Business Rules Forum 2007

Integrating Business Rules and Machine Learning Technologies

Dr. Jacob Feldman, OpenRules, Inc., Chief Technology Officer, jacobfeldman@openrules.com

Ian Graham, TriReme International Ltd, Principal Consultant, ian@trireme.com

TriReme
www.trireme.com

Outline

- Rule Learner + Rule Engine: Motivation
- Rules Discovery using Machine Learning techniques
- Rule Engine as an ML Trainer
- ML + BR Integration Scenarios
- Rules Compression and Optimization with ML

Motivation

- **Machine Learning (ML)**
  - Today Machine Learning offers powerful algorithms and tools for practical knowledge discovery
  - There are many powerful Rule Learners that use ML algorithms to extract decision rules and patterns from massive data sets
- **Business Rules (BR)**
  - Today Business Rules Management Systems (BRMS) are commonly used to represent, manage, and execute business rules efficiently using Rule Engines
  - Rules discovery is an important component of any BRMS; it is usually done by human experts
- **It seems only natural to combine ML and BR:**
  - A Rule Learner produces rules
  - A Rule Engine consumes them

Motivation – Rule Compression

- **Decision tables are a natural representation for this type of problem; but...**
- **With the usual approach, a BRMS will generate a rule per row. Here there are 18 rows (72 entries); 18 rules of the form:**

  If age is between 18 and 30
  and card type is standard
  then discount code is unknown

  If age is between 18 and 30
  and card type is gold
  then discount code is 2

- **1000 rows implies 1000 rules, &c.**
- **When you have hundreds of rows or attributes, you had better apply ML to do 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

Motivation – Rule Compression (2)

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

```
If card type is “Standard”
        then discount code is 1
        unless age is between 18 and 30
If card type is “Standard” and age is between 18 and 30
        then discount code is unknown
If card type is “Gold”
        then discount code is 2
        unless age is between 31 and 40
If card type is “Gold” and age is between 31 and 40
        then discount code is 1
If card type is “Platinum”
        then discount code is 3
        unless age is between 31 and 40
If card type is “Platinum” and age is between 31 and 40
        then discount code is 1
```

=

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

Just 6 rules – Less is more!

Real-World BRMS Problems

- Changes in data come so frequently that there is simply no time for manual rules harvesting and adjustment
- Business Rules Repositories grow quickly, become too complicated, and have to be compressed and optimized
- 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”
- Example:
  - A new booming movement known as “CEP” (Complex Event Processing) tends to ignore Rules Engines labeling them as incapable of reacting to high-volume data streams

Where BRMS Stops Short

- **Today BRMS are usually very good at addressing HOW-questions:**
  - *HOW business users can work in concert with technical users to create, modify, and manage business rules*
  - *HOW to execute the rules efficiently*
- **However, a BRMS usually ignores a WHAT-question:**
  - *WHAT changes should be made to existing rules to comply with ever changing data*
- **The vast majority of BR systems today do *only initial rules harvesting*, then just use resulting rules and rely on human experts to initiate and introduce changes in the rules**

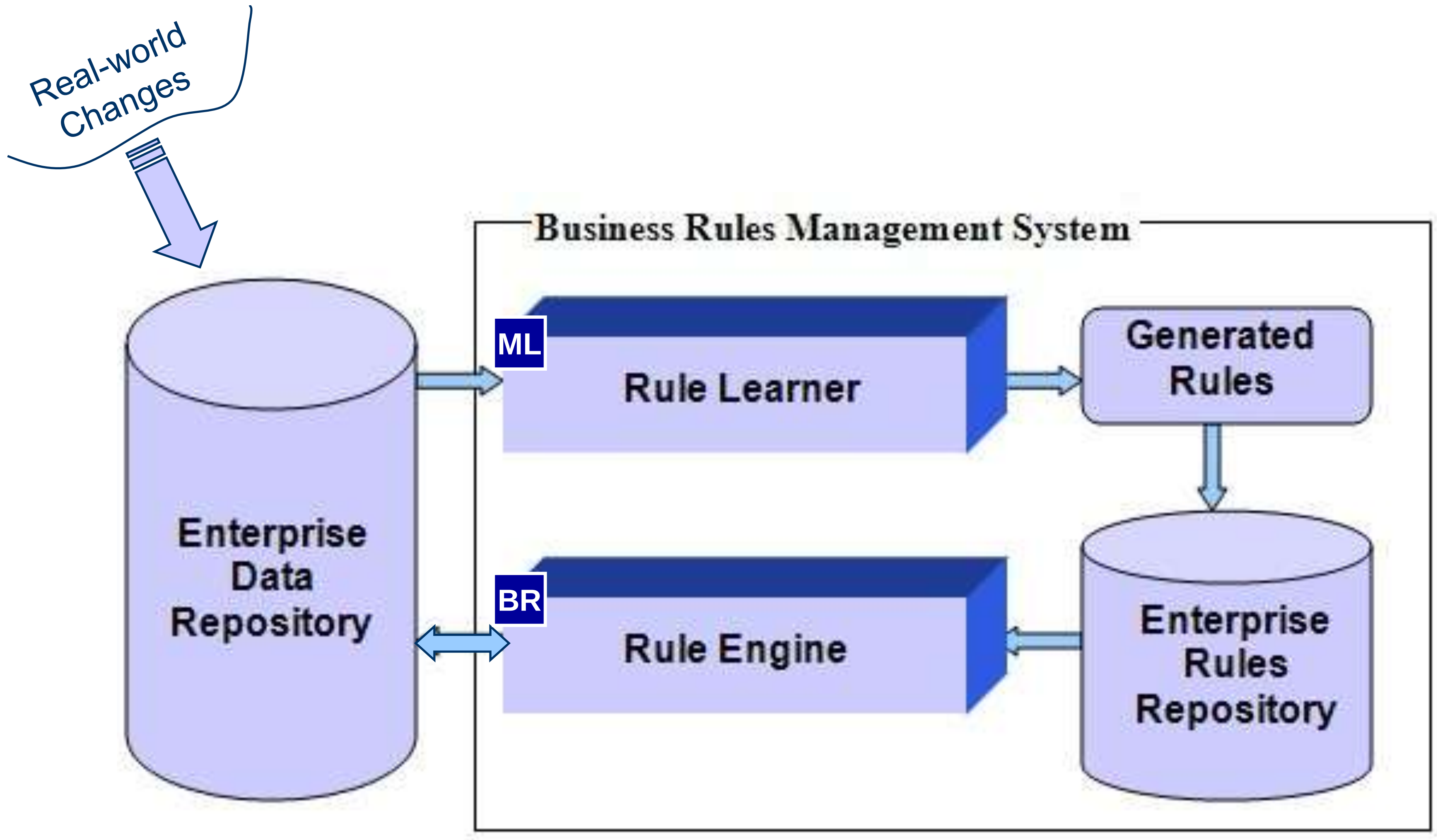
Never-Ending Rules Harvesting?

"Learning in humans and other animals is an ongoing process in which the agent learns many different capabilities, often in a sequenced curriculum, and uses these different learned facts and capabilities in a highly synergistic fashion. Why not build machine learners that learn in this same cumulative way, becoming increasingly competent rather than halting at some plateau? Can we build never-ending learners?"

Tom M. Mitchell, Carnegie Mellon University

- In the BR world this question is translated into the following:
  - **Can we build constantly-learning rule engines?**
- Real-world BR experience forces us to consider "never-ending" rules harvesting!
- To succeed over time, a rules-based system should become a "good employee" that is learning from experience and self-adjusting its own rules

High Level ML+BR Integration Scheme


Real-world Changes
Business Rules Management System
Enterprise Data Repository
ML
Rule Learner
Generated Rules
BR
Rule Engine
Enterprise Rules Repository

ML and Rule Learners

- The field of Machine Learning today covers a broad range of learning tasks, such as:
  - **how to design mobile robots that learn to navigate from their own experience**
  - **how to data-mine historical medical records to learn which future patients will respond best to which treatments**
  - **how to build customizable search engines that automatically adapt to their user's interests and many other factors.**
- ML has already produced a *rich set of learning algorithms* that are used successfully in speech recognition, computer vision, medical diagnosis, preventive maintenance, and commercial decision support
- Rule Learner is an ML-based component used in the field of data mining:
  - **To discover hidden regularities in vast volumes of data** and
  - **To present them in a form of humanly readable rules**

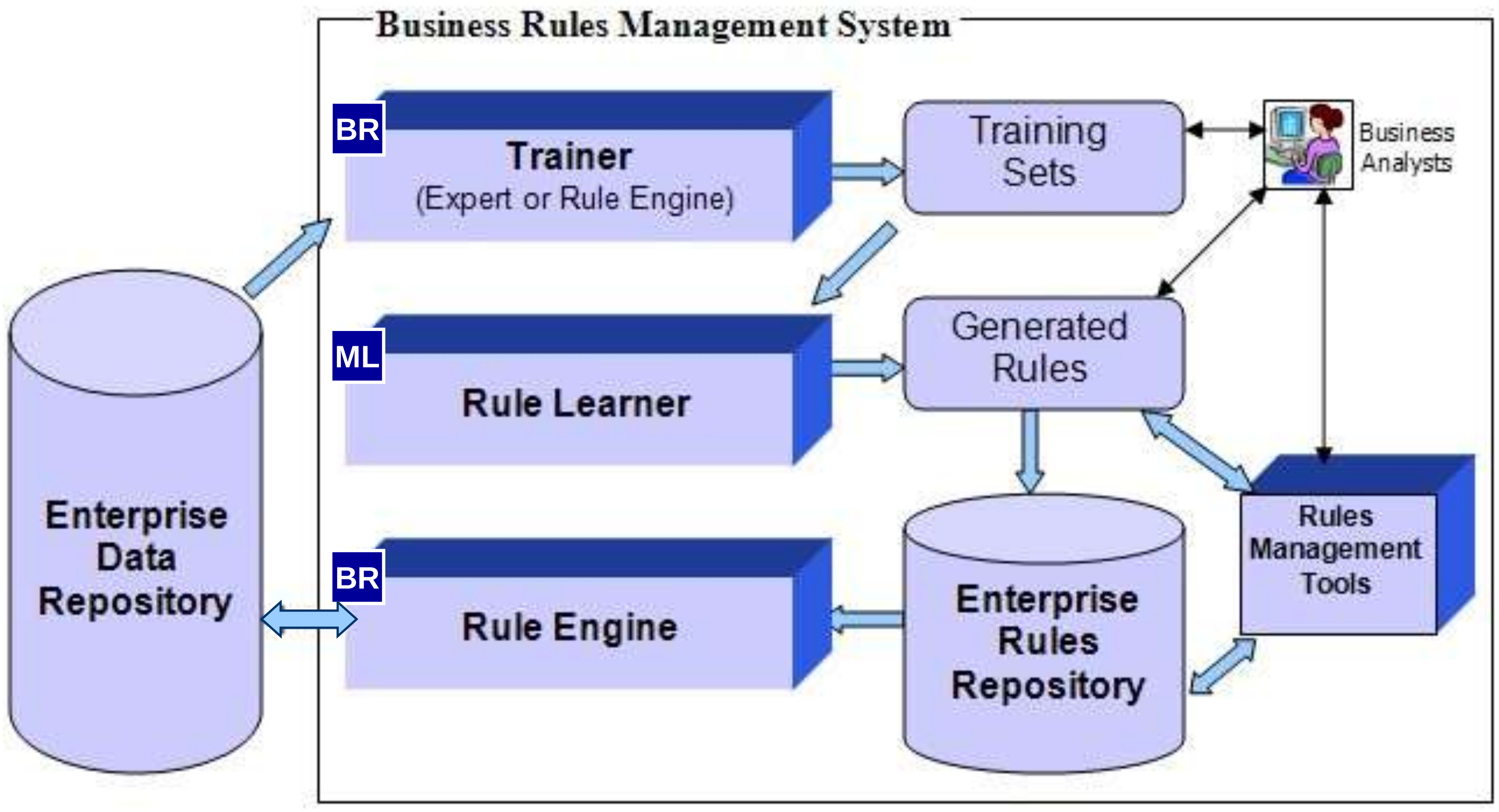
Where ML is Successful

- **Where the application is too complex for people to design the algorithm or define the processing rules manually** :
  - Example 1: All of us can easily label which photographs contain a picture of our mother, but none of us can write down an algorithm to perform this task
  - Example 2: Over 85% of handwritten mail in the US is sorted automatically using machine learning software *trained* to very high accuracy
  - Here it is relatively easy to collect labeled *training* data and relatively ineffective to try writing down a successful algorithm
- **Where the software has to be customized to its operational environment after it is fielded**:
  - Example1: Speech recognition systems that adapt to the user who purchases the software
  - Example2: Electronic stores that adapt to customer’s purchasing preferences
  - Example 3: Email readers that learn a customer’s definition of spam
  - Here Machine Learning provides the mechanism for adaptation

Supervised Learning

- **From the BR perspective we will focus on so called *supervised (or classification) learning* when a Rule Learner operates under supervision:**
  - Input: a training set of "labeled" data instances, e.g.
    - *Patients tagged with the correct diagnosis*
    - *Loan applications marked with a red flag*
  - Output: a set of learned rules that will label new (unlabeled) data instances accurately, e.g.
    - *Other patients in the selected population not diagnosed yet*
    - *New loan applications to be accepted or marked with red flags*
- **The success of learning can be judged**
  - By trying out the learned rules against test data for which the true classifications are known but not made available to the machine
  - In practical ML applications, success is measured subjectively in terms how useful the result appears to be to a human user

Adding Training Sets and Trainer



Training Sets

- **Training Set usually consist of examples (labeled data instances)**
  - Positive Examples

    indicate cases when the desired result has been achieved
  - Negative Examples

    indicate cases when the desired result has not been achieved
- **Training sets are used by Rule Learner to:**
  - discover and represent new rules
  - measure the accuracy and effectiveness of the rules once they have been learned
- **If the results are satisfactory, the rules can then be used to predict results for new previously unseen cases**
- **Training Sets are the key for learning success (or failure)**
  - Through training sets domain experts transfer their understanding of data to automatic Rule Learner
  - It is a well-known fact that with good training sets even very simple ML algorithms could produce amazingly good results

Trainer

- **Supervised learning requires a Trainer to create training sets**
- **Subject Matter Expert as a Trainer**
  - In most cases a trainer is an SME who has extensive experience dealing with the historical enterprise data and has the competence and skills to establish goals, concepts, and/or criteria for detecting patterns and rules
- **Rule Engine as a Trainer**
  - Trainer can be implemented as a special rule engine that automatically analyses large volumes of data in accordance with "*training rules*" created by SMEs and uses this data to generate new training sets
  - Automated trainer will keep a rule learner up to date with the latest changes in the enterprise data
- **In the integrated ML+BR environment it is business specialists (SMEs and NOT software developers) who are normally responsible for maintaining training sets and for evaluating the generated (learned) rules.**

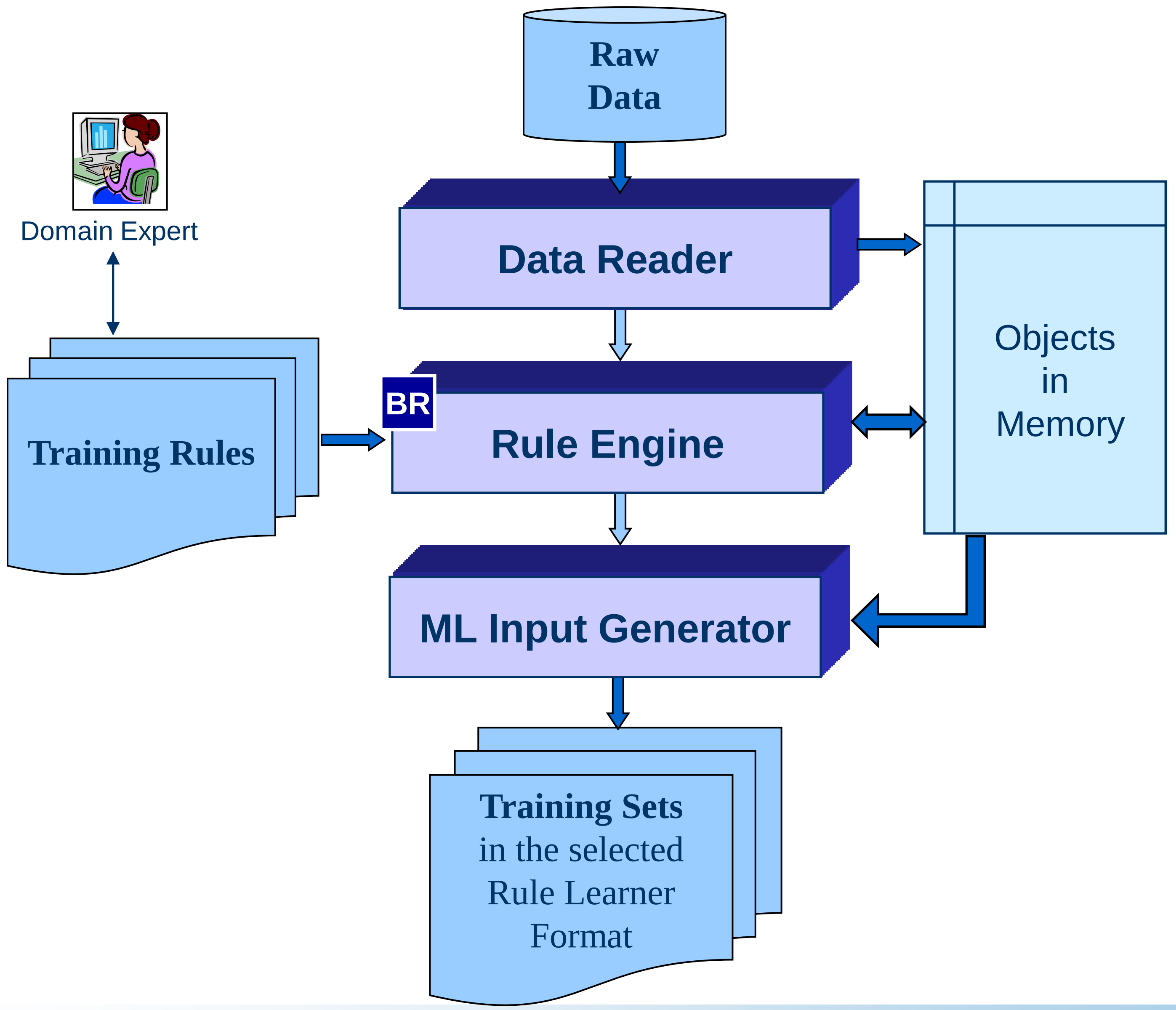
Learn Your Data

- ML algorithms may help you to learn your own data by conducting different experiments
- Automatically generated rules will almost certainly produce some errors by misclassifying certain instances
- Example:
  - Rule learner may find that "*ReadingAbility*" and "*ShoeSize*" are dependent attributes
  - If we bring an attribute "*Age*" into consideration, a rule learner will drop the relationship "*ReadingAbility - ShowSize*" in favour of "*ReadingAbilty - Age*"
- Thorough data analysis, cleaning, filtering, and selection of the right algorithm are key components of the successful machine learning
- Adding a rules-based trainer may improve essentially the quality of machine learning results

Incorporation of Domain Knowledge through Training Rules

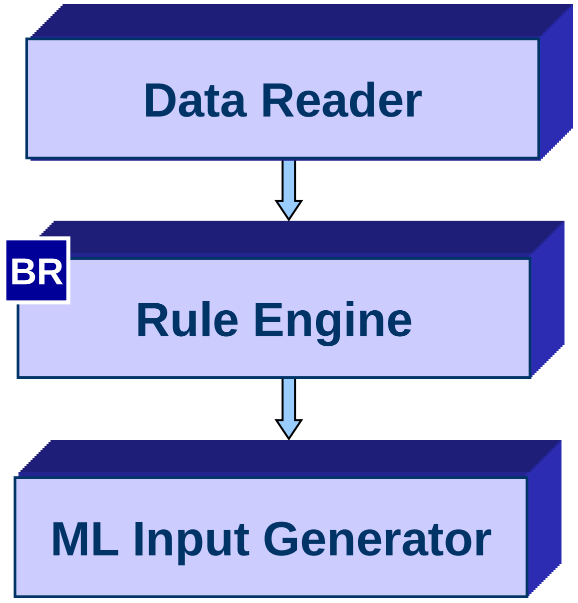
- No matter how good a machine learning algorithm is, its performance is always improved by applying domain knowledge to the problem
- Rules-based Trainer is a tool that allows domain experts to incorporate their knowledge into Rule Learner by presenting it in a form of *training rules*
- While Training Rules are domain-specific they may cover the following common data pre-processing tasks:
  - Selection of issues and attributes to be considered by a Rule Learner
  - Generating of new attributes that generalize the existing attributes by adding nominal attributes, ratios, etc.
  - Preliminary classification of issues
  - Instance filtering rules
  - Selection of a rule learner execution parameters
- Making changes in training rules domain experts effectively inform Rule Learner about the changes in the real-world environment

Generic ML Trainer Organization



Generic ML Trainer Components

- We use the word “generic” to stress the fact that the proposed Trainer could be vendor independent
- Generic “Data Reader” may be tuned to different data sources but has clearly defined objects that support different data structures
- Generic “Rule Engine” may come from different vendors but it may reuse the same object as input and output
- Generic “*ML Input Generator*” may generate training sets using different popular ML formats such as ARFF



Modern Machine Learning Algorithms

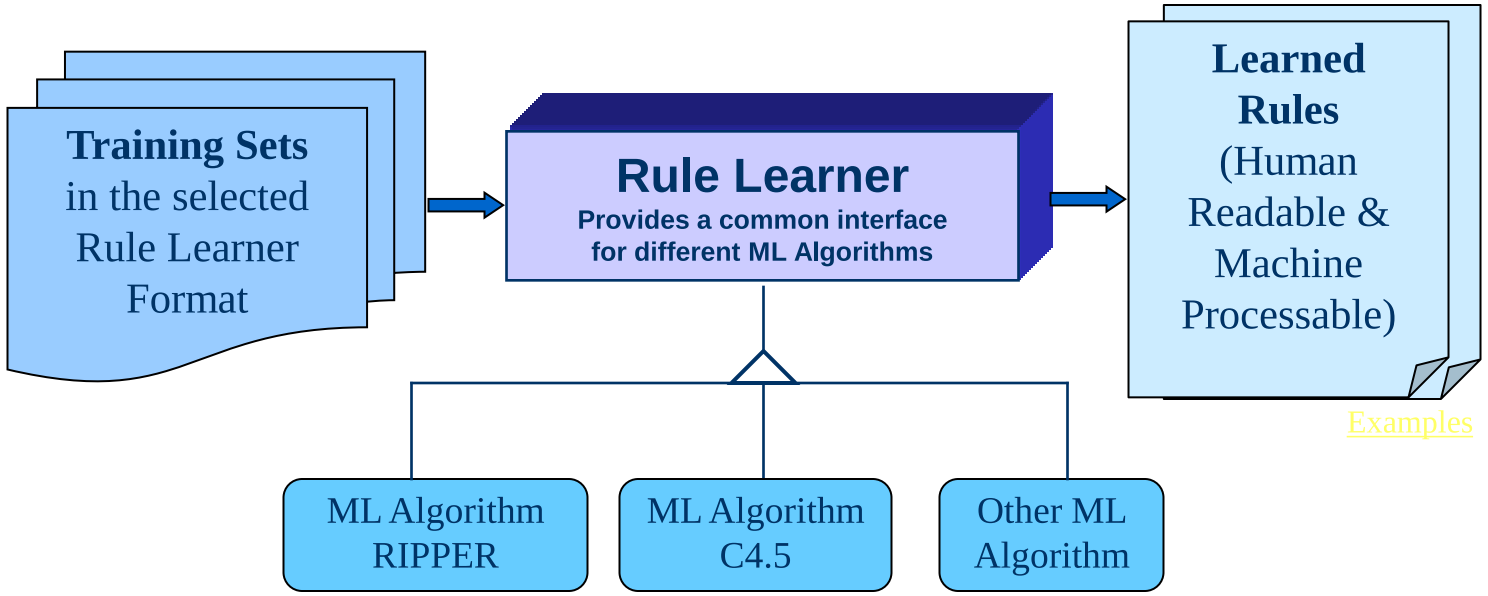
- Many different learning algorithms have been proposed and evaluated experimentally in different application domains. They are based on different ML schemas such as:
  - ***Decision Trees (e.g. C4.5)***
  - ***Classification Rules (e.g. RIPPER)***
  - ***Bayesian networks***
  - ***Instance-based learning***
  - ***Support vector machine***
  - ***Numeric prediction*, and more**

- *What is the relationship between different learning algorithms, and which should be used when?*

ML Algorithms and Rule Learner

- In spite of a variety of powerful ML algorithms, it is impossible to find one “magic” ML algorithm that fits all application domains in the best way
- A choice of the right ML algorithm is a learning process itself that highly depends on your data
- The best practical approach is to use a generic Rule Learner interface that is capable of switching between different ML algorithms based on concrete data
- We consider a popular Open Source ML toolkit “WEKA” (http://www.cs.waikato.ac.nz/ml/weka) as a good foundation for a generic Rule Learner. WEKA offers a variety of ML algorithms and allows a user to experiment with all of them while learning your own data dependencies
- There are plenty of other open source and commercial ML tools

Generic Rule Learner

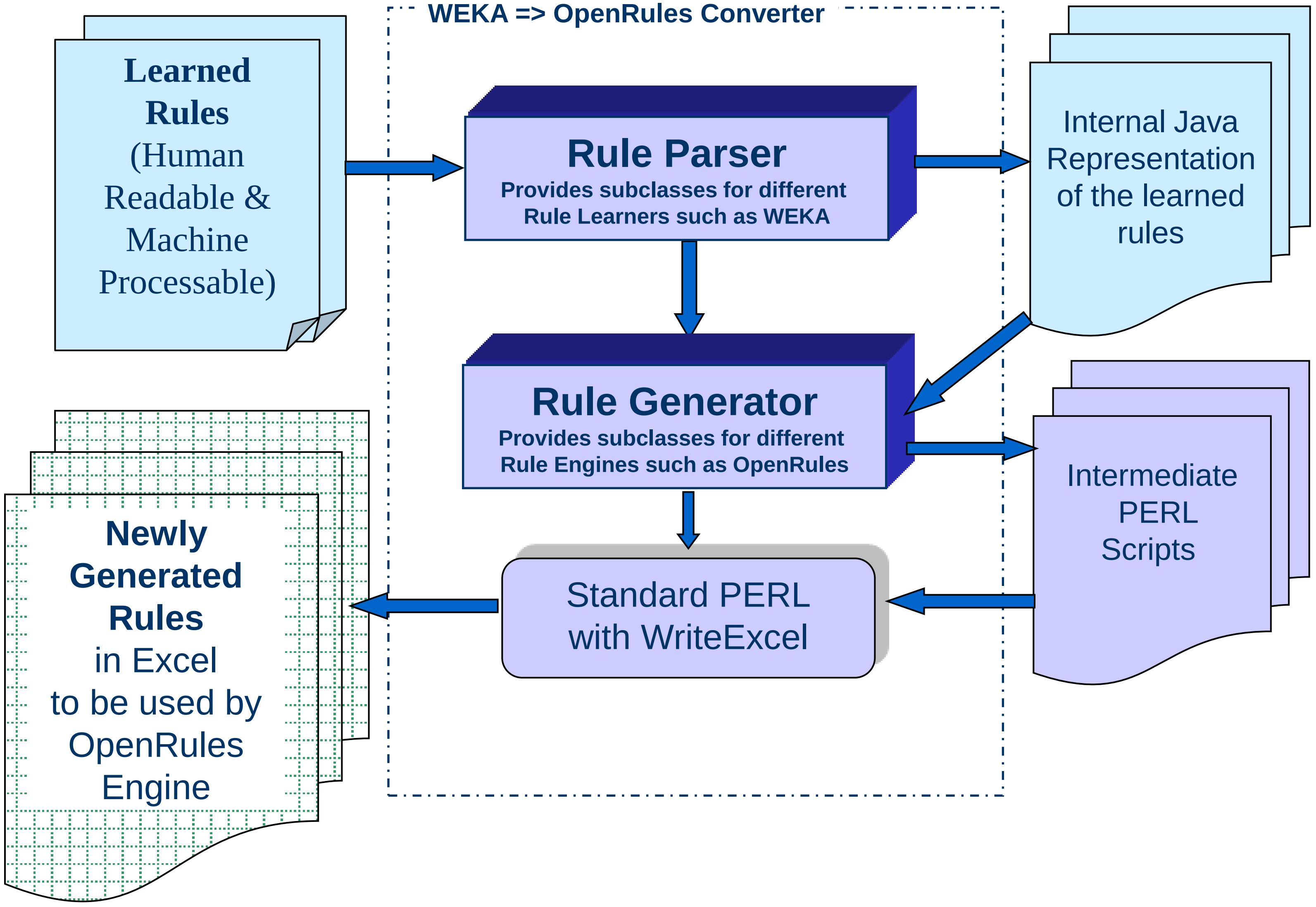


Training Sets in the selected Rule Learner Format
Rule Learner
Provides a common interface for different ML Algorithms
Learned Rules (Human Readable & Machine Processable)
Examples
ML Algorithm RIPPER
ML Algorithm C4.5
Other ML Algorithm

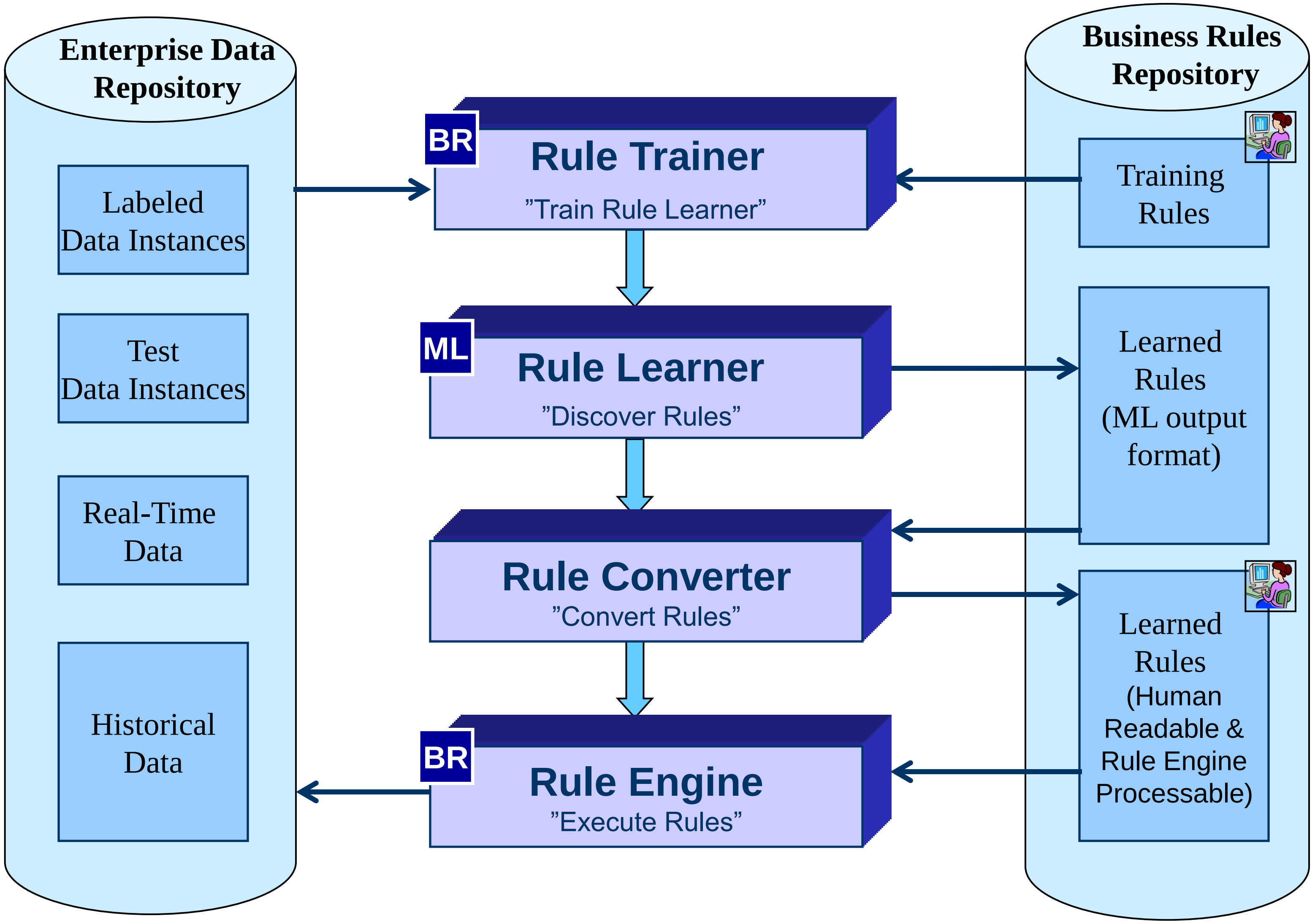
Transforming Learned Rules to BRMS

- **Classification ML algorithms usually produce rules in a human friendly form either as decision trees or as textual rules**
- **For a Rule Learner to be considered as a BRMS component it is crucial to provide an ability to *automatically* convert learned rules into a particular BRMS format**
- **The conversion of learned rules should preferably not depend on a particular vendor. The proper generic Rule Converter will execute two major steps:**
  1. **Parsing** Learned Rules into a generic intermediate format
  2. **Generating** Rules in the selected BRMS format
- **The next diagram shows an example of a concrete Rule Converter implementation that has been used successfully in real-world projects:**
  - "Rule Parser" transfer rules produced by WEKA-based Rule Learner into internal Java representation
  - "Rule Generator" uses these Java objects to generate PERL scripts and then executes PERL to produces Excel rule tables in the OpenRules format

Rule Converter Implementation


WEKA => OpenRules Converter
Learned Rules (Human Readable & Machine Processable)
Rule Parser
Provides subclasses for different Rule Learners such as WEKA
Internal Java Representation of the learned rules
Rule Generator
Provides subclasses for different Rule Engines such as OpenRules
Intermediate PERL Scripts
Standard PERL with WriteExcel
Newly Generated Rules in Excel to be used by OpenRules Engine

Resulting ML+BR Integration Schema



Integrated Framework

- The proposed ML+BR integration enables an enterprise to improve incrementally on the automatic extraction of rules and incorporation of those rules into the enterprise BRMS
- On the one hand, this system can be used as a black box. A magic wand can be waved and results produced. With no human intervention!
- On the other hand, we do not believe that this will produce the best results. Serious improvements, and for that matter, serious results can only be obtained by using an interactive framework to improve the enterprise’s domain knowledge and by incorporating that knowledge back into the framework
- This is a human-machine framework that supports a process of ongoing improvements!

No Lock to a Particulate Tool

- **The described ML+BR framework specifies and implements all necessary interfaces, while concrete underlying tools may differ**
- **All key modules are configurable:**
  - Rule Learner may use different open source or closed source machine learning algorithms and data mining tools
  - Rule Trainer and Rule Engine can be based on different open source or closed source BRMS

Example: Rules Discovered by ML

- 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 managed to convert this “gut feel” into rules with very concrete numeric thresholds*! The achieved results are comparable to the human experts and are easily interpreted by them
- Example of ML Output:

(BUSINESS_MILES_COUNTER >= 4100) and

(GROSS_RECEIPTS_AMOUNT <= 3772)

=> CAR_EXPENSE = RED

  - This car expense rule finds a solid relationship: as the number of business miles increases, so should gross receipts
  - Rule Learner selected these two attributes out of hundreds of other attributes considering around 50K data instances

Example: Learned Rules Converted to BRMS (OpenRules)

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

Example: ML-based Rules Compression

Rules void classifyInstance(TestInstance instance)

IF type is	AND adjust ment > $$$	AND adjust ment < $$$	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

ML Generated Rule:

IF amount <= -159

THEN classifiedAs=TOP

ELSE classifiedAs=BOTTOM

15 rules are replaced by 1!

Correctly Classified Instances:
2395 out of 2396

Conclusion

- ML+BR integration brings immediate improvements to BRMS by supporting never-ending rules harvesting
- The proposed ML+BR integration schemas allows:
  - Rules-based training for supervised rule learners
  - Automatic generation of new rules and adjustment of the existing ones by taking advantage of the steady stream of data flowing through BRMS
  - Automatic conversion of ML output into BR input
  - Rules compression and optimization
- Incorporation of Rule Learners into a BRMS is a natural step for converting rules-based decision engines into good self-learning “employees”
- We expect that, soon, rule learners will become a standard component of most BRMS

References

- ***The Discipline of Machine Learning, Tom M. Mitchell,*** *July 2006 CMU-ML-06-108* ***http://www.cs.cmu.edu/~tom/pubs/MachineLearningTR.pdf***
- **Data Mining: practical machine learning tools and techniques/ Ian H. Witten, Eibe Frank – 2nd ed., Morgan Kaufmann, 2005. http://www.cs.waikato.ac.nz/ml/weka/**
- ***Business Rules Management and Service Oriented Architecture: A Pattern Language*****, Ian Graham (Wiley, 2006)**
- **Ian Graham, "Improving Decision Table Rules with Data Mining,"** ***Business Rules Journal*****, Vol. 8, No. 3 (March 2007), http://www.brcommunity.com/b334.php**
- ***OpenRules Rule Learner*** ***http://www.openrules.com/RuleLearner.htm***