1 Credit Application Process

1.1 Process model

We created a fictive model as seen in Fig. 1. The constraints (including the additional data constraints) of the process model mean that:

- C_1: *response(ApplyForCredit, Assessment)*: If a person applies for a credit and has a salary lower or equal to 24,000 euros and requests a credit amount higher than 50,000 euros, then the assessment of the application will be complex, cost more than 100 euros and should be handled from a credit assessment officer.
- C_2: *response(ApplyForCredit, Assessment)*: If a person applies for a credit and requests a credit amount higher than 100,000 euros, then the assessment of the application will be complex and cost more than 100 euros.
- C_3: *response(ApplyForCredit, Assessment)*: If a person applies for a credit and has a salary higher or equal to 24,000 euros and requests a credit amount smaller than 100,000 euros, then the assessment of the application will be simple and cost less or equal to 100 euros.
- C_4: *precedence(OutcomeNotification, ApplyForCredit)*: If a person's credit application gets accepted, then this event must be preceded by a person that applies for a credit and has a salary higher than 12,000 euros.

Table 1: Fictive credit application process model in Declare notation with additional data constraints

ID	Constraint	A/T	Payload

1	response	ApplyForCredit
		Assessment
			Salary <= 24000 & Amount > 50000
			AssessmentType=Complex & AssessmentCost > 100 &
			Resource=CreditAssessmentOfficer

2	response	ApplyForCredit Assessment	Amount > 100000 AssessmentType=Complex & AssessmentCost > 100
3	response	ApplyForCredit Assessment	Salary > 24000 and Amount <= 100000 AssessmentType=Simple & AssessmentCost <= 100
4	precedence	OutcomeNotification ApplyForCredit	Result is Accepted Salary > 12000

1.2 Descriptive statistics about data models created in the feature extraction phase

In Table 2, we provide descriptive information about the data models that were extracted in the *Feature Extraction* phase. The credit application example consists of two different Declare constraints where for each constraint, a data model, evaluated on the constraint itself, is extracted. Hence, the first column (*Declare Const.*) stands for Declare constraints where we shortcuted constraints due to lack of space (shortcuts are explained in Table **??**). The second and third columns (*#Activ Attr.* and *#Targ Attr.*, respectively) stand for the number of attributes in activation and target views, respectively. The fourth and fifth columns (*#Positive Ent.* and *#Deviant Ent.*) stand for the number of entities or in other words the length of activation and target views, respectively. We use this information to emphasize the values of measures in section *Redescription Quality Evaluation.*

Table 2: Descriptive Statistics about Feature Extraction Data Models (AC=ApplyForCredit, Ass=Assessment and ON=OutcomeNotification)

Declare Const.	**#Activ_Att**	**#Target_Att**	**#Positive Ent.**	**#Negative Ent.**
resp(AC, Ass)	2	3	1769	24
prec(AC, ON)	1	2	337	24

1.3 Redescription Quality Evaluation

How to interpret the plots: Every figure contains at most three plots in one view. Each plot has a title which refers to the Declare constraint from which the data model is created. In the x-axis, the different mining algorithms are shown where in parentheses the number of the discovered redescriptions per algorithm is displayed. In the y-axis, the values of measures such as Jaccard Index, p-value, AEJ, AAJ, Information Gain, Lift and Pearson's correlation coefficient are shown. Moreover, the box plot displays the minimum, the median and the maximum accuracy and also the quartiles which state where the most of accuracies lie in. The minimum value is represented by the bottom line of the box plot, the median by the middle line of box plot and the maximum by the top line of the box plot. Moreover, the first quartile starts from the minimum value line till the bottom line of the box and it states that 25% of accuracy values lie in that section, then the second quartile is located within the boundaries of the box stating that 50% of accuracy values lie in that section where the middle line is the median value of the overall data. Lastly, the last quartile states that 25% of values lie within that group.

In Table 3, we have displayed the discovered redescription rules from every algorithm with respect to the MP-Declare Constraints that they correspond to. It can be seen from the table that RF-SplitT has discovered redescriptions that relate to the Declare constraints that were used to generate the respective event logs, whereas for the other algorithms is not the case. In the paragraphs below, we discuss the quality of the redescriptions:

Table 3: Credit Application Process redescription rules

C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**
C_2	ApplyForCredit Assessment	108071<Amount<111552 80<Cost<93	109448<Amount & Amount<117158 & 115169<Amount Type=Complex	-
C_3	ApplyForCredit Assessment	91568<Salary<109914 80<Cost<84	Salary<7714 Cost<225 & 84<Cost & Cost<127	!109351<Amount & 105320<Amount & 13839<Salary !92<Cost & !Type=Complex
C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**
C_1	ApplyForCredit Assessment	-	50178<Amount & Salary<23558 Type=Complex	(!34617<Salary & 50757<Amount) — (99627<Amount & 34617<Salary & 50757<Amount) (! 98<Cost & Type= Complex & ! 101< Cost) — (! Type= Simple & 101<Cost) — (!162<Cost & Type= Simple & 101<Cost)

C_2	ApplyForCredit
	Assessment
		46630<Amount
		72<Cost
			93423<Amount
			100<Cost &
			Type=Complex &
			Resource=CreditOfficer
				extensive

C_3	ApplyForCredit Assessment	26485<Salary<101546 $\mid$ Amount<49978 Type=Simple & Cost<101	Amount<50705 & 2388<Salary Cost<102	extensive
C_4	OutcomeNotif. ApplyForCredit	Result=Rejected Salary<3054	Result=Accepted 7274<Salary	Result=Rejected Salary<6732

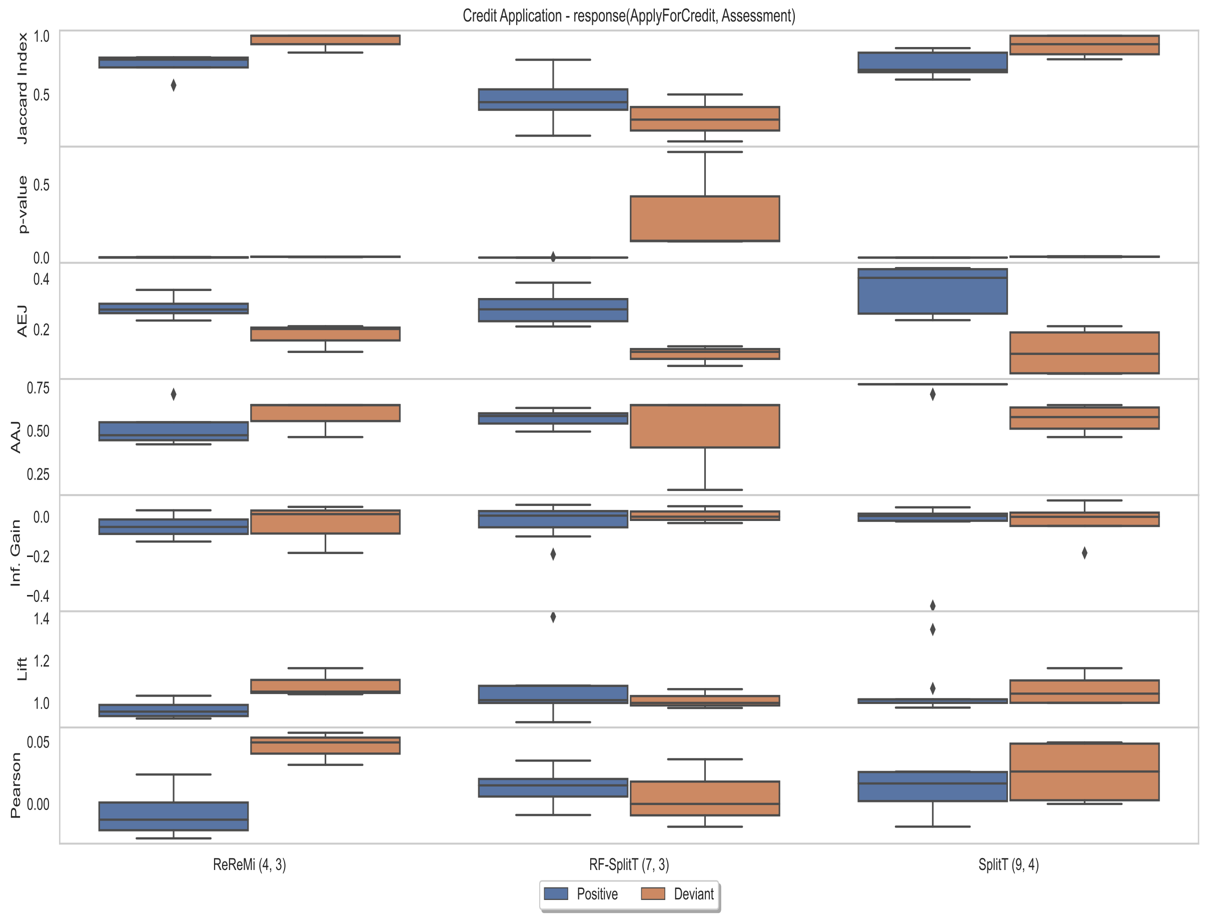

Figure 1: Credit Application redescriptions discovered for response(ApplyForCredit, Assessment). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

response(ApplyForCredit, Assessment): Observations regarding the quality of the redescriptions discovered from the respective constraint (cf. Fig. 1):

1. The behaviour between the algorithms over the different measurements is disimilar, this is a new behaviour with regard to previous event logs.

2. The RF-SpliT's rules have a lower accuracy, here the SIREN algorithms thrive.
3. The SplitT algorithm was able to discover more redescriptions, however, its rules are extensive and not at all understandable and could not be used for deviance analysis (cf. Table 3).

precedence(ApplyForCredit, OutcomeNotification): Observations regarding the quality of the redescriptions discovered from the Credit Application positive event log (cf. Fig. 2):

1. The ratio of discovered rules discovered from RF-SplitT is higher than the ones from the SIREN algorithms.
2. The RF-SplitT rules have a higher accuracy compared to the other algorithms.
3. None of the algorithms was able to discover rules from the deviant traces with respect to the constraint.

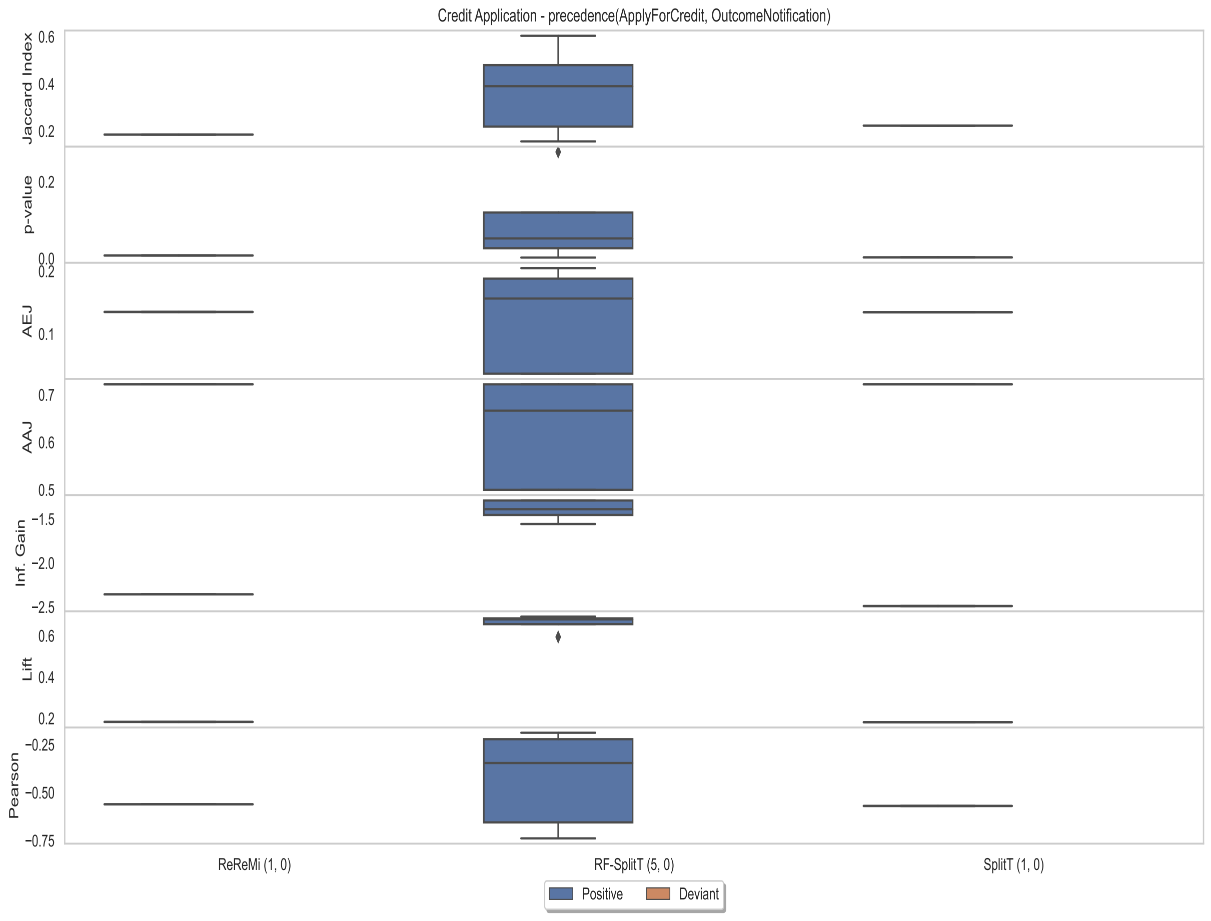

Figure 2: Credit Application redescriptions discovered for precedence(ApplyForCredit, OutcomeNotification). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

1.4 Redescription Execution Time Analysis

In order to estimate the time required for deviance analysis in practice, we performed a runtime measurement for the different redescription algorithms. The results obtained are shown in Table 4. To allow a fair comparison, we run all algorithms on the same data models. Our measurements handle the time that an algorithm requires to discover the respective rules on a given input dataset. For the Credit Application example, we found that the SplitT algorithm has the lowest execution time, while the ReReMi algorithm is the worst one. The reason why the ReReMi algorithm takes more time is because of its greedy behaviour, where it traverses all attributes and literals in the data model. The, SplitT algorithm, on the other hand, is a decision tree based algorithm where the trees grow based on the entropy calculation, resulting in faster tree discovery. The RF-SplitT algorithm is ranked second. This was also to be expected since it is an ensemble of decision trees and therefore takes more time than the SplitT algorithm. In summary, all execution times are in a range that is unproblematic for practical use.

Table 4: Execution times for Credit Application Process

	C_1,C_2,C_3		C_4		Time (s)		
	P	N	P	N	P	N	Rank
ReReMi	312.73	2.06	2.73	0.11	315.46	2.17	3
RF-SplitT	20.72	15.77	7.68	0.76	28.4	16.53	2
SplitT	3.07	0.78	0.67	0.45	3.74	1.23	1

1.5 Comparisons and Analysis of Discovered Redescriptions Evaluation

The final result of the deviance analysis pipeline is a detailed report in form of a human-readable text that explains the reasons of the deviance in general, as well as the causes for the deviance of the single process execution (cf. ReReMi (Table 5), RF-SplitT (Table 6) and SplitT (Table 7)). The tables have two sections, the *'Deviances of the event logs in general:'* and *'Analysing traces in detail:'*. In the *'Deviances of the event logs in general:'* section, the comparisons and analysis of deviant redescriptions with positive ones are handled. For example, in the Table 5 is detected that deviant traces that have appliers of credit requiring a credit amount between 108,071 euros and 111,552 euros lead to an assessment cost between 80 and 93 euros. However, in a positive traces that assessment cost takes place only when the required credit amount is lower than 49,978 euros. Moreover, as part of providing reasons on why particular deviant traces occurred, we can see that in Table 5 the trace with ID 397 has deviated because the salary of the applier was not in the range where credit applications are accepted.

Table 5: Excerpt of the deviance report in natural language text (Credit Application Process - ReReMi)

Deviances of the event logs in general:

The first mined negative rule is 'If the credit amount varies between 108071 and 111552 that implicates that the assessment cost alternates between 80 and 93. (r1)' and its subrules comparisons to the positive subrules are below:

- The amount for the event 'Apply For Credit' varies between 108071 and 111552 in the negative rule, while in the positive rule r21, it is lesser than 49978.

The second mined negative rule is 'If the credit salary varies between 3421 and 12413 that implicates that the assessment cost varies between 87 and 100. (r2)' and its subrules comparisons to the positive subrules are below:

- The assessment cost for the event 'Assessment' varies between 87 and 100 in the negative rule, while in the positive rule r14, it doesn't go lower than 94.
- The salary for the event 'Apply For Credit' varies between 3421 and 12413 in the negative rule, while in the positive rule r21, it alternates between 26485 and 101546.

The third mined negative rule is 'If the credit salary ranges from 91568 to 109914 that implicates that the assessment cost varies between 80 and 84. (r0)' and its subrules comparisons to the positive subrules are below:

- The salary for the event 'Apply For Credit' ranges from 91568 to 109914 in the negative rule, while in the positive rule r14, it is below 21267.
- The assessment cost for the event 'Assessment' varies between 80 and 84 in the negative rule, while in the positive rule r14, it is bigger than 94.

Analysing traces in detail:

- The process execution with 'Case No. 379' is deviant because the notification result differs from Rejected (r32).
- The process execution with 'Case No. 385' is deviant because the credit salary does not vary between 26485 and 101546 (r21), the assessment cost is smaller than 72 (r10) and the process is not executed by BankManager (r14).
- The process execution with 'Case No. 397' is deviant because the credit salary doesn't alternate between 26485 and 101546 (r21).

Table 6: Excerpt of the deviance report in natural language text (Credit Application Process - RF-SplitT)

Deviances of the event logs in general:

The first mined negative rule is 'If the credit amount is bigger than 109448 and the credit amount is below 117158 and the credit amount is bigger than 115169 that implicates that the assessment type is Complex. (r0)' and its subrules comparisons to the positive subrules are below:

- The amount for the event 'Apply For Credit' is bigger than 115169 in the negative rule, while in the positive rule r4, it is smaller than 50705.
- The assessment type for the event 'Assessment' is Complex in the negative rule, while in the positive rule r10, it is equal to Simple.

The second mined negative rule is 'If the credit salary is bigger than 32780 that implicates that the assessment cost is lower than 225 and the assessment cost doesn't go lower than 84 and the assessment cost is bigger than 127. (r5)' and its subrules comparisons to the positive subrules are below:

- The salary for the event 'Apply For Credit' is bigger than 32780 in the negative rule, while in the positive rule r1, it is lower than 23558.
- The assessment cost for the event 'Assessment' is bigger than 127 in the negative rule, while in the positive rule r4, it is below 102.

The third mined negative rule is 'If the credit salary is below 7714 that implies the assessment cost is lower than 225 and the assessment cost doesn't go lower than 84 and the assessment cost is lower than 127. (r3)' and its subrules comparisons to the positive subrules are below:

- The assessment cost for the event 'Assessment' is lower than 225 in the negative rule, while in the positive rule r3, it is lower than 166.
- The salary for the event 'Apply For Credit' is below 7714 in the negative rule, while in the positive rule r4, it doesn't go lower than 2388.

Analysing traces in detail:

- The process execution with 'Case No. 292' is deviant because the notification result differs from Accepted (r24), the notification result is not Accepted (r23) and the credit salary is not below 4573 (r22).
- The process execution with 'Case No. 386' is deviant because the credit salary is ,above 11521 (r12) the credit salary is above 23558 (r1) and the assessment cost goes lower than 100 (r21).
- The process execution with 'Case No. 399' is deviant because the credit amount is above 50705 (r4), the assessment cost is not below 100 (r10) and the credit amount is higher than 86833 (r14).

Table 7: Excerpt of the deviance report in natural language text (Credit Application Process - SplitT)

Deviances of the event logs in general:

The first mined negative rule is 'If the credit amount doesn't exceed 106411 and the credit salary exceeds 3570 and the credit salary is not bigger than 14742 that implicates that the assessment cost goes lower than 125 and the assessment cost doesn't go lower than 100. (r37)' and its subrules comparisons to the positive subrules are below:

- The salary for the event 'Apply For Credit' exceeds 3570 in the negative rule, while in the positive rule r73, it is smaller than 7533.
- The salary for the event 'Apply For Credit' exceeds 3570 in the negative rule, while in the positive rule r73, it exceeds 7676.
- . . .

The second mined negative rule is 'If the credit amount goes lower than 109351 and the credit amount doesn't go lower than 105320 and the credit salary is bigger than 13839 that implicates that the assessment cost doesn't exceed 92 and the assessment type is not Complex. (r50)' and its subrules comparisons to the positive subrules are below:

- The amount for the event 'Apply For Credit' goes lower than 109351 in the egative rule, while in the positive rule r73, it is not bigger than 118656.
- The amount for the event 'Apply For Credit' doesn't go lower than 105320 n the negative rule, while in the positive rule r73, it is not bigger than 118656.
- . . .

The third mined negative rule is 'If the credit amount goes lower than 114743 and the credit amount is bigger than 108071 that implicates that the assessment cost doesn't go lower than 80 and the assessment cost goes lower than 94. (r39)' and its subrules comparisons to the positive subrules are below:

- The amount for the event 'Apply For Credit' goes lower than 114743 in the egative rule, while in the positive rule r73, it goes lower than 118656.
- The amount for the event 'Apply For Credit' is bigger than 108071 in the egative rule, while in the positive rule r73, it goes lower than 118656.
- . . .

The fourth mined negative rule is 'If the credit salary is bigger than 3421 and the credit salary is not bigger than 12556 that implicates that the assessment cost varies between 87 and 100. (r42)' and its subrules comparisons to the positive subrules are below:

- The salary for the event 'Apply For Credit' is bigger than 3421 in the egative rule, while in the positive rule r73, it is smaller than 7533.
- The salary for the event 'Apply For Credit' is bigger than 3421 in the egative rule, while in the positive rule r73, it exceeds 7676.
- . . .

Analysing traces in detail:

- The process execution with 'Case No. 293' is deviant because the credit salary is higher than 6732 (r168).
- The process execution with 'Case No. 382' is deviant because the notification result is not Rejected (r168).
- The process execution with 'Case No.399' is deviant because the credit salary doesn't exceed 26485 (r81), the process is not executed by BankManager (r114) and the process is not executed by BankManager (r85).

2 Road Traffic Fines Process

2.1 Process model

Table 8: Road Traffic Fines Declare Constraints

ID	Constraint	A/T	Payload
1	response	InsertFineNotification AddPenalty	NotificationType=P Amount $<$ 5000
2	response	InsertFineNotification AddPenalty	NotificationType=C Amount $>$ 5000
3	response	AddPenalty SendForCreditCollection	Amount $>$ 5000 Resource=Admin
4	response	AddPenalty SendForCreditCollection	Amount $<$ 5000 Resource=Police

To build this process model, we took the event log [1] and mined a declarative process model out of it. A subset of the mined model which is also used in this paper is shown in Table. 8. The constraints (including the additional data constraints) of the process model mean that:

- C_1: *response(InsertFineNotification, AddPenalty)*: If a notification of fine insertion is of type P, then the amount of the penalty is smaller than 5,000 euros.
- C_2: *response(InsertFineNotification, AddPenalty)*: If a notification of fine insertion is of type C, then the amount of the penalty is higher than 5,000 euros.
- C_3: *response(AddPenalty, SendForCreditCollection)*: If the amount of a penalty is higher than 5,000 euros, then the collection of the credit should be handled by an admin staff.
- C_4: *response(AddPenalty, SendForCreditCollection)*: If the amount of a penalty is lower than 5,000 euros, then the collection of the credit should be handled by a police staff.

2.2 Descriptive statistics about data models created in the feature extraction phase

In Table 9, we provide descriptive information about the data models that were extracted in the *Feature Extraction* phase. The credit application example consists of two different Declare constraints where for each constraint, a data

[1]DOI: 10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5

model, evaluated on the constraint itself, is extracted. Hence, the first column (*Declare Const.*) stands for Declare constraints where we shortcuted constraints due to lack of space (shortcuts are explained in Table **??**). The second and third columns (*#Activ Attr.* and *#Targ Attr.*, respectively) stand for the number of attributes in activation and target views, respectively. The fourth and fifth columns (*#Positive Ent.* and *#Deviant Ent.*) stand for the number of entities or in other words the length of activation and target views, respectively. We use this information to emphasize the values of measures in section *Redescription Quality Evaluation.*

Table 9: Descriptive Statistics about Feature Extraction Data Models (IFN= InsertFineNotification, AP= AddPenalty and SCC= SendForCreditCollection)

Declare Const.	**#Activ_Att**	**#Target_Att**	**#Positive Ent.**	**#Negative Ent.**
resp(IFN, AP)	2	1	659	150
response(AP, SCC)	1	1	973	16

2.3 Redescription Quality Evaluation

In Table 10, we have displayed the discovered redescription rules from every algorithm with respect to the MP-Declare Constraints that they correspond to. It can be seen from the table that every algorithm has discovered redescriptions that relate to the Declare constraints that were used to generate the respective event logs. In the paragraphs below, we discuss the quality of the redescriptions:

response(InsertFineNotification, AddPenalty): Observations regarding the quality of the redescriptions discovered from the respective constraint (cf. Fig. 3):

1. The algorithms in general discover more rules from the positive traces, however RF-SplitT has discovered in general more, thus depicting different deviant behaviors.
2. In general, all algorithms have reached a good accuracy as part of the quality of the redescriptions.
3. The discovered redescription from ReReMi and RF-SplitT have no correlation with each other which is a good point, since we do not want to discover many rules that explain this behaviour. However, that is this not the case for the SplitT algorithm where one can see small negative correlation between its rules.

response(AddPenalty, SendForCreditCollection): Observations regarding the quality of the redescriptions discovered from the Credit Application positive event log (cf. Fig. 4):

Table 10: Road Traffic Fines Process redescription rules

C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**
C_1	InsertFineNotification	NotificationType=P	NotificationType=P	NotificationType=P
	AddPenalty	5000<Amount	4974<Amount	5000<Amount
C_2	InsertFineNotification	NotificationType=C	NotificationType=C	! NotificationType=P
	AddPenalty	Amount<4950	Amount<4975	Amount<4950
C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**
C_1	InsertFineNotification	NotificationType=P	NotificationType=C & Resource=Police	NotificationType=P
	AddPenalty	Amount<4909	Amount<5903	! 5000<Amount
C_2	InsertFineNotification	NotificationType=C	NotificationType=C	NotificationType=C
	AddPenalty	5000<Amount	4953<Amount	5000<Amount
C_3	AddPenalty	5000<Amount	4973<Amount	5000<Amount
	SendForCreditCollec.	Resource=Admin	Resource=Admin	! Resource=Police
C_4	AddPenalty	Amount<4951	Amount<4973	! 5000<Amount
	SendForCreditCollec.	Resource=Police	Resource=Police	Resource=Police

1. The ratio of discovered rules discovered from RF-SplitT is 2:1 compared to other SIREN algorithms.

2. The SplitT rules have a higher information gain which normally does not happen, but it has discovered for this constraint very compact rules. The Table 10 displays its rules.

3. None of the algorithms was able to discover rules from the deviant traces with respect to the constraint.

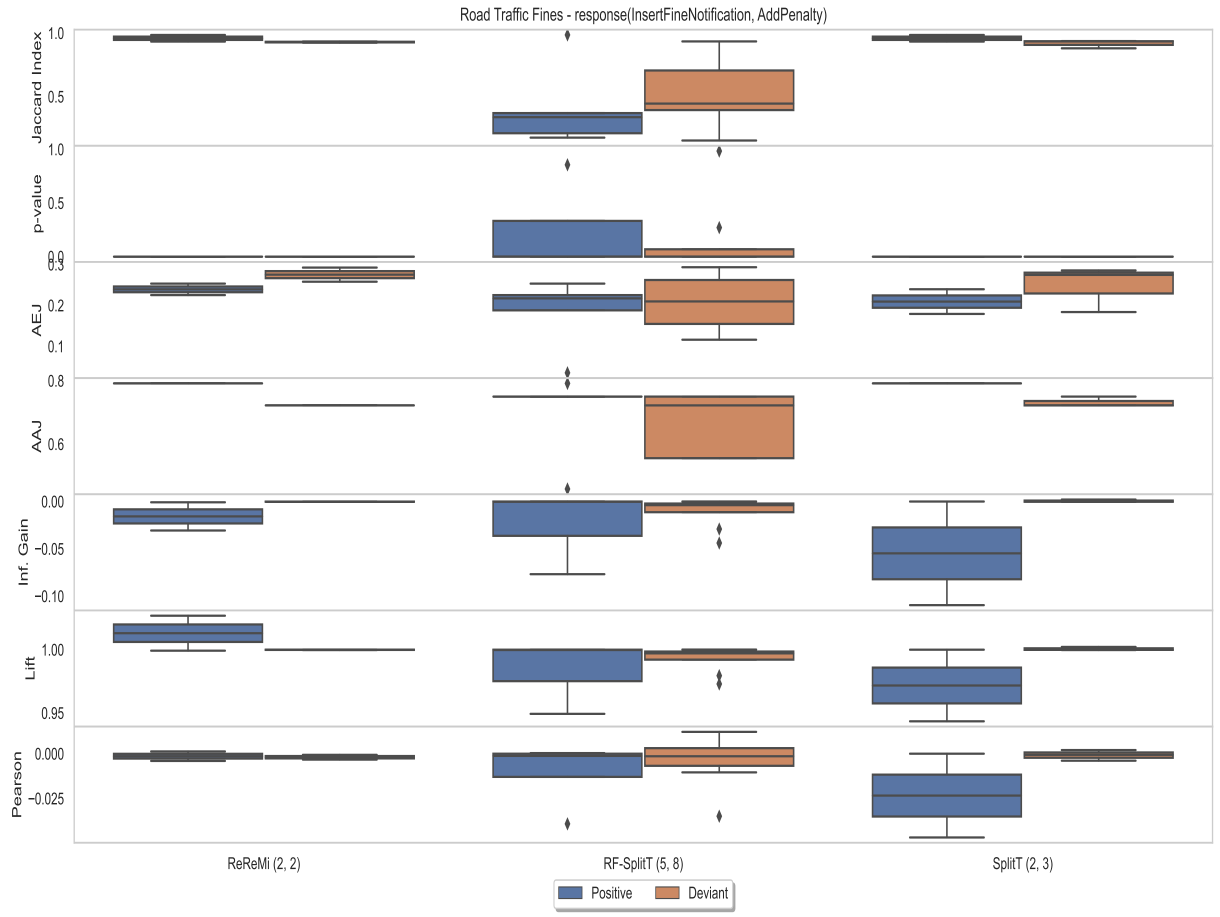

Figure 3: Road Traffic Fines redescriptions discovered for response(InsertFineNotification, AddPenalty). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

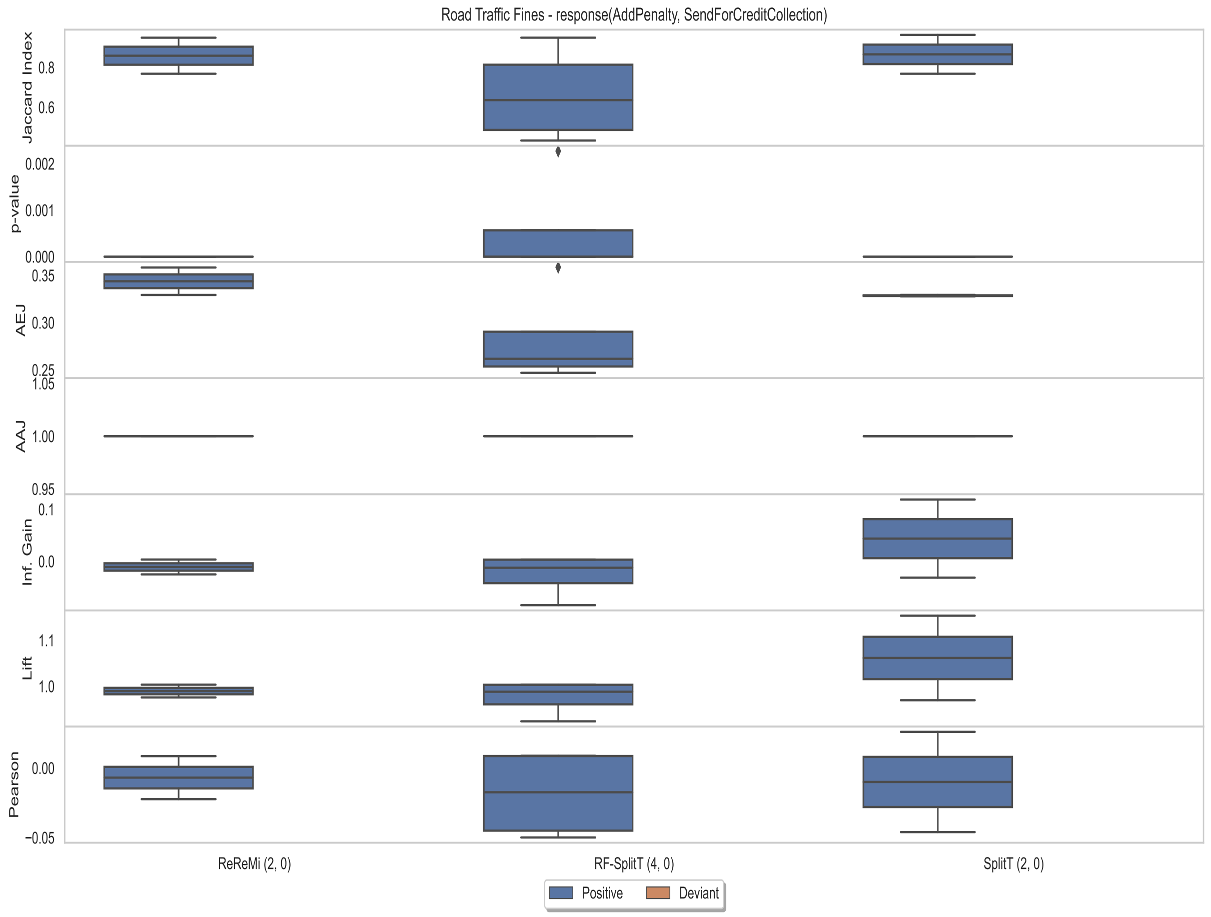

Figure 4: Road Traffic Fines redescriptions discovered for response(AddPenalty, SendForCreditCollection). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

2.4 Redescription Execution Time Analysis

In order to estimate the time required for deviance analysis in practice, we performed a runtime measurement for the different redescription algorithms. The results obtained are shown in Table 11. To allow a fair comparison, we run all algorithms on the same data models. Our measurements handle the time that an algorithm requires to discover the respective rules on a given input dataset. For the Road Traffic Fines example, we found that the SplitT algorithm has the lowest execution time, while the RF-SplitT algorithm is the worst one. The reason why the ReReMi algorithm is slightly faster than RF-SplitT is because the discovered rules from ReReMi are singleton queries meaning there was no need for much computations. We know from the paper that singelton queries are the base queris of the algorithm. The, SplitT algorithm, on the other hand, is a decision tree based algorithm where the trees grow based on the entropy calculation, resulting in faster tree discovery. The RF-SplitT algorithm is ranked third. This was also to be expected since it is an ensemble of decision trees and therefore takes more time than the SplitT algorithm.

Table 11: Execution times for Road Traffic Fines Process

Execution time for Road Traffic Fines Process

	C_1,C_2		C_3,C_4		Time (s)		
	P	N	P	N	P	N	Rank
ReReMi	6.77	0.74	5.32	0.13	12.09	0.87	2
RF-SplitT	9.99	5.98	3.5	0.42	13.49	6.4	3
SplitT	0.82	0.58	0.81	0.17	1.63	0.75	1

2.5 Comparisons and Analysis of Discovered Redescriptions Evaluation

The final result of the deviance analysis pipeline is a detailed report in form of a human-readable text that explains the reasons of the deviance in general, as well as the causes for the deviance of the single process execution (cf. ReReMi (Table 12), RF-SplitT (Table 13) and SplitT (Table 14)). The tables have two sections, the *'Deviances of the event logs in general:'* and *'Analysing traces in detail:'*. In the *'Deviances of the event logs in general:'* section, the comparisons and analysis of deviant redescriptions with positive ones are handled. For example, in the Table 13 is detected that deviant traces that have a penalty amount smaller than 6,393 euros, then the insertion of the fine is done by a person from an admin staff. However, in a positive traces the insertions of fines with an amount smaller than 6,393 euros are done from the police staff. Moreover, as part of providing reasons on why particular deviant traces occurred, we can see that in Table 13 the trace with ID 8999 has deviated because the notification type is not C and therefore not executed by a police staff.

Table 12: Excerpt of the deviance report in natural language text (Road Traffic Fines Process - ReReMi)

Deviances of the event logs in general:

The first mined negative rule is'If the notification type is C that implicates that the penalty amount is lesser than 4950. (r0)' and its subrules comparisons to the positive subrules are below:

- The amount for the event 'Add Penalty' is lesser than 4950 in the negative rule, while in the positive rule r2, it is bigger than 5000.
- The notification type for the event 'Insert Fine Notification' is C in the negative rule, while in the positive rule r3, it is equal to P.
- The amount for the event 'Add Penalty' is lesser than 4950 in the negative rule, while in the positive rule r3, it is below 4909.

The second mined negative rule is 'If the notification type is P that implicates that the penalty amount exceeds 5000. (r2)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is P in the negative rule, while in the positive rule r2, it is equal to C.
- The amount for the event 'Add Penalty' exceeds 5000 in the negative rule, while in the positive rule r3, it is lesser than 4909.

Analysing traces in detail:

- The process execution with 'Case No. 50' is deviant because the process is not executed by Police (r7), the notification type is unequal to P (r3) and the penalty amount doesn't exceed 5000 (r2).
- The process execution with 'Case No. 84' is deviant because the notification type is unequal to P (r3), the penalty amount is smaller than 5000 (r2) and the process is not executed by Police (r7).
- The process execution with 'Case No. 891' is deviant because the process is not executed by Admin (r6) and the penalty amount is not below 4951 (r7).

Table 13: Excerpt of the deviance report in natural language text (Road Traffic Fines Process - RF-SplitT)

Deviances of the event logs in general:

The first mined negative rule is 'If the notification type is equal to C or the notification type is c and the process is executed by Admin or the notification resource is System that implies that the penalty amount is smaller than 6393. (r0)' and its subrules comparisons to the positive subrules are below:

- The resource for the event 'Insert Fine Notification' is executed by Admin in the negative rule, while in the positive rule r1, it is executed by Police.
- . . .

The second mined negative rule is 'If the notification type is C that implies that the penalty amount is lesser than 4975. (r9)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is C in the negative rule, while in the positive rule r2, it is P.
- . . .

The third mined negative rule is 'If the notification type is P that implies the penalty amount exceeds 4974. (r8)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is P in the negative rule, while in the positive rule r1, it is equal to C.
- . . .

The fourth mined negative rule is 'If the notification type is P and the process is executed by Admin or the process is executed by System that implies that the penalty amount is smaller than 6393. (r1)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is P in the negative rule, while in the positive rule r1, it is C.
- . . .

The fifth mined negative rule is 'If the notification type is P and the notification resource is Police that implies the penalty amount exceeds 6515. (r3)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is P in the negative rule, while in the positive rule r1, it is equal to C.
- . . .

. . .

Analysing traces in detail:

- The process execution with 'Case No. 305' is deviant because the process is not executed by Police (r1), the process is not executed by System (r5) and the process is not executed by System (r3).
- The process execution with 'Case No. 882' is deviant because the process is not executed by Admin (r10), the process is not executed by Admin (r2) and the process is not executed by Police (r1).
- The process execution with 'Case No. 899' is deviant because the notification type differs from C (r7), the process is not executed by Police (r1) and the process

Table 14: Excerpt of the deviance report in natural language text (Road Traffic Fines Process - SplitT)

Deviances of the event logs in general:

The first mined negative rule is 'If the notification type is unequal to P that implicates that the penalty amount is lower than 4950. (r1)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is unequal to P in the negative rule, while in the positive rule r28, it is equal to C.
- The amount for the event 'Add Penalty' is lower than 4950 in the negative rule, while in the positive rule r28, it exceeds 5000.
- The amount for the event 'Add Penalty' is lower than 4950 in the negative rule, while in the positive rule r32, it is smaller than 5000.

The second mined negative rule is 'If the notification type is equal to P that implicates that the penalty amount is bigger than 5000. (r5)' and its subrules comparisons to the positive subrules are below:

- The notification type for the event 'Insert Fine Notification' is equal to P in the negative rule, while in the positive rule r28, it is C.

Analysing traces in detail:

- The process execution with 'Case No. 17' is deviant because the penalty amount is smaller than 5000 (r57), the notification type is unequal to P (r32) and the process is not executed by Police (r61).
- The process execution with 'Case No. 41' is deviant because the notification type differs from P (r32), the process is not executed by Police (r61) and the penalty amount is smaller than 5000 (r57).
- The process execution with 'Case No. 893' is deviant because the process is not executed by Police (r57) and the penalty amount is above 5000 (r61).

3 Repair Example

3.1 Process model

Table 15: Repair Example Declare Constraints

ID	Constraint	A/T	Payload

1	response	Register
		InformUser
			Resource=Tester
			Resource=System

2	response	Register
		InformUser
			Resource=Solver
			ComplexityOfRepair=Complex

3	response	InformUser ArchiveRepair	Resource=Tester NumberRepairs $>$ 2 & DefectFixed=false & Resource=Solver
4	response	InformUser ArchiveRepair	Resource=System NumberRepairs <= 2 & DefectFixed=true

5	response	InformUser
		ArchiveRepair
			ComplexityOfRepair=Complex
			NumberRepairs = 3

To build this process model, we took the event log [2] and mined a declarative process model out of it. A subset of the mined model which is also used in this paper is shown in Table. 15. The constraints (including the additional data constraints) of the process model mean that:

- C_1: *response(Register, InformUser)*: If an activity for registering a repair is executed by a Tester, then the informing of the user will be done by System.
- C_2: *response(Register, InformUser)*: If an activity for registering a repair is executed by a Solver, then the complexity of the repair will be Complex.
- C_3: *response(InformUser, ArchiveRepair)*: If the informing of the user will be done by a Tester, then an archive of repair will occur where number of repairs are higher than two, defect is not fixed and the activity is executed by a Solver.
- C_4: *response(InformUser, ArchiveRepair)*: If the informing of the user will be done by System, then an archive of repair will occur where number of repairs are smaller or equal to two and defect is fixed.
- C_5: *response(InformUser, ArchiveRepair)*: If the a repair is complex, then the number of repairs will be three.

[2]DOI: 10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1

3.2 Descriptive statistics about data models created in the feature extraction phase

In Table 16, we provide descriptive information about the data models that were extracted in the *Feature Extraction* phase. The credit application example consists of two different Declare constraints where for each constraint, a data model, evaluated on the constraint itself, is extracted. Hence, the first column (*Declare Const.*) stands for Declare constraints where we shortcuted constraints due to lack of space (shortcuts are explained in Table **??**). The second and third columns (*#Activ Attr.* and *#Targ Attr.*, respectively) stand for the number of attributes in activation and target views, respectively. The fourth and fifth columns (*#Positive Ent.* and *#Deviant Ent.*) stand for the number of entities or in other words the length of activation and target views, respectively. We use this information to emphasize the values of measures in section *Redescription Quality Evaluation.*

Table 16: Descriptive Statistics about Feature Extraction Data Models (R= Register, IU= InformUser and AR= ArchiveRepair)

Declare Const.	**#Activ_Att**	**#Target_Att**	**#Positive Ent.**	**#Negative Ent.**
response(R, IU)	1	2	809	95
response(IU, AR)	2	3	579	4

3.3 Redescription Quality Evaluation

In Table 17, we have displayed the discovered redescription rules from every algorithm with respect to the MP-Declare Constraints that they correspond to. It can be seen from the table that for this process algorithms are unable to discover redescriptions that relate to the Declare constraints that were used to generate the respective event logs. Note that RF-SplitT covers most of the Declare constraints. In the paragraphs below, we discuss the quality of the redescriptions:

response(Register, InformUser): Observations regarding the quality of the redescriptions discovered from the respective constraint (cf. Fig. 5):

1. The RF-SplitT has discovered in general more redescriptions, thus depicting different deviant behaviors.
2. In general, all algorithms have reached a good accuracy as part of the quality of the redescriptions.
3. The discovered rules from the RF-SplitT are high in amount and the p-value in some cases very high in which case we can not ignore the null hypothesis. That implies that our data would have occurred by random

Table 17: Repair Example Process redescription rules

C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**
C_1	Register InformUser	Resource=Tester Resource=Tester	Resource=Tester Resource=Tester	Resource=Tester Resource=Tester

C_2	Register
	InformUser
		-	Resource=Solver
			Complexity=Simple
				-

C	**A/T**	**ReReMi**	**RF-SplitT**	**SplitT**

C_1	Register
	InformUser
		-	Resource=Tester
			Resource=System
				-

C_2	Register
	InformUser
		Resource=Solver
		Complexity=Complex
			Resource=Solver
			Complexity=Complex
			& Resource=Solver
				Resource=Solver
				Complexity=Complex

C_3	InformUser ArchiveRepair	-	-	-
C_4	InformUser ArchiveRepair	Resource=System DefectFixed	Resource=System (Resource=System — Resource=Tester) & DefectFixed & 1<NumberRepairs	-
C_5	InformUser ArchiveRepair	-	-	-

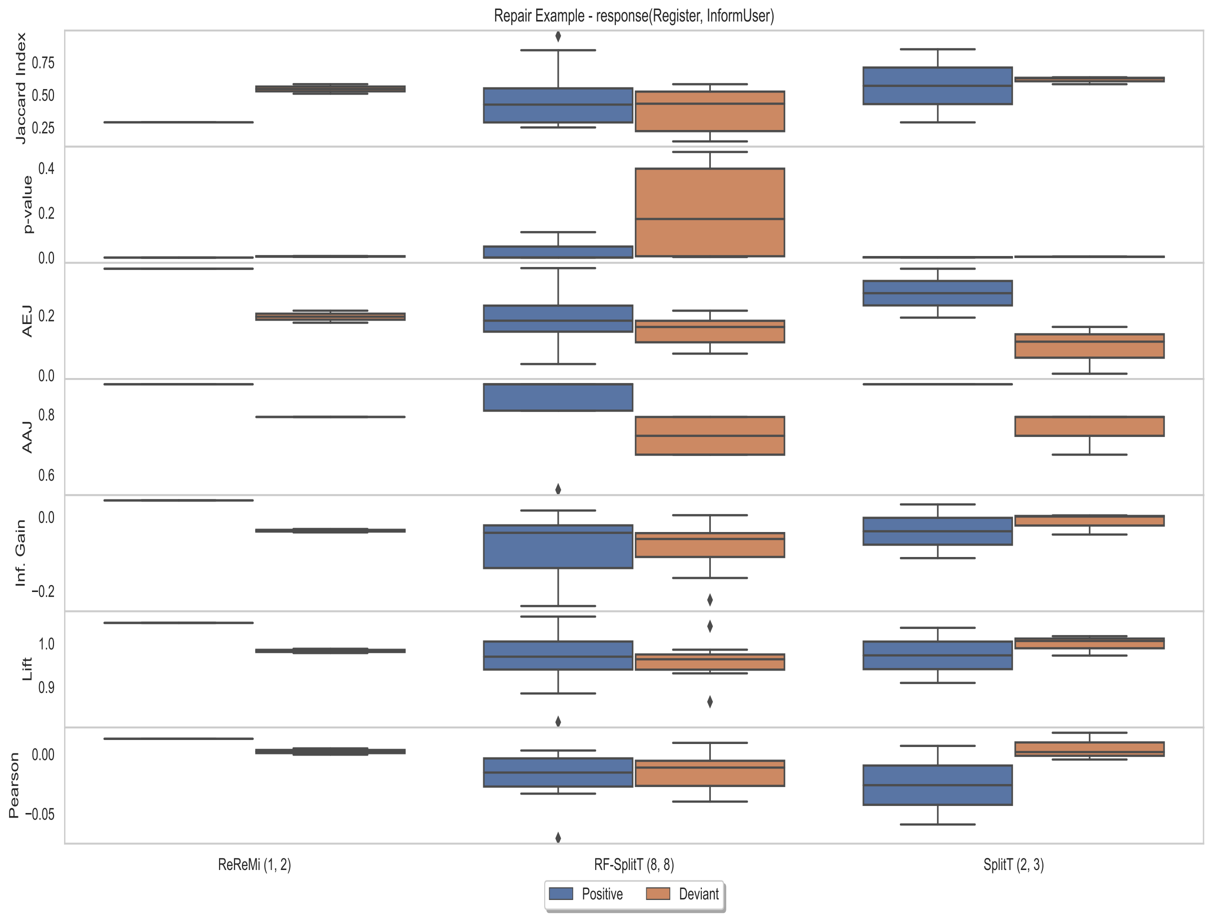

Figure 5: Repair Example redescriptions discovered for response(Register, InformUser). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

chance. This is a good thing that RF-SplitT was able to discover rule for such random behaviors.

response(AddPenalty, SendForCreditCollection): Observations regarding the quality of the redescriptions discovered from the Credit Application

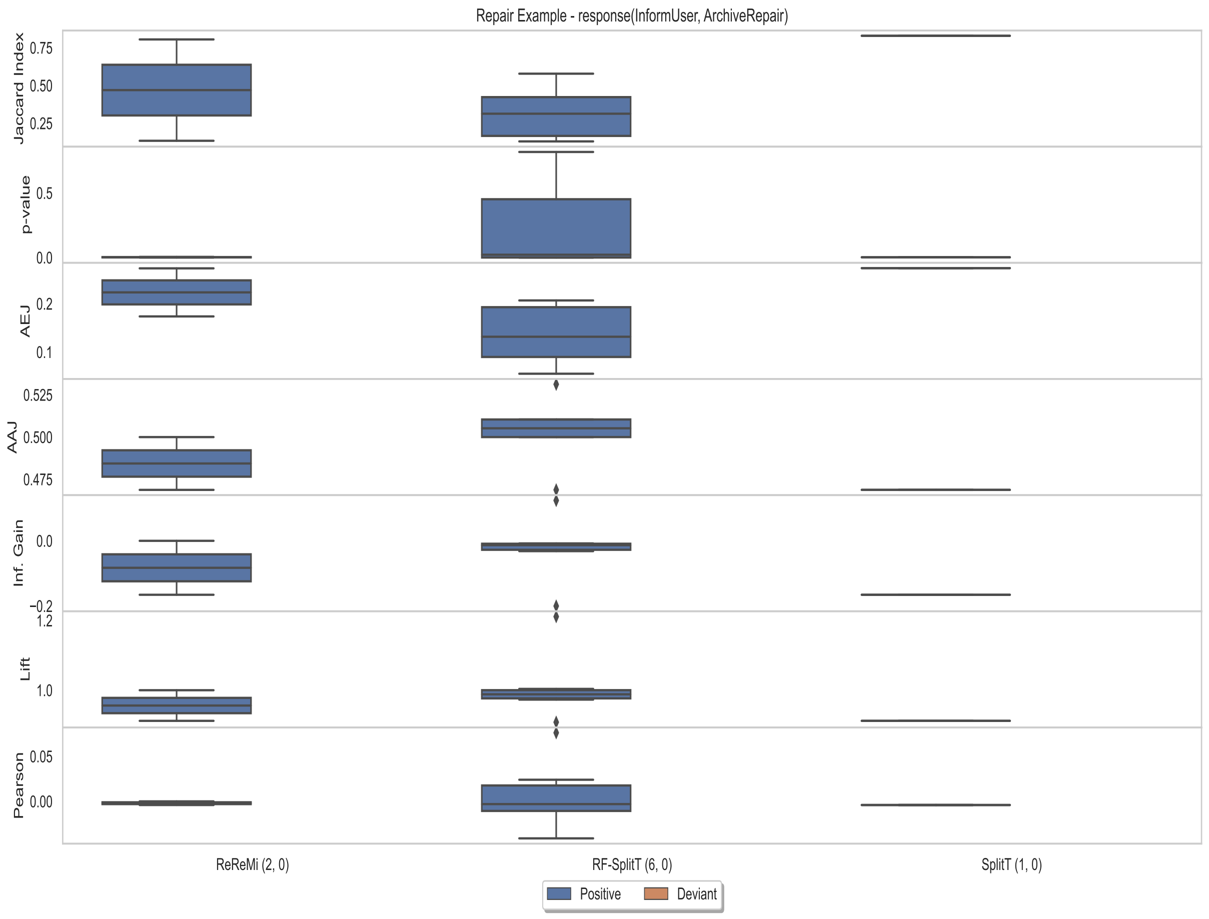

Figure 6: Repair Example redescriptions discovered for response(InformUser, ArchiveRepair). An explanation how to read the box plots can be found in the Appendix (cf. Sec. 1.3). We denoted the number of positive (p) and (d) redescriptions identified by the respective algorithm in form of a tuple (p, d) behind its name.

positive event log (cf. Fig. 6):

1. The ratio of discovered rules discovered from RF-SplitT is 3:1 compared to other SIREN algorithms.
2. The ReReMi and SplitT rules have a higher accuracy (Jaccard index).

3. None of the algorithms was able to discover rules from the deviant traces with respect to the constraint.
4. Also, for this constraint the discovered rules from the RF-SplitT are high in amount, but the p-value has a high value which means that our data have occurred by random chance. However, this is a good thing that RF-SplitT was able to discover rule for such random behaviors.

3.4 Redescription Execution Time Analysis

In order to estimate the time required for deviance analysis in practice, we performed a runtime measurement for the different redescription algorithms. The results obtained are shown in Table 18. To allow a fair comparison, we run all algorithms on the same data models. Our measurements handle the time that an algorithm requires to discover the respective rules on a given input dataset. For the Repair Example, we found that the SplitT algorithm has the lowest execution time, while the ReReMi algorithm is the worst one. The reason why the ReReMi algorithm was slower than the other algorithms is its greedy behavior to go through each attribute therefore needing more computations.The, SplitT algorithm, on the other hand, is a decision tree based algorithm where the trees grow based on the entropy calculation, resulting in faster tree discovery. The RF-SplitT algorithm is ranked third. This was also to be expected since it is an ensemble of decision trees and therefore takes more time than the SplitT algorithm.

Table 18: Execution times for Repair Example Process

Execution time for Repair Example

	C_1,C_2		C_3,C_4,C_5		Time (s)		
	P	N	P	N	P	N	Rank
ReReMi	0.18	0.2	0.32	0.09	0.5	0.29	1
RF-SplitT	6.34	5.62	12.65	1.41	18.99	7.03	3
SplitT	0.41	0.31	0.45	0.15	0.86	0.46	2

3.5 Comparisons and Analysis of Discovered Redescriptions Evaluation

The final result of the deviance analysis pipeline is a detailed report in form of a human-readable text that explains the reasons of the deviance in general, as well as the causes for the deviance of the single process execution (cf. ReReMi (Table 19), RF-SplitT (Table 20) and SplitT (Table 21)). The tables have two sections, the *'Deviances of the event logs in general:'* and *'Analysing traces in detail:'*. In the *'Deviances of the event logs in general:'* section, the comparisons and analysis of deviant redescriptions with positive ones are handled. For example, all algorithms have detected that traces are deviant when the activity register is not executed by a tester and inform the user is not executed by a test. However, in positive traces, the execution of activity inform user is done by the solver. Moreover, as part of providing reasons on why particular deviant traces occurred, we can see that in Table 19 the trace with ID 935 has deviated because the repair is not complex and the process is not executed by solver.

Table 19: Excerpt of the deviance report in natural language text (Repair Example Process - ReReMi)

Deviances of the event logs in general:

The first mined negative rule is 'If the register resource is Solver that implicates that the process is executed by System. (r2)' and its subrules comparisons to the positive subrules are below:

- The resource for the event 'Inform User' is executed by System in the negative rule, while in the positive rule r2, it is Solver.

The second mined negative rule is 'If the register resource is Tester that implicates that the user resource is Tester. (r0)' and its subrules comparisons to the positive subrules to the positive subrules are below:

- The resource for the event 'Register' is Tester in the negative rule, while in the positive rule r2, it is executed by Solver.
- The resource for the event 'Inform User' is Tester in the negative rule, while in the positive rule r2, it is Solver.

Analysing traces in detail:

- The process execution with 'Case No. 372' is deviant because the process is not executed by Solver (r1) and the process is not executed by Solver (r2).
- The process execution with 'Case No. 376' is deviant because the user complexity ,of repair differs from Complex (r1).
- The process execution with 'Case No. 935' is deviant because the user complexity of repair is not Complex (r1) and the process is not executed by Solver (r2).

Table 20: Excerpt of the deviance report in natural language text (Repair Example Process - RF-SplitT)

Deviances of the event logs in general:

The first mined negative rule is ’If the process is executed by Solver or the process is executed by System that implies that the user complexity of repair is Medium. (r5)’ and its subrules comparisons to the positive subrules are below:

- The resource for the event ’Register’ is executed by Solver in the negative rule, while in the positive rule r0, it is Tester.
- . . .

The second mined negative rule is ’If the process is executed by Solver or the process is executed by System that implies the process is executed by System. (r0)’ and its subrules comparisons to the positive subrules are below:

- The resource for the event ’Register’ is executed by Solver in the negative rule, while in the positive rule r0, it is executed by Tester.
- . . .

The third mined negative rule is ’If the process is executed by System or the register resource is Tester that implies the user complexity of repair is equal to Complex. (r4)’ and its subrules comparisons to the positive subrules are below:

- The resource for the event ’Register’ is executed by System in the negative rule, while in the positive rule r0, it is Tester.
- . . .

The fourth mined negative rule is ’If the register resource is Solver that implicates that the user complexity of repair is equal to Simple. (r3)’ and its subrules comparisons to the positive subrules are below:

- The resource for the event ’Register’ is Solver in the negative rule, while in the positive rule r0, it is executed by Tester.
- . . .

The fifth mined negative rule is ’If the register resource is Solver that implies that the user resource is Solver. (r1)’ and its subrules comparisons to the positive subrules are below:

- The resource for the event ’Register’ is Solver in the negative rule, while in the positive rule r0, it is Tester.
- . . .

...

Analysing traces in detail:

- The process execution with ’Case No. 207’ is deviant because the user complexity of repair is not Medium (r5), the process is not executed by Solver (r8) and the process is not executed by Tester (r7).
- The process execution with ’Case No. 461’ is deviant because the user complexity of repair is not Simple (r3), the process is not executed by Solver (r16) and the process is not executed by Solver (r4).
- The process execution with ’Case No. 799’ is deviant because the process is not executed by System (r8), the process is not executed by Solver (r1) and the process is not executed by System (r3).

Table 21: Excerpt of the deviance report in natural language text (Repair Example Process - SplitT)

Deviances of the event logs in general:

The first mined negative rule is 'If the register resource is not Tester that implicates that the user resource is not Tester. (r7)' and its subrules comparisons to the positive subrules are below:

- The resource for the event 'Register' is not Tester in the negative rule, while in the positive rule r37, it is not System.
- The resource for the event 'Inform User' is not Tester in the negative rule, while in the positive rule r37, it is executed by Solver.

The second mined negative rule is 'If the process is executed by Tester that implicates that the user resource is Tester. (r3)' and its subrules comparisons to the positive subrules are below:

- The resource for the event 'Register' is executed by Tester in the negative rule, while in the positive rule r37, it is not executed by System.
- The resource for the event 'Inform User' is Tester in the negative rule, while in the positive rule r37, it is Solver.

Analysing traces in detail:

- The process execution with 'Case No. 1' is deviant because the process is not executed by System (r17), the user complexity of repair is not Complex (r21) and the process is not executed by Solver (r37).
- The process execution with 'Case No. 375' is deviant because the process is not executed by System (r17) and the user complexity of repair is unequal to Complex (r21).
- The process execution with 'Case No. 888' is deviant because the process is not executed by System (r17), the user complexity of repair differs from Complex (r21) and the process is not executed by Solver (r37).