without any parameters gives a good model in terms of replaying the log but is worse than the inductive miner with a noise threshold of $0.1 - 0.2$. The IMf gives us a simple log which has good fitness, is not overfitting and represents the process the best. Further, we ran the ILP Miner but ended up with a model, having many arcs and not good precision and fitness after analyzing with ProM. Lastly, we tried the eST Miner but it could not finish computing.

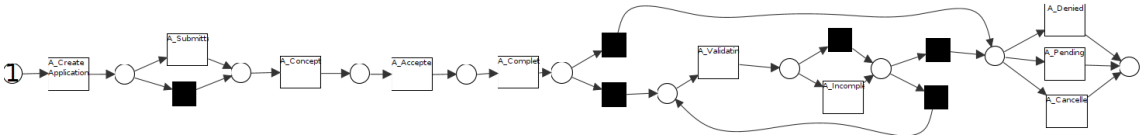


Figure 2: M_1^A mined with IMf and a noise threshold of 0.2

We go the same way as before and first run die Alpha Miner on the filtered log for branch two. This gives us a workflow net this time, but it is not sound, contains too many implicit places and is not simple. Speaking of simplicity, we again have that the ILP miner is not simple and therefore not further contemplated. The IMf with frequency thresholds of $0.1 - 0.2$ is precise but too general, which is why we recommend the normal IM (see *Figure 3*). The Heuristic Miner would also be a good choice, because if its statistics. It has a fitness of 0.99 and a precision of one but is not as simple as the Inductive Miner, who has a fitness of 1, by construction, but a precision of 0.956. For reasons of simplicity we choose the Inductive Miner in its classical variant. Again, the eST Miner could not finish computing.

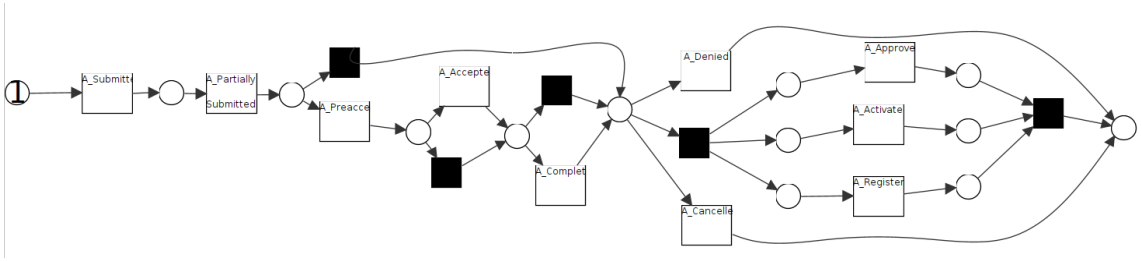
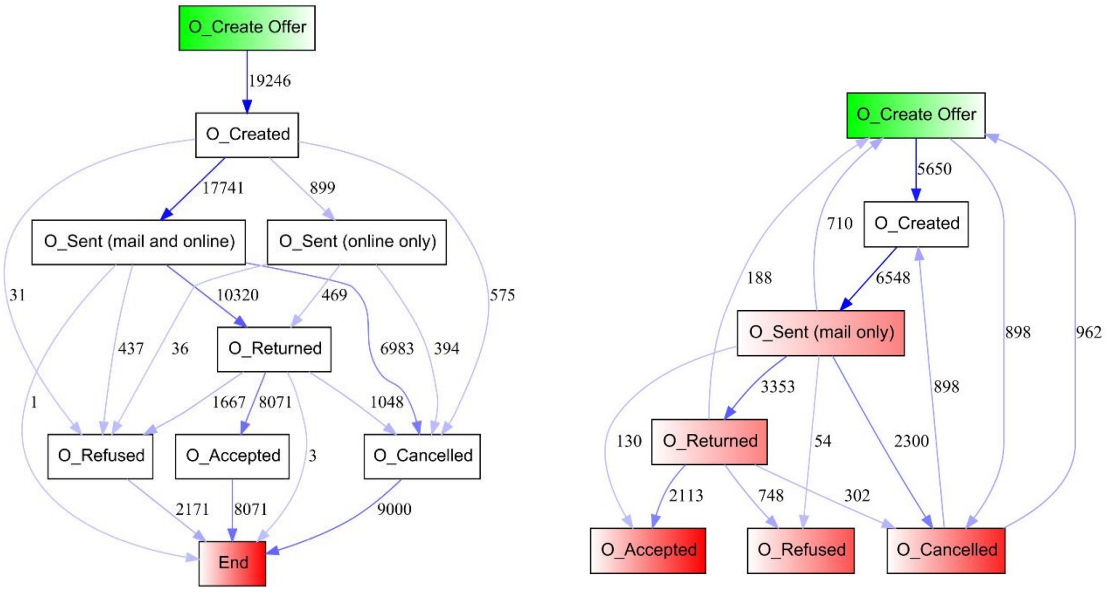


Figure 3: M_2^A mined with IM

Comparing the two models, we see that they look similar by their structure. The first model starts with creating an application and then choosing between submitting or directly moving towards A_Concept. This is not observed in the second model, which is due to the fact that submission is only possible via an online application.

Confusingly, the second branch has also a submitted state which is why we cannot rely on names when analyzing (see enhancement strategies). Now the "low" precision of M_2^A comes to shine when we observe that multiple activities can be skipped by tau transitions. This is not highly observed in the first model, where we only have a few decisions and applications are always first accepted and completed (which can be skipped in the first model). The second model has a pre accepting state which cannot be seen in the first model. This might be a result of the difference in the information systems. Towards the end M_1^A allows for looping between validation and marking incomplete files, which cannot be seen in the second model (according to the model the second branch has to have complete files at some point in time). At the end we can choose between denying and canceling and the state A_Pending in the first model. The difference to the second model is here that branch two allows for a parallel run between A_Approved, A_Activated and A_Registered.

Process of Offer States As requested we do not filter the offer logs. The directly follows graphs (abbrev. DFG) can be looked at below.



(a) DFG L_1^O (b) DFG L_2^O

Figure 4: Comparison of the DFG in two branches.

The main difference is that the DFG for the first log has an end activity which cannot be seen in the second log. Further, the DFG for the first log has no loops and therefore does not allow for repetitions in an offer process. The other DFG allows for the creation of multiple offers after an offer has been canceled, which

could be the result of negotiations, allowed by the second branch. Hence, acceptance and refusing could be considered the two end activities for an offer process on the second branch, but *not quite*. Looking at the colors we observe that the first branch end always in end but in the second branch some offer processes end in O_Sent (mail only) and O_Returned, representing incomplete processes. For completeness we should consider the first DFG, nevertheless does the second DFG represent a realistic behavior, with customers simply not responding and rejecting offers. For *analyzing* the process we choose the second DFG (because we can analyze where the problems exactly lie to enhance the process) but for a desirable behavior and a *discovery* we would like to see the first DFG, since it avoids loops and has individual start and end activities. The reasons for the differences could be that the first branch has better tracking of the process. This is supported by the fact that the first branch offers one additional technology and therefore might also have better technology in general when it comes to control of the workflow.

Process of Workflow States Lastly, we analyze the workflow of the process. To do so, we mine the two models M_1^W and M_2^W with the IMflc Miner, using a noise threshold of 0.05 for the first model and a threshold of 0.2 for the second one. We omit the IMlc Miner since it yields too large and not simple models.

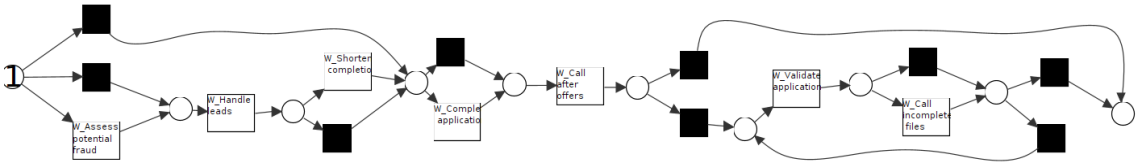


Figure 5: M_1^W mined with IMflc and a noise threshold of 0.05

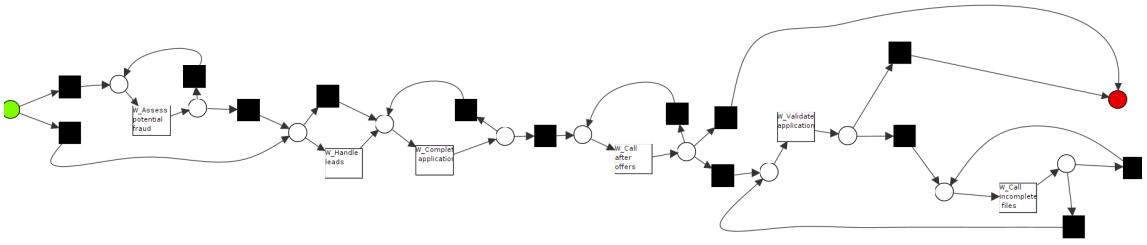


Figure 6: M_2^W mined with IMflc and a noise threshold of 0.2

Comparing the two models we can see that both allow for many choices. Still, the second model required for a complete application whereas this step can be skipped in the first log, but both require a call after offers. A call after offers can be looped in the second model, which is realistic. The end part for both workflows is the same as both models either finish with call after offers or go for validating the application

and possibly calling and looping for incomplete files. The first branch has less choices than the second one and follows therefore a stricter order, which has been already mentioned through the course of this report. The fact that the second model has more loops shows a general trend, as loops were also seen in the DFG for the second log in the offer state.

Conformance Checking

This section covers findings of behavior that was not expected and should not happen, as it represents a deviation. For this, we use the process explorer in ProM and compute alignments between the models M_1^A (see *Figure 2*) and M_2^A (see *Figure 3*). Further, we apply token based replay to support our findings and also run the inductive visual miner for optimal results.

Deviations between L_1^A and M_1^A In this case we observe one deviation when looking at alignments. This means, that an activity, e.g. A_Denied, has been per-

Figure 7: A_Complete appearing only in the model and not in the log

formed at the process without A_Complete being observed in the log. However, *Figure 9* shows that this is the case because A_Complete is *forced* after A_Accepted. This can be also observed using the inductive visual miner and choosing the option

Figure 8: Reason for the above deviation

"paths and deviations". Further, it can be seen by applying token based replay and examining the missed tokens (and the remaining tokens before A_Complete) before the last three activities.

Deviations between L_2^A and M_2^A The model has perfect fitness but is not completely precise as it allows for behavior not observed in the log. This can be seen by

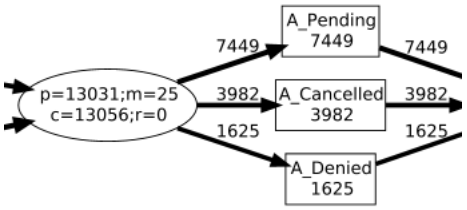


Figure 9: Another reason for the above deviation

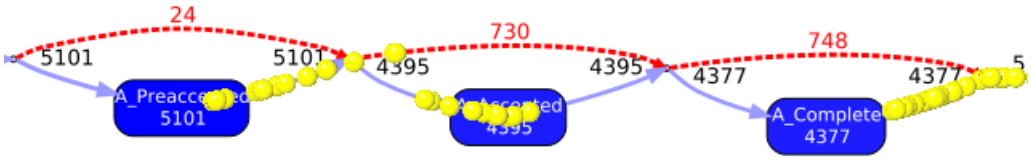


Figure 10: Deviations on the preprocessed log and M_2^A

applying the Inductive Visual Miner in the figure below. Computing alignments on the preprocessed log gives us zero deviations. However, for completeness applying the same methods as before, e.g. not filtered log and same miner on the not filtered log, we do observe the following deviations. Further we see that A_Preaccepted is

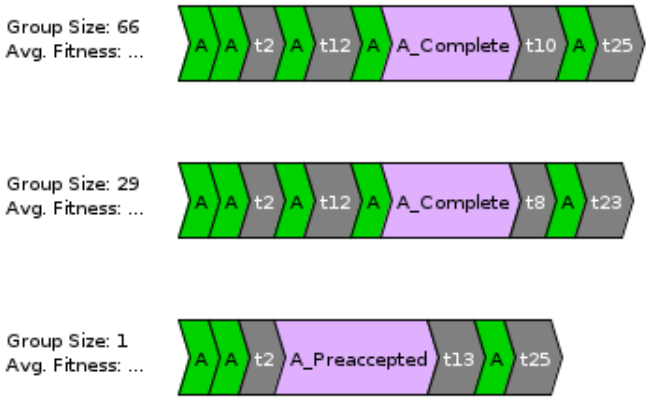


Figure 11: A_Complete and A_Preaccepted appearing only in the model and not in the log

also a model move and not seen in the log. The same is the case for A_Validating.

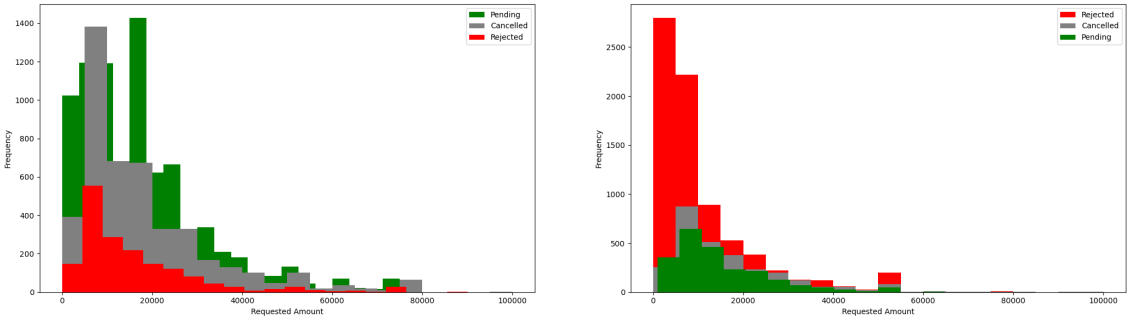
Figure 12: A_Validating appearing only in the model and not in the log

Additional Insights

While analyzing the process we found three additional insights we would like to share with you in order to optimize the process. A summary and enhancement technique is found in the summary.

Rejection of Applications

Our first finding is concerned with the applications together with how much money a customer requested. While the amount of approved and cancelled orders is roughly



(a) Branch one

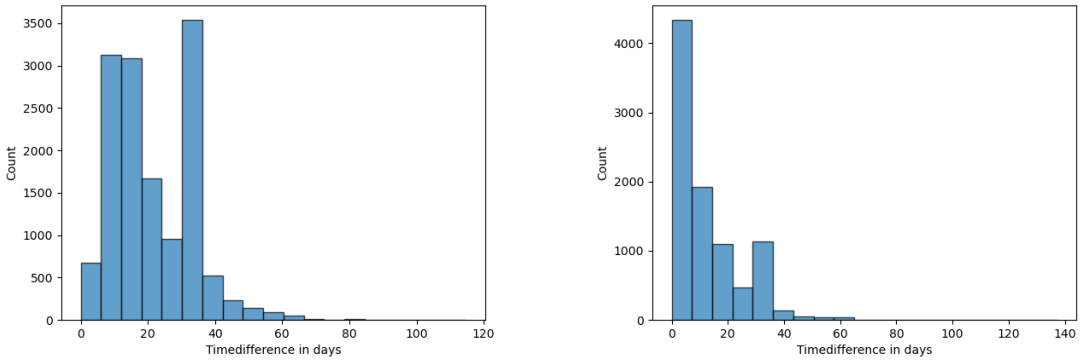
(b) Branch two

Figure 13: Comparison of the denied, cancelled and pending applications after requested amount for each branch

the same in both branches, the amount of denied orders in branch two surpasses the ones in branch one by far. Interestingly, this is not even depending on the amount of money requested, which is is important because if the applications rejected in branch two would not be rejected in branch one the bank would be missing out a lot of money. Both branches have a peak of cancelled orders at roughly 5000 requested dollars. Maybe other banks have better conditions for this range of requested money and our bank needs to adapt.

Duration of the process

The second finding presents the duration of the application of the two branches. To support our arguments we provide the following two diagrams.



(a) Branch 1 (b) Branch 2

Figure 14: Comparison of the time difference between the first and last event of the workflow

In general less time is spend for each application in branch two. When looking further into this, one can see that the time spend for approved applications is similar. When looking at the cancelled and denied applications though, branch two spends way less time on them on average .
A takeaway could be that branch two denies far more applications from the start. Therefore they save more time, compared to branch one. On the other hand as seen in the first part of this task branch two is likely to deny too many applications. Maybe when combining both approaches of denying, one would achieve the optimum between time spend on unprofitable applications and profitable applications approved.

Applications after 30 days

Interestingly both branches have some sort of automatic cancellation after roughly 30 days . In both branches the applications which are still open after 30 days are cancelled in over 75% of the applications.
Since branch 2 is denying way more applications from the start, these cancelled applications after 30 days only makeup 8.3% of total applications. For branch 1 on the other hand these cancelled applications make up 24.7% of all applications. This is significant because branch 2 seems to be much better at denying customers, who would (be) cancel(led) later on anyways. Still this might come at a risk of missing out on potentially successful applications.

Conclusion

In this report we have analyzed the loan application of two branches, which follow the same process. We analyzed two logs, one of each branch. While both logs are from the same timeline and have a similar amount of applications, the number of events in the log from branch one exceeds those from branch two by far. This is due to the different granularity of logging performed. Additionally different activities occurred in the logs, which should be avoided. Our suggestion would be to use the logging of the first branch, since it allows for better insights and analysis.

We mined various models for the logs and examined which one explains the process the best in terms of simplicity, generalization, fitness and precision. We saw that the inductive miner (IM and IMf) succeed to represent the process the best. We deliver to you multiple models, which allow to visualize the process in different states (W, A, O). We have checked for deviations with these models and could only find a few. These indicate that the log not always follows a strict order. This should be avoided at best.

We also found some room for improvement. In the second branch way more applications are denied, while in the first branch way more applications are approved. This is likely caused by the automatic assessment, which seems to be much stricter in branch 2.
As a cause of that, fewer applications are cancelled in branch 2 after 30 days which is a good thing, since these seem to be removed by the assessment. Still branch 1 has more approved applications, which are probably removed by the assessment in branch 2. In both branches this automatic assessment should be reevaluated in order to find the best between both.

Lastly branch 1 spends more time on the applications compared to branch 2. On approved applications they spend the same amount of time in general, but on cancelled or denied applications branch 2 is much faster. This also explains why branch 2 can handle almost the same amount of applications while only having half the resources compared to branch 1.

Right now we can not decide which assessment is better, since we cant compute the profit for an application in log 2, since key attributes are missing. Please use the same attributes as in log 1 for the second log, in order to get more advanced insight.

Appendix

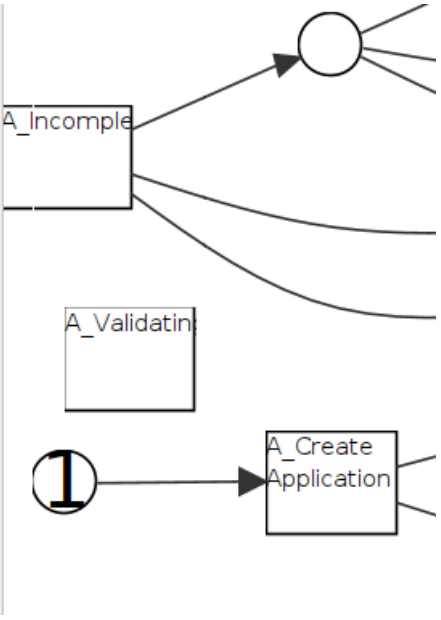


Figure 15: Reason to not choose the Alpha Miner for L_1^A

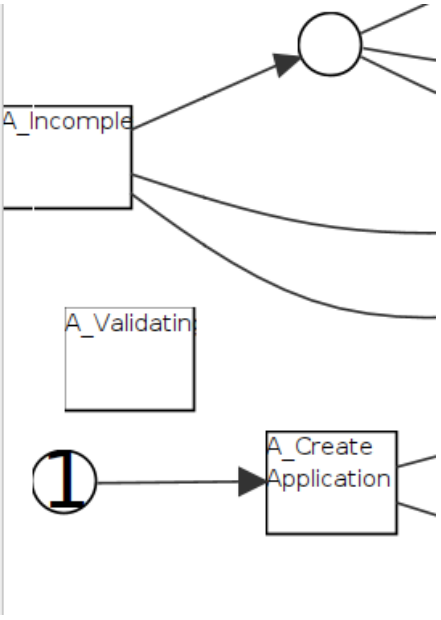


Figure 16: Reason to not choose the Alpha Miner for L_2^A

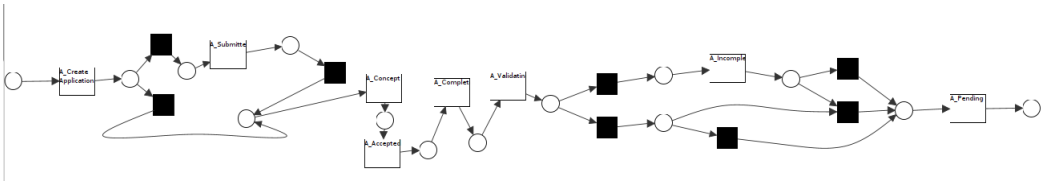


Figure 17: Reason to not choose the Heuristic Miner for L_1^A

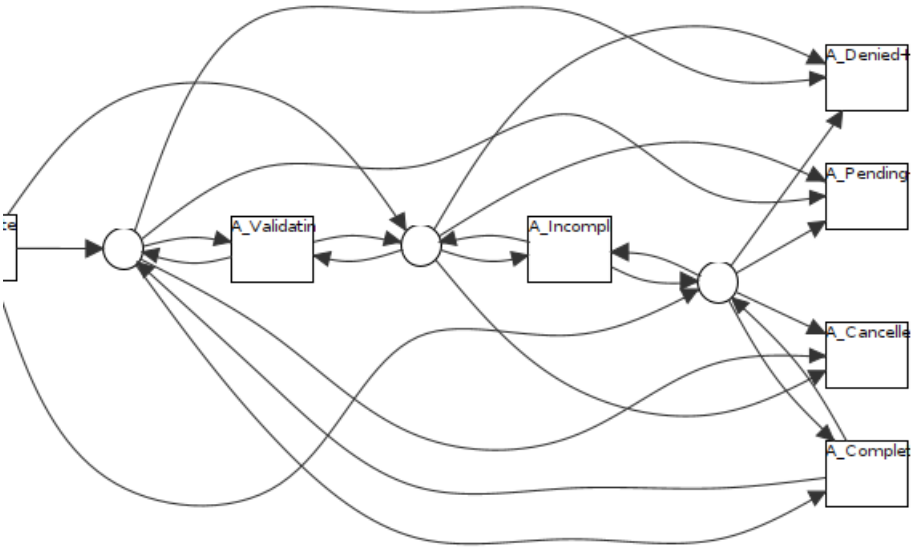


Figure 18: Reason to not choose the ILP Miner for L_1^A

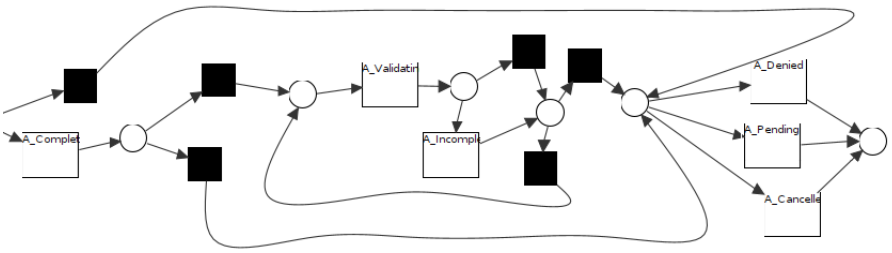
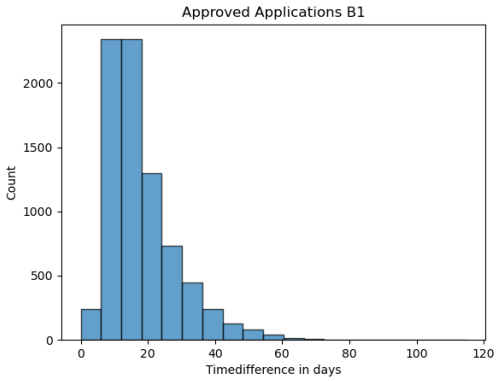
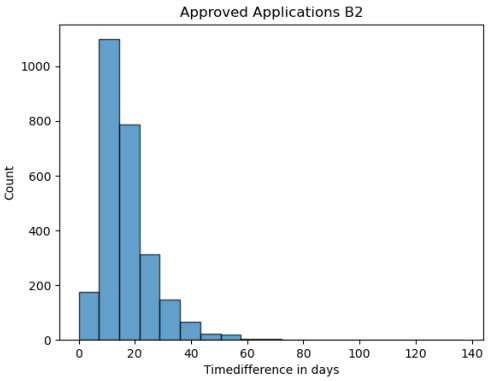


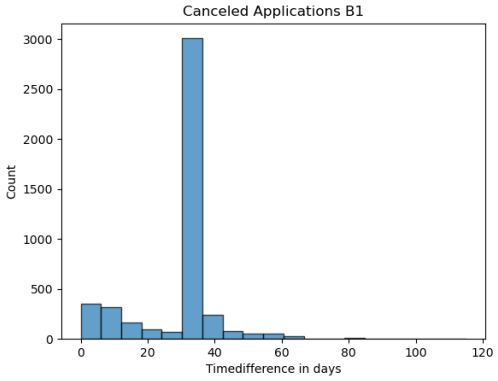
Figure 19: Reason to not choose the Inductive Miner for L_1^A



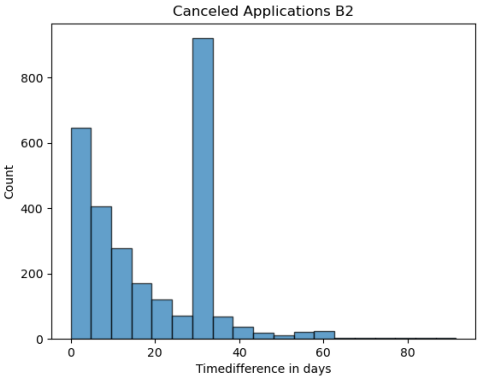
(a) Approved - Branch 1



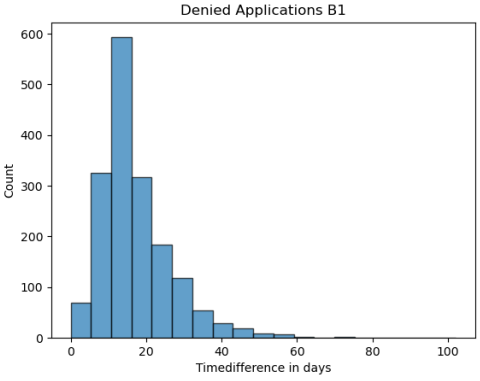
(b) Approved - Branch 2



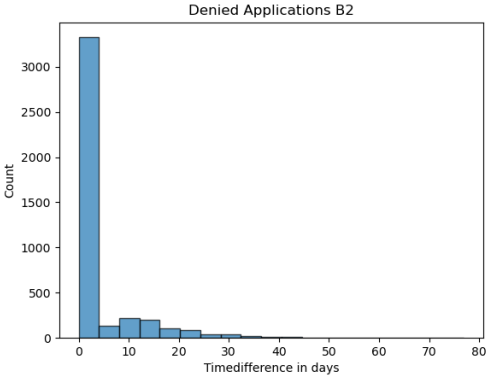
(c) Cancelled - Branch 1



(d) Cancelled - Branch 2



(e) Denied - Branch 1



(f) Denied - Branch 2

Figure 20: Comparison of the time difference between the first and last event of the workflow sorted after approved, denied and cancelled applications