Rule Compressor™

Using Machine Learning for Compression of Large Classification Rulesets

Jacob Feldman, PhD
Chief Technology Officer

BR & ML

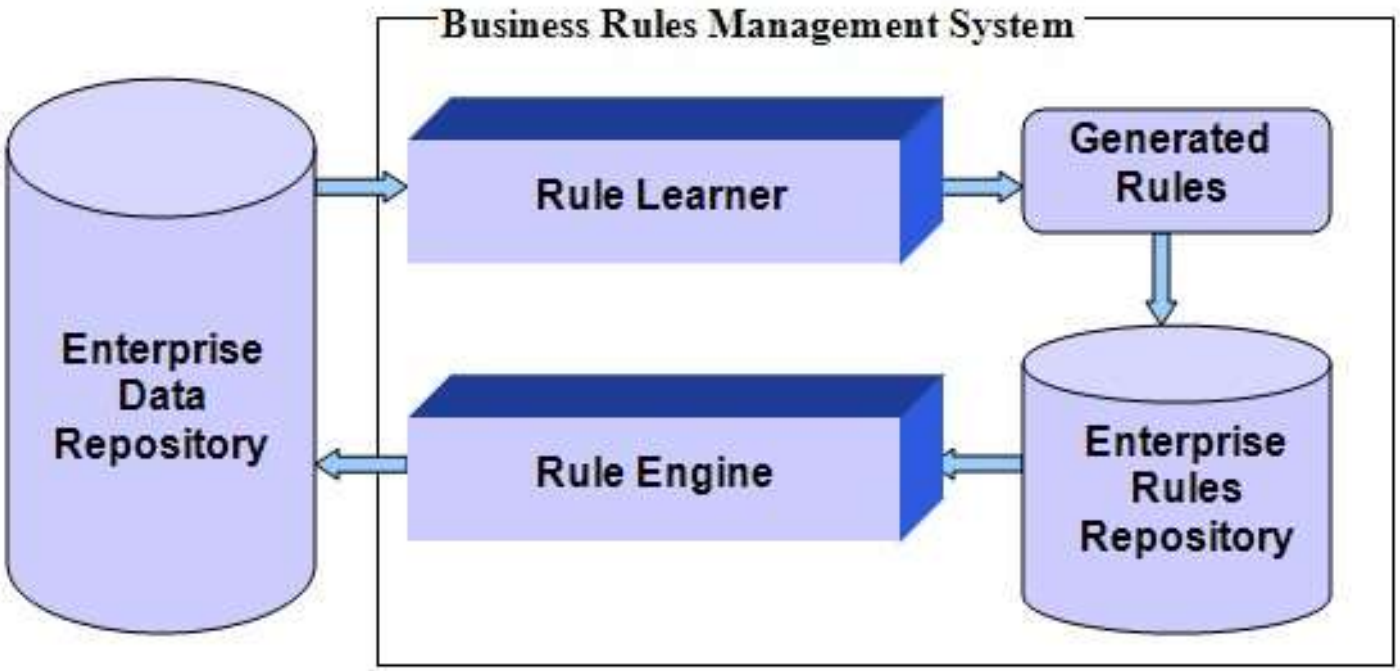
- Business Rules (BR)
  - Business Rules and Decision Management Systems are commonly used to represent, manage, and execute business logic efficiently using Rule Engines
- Machine Learning (ML)
  - Machine Learning offers powerful algorithms and tools for practical extraction of rules (patterns) from data

Real-World BR&ML Problems

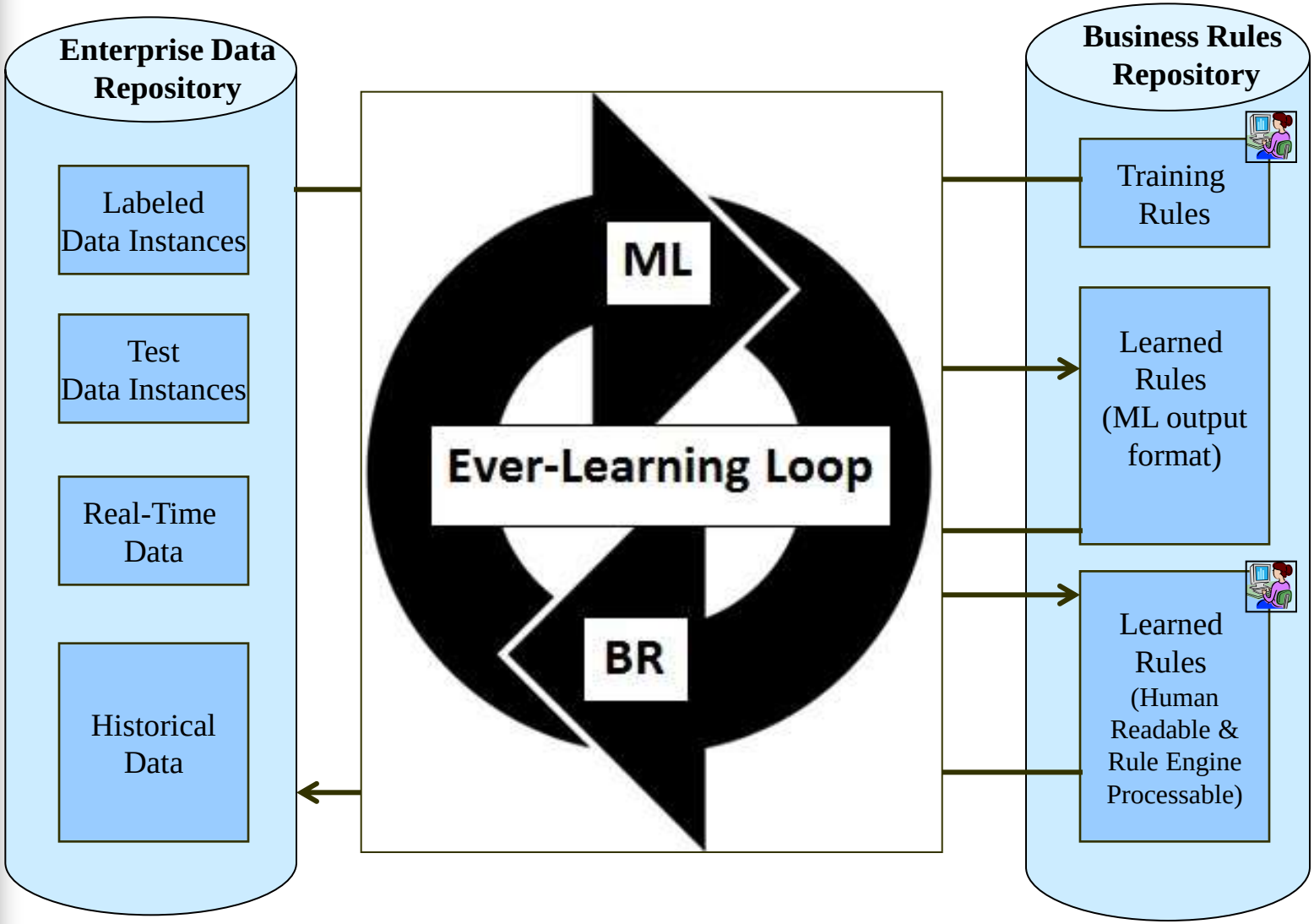
- It becomes increasingly important to find previously unknown dependencies inside data streams
- Online multi-transactional processing systems require new rules to be discovered “on-the-fly”
- Business Rules Repositories grow quickly, become too complicated, and have to be compressed and optimized

ML + BR Integration

- **Rule Learner discovers and produces rules**
- **Rule Engine consumes/executes them**



OpenRules Rule Learner: ML+BR Integration Schema



Never-Ending Rules Learning

"Why not build machine learners that learn in the same cumulative way as humans, becoming increasingly competent rather than halting at some plateau? Can we build never-ending learners?"

Tom M. Mitchell, Carnegie Mellon University

Real-World Examples

- In a real-world application, humans experts classify their data using a "*gut feel*" based on their past experience in working with the data.
- Rule Learner helps to convert this "gut feel" into rules with very specific numeric thresholds!
- Example of generated red-flag rules for a large government agency:

Rules String classifyCarExpense(Record r)	Rule 1	Rule 2	Rule 3
if CAR_EXPENSE_AMOUNT	>= 2758		
and BUSINESS_MILES_COUNT		>= 4100	
and GROSS_RECEIPTS_AMOUNT		<= 3772	
then CAR_EXPENSE	RED	RED	GREEN

  - Rule Learner selected only a few essential attributes out of hundreds considering around 50K data instances

Motivation for BR Compression

- Decision Tables and other rulesets have a tendency to grow quickly
  - Attempts to covers all possible combinations of decision variable lead to a combinatorial explosion
  - A decision table with 5-10 columns may end up with 1000's rules
- Question: Can ML help with compression of large rulesets?

Manual Rules Compression

Min Age	Max Age	Card Type	Discount Code
18	30	Standard	0
18	30	Gold	2
18	30	Platinum	3
31	40	Standard	1
31	40	Gold	1
31	40	Platinum	1
41	50	Standard	1
41	50	Gold	2
41	50	Platinum	3
51	60	Standard	1
51	60	Gold	2
51	60	Platinum	3
61	70	Standard	1
61	70	Gold	2
61	70	Platinum	3
71	120	Standard	1
71	120	Gold	2
71	120	Platinum	3

Card Type	Min Age	Max Age	Discount Code
Standard			1
Standard	18	30	0
Gold			2
Gold	31	40	1
Platinum			3
Platinum	31	40	1

18 rules => 6 rules

(with overrides)

One cannot do it manually when there are much more attributes and rules – we need an automatic compression!

Automatic Rules Compression

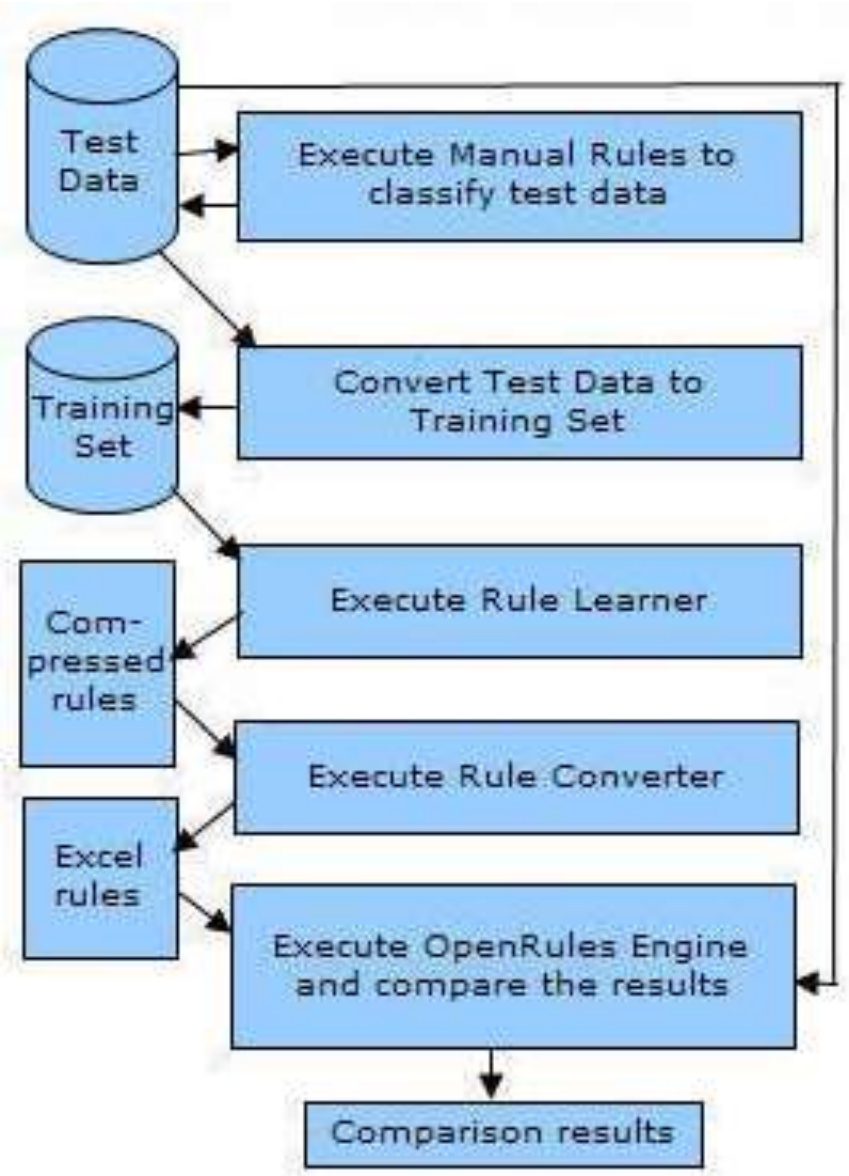
Rules void classifyInstance(TestInstance instance)					
IF type is	AND adjustment > $$$	AND adjustment < $$$	AND amount < $$$	AND amount >= $$$	THEN Classify Instance as
					NONE
31	$200		-$150		TOP
31		$200		-$189	BOTTOM
32	$500		-$1,000		TOP
32		$500		-$99	BOTTOM
33	$500		-$1,000		TOP
33		$500		-$100	BOTTOM
34	$500		-$1,000		TOP
34		$500		-$100	BOTTOM
35	$500		-$800		TOP
35		$500		-$100	BOTTOM
36	$500		-$800		TOP
36		$500		-$100	BOTTOM
37	$500		-$2,000		TOP
37		$500		$0.0	BOTTOM

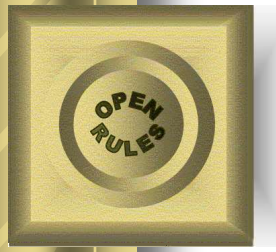
```
IF amount <= -159
THEN
    classifiedAs=TOP
ELSE
    classifiedAs=BOTTOM
```

15 rules => 1 rule!

Correctly Classified Instances:
2,395 out of 2,396

How Rule Compressor Works





Automatic Rules Generation: Important Warning

- Positive effect:
  - Smaller and easy to maintain rulesets
- Negative effect:
  - Unavoidable errors: always validate if generated rules are acceptable
    - Could be OK for small insurance claims
    - Could be a disaster for medical diagnostics
  - Lost business knowledge

Conclusion

- ML+BR integration brings immediate improvements to BR systems by supporting never-ending rules discovery and adjustment
- Rule Compressor allows compressing large rules sets
- It is always necessary to evaluate if unavoidable errors are acceptable