BSI - CHALLENGE 2020

Marco Incerti
Università degli Studi di Milano
Matricola 989428
Prof. Paolo Ceravolo

07.2022

1 Scenario

This challenge focuses on studying and transforming the real data that has been provided to us by whatever means possible. In many organizations, staff members travel for work. They travel to customers, to conferences or to project meetings and these travels are sometimes expensive. As an employee of an organization, you do not have to pay for your own travel expenses, but the company takes care of them.

At Eindhoven University of Technology (TU/e), this is no different. The TU/e staff travels a lot to conferences or to other universities for project meetings and/or to meet up with colleagues in the field. And, as many companies, they have procedures in place for arranging the travels as well as for the reimbursement of costs.

On a high level, we distinguish two types of trips, namely domestic and international.

For domestic trips, no prior permission is needed, i.e. an employee can undertake these trips and ask for reimbursement of the costs afterwards.

For international trips, permission is needed from the supervisor. This permission is obtained by filing a travel-permit and this travel permit should be approved before making any arrangements.

To get the costs for a travel reimbursed, a claim is filed. This can be done as soon as costs are actually payed (for example for flights or conference registration fees), or within two months after the trip (for example hotel and food costs which are usually paid on the spot).

1.1 The Process Flow

The various declaration documents (domestic and international declarations, prepaid travel costs and requests for payment) all follow a similar process flow. After submission by the employee, the request is sent for approval the travel administration. If approved, the request is then forwarded to the budget owner and after that to the supervisor. If the budget owner and supervisor are the same person, then only one of the these steps it is taken. In some cases, the director also needs to approve the request.

In all cases, a rejection leads to one of two outcomes. Either the employee resubmits the request, or the employee also rejects the request.

If the approval flow has a positive result, the payment is requested and made.

The travel permits follow a slightly different flow as there is no payment involved. Instead, after all approval steps a trip can take place, indicated with an estimated start and end date. These dates are not exact travel dates, but rather estimated by the employee when the permit request is submitted. The actual travel dates are not recorded in the data, but should be close to the given dates in most cases.

After the end of a trip, an employee receives several reminders to submit a travel declaration.

After a travel permit is approved, but before the trip starts, employees can ask for a reimbursement of prepaid travel costs. Several requests can be submitted independently of each other. After the trip ends, an international declaration can be submitted, although sometimes multiple declarations are seen in specific cases.

It's important to realize that the process described above is the process for 2018. For 2017, there are some differences as this was a pilot year and the process changed slightly on several occasions.

2 Data

The data is split into travel permits and several request types, namely domestic declarations, international declarations, prepaid travel costs and requests for payment, where the latter refers to expenses which should not be related to trips (think about representation costs, hardware purchased for work, etc.).

The data is anonymized in such a way that no TU/e internal IDs are visible in the final dataset, i.e. all identifiers are freshly generated. Furthermore, the amounts mentioned in the data are not exact amounts. However, adding declarations referring to the same travel permit and then comparing them to the original budget/estimate is still possible. Furthermore, on large enough samples of the data, the summed/averaged amounts should be roughly correct.

Staff members cannot be identified in the data. Instead, for all steps, the role of

the person executed the step is recorded. The resource recorded in the data is either the SYSTEM, a STAFF MEMBER or UNKNOWN, or, on occasion, the data is MISSING. Within each file we have logs. Each log consists of all the events that make up the path of that log (of that request).

The available datasets are:

- **Requests for Payment** (should not be travel related): 6,886 cases, 36,796 events.
- **Domestic Declarations**: 10,500 cases, 56,437 events.
- **Prepaid Travel Cost**: 2,099 cases, 18,246 events.
- **International Declarations**: 6,449 cases, 72151 events.
- **Travel Permits** (including all related events of relevant prepaid travel cost declarations and travel declarations): 7,065 cases, 86,581 events.

3 Organization Goal

The goal of my project is to be able to decrease organizational costs through data analysis. Using processes, libraries, and software to try to extract useful information so that we can find an answer to these questions:

- What is the throughput of a travel declaration from submission (or closing) to paying?
- Is there are difference in throughput between national and international trips?
- Where are the bottlenecks in the process of a travel declaration?
- Are there any double payments?
- Create a model to predict whether a trip will be overspent.

4 Knowledge Uplift Model

	Input	Analytics/Model	Type	Output
Step 1	5 event Logs	First analysis on process map with Disco and pm4py	Descriptive	Fields and events explanation
Step 2	5 event Logs	Temporal filtering	Prescriptive	Filtered event logs
Step 3	Step 2	Filtering logs on top variants	Prescriptive	Filtered event logs
Step 4	Step 2	Throughput for payed request	Descriptive	Statistics
Step 5	Step 2 Step 4	Calculated the time lost in each individual activity	Descriptive	Statistics
Step 6	Step 2 Step 4 Step 5	Bottlenecks research	Descriptive	Statistics
Step 7	Step 1 Step 2	Checking double payments	Descriptive	Statistics
Step 8	Step 2 Step 3	Find best models for each log	Prescriptive	Petri net and and frequencies for each log
Step 9	Step 8	Conformance Checking for each model	Descriptive	Anomalous cases for each log
Step 10	Step 2 Step 3	Predicting overspent declarations	Predictive	Predictive model

Figure 1: Knowledge Uplift Model

5 Project Solution

5.1 Temporal filtering

As indicated on the BPI challenge 2020 page, the process described is represented by the event logs attached starting from 2018, as before then only two units were logging data as per description.

```
data_log_from_2017 = {k:len(data_log[k]) for k in data_log.keys()}
```

```
from pm4py.algo.filtering.log.timestamp import timestamp_filter
data_log_from_2018 = {
    k:timestamp_filter.filter_traces_contained(data_df[k], "2018-01-01 00:00:00", "2022-07-01 23:59:59",
                parameters={timestamp_filter.Parameters.TIMESTAMP_KEY: "time:timestamp"}) for k in data_df.keys()
                }
```

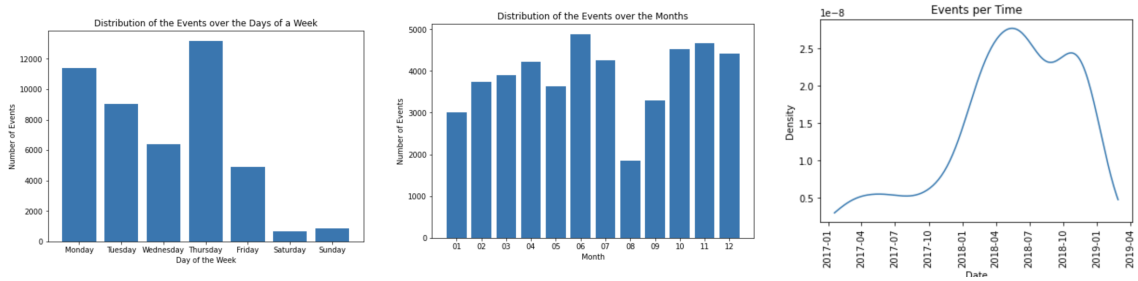
I then filtered the five event logs using PM4PY2, a piece of the procedure3 used for filtering. In the table below we can see the number of logs from 2017 and 2018, along with the difference.

LOG	Cases from 2017	Cases from 2018	Difference
Domestic Declarations	10500	8260	2240
International Declarations	6449	4952	1497
Permit Log	7065	5598	1467
Prepaid Travel Cost	2099	1776	323
Request For Payment	6886	5778	1108

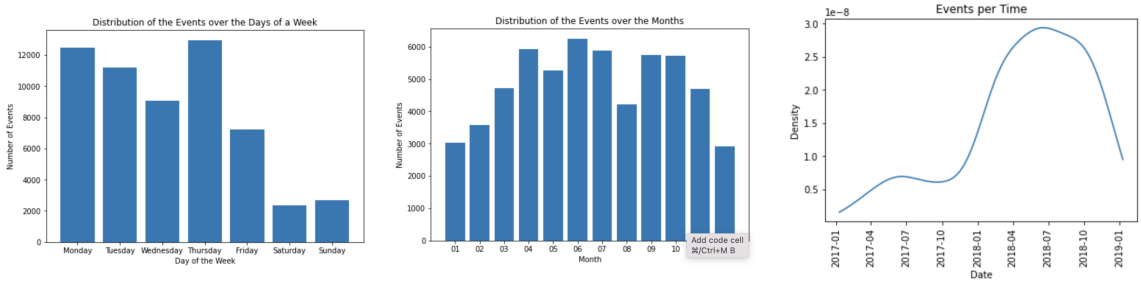
Figure 2: Temporal Filtering

5.2 Distribution of declarations over time

I calculated the onset of requests on days of the week, month, and throughout the year. Below are the different graphs:
Domestic Declarations:



International Declarations:



From the graphs we can see that there is no difference between the two types of travel, but this analysis can help us understand when and where more work is needed to be able to meet the work in the fastest way. So put more staff in the periods with more declarations and less in the periods with less declarations.

5.3 Fields and events analysis

In this section initially I want to understand in detail how the process is managed and through which events it is carried out. In this case I used Disco4, an academically licensed software that allows you to easily apply filters, analyze different variants, look at different field statistics, etc. The image below shows a trace for the variants with multiple cases of the DomesticDeclarations dataset, it is not shown to avoid redundancy but most of the processes that come to an end (ie where employees receive the refund) start with a “Declaration SUBMITTED by EMPLOYEE” event (in some cases preceded by two events: “Start Trip” and “End Trip”; most successful refund processes end with “Request Payment” followed by “Payment Handled”.

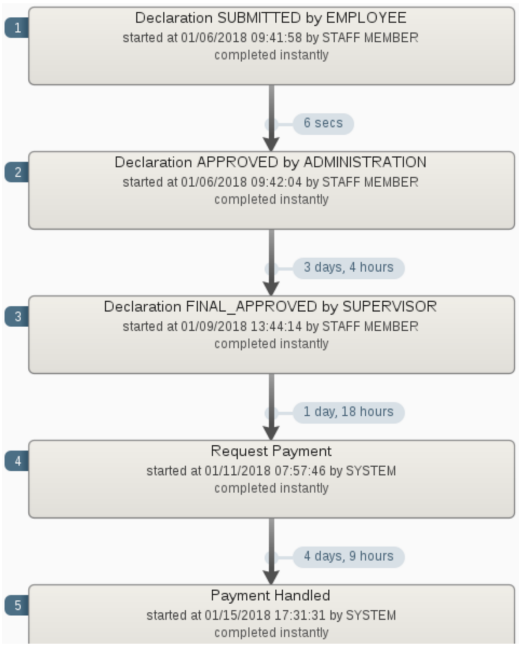


Figure 3: Events of one case of most popular variants in DomesticDeclarations.

In order to check multiple payments, it is necessary to understand which field corresponds to the amount of money requested in a declaration or request for reimbursement.

Also through Disco I noticed that the events in their respective datasets contain the field:

- Domestic Declarations → “RequestedAmount”;
- International Declarations → “RequestedAmount”;

- Permit Log → “TotalDeclared”;
- Prepaid Travel → “RequestedAmount”;
- Request For Payment → “RequestedAmount”.

These fields are populated to 0 when the request is unsuccessful.

5.4 Throughtput

At this stage, I calculated the average time over the entire procedure by comparing the duration between domestic and international trips, showing a significant difference between the two results.

```
import pm4py

all_case_durations = {
    k:pm4py.get_all_case_durations(data_log_from_2018[k]) for k in data_df.keys()
                }
max_case_durations = {
    k:sum(all_case_durations[k])/len(all_case_durations[k]) for k in data_df.keys()
                }
```

```
{'DomesticDeclarations': 11.5,
 'InternationalDeclarations': 76.4,
 'PermitLog': 84.1,
 'PrepaidTravelCost': 36.6,
 'RequestForPayment': 12.3}
```

5.5 More analysis on Throughput

I noticed a big difference between domestic and international flights so I went to do more in-depth analysis. For each case I want to isolate the events between "Declaration SUBMITTED by EMPLOYEE" and "Payment Handled" via this pm4py function (I only show the functions applied to domestic travel to avoid redundancy):

```
data_df = log_to_df(data_log_from_2018["DomesticDeclarations"])
filtered_log = pm4py.filter_between(data_df, "Declaration SUBMITTED by EMPLOYEE", "Payment Handled")
```

Now I have calculated the average duration of each payment request within the two states described above. So we can see that the average duration from request to payment for domestic trips and it is also possible to answer the question on the difference in throughtput between national (Domestic) and international declarations, they have a very similar average time in days.

```
{'DomesticDeclarations': 11.624,
 'InternationalDeclarations': 15.130}
```

The difference is not much so I decided to go deeper. So I calculated the average lead time for each individual activity that is present between the two phases listed earlier. This allows me to better analyze where there is a large waste of time by increasing the fineness of my analysis.

```
from pm4py.objects.log.util import interval_lifecycle

enriched_log = interval_lifecycle.assign_lead_cycle_time(df_to_log(filtered_log))
enriched_log
data_df = log_to_df(enriched_log)
df_new = data_df[['concept:name', '@@approx_bh_overall_wasted_time']]
df_new.groupby('concept:name', as_index=False)['@@approx_bh_overall_wasted_time'].mean()
```

Result:

	MEAN		
Logs	SUBMITTED by EMPLOYEE to Payment Handled	Request Payment to Payment Handled	FINAL APPROVED by SUPERVISOR to Request Payment
Domestic Declarations	10.35	3.45	3.24
International Declarations	12.38	3.41	3.26

Figure 4: Average days to pass from one phase to another, indicated in the columns.

5.6 Bottlenecks

I noticed a very large expenditure of time in the final parts of the process. So I decided to isolate other cases, specifically from "Request Payment" to "Payment Handled" and from "Declaration FINAL_APPROVED by SUPERVISOR" to "Request Payment".

In order to speed up the process it is interesting to analyze in detail if there are any bottlenecks during the various stages of the process. The code snippet shows the code to identify the 25th percentile, 50th percentile, 75th percentile and the maximum time, expressed in days.

```
d_log = {n:get_logs(f) for n,f in zip(list_name, lis\t_xes)}
for k in d_log.keys():
    list_res = []
    for trace in d\_log[k]:
        start = None
        end = None
        for event in trace:
            if event['concept:name'] == 'Declaration SUBMITTED by EMPLOYEE':
                start = event['time:timestamp']
            elif event['concept:name'] == 'Payment Handled':
                end = event['time:timestamp']
                break
        if start is not None and end is not None:
            list\_res.append((start, end, trace.attributes['concept:name']))

```

```
    print(k,"\n", pd.Series([(end-start).total_seconds()/60/60/24 for (start, end, _) in list_res]).describe()
    [["25%","50%", "75%"]])
```

```
DomesticDeclarations
 25%      5.919103
50%       7.268623
75%      11.257731
```

```
InternationalDeclarations
 25%      6.182905
50%       9.096863
75%      14.118640
```

```
PermitLog
 25%      6.151730
50%       9.073310
75%      14.109491
```

"Request Payment" to "Payment Handled"

```
DomesticDeclarations
 25%     2.318148
50%     3.240475
75%     4.280729
```

```
InternationalDeclarations
 25%     2.307937
50%     3.218652
75%     4.259711
```

```
PermitLog
 25%     2.310336
50%     3.223032
75%     4.264769
```

We can see that the general slowest sequence is Request Payment to Payment Handled. Ignoring the outliers present in the maximums, we can see that the DomesticDeclarations have the same timing as InternationalDeclaration despite the latter, we have a more complicated approval process that also requires the passage of Permit.

5.7 Checking double payments

Now we approach the economic part. We focus on finding solutions that save us money. I want to check for duplicate payments in the same trace. I started by writing a small script to verify the presence or absence of trace with this behavior, I report the code with its output.

```
dict_decl = set()
double_time = []
for trace in data_log["PermitLog"]:
  for event in trace:
    if event['concept:name'] == 'Payment Handled':
      dec = trace.attributes['concept:name']
      if dec in dict_decl:
        double_time.append(dec)
      else:
        dict_decl.add(dec)

set_double = list(set(double_time))
print(set_double.count)
#print "1017"
```

Since the PermitLog shows 1017 traces with multiple Payment Handled events, I specifically investigated this log to understand how much money it was. The suspicious traces are collected and subsequently analyzed. I then added the LOSS value for each line to get an amount equivalent to 4478912.

```
res_dict = {k:[] for k in set_double}
for trace in data_log["PermitLog"]:
    if (trace.attributes["concept:name"]) in set_double:
        for event in trace:
            if event['concept:name'] == 'Payment Handled':
                res_dict[trace.attributes["concept:name"]].append(trace.attributes["TotalDeclared"])
                #ogni quando veniva rilasciato un altro pagamento per poi fare una media - poco utile
                #res_dict[trace.attributes["concept:name"]].append(trace.attributes["time:timestamp"])
tmp = []
#print(res_dict)
for k, v in res_dict.items():
    tmp.append([k, round(v[0]), len(v)])
#print(tmp)
df_res = pd.DataFrame(tmp, columns=["ID", "AMOUNT", "TIMES"])
df_res["LOSS"] = df_res["AMOUNT"].to_numpy()*(df_res["TIMES"].to_numpy()-1)
df_res.sort_values("LOSS", ascending=False).head(10)
```

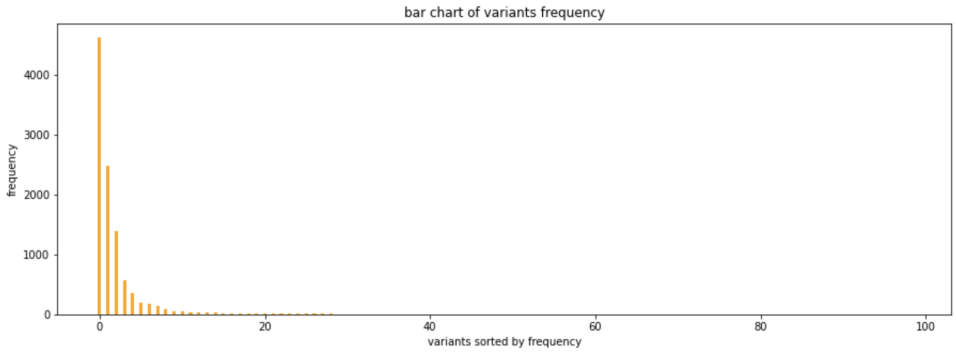
This table shows the top 10 traces where the most money was lost

	ID	AMOUNT	TIMES	LOSS
378	travel permit 54518	1458579	2	1458579
1183	travel permit 22776	12232	14	159016
788	travel permit 24054	9246	12	101706
206	travel permit 53343	4811	14	62543
1207	travel permit 73562	3375	17	54000
644	travel permit 57181	4652	9	37216
15	travel permit 33319	6177	6	30885
167	travel permit 45672	6626	5	26504
677	travel permit 66595	5043	6	25215
150	travel permit 28756	4133	7	24798

5.8 Filtering logs on top k variants

In this section I have filtered the log files to get the top-k popular variants. Below there is shown the code to perform the filter and the relative coverage percentages with respect to the total traces and the number of traces kept.

In addition, we can see a graphic representation on the distribution of variants. (Only the one of domestic trips so as not to create redundancy)




```
import pm4py
log_top = {k:pm4py.filter_variants_top_k(data_log[k], 5) for k in data_log.keys()}
log_top['InternationalDeclarations'] = pm4py.filter_variants_top_k(data_log['InternationalDeclarations'], 50)
log_top['PermitLog'] = pm4py.filter_variants_top_k(data_log['PermitLog'], 90)
log_top['PrepaidTravelCost'] = pm4py.filter_variants_top_k(data_log['PrepaidTravelCost'], 30)

for k,v in log_top.items():
    print (f"{k} --> {round(100*(len(v)/len(data_log[k])))}% - {len(v)}/{len(data_log[k])}")
```

DomesticDeclarations –> 90% - 9403/10500
InternationalDeclarations –> 76% - 4912/6449
PermitLog –> 71% - 4996/7065
PrepaidTravelCost –> 86% - 1808/2099
RequestForPayment –> 88% - 6039/6886

We were able to reduce the number of traces while maintaining excellent coverage.

5.9 Best model for each log

We now want to find a good model that has a good trade-off between the metrics seen in class, namely:

- **Replay Fitness**

- **Precision**
- **Generalization**
- **Simplicity**

I used the **three miners** available in the PM4PY package to get the full picture:

- **Alpha Miner**
- **Inductive Miner**: Infrequent and Inductive Miner directly-follows, trying different values for NOISE parameter.
  For NOISE I tried this values {0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1}.
- **Heuristic Miner**: trying different combinations for the hyperparameters. I tried to change the following parameters:
  • DEPENDECY TRESH 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1
  • MIN ACT COUNT 1, 20, 100, 200, 300
  • MIN DFG OCCURRENCES 1, 20, 100, 200, 300

I wrote some code to test the different combinations of parameters and patterns and then put the results in a DataFrame. I then filtered the results of the table obtained for each log with respect to the metrics indicated above to obtain the best model with the best parameters, finally I created a Petri net for each log. The best results obtained for each log file are shown below.

	MODEL	PARAMETERS	FITNESS	PRECISION	GENERALIZATION	SEMPLICITY	LOG
2	IMf	[0.0]	92.24	0.92	0.96	0.87	PrepayedTravelCost
14	HEU	[0.1, 1, 1]	100.00	0.99	0.98	0.85	DomesticDeclaration
14	HEU	[0.1, 1, 1]	100.00	0.99	0.97	0.85	RequestForPay
17	HEU	[0.1, 1, 200]	94.68	0.92	0.97	0.84	InternationalDeclaration
17	HEU	[0.1, 1, 200]	92.22	0.89	0.97	0.83	PermitLog

To make the visualization easier, in the report I preferred to attach the heuristic nets, both the Petri nets and the heuristic nets of each log file are on Colab.

For each log I make a comparison between before taking the top variations and after

5.10 Domestic Declarations

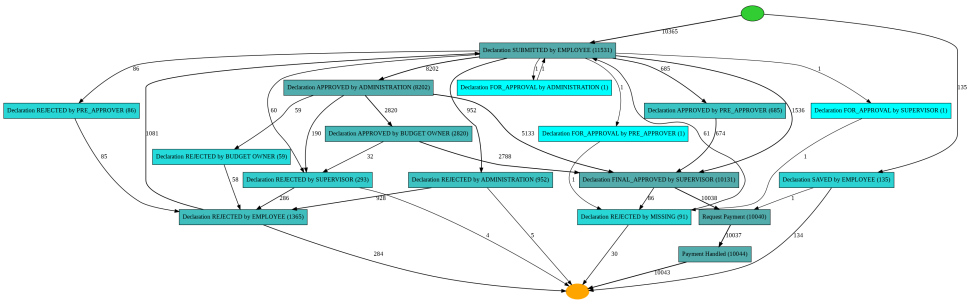
The heuristic net related to Domestic Declarations we can see that the model has learned the process well, this net covers the various types that can exist in this type of process:

• Everything goes well, it starts with the declaration of the employee who receives the various approvals and arrives at the payment

• After the declaration of the employee you get a reject from the administration followed by reject from the employee, here are two options: 1. the process ends, 2. the employee creates a new declaration and the process starts again.

In this process it is also evident that the approval of the budget owner is optional and for the cases analyzed, about two thirds of the cases do not need this approval. I also

did a comparison using all logs and logs filtered by variants. We can see that the net is much more confusing and not very useful.



5.11 International Declarations

The Heuristic net related to International Declarations is less precise, we immediately notice that reimbursement rejects are not covered because from the rejected declaration stage we then have only 2 options:

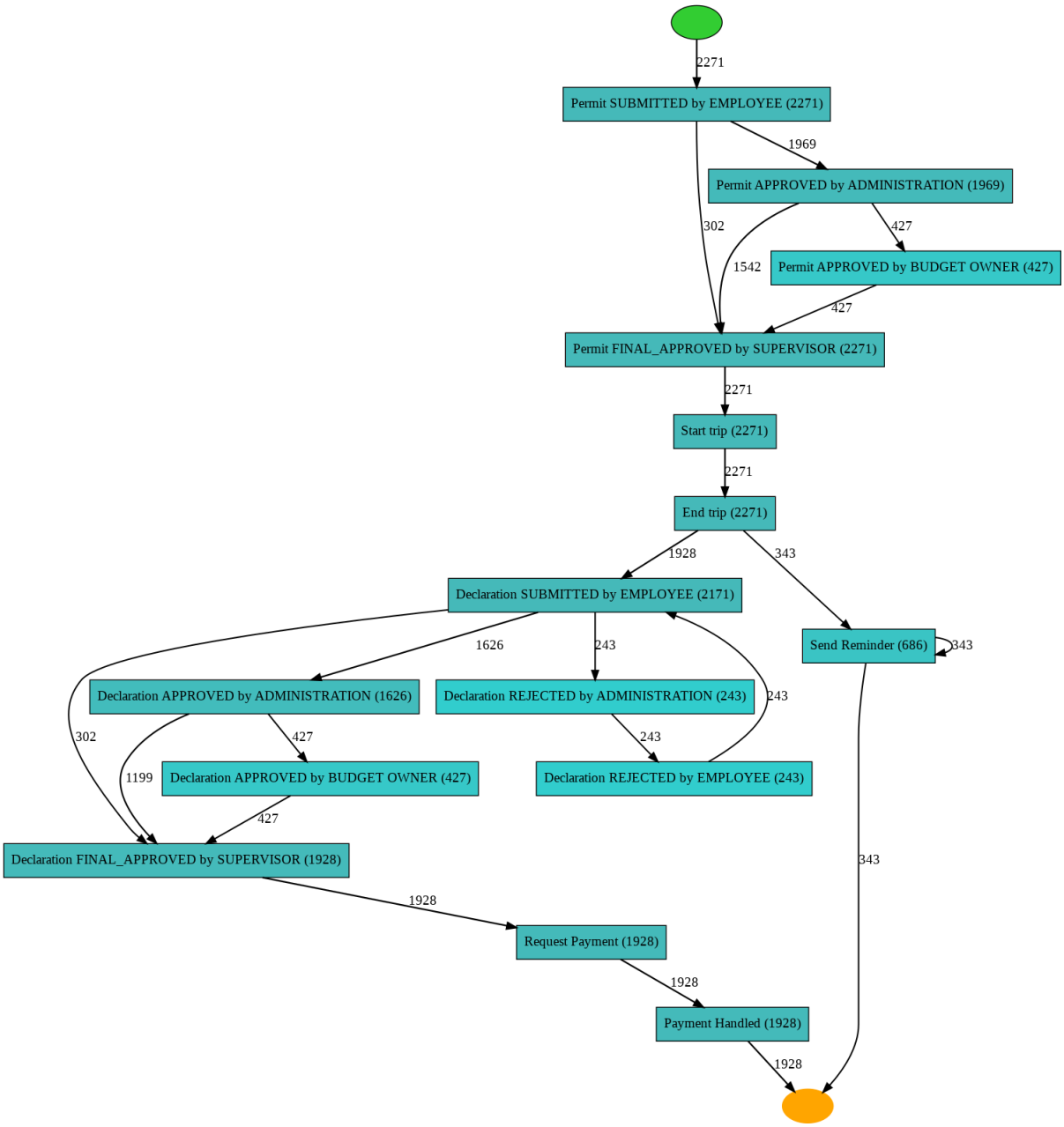
• Re-execute the request by the employee which can be rejected by going in the loop

• Accepted and yes advance until payment.

Unlike Domestic Declaration, a permit procedure takes place before the various declarations. Permit also follows steps very similar to the declaration, i.e. submitted by employee to then receive the various approvals. Also in this case we note that the approval of the budget owner is optional.

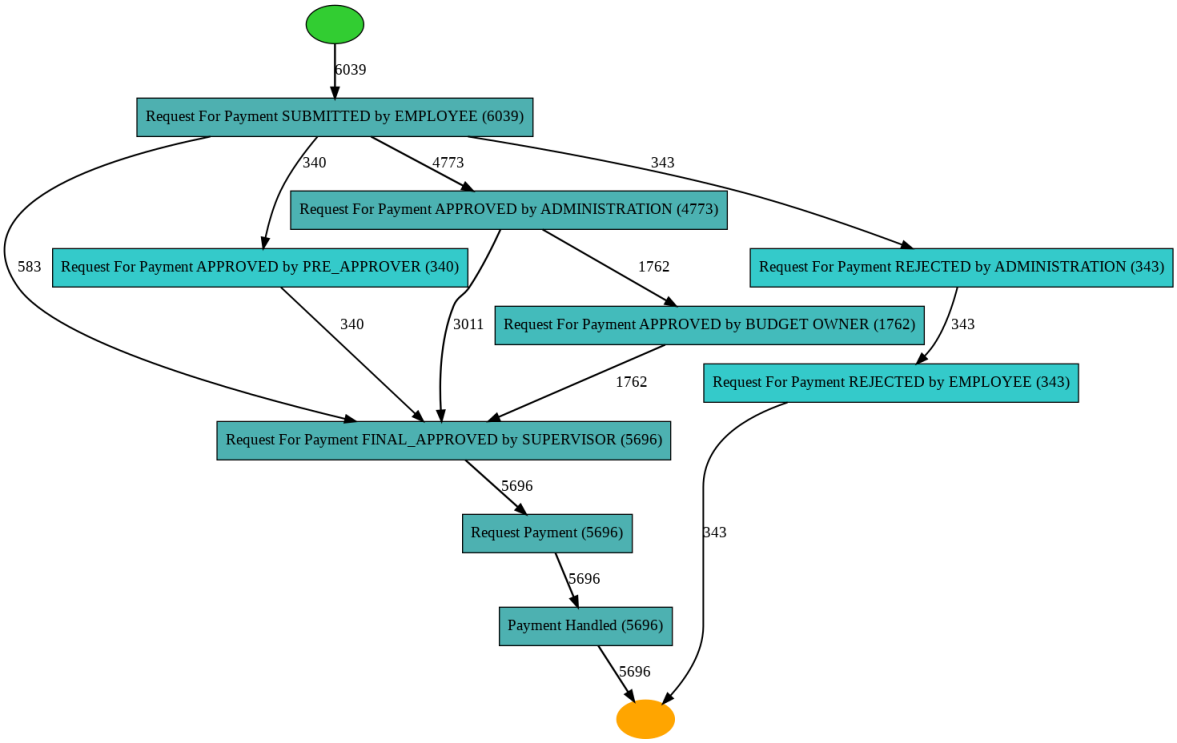
5.12 Permit Log

The heuristic net related to Permit Log. The net is a bit more complicated than the previous ones but still quite simple, very similar to the behavior in International Declarations with the addition of the Send Reminder event, which loops with itself and then ends. Here, too, the Budget Owner in the approvals remains optional.



5.13 Request For Payment

For Request For Payment we use a Heuristic net. Process very similar to Domestic Declaration, it seems to differ only in the prefix of the events that become “Request For Payment”. Also here, the Budget Owner in the approvals remains optional.



6 Conformance Checking

Conformance checking is a techniques to compare a process model with an event log of the same process. The goal is to check if the event log conforms to the model, and, vice versa. In PM4PY, two fundamental techniques are implemented: token-based replay and alignments.

6.1 Domestic Declaration and Request For Payment

For Domestic Declarations and Request For Payment there are no anomalous cases with respect to the two Petri nets found in the Best Model section.

```
ALIGNMENTS
Number of traces 7786
Number of anomalous traces  0
Number of normal traces  7786
Percentage of anomalous traces 0.0 %
Odds of anomalous traces 0.0
```

6.2 International Declarations

For International Declarations I find 140 anomalous cases and analyzing these traces I realized that all non-aligned cases are those that start from “Start trips” and then follow the classic inter of submissions and approvals, a case that perhaps shouldn’t be possible?

```
ALIGNMENTS
Number of traces 2632
Number of anomalous traces  140
Number of normal traces  2492
Percentage of anomalous traces 5.319148936170213 %
Odds of anomalous traces 0.06
```

6.3 Permit Log

For Permit Log I find 166 anomalous cases, analyzing these traces I realized that the cases not covered are those that have “Request For Payment SUBMITTED by EMPLOYEE” to “Request Payment”. The question I ask myself is, “Is it correct that there are cases that come in demand for payment without the various approvals of higher-level employees?”

```
ALIGNMENTS
Number of traces 2135
Number of anomalous traces  166
Number of normal traces  1969
Percentage of anomalous traces 7.775175644028103 %
Odds of anomalous traces 0.08
```

6.4 Prepaid Travel Cost

For Permit Log I find 93 anomalous cases, analyzing these traces I realized that all non-aligned cases are those that start “Request For Payment SUBMITTED by EMPLOYEE”, practically employees ask for money before submitting a permit? In this case

```
ALIGNMENTS
Number of traces 1199
Number of anomalous traces  93
Number of normal traces  1106
Percentage of anomalous traces 7.756463719766472 %
Odds of anomalous traces 0.08
```

I have also analyzed a particular case via Disco and actually it seems that the employees start asking for money from the company.

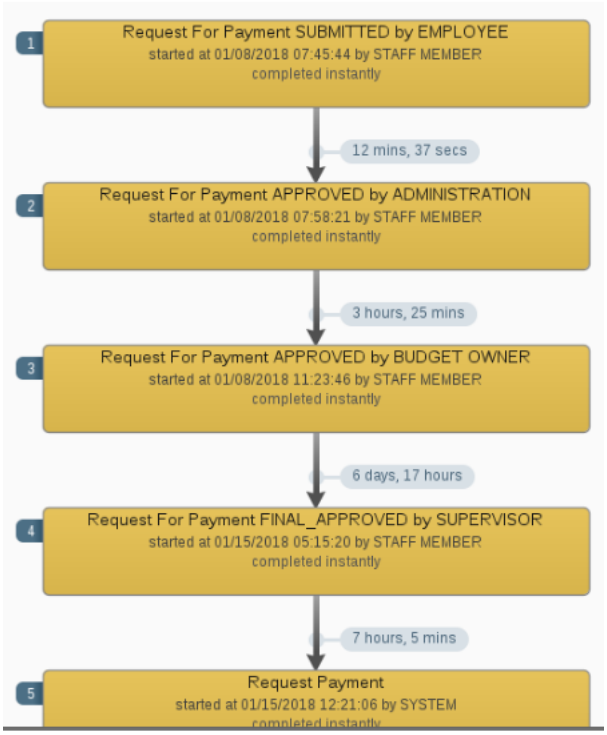


Figure 5: Anomalous cases for PrepaidTravelCost, request 185740.

7 Predicting overspent declarations

In this section I initially try to carry out an analysis on the Permit Log and International Declarations columns in order to create a single table through a join, then extract the columns relating to the duration of the requests, the travel time and various available information relating to the budget required for the creation of a prediction model to try to predict whether or not the request will incur an overspent.
International declarations The useful columns are:

- “case: concept: name”: key of the table;
- “case: Permit RequestedBudget”: requested budget;
- “case: Permit ID”: key of the PermitLog table.

To these fields I have added a column that contains, for each declaration, the time required for approval.
PermitLog The useful columns are:

- “case: concept: name”: key of the table;
- “case: RequestedBudget”: requested budget;
- “case: Overspent”: flag for budget overrun;
- “case: OverspentAmount”: amount of additional money spent;
- “case: TotalDeclared”: total of the declaration

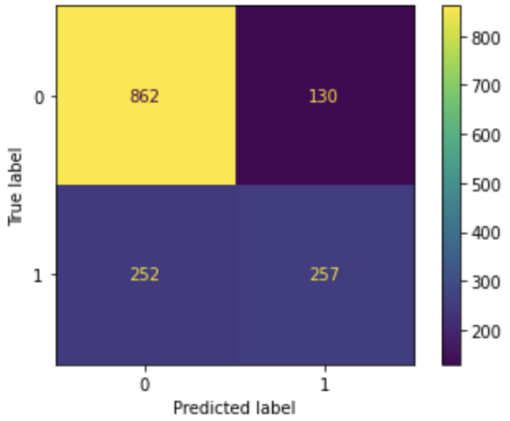
Also in this case I added a column that contains, for each declaration, the time required for approval. I also added a column that contains the duration of the trip, i.e. the timestamp difference (in days) between Start Trip and End Trip.

Then I created the table through the pandas merge function and printed the correlations between the various attributes. As expected, there is a strong correlation between the “case: Overspent” flag and the “case: OverspentAmount” attribute, for this reason I will not use OverspentAmount as the model input as it would result in a bias.

The attributes used are therefore “case: RequestedBudget”, “case: Permit RequestedBudget”, “case: TotalDeclared” and three timestamps in days, that is: declaration approval time, permit approval time, time between departure and return. To create a train and a test set I used train test split from sklearn, it takes the feature columns and the label (flag overspent) as input and splits it using 75% of the data as a train and 25% as a test.

For my analysis I tested some models, below you can see their results:

RandomForestClassifier

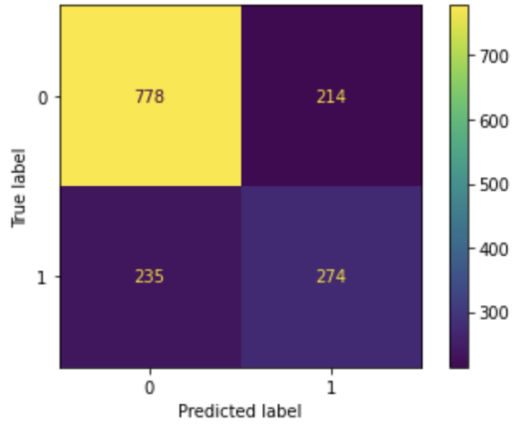



```
Accuracy:  0.7455029980013325
Recall:  0.5049115913555993
Precision: 0.6640826873385013
CL Report:               precision    recall  f1-score   support

       False       0.77      0.87      0.82       992
        True       0.66      0.50      0.57       509

    accuracy                           0.75      1501
   macro avg       0.72      0.69      0.70      1501
weighted avg       0.74      0.75      0.74      1501
```

KNeighborsClassifier(3)

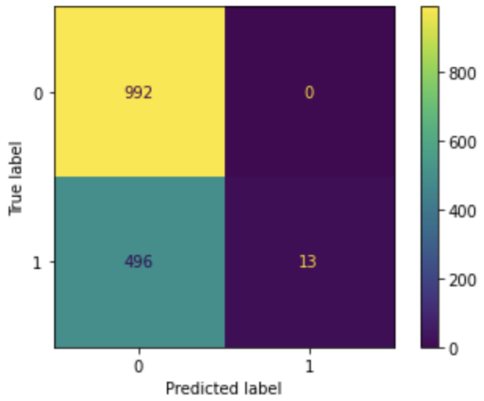



```
Accuracy:  0.7008660892738174
Recall:  0.5383104125736738
Precision: 0.5614754098360656
CL Report:               precision    recall  f1-score   support

       False       0.77      0.78      0.78       992
        True       0.56      0.54      0.55       509

    accuracy                           0.70      1501
   macro avg       0.66      0.66      0.66      1501
weighted avg       0.70      0.70      0.70      1501
```

SVC(gamma=2, C=1)




```
Accuracy:  0.6695536309127248
Recall:  0.025540275049115914
Precision: 1.0
CL Report:               precision    recall  f1-score   support

       False       0.67      1.00      0.80       992
        True       1.00      0.03      0.05       509

    accuracy                           0.67      1501
   macro avg       0.83      0.51      0.42      1501
weighted avg       0.78      0.67      0.55      1501
```

SVC(kernel="linear", C=0.025)
Available in Colab
GaussianProcessClassifier(1.0 * RBF(1.0))
Available in Colab
DecisionTreeClassifier(max_depth=5)
Available in Colab
MLPClassifier(alpha=1, max_iter=1000)
Available in Colab
AdaBoostClassifier()
Available in Colab
QuadraticDiscriminantAnalysis()
Available in Colab

In the best confusion map we can see the performances indicate that, on average, the model correctly predicts about three out of four cases of label overspent. By placing more emphasis on the True label (i.e. when you go over budget) performance drops to about seven out of ten correct predictions.

8 Technologies used

The technologies I used for my project include:

- Google Colab: used to develop my analysis and solutions.
- PM4PY: python library that contains the functions used for analysis. All the functions used can be found in its documentation.
- Disco: Disco is a great process mining tool that simply works: it is able to deal with large event logs and complex models and conversion and filtering are made easy. Performance metrics are shown in a direct and intuitive manner and the history can be animated on the model.

Conclusion

In this report, I conducted analyses on several aspects. First, I tried to figure out what the most common variants were (also helping me with Disco) and patterns to help me and to analyze the log workflow. Here I found that some requests take a long time to complete. Also, the average time is not high and there is not a big difference between different trip types. One solution I have come up with to decrease the wasted time may be to decrease the steps within an event, so as to standardize the process making it smoother and faster. Also alerts are essential to ensure that the process goes on.
The second analysis I did is on the money lost. As you can see in the report, I found that many claims are paid multiple times, causing the organization to lose a lot of money. So you need software that keeps this information up to date so that it is no longer allowed.
Last Analysis is about creating a predictive model to estimate whether a trip will need new logs. It is not super accurate, but with more data and perhaps a more complex learning method for regression we can achieve good results.