Homework 1

Miriam Wagner
373045

18. Mai 2018

Question 1

I used for solving this exercise Matlab (with my own written code no packages). For the distance I used euclidian distance.
My clustering algorithm calculates for every Datavector the distances to every centroid and then checks which centroid is closed. The vector is add to the closest cluster. When all datavectors are put in a cluster the new cluster centroids are calculated.
My abort criterion is the change between the centroids. If they still change I will apply the algorithms again. Because of the computer accuracy I do not use 0, but 10^{-15}. I also let the algorithm stop after 3 rounds, but it is already would end after 4 rounds.
After the first round all data is clustered in cluster 2 and 3. I decided to let be the first centroid $(-1000, -1000, -1000)$. Than it does not influence the result anymore. But two other options would be

1. Set the centroid to 0 or 1 or do not change it
2. random restart of just this or all cluster centroids

When I would have to choose another option I would use a restart for improvement, but it was not said, that it has to be 3 clusters, so I decided it would be okay like this and it takes less time like this.

Tabelle 1: Distances first round

distance_to_cluster_1	distance_to_cluster_2	distance_to_cluster_3
43000,00002	3000,002694	23000,00899
35000,07642	5000,486525	15000,09671
26888,00021	13112,00189	6888,039561
44000,00413	4000,028178	24000,00035
40000,00664	20,15113235	20000,002
47500,00003	7500,000631	27500,00685
31000,02992	9000,088284	11000,02773
45000,0265	5000,212648	25000,02102
37000,00023	3000,000006	17000,00789
15000,23523	25000,12893	5000,409169
45000,00443	5000,027278	25000,00028
42500,00031	2500,000209	22500,00527
44000,00319	4000,02221	24000,00046
43500,00003	3500,002355	23500,00885
25000,03222	15000,04597	5000,051895
40000,00014	7,034833476	20000,01362
42200,02875	2200,477885	22200,02037
41500,00014	1500,016496	21500,01267
33000,00058	7000,000319	13000,00815
47000,00118	7000,004581	27000,0024

Tabelle 2: centroids after first round

centroid_1	centroid_2	centroid_3
-10000	8752,941176	27704
-10000	7,779844647	13,629704
-10000	22,82352941	44

Tabelle 3: Distances second round

distance_to_cluster_1	distance_to_cluster_2	distance_to_cluster_3
22116,19085	1753,037892	20704,03917
28758,56529	6247,302602	12704,04405
36006,45169	14359,07461	4592,203257
21365,89781	2752,952894	21704,01257
24510,81085	1247,073587	17704,00916
18878,30691	6252,966011	25204,03083
32284,64122	10247,08985	8704,002232
20652,19999	3753,0757	22704,00565
27003,89625	4247,084395	14704,04516
47192,82789	26247,14247	7296,135236
20631,35796	3752,941811	22704,00875
22505,10872	1253,018169	20204,03122
21367,51278	2752,941595	21704,01232
21734,0764	2253,017414	21204,03843
37765,65939	16247,07449	2704,012108
24496,12265	1247,240632	17704,05272
22765,44019	953,4550448	19904,00236
23287,55072	253,8360288	19204,0486
30483,445	8247,069583	10704,05649
19220,80594	5752,946555	24704,01792

$$\text{cluster} = \{2, 2, 3, 2, 2, 2, 3, 2, 2, 3, 2, 2, 2, 2, 3, 2, 2, 2, 3, 2\}$$

Where the order of the vector is the order of the data.

Tabelle 4: centroids second round

centroid_1	centroid_2	centroid_3
-10000	8112,5	25528
-10000	7,029701813	15,1678105
-10000	21,4375	44,25

Tabelle 5: Distances third round

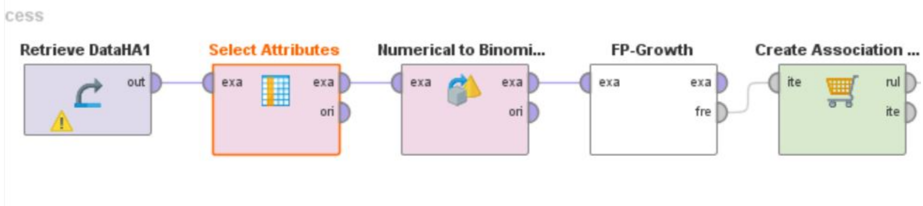
distance_to_cluster_1	distance_to_cluster_2	distance_to_cluster_3
22116,19085	1112,627498	18528,04545
28758,56529	6887,73384	10528,0504
36006,45169	14999,51298	2416,39972
21365,89781	2112,51454	19528,01534
24510,81085	1887,5148	15528,01088
18878,30691	5612,522945	23028,03511
32284,64122	10887,53297	6528,001674
20652,19999	3112,67816	20528,00509
27003,89625	4887,517724	12528,05539
47192,82789	26887,5854	9472,102143
20631,35796	3112,502093	20528,01042
22505,10872	612,6238331	18028,03664
21367,51278	2112,50014	19528,0146
21734,0764	1612,589208	19028,04449
37765,65939	16887,51723	528,0438334
24496,12265	1887,603135	15528,0622
22765,44019	314,2232448	17728,00213
23287,55072	388,0020635	17028,05672
30483,445	8887,50778	8528,074443
19220,80594	5112,504053	22528,02049

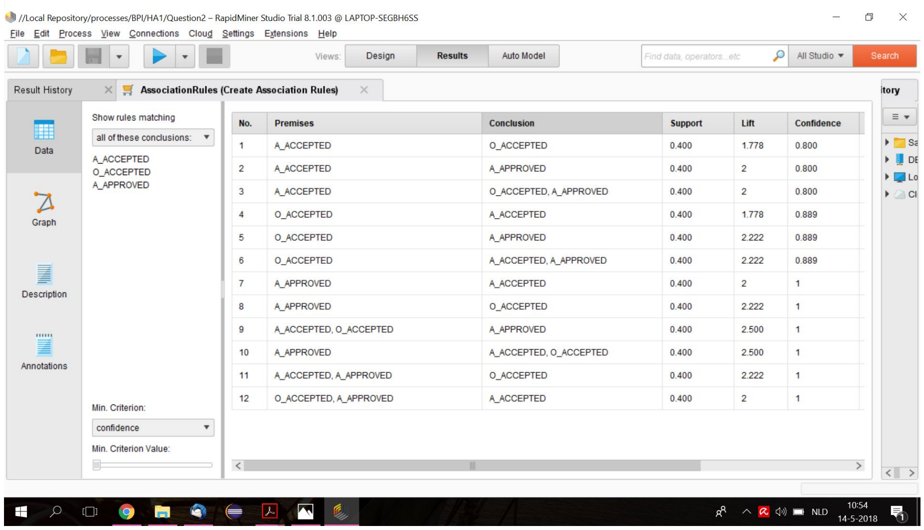
Tabelle 6: centroids third round

centroid_1	centroid_2	centroid_3
-10000	7520	23822,4
-10000	7,497431933	12,1369984
-10000	22,06666667	37,8

Question2

The process is built up like in the screenshot. After I imported the data I selected the attributes I am assigned to consider (**Select Attributes**). Then the numerical values are translated to binomial values, because this is needed for the association rules. (**Numerical to Binomial**) RapidMiner expectes a FrequentItemSet for the **Create Association Rule**, so before I could use that I also had to use **FP-growth**.

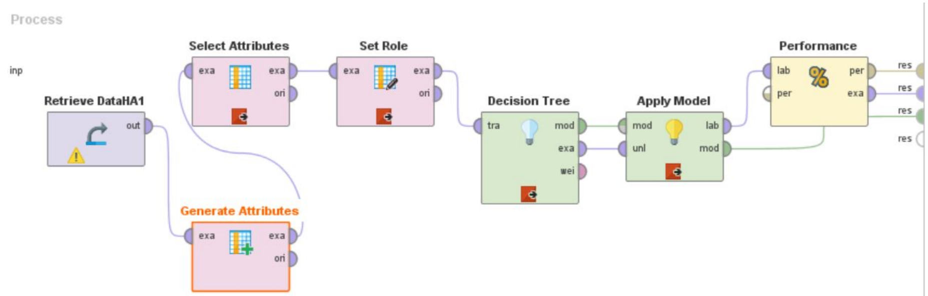




No.	Premises	Conclusion	Support	Lift	Confidence
1	A_ACCEPTED	O_ACCEPTED	0.400	1.778	0.800
2	A_ACCEPTED	A_APPROVED	0.400	2	0.800
3	A_ACCEPTED	O_ACCEPTED, A_APPROVED	0.400	2	0.800
4	O_ACCEPTED	A_ACCEPTED	0.400	1.778	0.889
5	O_ACCEPTED	A_APPROVED	0.400	2.222	0.889
6	O_ACCEPTED	A_ACCEPTED, A_APPROVED	0.400	2.222	0.889
7	A_APPROVED	A_ACCEPTED	0.400	2	1
8	A_APPROVED	O_ACCEPTED	0.400	2.222	1
9	A_ACCEPTED, O_ACCEPTED	A_APPROVED	0.400	2.500	1
10	A_APPROVED	A_ACCEPTED, O_ACCEPTED	0.400	2.500	1
11	A_ACCEPTED, A_APPROVED	O_ACCEPTED	0.400	2.222	1
12	O_ACCEPTED, A_APPROVED	A_ACCEPTED	0.400	2	1

I would pick the rule $\{A\_ACCEPTED, O\_ACCEPTED\} \Rightarrow \{A\_APPROVED\}$ and $\{A\_APPROVED\} \Rightarrow \{A\_ACCEPTED, O\_ACCEPTED\}$, because they have the highest lift, confidence and support. When you have a closer look you will see that the sets are probably of the rules have a back- and forth relationsip, so they are symmetric. It is an interesting behaviour. Such that you could summarize it in one rule.
Then you could look for the next best rules, where lift, support and confidence is the highest.

Question 3



The process
first changes the numerical attribute TotalActivites to a nominal one by **Generate Attributes** and **Select Attributes**. The generating consideres the old TotalActivites and uses the rule that all data, where TotalActivites is lower or the same as 40 it is assigned to "Low", otherwise "High". The **Set Role** gives the new attribute role label, so RapidMiner knows what should have to be the outcome of the Decision Tree. **Decision Tree** generates the decision Tree. Then **Apply Model** for **Performance** checking. The output is then the model and the performance of the model on the data.
The resulting decision tree is

If you check the Confusionmatrix

accuracy: 100.00%

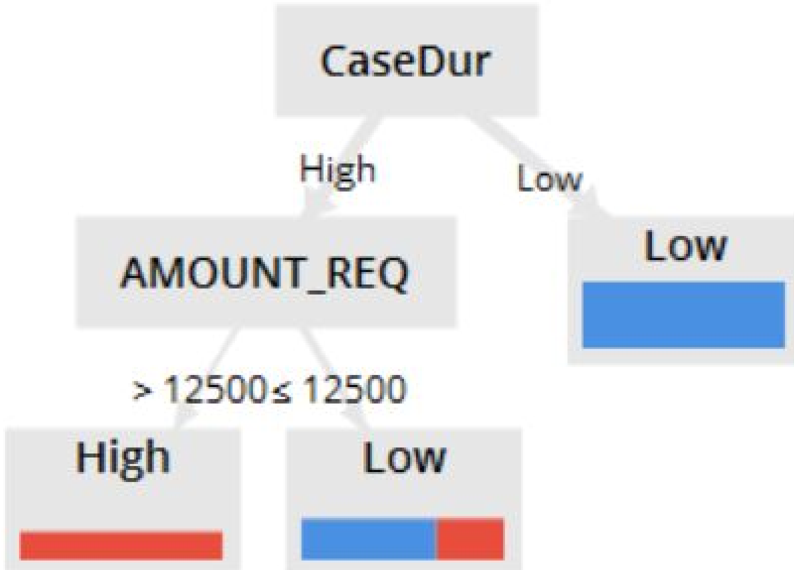
	true Low	true High	class precision
pred. Low	14	0	100.00%
pred. High	0	6	100.00%
class recall	100.00%	100.00%	

you see, that this decision tree classifys the data perfectly. So you can predict by just knowing the case duration the nominal outcome for total activities. If the case duration is higher, than also the total activities are high. This seems to be logical, if you have to do a lot this takes most of the times longer and otherwise around, if you do not need long you mostly did not do a lot of different things in the time.
For completeness I checked also what happens for different setting:

1. Changing case duration to nominal with high low (low, if caseduration<=1/6/15)

2. Changing case duration to nominal with high,low,middel ((low, if caseduration<=6, middel if 6<caseduration<=15,otherwise high))

The first case gives in all versions the same slightly different solution from the first one, but also it gets worse.
The second version again gives a tree just deciding with the case duration.



I also treid what happens without case duration, but decided that this is not helpful for the exercise, because we have the information. In my opinion it is like said above a dependency bewtween duration and activity.

Question 4

1.

$L1 = [\langle a,b,e,f\rangle, \langle a,b,e,c,d,b,f\rangle, \langle a,b,c,e,d,b,f\rangle, \langle a,b,c,d,e,b,f\rangle, \langle a,e,b,c,d,b,f\rangle]$

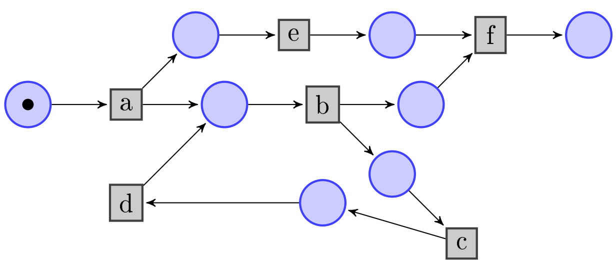
The $\alpha-$Algorithm gives the following:

$$T_L = \{a,b,c,d,e,f\}$$
$$T_I = \{a\}$$
$$T_O = \{f\}$$

	a	b	c	d	e	f
a	#	→	#	#	→	#
b	←	#	→	←	||	→
c	#	←	#	→	||	#
d	#	→	←	#	||	#
e	←	||	||	||	#	→
f	#	←	#	#	←	#

$X_L = \{(\{a\},\{b\}), (\{a\},\{e\}), (\{b\},\{c\}), (\{b\},\{f\}), (\{b\},\{f,c\}), (\{c\},\{d\}), (\{d\},\{b\}), (\{e\},\{f\}),$

$Y_L = \{(\{a\},\{b\}), (\{a\},\{e\}), (\{d\},\{b\}), (\{c\},\{d\}), (\{b\},\{f,c\})\}$



This model is sound, because it is bounded and live.

2.

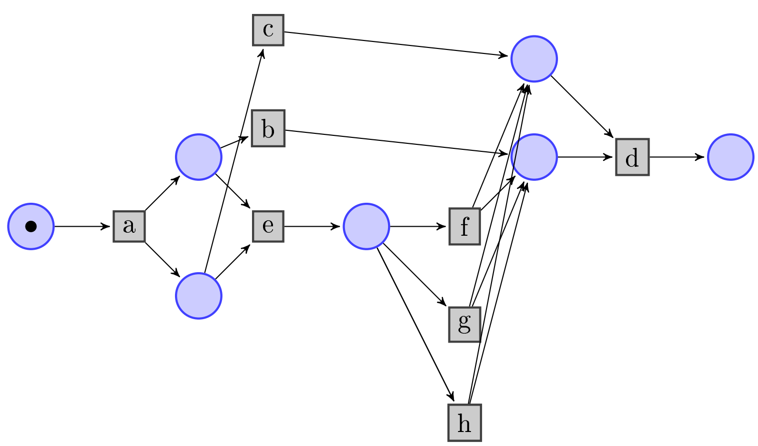
$$L2 = [\langle a,b,c,d\rangle, \langle a,c,b,d\rangle, \langle a,e,f,d\rangle, \langle a,e,g,d\rangle, \langle a,e,h,d\rangle]$$

The $\alpha-$Algorithm gives the following:

$$T_L = \{a,b,c,d,e,f,g,h\}$$
$$T_I = \{a\}$$
$$T_O = \{d\}$$

	a	b	c	d	e	f	g	h
a	#	→	→	#	→	#	#	#
b	←	#	||	→	#	#	#	#
c	←	||	#	→	#	#	#	#
d	#	←	←	#	#	←	←	←
e	←	#	#	#	#	→	→	→
f	#	#	#	→	←	#	#	#
g	#	#	#	→	←	#	#	#
h	#	#	#	→	←	#	#	#

$X_L = \{(\{a\},\{b\}),(\{a\},\{c\}),(\{a\},\{e\}),(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{b\},\{d\}),$
$(\{c\},\{d\}),(\{e\},\{f\}),(\{e\},\{g\}),(\{e\},\{h\}),(\{e\},\{f,g\}),(\{e\},\{f,h\}),(\{e\},\{g,h\}),(\{e\},\{f,g,h\}),$
$(\{g\},\{d\}),(\{h\},\{d\}),(\{f\},\{d\}),$
$(\{g,h\},\{d\}),(\{g,f\},\{d\}),(\{h,f\},\{d\}),(\{g,f,h\},\{d\}),$
$(\{g,b\},\{d\}),(\{g,c\},\{d\}),(\{h,b\},\{d\}),(\{h,c\},\{d\}),(\{f,b\},\{d\}),(\{f,c\},\{d\}),$
$(\{g,f,b\},\{d\})\},(\{g,f,c\},\{d\}),(\{g,b,h\},\{d\}),(\{g,c,h\},\{d\}),(b,f,h\},\{d\}),(\{c,f,h\},\{d\}),$
$(\{g,f,h,b\},\{d\}),(\{g,f,h,c\},\{d\})\}$
$Y_L = \{(\{a\},\{b,e\}),(\{a\},\{c,e\}),(\{e\},\{f,g,h\}),(\{g,f,h,b\},\{d\}),(\{g,f,h,c\},\{d\})\}$



This one is sound, because it is bounded and live. If you try every possible trace you see that easily.

3.

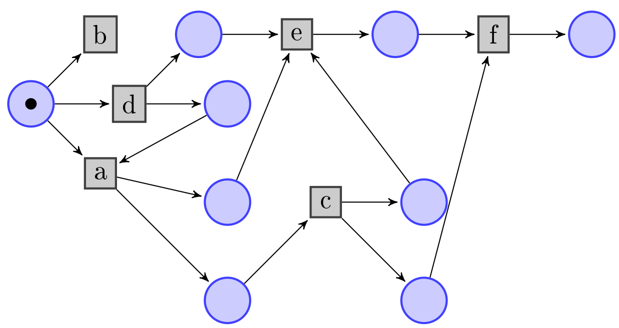
$$L3 = [\langle d,c,b,e,f\rangle,\langle a,e,f\rangle,\langle d,b,b,c,ef\rangle,\langle a,b,c,d,e,f\rangle,\langle b,d,a,c,f\rangle]$$

The $\alpha-$Algorithm gives the following:

$$T_L = \{a, b, c, d, e, f\}$$
$$T_I = \{a, b, d\}$$
$$T_O = \{f\}$$

	a	b	c	d	e	f
a	#	→	→	←	→	#
b	←	||	||	||	→	#
c	←	||	#	||	→	→
d	→	||	||	#	→	#
e	←	←	←	←	#	→
f	#	#	←	#	←	#

$X_L = \{(\{a\},\{c\}),(\{a\},\{e\}),(\{c\},\{e\}),(\{c\},\{f\}),(\{e\},\{f\}),(\{d\},\{a\}),(\{d\},\{e\})\}$
$Y_L = X_L$



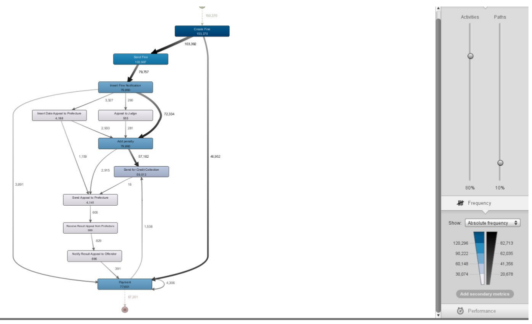
It is not live, because it is never possible to fire a, so it is neither sound.

Question 5

a)

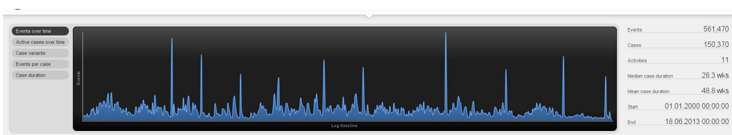
The process has 150370 cases and 561470 events. Median number of events is 5. The average duration is 48.8 weeks.

b)



The loop tells us, that a part of all people had to pay at least two times. If you check the log data, you can see, that mostly it happens in the next 30 days. Maybe they had a plan how to pay over a few weeks or had to pay penaltys.
It happen 4306 and for 4014 cases. So there are cases, where it happens more than one time. You also can see, that it happens at most 14 times.

c)



You can see that the distribution is going up and down a little bit, but there are 10 peaks to see where more happens. In the end activity gets lower and the distance between the 6th and 7th peak is higher than between the others. It is always around the typical paydays, so probably a lot of people then have the money to pay the fine.
The peakes are on: 06.04.2002 (Sa), 06.01.2003 (Mo), 05.01.2004 (Mo), 24.12.2004 (Fr), 20.02.2006 (Mo), 18.02.2007 (Su), 27.03.2009 (Fr), 08.10.2010 (Fr), 22.03.2012 (Thu), 19.04.2013 (Fr)

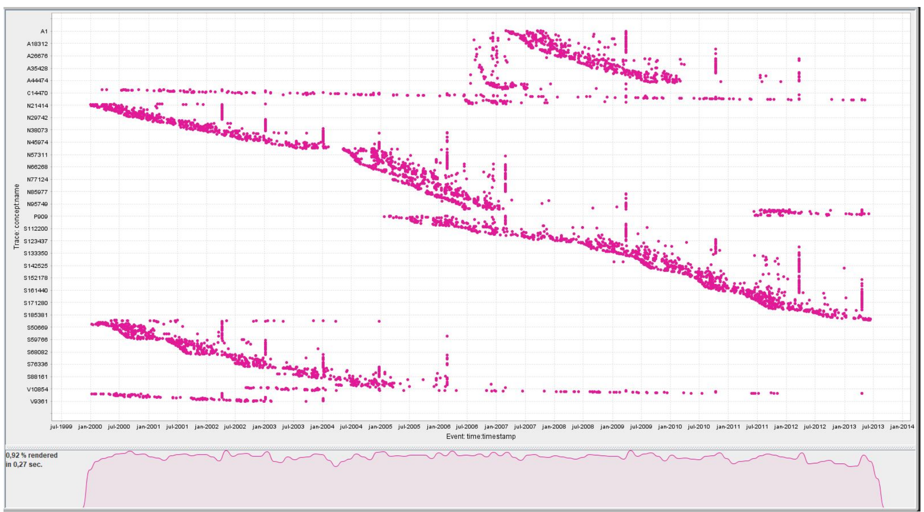
d)

There are 231 variants. The third most frequent variant has 20385 instances. It just contains the behaviour **create** and **send fine**. Nothing more happens there.

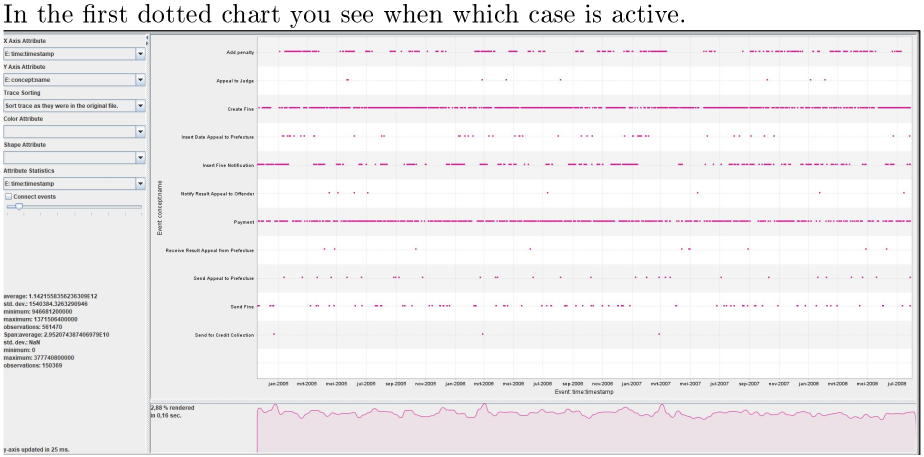
e)

It is just possible to 43% or 56%. The average case duration shorts to 45.2 weeks from 48.8 weeks and the median to 20.9 weeks from 28.3 weeks in both cases. So the most common cases are in average faster finished than all cases in average. The median is the instance in the middel. So if you write down all instances sorted, it is the middel one. The average is the sum of all instances divided by the number of instances. The average can change a lot for big or extree small oultiers. The median shows more a real duration in the middle of all instance durations.

f)



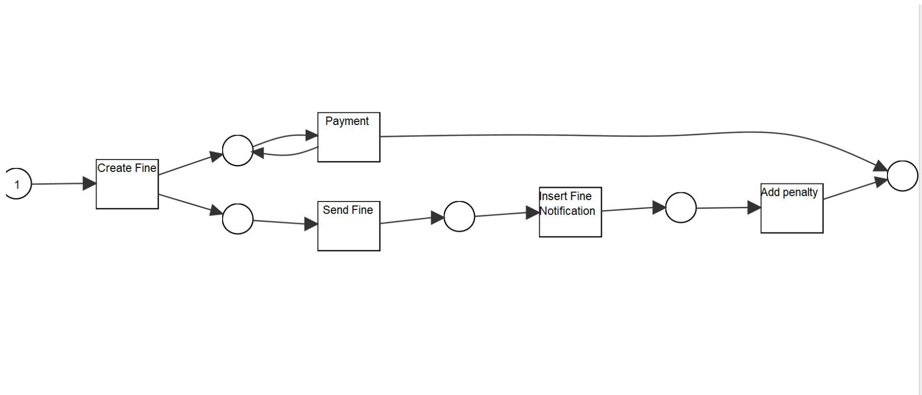
In the first dotted chart you see when which case is active.



For interpreting the dot chart for the question c) I changed the y-axis to the event names so I can see when which events happening. I also zoomed in so

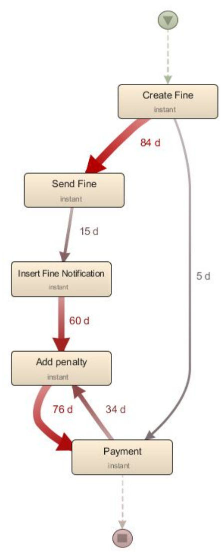
I can see the months of the years. Now you can check better the dates of the peaks.
You can see that **Send for Credit Collection** is strongly connected to the peakes to see in c).
Payment happens mostly always, but still it is a little bit bundled at the peaks.
Furthermore I would say, that in disco it is more easy to see when peaks happen, but in prom better to see what is happen on the peak days.

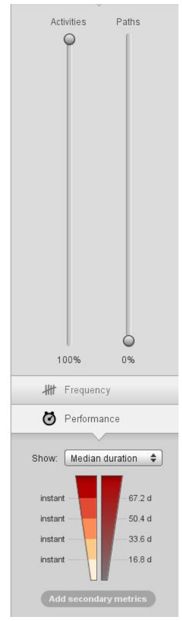
g)



If you apply the alpha-algoirthm on the not filtered data you can not see so clear, that in the resulting chart the payment can happen always and also infinite often.
It sounds weird, because you do not expect someone paying before he gets the fine, but looking at the data it happens.
Also in the filtered version the people or pay after the fine is created or pay to late, so that they get a penalty.

h)





84days is the median for send fine. I chose the median, because then you get a better idea what happens most of the times. The biggest mean is 23.3 weeks between add penalty and payment.
What you also can see, is that after 60days always a penalty is added.