Paper SAS579-2017

Use Machine Learning to Discover Your Rules

David R. Duling, SAS Institute Inc.

ABSTRACT

Machine learning is not just for data scientists. Business analysts can use machine learning to discover rules from historical decision data or from historical performance data. Decision tree and logistic regression scorecard learning are available for standard data tables, and associations' analysis is available for transactional event tables. These rules can be edited and optimized for changing business conditions and policies, and then deployed into automated decision making systems. Users will see demonstrations using real data and learn how to apply machine learning to business problems.

INTRODUCTION

Business rules are a dominant feature in automated business decision making systems. These rules encode the policies and procedures of business operations in all fields including processing credit applications, insurance claims, debt collections, airline, and hotel reservations, factory scheduling, data quality and remediation, sales revenue attribution, auto repossessing, truck maintenance scheduling, national customs import/ export forms, and even managing training programs at NASA. To say that the list is unlimited would be limiting.

However, there are several challenges to using business rules for complex scenarios. The set of rules grows rapidly as new business conditions are encountered and maintaining the rules can be laborious and inefficient. These systems are observed to contain several thousands of terms and rules. Manual generation of business rules is both time consuming and error prone. Many rules become redundant and add confusing complexity to the system. Each rule requires domain expertise to understand the relationship between conditions and actions. The execution of an excessively large set of rules can create run-time performance issues. The final result is that in some cases the business is unsure what results are produced by the business rules system.

Machine learning algorithms can be used to discover patterns in historical or generated data that can often be used to create new business rules. Modern algorithms have a variable selection behavior that rejects redundant or rarely useful variables in favor of terms that have more broad impact on the model. Using fewer terms in the rules makes them easier to maintain and more efficient to deploy to run-time systems. Rule sets can be learned much more quickly than developed by manual data entry. The basis set of rules can be easily modified by rule developers and extended to cover special cases or rare events as needed. Custom rules can be added as separate rule sets so that the machine learning rule sets can be relearned when new data is available.

This paper demonstrates the theory and practice of creating business rules from machine learning algorithms using SAS® software. The SAS® Business Rules Manager product has been available since 2012 and has been implemented in locations around the world.

BUSINESS RULES

The main attraction of business rules systems is that the simple form can be edited and maintained by business analysts who have limited technical programming experience. These users typically interact with spreadsheets or a dedicated rule editing system. The execution of the rules is managed separately.

Business rules are also directly interpretable. Run-time systems should record which rules were fired for each transaction. These records can be used to explain decisions when the results are challenged or improvement is needed.

The basic form of a business rule is relatively simple. It is common to use the names *condition terms* and *action terms* for input variables and output variables, respectively. The local collection of *terms* is called a *vocabulary*. In Table 1 we can see a single rule for determining the maintenance needs of a delivery truck.

ID	Conditions		Actions
	Trip Distance	Vibration	Maintenance Flag
1	>= 170 and < 197	>= 250	1

Table 1. A Typical Small Business Rule

This rule is evaluated as the following run-time code:

```
IF Trip_Distance >= 170 and Trip_Distance < 197 and Vibration >= 250 THEN
Maintenance_flag= 1;
```

Each evaluation of a condition is named an *expression*, for example: *Vibration >= 250.* These expressions can become very complex and contain many functions commonly found in the SAS language. A single rule can have dozens of expressions. Hundreds of related rules sharing a common set of input terms are packaged into a *rule set.*

A business rules management system also requires an execution environment where these rules can be evaluated in both large data batch processing jobs and in online web services used for transaction processing. The simple form of business rules makes them ideal for portable execution in many environments.

IMPLEMENTATION

Developing business rules manually requires developing extensive design documents, defining a vocabulary, and then carefully organizing and entering rules that match the design specification. Developing business rules by machine learning requires a very different process represented by the following steps. These are standard steps for any analytical or data mining process.

1. Identify the objectives for the rules. What business decisions do you need to make? In our example we need to identify and recommend trucks for maintenance. We have to balance the decisions by the seriousness of the problem and the impact on overall deliveries.
2. Identify what input data (*conditions*) will be available for making these decisions in the operational environment.
3. Identify stored data that represents a history of the conditions in representative situations. If this data is not available, can it be simulated? Can it be approximated by similar data?
4. Identify stored data that represents a corresponding set of decisions that have been made. If this data is not available, can the conditions data be labeled with ideal decisions?
5. Select and use the appropriate machine learning technique. For problems where the historical or ideal decisions are known, we recommend starting with a basic decision tree algorithm. The output will be a model form that can be converted to a set of business rules.
6. Convert the model into a set of rules and load them into the business rules management system.
7. Review the business rules. Identify and revise rules that might violate business policy or governance regulations. Create new rule sets to add conditions and expressions that as needed.
8. Test the rules on historical data and samples of new data to validate that the results generated are providing the intended value.

Following these steps can quickly produce useful and maintainable business rules.

SAMPLE DATA

We are using a variation of the same sample data that was used in the 2016 SAS Global Forum paper on deploying models to SAS® Event Stream Processing. In that case, we deployed the model score code directly to the event stream engine. In this case we are generating business rules that can be modified and deployed to any SAS execution engine. The measurements represent data that was collected in real time from each truck during its routes. During routine maintenance checks, each truck was labeled with a

Maintenance_flag if it required any maintenance work. The data contains 8305 records from 54 trucks divided into 12 fleets. The complete vocabulary is shown in the following table.

Binary Response Variable		
Maintenance_flag		
Numeric Measurement Variables (&VARS)		
Accel_Pedal_Pos_D	Ambient_air_temp	Accel_Ssor_Total
CO2_in_g_per_km_Inst	CO_emission	Emission_alert
Engine_Coolant_Temp	Engine_Load	Engine_Oil_Temp
Engine_RPM	Engine_control	Fog_control
Fuel_level	GPS_Altitude	GPS_Bearing
GPS_Latitude	GPS_Longitude	HC_emission
Intake_Air_Temp	Intake_Manifold_Pressure	Litres_Per_100km_Ins
Mass_Air_Flow_Rate	NOx_emission	Oil_life
PM_emission	Speed_GPS	Speed_OBD
Throttle_Pos_Manifold	Trip_Distance	Trip_Time_journey
Turbo_Boost_And_Vcm_Gauge	Turbo_alertVehicle_speed_sensor	
Vibration	Vibration_alert	Vibration_alert_recent
Voltage_Control_Module	Vibration_alert_since_start	

Table 2. The Complete Vocabulary for the Sample Data

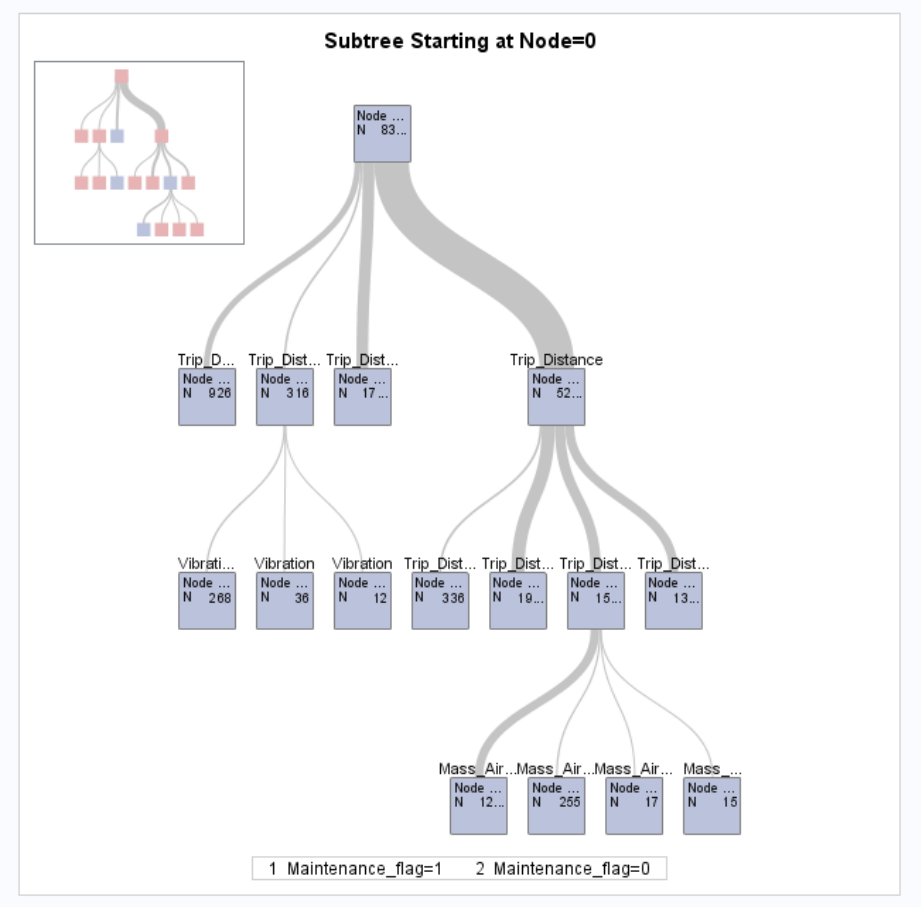
DECISION TREE

The most common machine learning models for producing business rules are the family of decision tree models. These are supervised models that require a dependent variable, also known as a target or response, and a set of input variables. Decision trees trace their lineage to work by professors Leo Breiman and Jerome Friedman. SAS has implemented decision tree algorithms in SAS® Enterprise Miner™, SAS/STAT® software, and SAS® Visual Data Mining and Machine Learning. These models are sometimes called rule learning models or recursive partitioning since they implement extremely efficient algorithms for discovering the best rules that partition the data into small 'nodes' that have a similar response value. The following SAS code builds a simple tree on the truck maintenance data.

```
PROC HPSPLIT data =sgf2017.fleet_train maxleaves=100
maxbranch=4 leafsize=10 intervalbins=20 mincatsize=10
missing=popularity ;
      criterion fastchaid ;
      target Maintenance_flag / level=nom ;
run ;
```

One of the nice properties of decision tree algorithms is that rules are only created for variables that have a statistically significant effect on the accuracy of the model. In this final model only three variables are selected: *Trip_Distance, Vibration*, and *Mass_Air_Flow_Rate*. The tree model has 12 leaves, which are labeled as either predicted Maintenance_flag is 1 (true) or 0 (false). The top of the tree represents all rows of data. Each branch is a single-variable WHERE clause based on *Trip_Distance*, *Vibration*, or *Mass_Air_Flow_Rate*.

Output 1 displays the resulting decision tree. The top node represents all the data. Each branch represents a rule expression such as where 250 < Trip_Distance < 300. Each path from the top root node to a terminal leaf can be converted into a complete business rule. The conditions of the rule are the terms and expressions that define each branch. The action of the rule is the prediction made in the terminal leaf. Rules sets can easily be generated for this model. These rule sets are easier to manage, modify, and interpret than the corresponding score code generated by the algorithm.



Output 1. Decision Tree Model for Generation of Business Rules

We can also see how this result is more efficient than rules manually created by humans. The resulting twelve rules are stored in the application database as shown in Table 3.

ID	Conditions			Actions
	Trip Distance	Vibration	Mass Air Flow Rate	Maintenance Flag
1	< 161.92097			0
2	>= 170.71429 AND < 197.09424			1
3	>= 161.92097 AND < 170.71429	<248.54914 OR MISSING		0
4	>= 161.92097 AND < 170.71429	>= 248.54914 AND < 250.61854		0
5	>= 161.92097 AND < 170.71429	>=250.61854		1
6	>= 197.09424 AND < 205.88755			0
7	>= 205.88755 AND < 267.44077			0

	OR MISSING			
8	>=293.82072			0
9	>= 267.44077 AND < 293.82072		<29.4775 OR MISSING	1
10	>= 267.44077 AND < 293.82072		>= 29.4775 AND < 70.746	0
11	>= 267.44077 AND < 293.82072		>= 70.746 AND < 82.537	0
12	>= 267.44077 AND < 293.82072			0

Table 3. Rules Created by Decision Tree Algorithm

These business rules would execute correctly in a decision making application; however, we want to make changes to the rules to fit our application. The first observation is that the nine rules that result in an action of *Maintenance_flag = 0* can be discarded, leaving three useful rules. The next observation is that the model only considered data where Trip Distance is less than 293.8072. Within the final rules we can eliminate < 293.8072 from any expression. Finally, we want to round any numbers to make the rules more readable and maintainable without significantly affecting the final decisions. The resulting set of rules is shown in Table 4.

ID	Conditions			Actions
	Trip Distance	Vibration	Mass Air Flow Rate	Maintenance Flag
2	>= 170 AND < 197			1
5	>= 161 AND < 170	>=250		1
9	>= 267		<29 OR MISSING	1

Table 4. Final Rules after Editing

This final set of rules is likely to be much more efficient and effective than a set of rules created manually by independent analysis and expertise. If additional conditions are uncovered, they can be incrementally added to the system.

To illustrate the process, we have shown the details of the individual steps. This process is also embedded into SAS Business Rules Manager. The following examples will use the software directly and eliminate some details for brevity.

SCORECARD

Another common analytical technique appropriate for business rules generation is an analytical scorecard. These models are used in fields such as credit scoring and retail marketing where a simple expression with a numerical result and complete interpretability is needed. These are also supervised models that require a set of input variables and a single categorical target variables.

The basic form is a scorecard is a set of rules that adds a score value to the score total. Each rule is based on some data feature. The final score is the sum of the score values of all the rules. Generally, a higher score indicates greater importance or likelihood, such as greater need for truck maintenance. The

following table shows a few sample rules based on the same sample data. In this case, the rules are listed vertically.

Id	Type	Condition	Expression	Action
1	IF	Vibration	= MISSING	Scorepoints+148
2	ELSE	Vibration	<= 244	Scorepoints+268
3	ELSE	Vibration	>244 and <=300	Scorepoints+159
4	ELSE	Vibration	>300	Scorepoints+100
5	IF	Trip Distance	<=260	Scorepoints+137
6	ELSE	Trip Distance	>260 and <=400	Scorepoints+160
7	ELSE	Trip Distance	>400	Scorepoints+190

Table 5. Sample Scorecard Rules

Scorecards can be created either manually or by machine learning. The most common function of learning the terms in a scorecard is an algorithm based on transformations and logistic regression. The basic steps of the computation are described by this list.

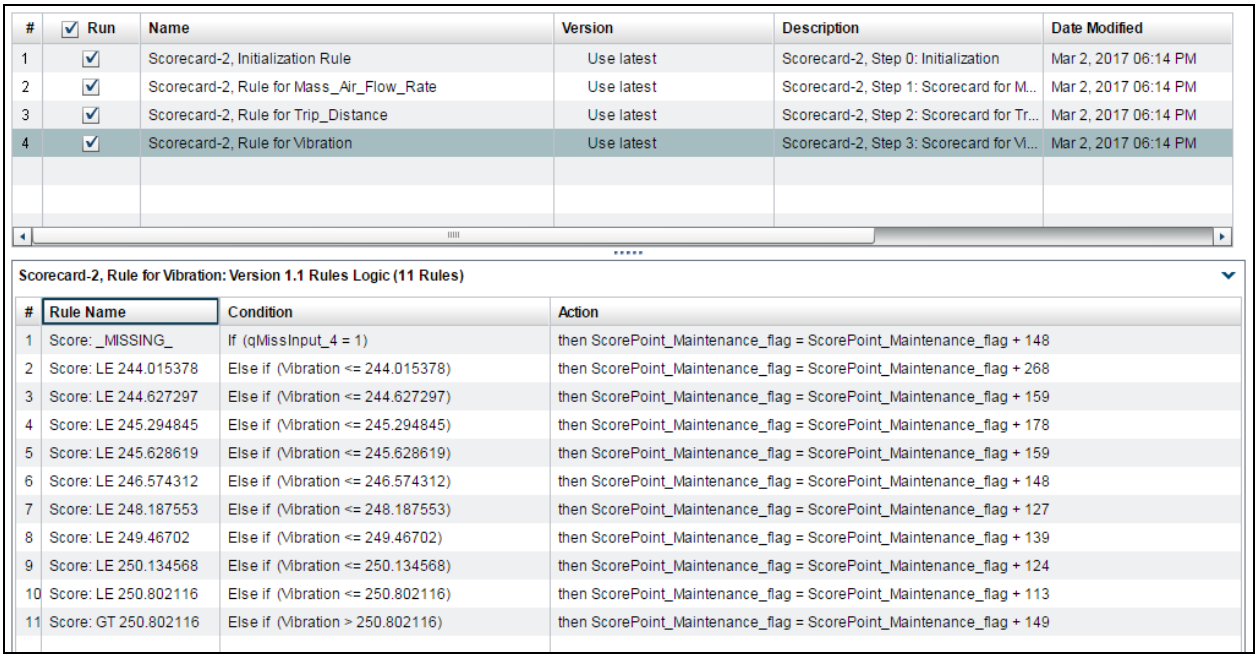
1. Discretize each numeric attribute. In the rules shown in Table 5, the variable Vibration has been binned into 4 different buckets. There are many functions for discretizing data and SAS Business Rules Manager reuses the decision tree algorithm.
2. Compute a Weight of Evidence (WOE) variable for each discretization. WOE is computed as the log odds ratio of the target variable values for each discretized value of each input variable. In the sample of rules shown above, we will produce a WOE variable for Vibration and a WOE variable for Trip Distance.
3. Compute a logistic regression on the data where the WOE variables are used as inputs. This will result in a set of coefficients, *Beta,* for the WOE variables and an *intercept*.
4. Finally, we compute the score points by the following steps. Min and Max are defined as the expected minimum and maximum scorecard values, i is the index of the input variable, j is the index of the bins of the input variable, and k is the number of variables.
    a. Score(i,j) = WOE(i,j) * *Beta*(i) + *intercept* / k — raw score for each input variable bin
    b. MaxScore(i) = MAX(Score(i,j) — max score for each input variable
    c. MinScore(j) = MIN(Score(i,j)) — min score for each input variable
    d. MaxScore = SUM(MaxScore(i)) — max possible raw score total
    e. MinScore = SUM(MinScore(i)) — min possible raw score total
    f. Factor = (Max – Min) / (MaxScore – MinScore) — score scaling factor
    g. Offset = Max – Factor * MaxScore — score offset
    h. ScorePoint(i,j) = Factor * Score(i,j) + Offset / k — final score points for each variable bin.

The end result is a scorecard that assigns points for each variable value and a score point total for each row of data. Within SAS Business Rules Manager, the user follows these steps.

5. Select an input table with rows of data for learning the rules.
6. Select the min and max values of the score card. The default values are 200 and 1000.
7. Select the target variable, in this case *Maintenance_flag*.
8. Select the input variables. For comparison purposes, we selected only *Vibration*, *Trip_Distance*, and *Mass_Air_Flow_Rate*.

The software will now execute the machine learning process for constructing a scorecard. In addition to the steps listed above, the software will run multiple variable selection steps run to produce a scorecard with a reasonable number of rules. Display 1 shows the final set of rules in the user interface. One rule set has been generated for initialization and three rule sets have been generated for the input variables. The details of the rule set for V*ibration* are also displayed.

Since the scorecard has been entered into the business rules software, users can now modify the rules and extend them with additional custom logic for their decision making processes.



#	Run	Name	Version	Description	Date Modified
1	☑	Scorecard-2, Initialization Rule	Use latest	Scorecard-2, Step 0: Initialization	Mar 2, 2017 06:14 PM
2	☑	Scorecard-2, Rule for Mass_Air_Flow_Rate	Use latest	Scorecard-2, Step 1: Scorecard for M...	Mar 2, 2017 06:14 PM
3	☑	Scorecard-2, Rule for Trip_Distance	Use latest	Scorecard-2, Step 2: Scorecard for Tr...	Mar 2, 2017 06:14 PM
4	☑	Scorecard-2, Rule for Vibration	Use latest	Scorecard-2, Step 3: Scorecard for Vi...	Mar 2, 2017 06:14 PM

Scorecard-2, Rule for Vibration: Version 1.1 Rules Logic (11 Rules)

#	Rule Name	Condition	Action
1	Score: _MISSING_	If (qMissInput_4 = 1)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 148
2	Score: LE 244.015378	Else if (Vibration <= 244.015378)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 268
3	Score: LE 244.627297	Else if (Vibration <= 244.627297)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 159
4	Score: LE 245.294845	Else if (Vibration <= 245.294845)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 178
5	Score: LE 245.628619	Else if (Vibration <= 245.628619)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 159
6	Score: LE 246.574312	Else if (Vibration <= 246.574312)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 148
7	Score: LE 248.187553	Else if (Vibration <= 248.187553)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 127
8	Score: LE 249.46702	Else if (Vibration <= 249.46702)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 139
9	Score: LE 250.134568	Else if (Vibration <= 250.134568)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 124
10	Score: LE 250.802116	Else if (Vibration <= 250.802116)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 113
11	Score: GT 250.802116	Else if (Vibration > 250.802116)	then ScorePoint_Maintenance_flag = ScorePoint_Maintenance_flag + 149

Display 1. Scorecard Rules Displayed in SAS Business Rules Manager

MARKET BASKETS

In cases where a target variable is not present, we can use a common association analysis algorithm to find sets of conditions that frequently co-occur and their most likely resulting action. The famous, and probably fictional, case of baby diapers and beer comes to mind. This analysis requires transactional data containing discrete event codes. A good example is a history of sales records where each transaction contains one or more items. The algorithm will find sets of products that are sold together more frequently than expected and naturally produces a set of rules for providing product recommendations. The following table shows one set of sample rules.

ID	Conditions			Actions
	Item1	Item2	Item3	recommend
1	Bread	Fish	Dessert	Wine
2	Bread	Eggs	Cheese	Juice
3	Bread	Meat	Cheese	Soda

Table 6. Sample Market Basket Rules

These rules can be interpreted with the following code:

```
IF Item1 eq 'Fish' and Item2 eq 'Bread' and Item3 eq 'Dessert'
then recommend = 'Wine' ;
IF Item1 eq 'Bread' and Item2 eq 'Eggs' and Item3 eq 'Cheese'
then recommend = 'Juice' ;
```

```
IF Item1 eq 'Bread' and Item2 eq 'Meat' and Item3 eq 'Cheese'
then recommend = 'Soda' ;
```

Machine learning includes a class of algorithms targeted at frequent item sets analysis. One of the most common is the Apriori algorithm, which can be found in the SAS Enterprise Miner. The algorithm produces rules with multiple conditions, multiple actions, and numeric measures of rule importance known as Support and Confidence. Support is defined as the percentage of transactions that contain the basket of items. Confidence is defined as the percentage of transactions that contain the condition items which also contain the action items. Analysts often look for rules that have a high confidence and low support, indicating they are both rare and reliable. These rules represent opportunities that might not have been previously noticed.

Market baskets are also used in SAS Business Rules Manager to produce business rules. For this sample we will use a different data set. The MSEQ2 data set represents a set of customer events at a retail bank. These events include both personal events such as the birth of a child or a new job, and also banking events such as opening a new checking account. The table contains three columns: Datetime, Customer ID, and Action. Overall, the data contains 164 customers, 10 different events, and only 564 transactions. However, this data is large enough to illustrate market basket rule discovery.

Within SAS Business Rules Manager, the user follows these steps.

9. Select an input table with rows of data for learning the rules.
10. Select the ID variable and the Target variable.
11. Select the maximum number of rules. This parameter is needed because the market basket algorithm can create a very large number of rules when analyzing data with a large number of transactions and items. The rules returned are sorted by descending confidence.

Due to the high number of items and the relatively small size of the rules, the discovered rule sets are very sparse. The positive result is that rules with smaller numbers of items are easier to manage; however, sparse rules are also difficult to visualize and manage. We use a combination of proprietary clustering techniques, which are also machine learning algorithms, to group the rules into more homogeneous rule sets to simplify the rule editing and management for end users.

For execution efficiency, a new temporary variable is created for each possible condition item and for each possible action item. Some items can appear as both condition items and action items. Market basket rules are executed with SAS by group processing on a stream of transactional data. The execution flow can be described as the following process.

12. In the first.customer block, a rule set is executed that initializes the value of each temporary condition item flag variable to the value false. An example is: *C_new_job= 0;* which indicates a new job has not been reported for this customer. Temporary action item flag variables are also set to false; an example is *A_safe_deposit=0*;
13. In the main block, the temporary item flag variables are set to true if the item is detected. An example is*: IF item eq 'new_job' then C_new_job=1;* which indicates that a new job has been reported.
14. In the last.customer block, the market basket rules are applied to the temporary item flag variables. An example is: *IF C_new_job and C_money_market then A_safe_deposit=1;*

This sample analysis produced 140 unique rules divided into eight clusters. For each customer ID, one or more rules fire to create new recommendations (actions). Display 2 shows the last.customer rules associated with cluster number 5 of the sample analysis.

	R..	Operator	Term	Value
▼	I...			
		IF	C_new_job	= 1
		THEN	SUPPORT	= 11.21
		SET	CONFIDENCE	= 100
		SET	RULE_CLUSTER_INDEX	= 5
		SET	A_credit_card	= 1
		SET	A_mutual_fund	= 1
▼	I...			
		IF	C_IRA	= 1
		AND	C_money_market	= 1
		THEN	SUPPORT	= 14.66
		SET	CONFIDENCE	= 100
		SET	RULE_CLUSTER_INDEX	= 5
		SET	A_new_home	= 1
		SET	A_safe_deposit	= 1
▼	I...			
		IF	C_IRA	= 1
		AND	C_open_loan	= 1
		THEN	SUPPORT	= 10.34
		SET	CONFIDENCE	= 100
		SET	RULE_CLUSTER_INDEX	= 5
		SET	A_new_home	= 1
		SET	A_safe_deposit	= 1

Display 2. Market Basket Rules Displayed in List Format

ADDITIONAL MODELS

Advanced users can create business rules using alternative machine learning algorithms that can be converted to business rules. SAS Business Rules Manager includes SAS macros for importing and exporting rules flows. The transport format is comma separated value (CSV) files that can be easily generated and edited in SAS code, text editors, and Microsoft Excel spreadsheets. Users can execute their own machine learning algorithms and convert the models into business rules encoded in the CSV format recognized by the software. The macros and CSV format are described in the software documentation.

CONCLUSION

While business rules are ubiquitous in decision making applications, creating, maintaining, and deploying large numbers of rules can be daunting and inefficient. Machine learning can help with this process by generating rules based on analysis of either historical data or managed data. These rules can be used to bootstrap a new decision management system, or to adapt to changing conditions found in new data. We have shown how to use three very different machine leaning algorithms – decision tree, scorecard, and market baskets – to generate rules for different common business scenarios. All three techniques are embedded into SAS Business Rules Manager software. The resulting business rules are more easily interpretable and manageable than the original models. The simple form of the rules can be easily deployed to batch and real-time operational systems.

REFERENCES

Agrawal, Rakesh, and Ramakrishnan Srikant. 1994. “Fast algorithms for mining association rules in large databases.” *Proceedings of the 20th International Conference on Very Large Databases*. Pages 487-499. San Francisco, CA: Morgan Kaufmann Publishers Inc.

Breiman, Leo, Jerome Friedman, Charles J Stone, and R. A. Olshen. 1984. *Classification and Regression Trees*. Chapman and Hall/CRC.

Date, C. J. 2000. *What Not How: The Business Rules Approach to Application Development*. Addison-Wesley Professional.

Davenport, Thomas H, and Rich Masi. 2016. "'Printing Money' with Operational Machine Learning." DataInformed. Available at http://data-informed.com/printing-money-with-operational-machine-learning/

deVille, Barry, and Padraic Neville. 2013. *Decision Trees for Analytics Using SAS® Enterprise Miner™*. Cary, N.C.: SAS Institute Inc.

Guruswamy, Karthik. 2015. "Data Science: Machine Learning Vs. Rules Based Systems." *Forbes*. Available at https://www.forbes.com/sites/teradata/2015/12/15/data-science-machine-learning-vs-rules-based-systems/#283492b02119

McNeill, Fiona, David Duling, and Stephen Sparano. 2016. "Streaming Decisions: How SAS® Puts Streaming Data to Work." *Proceedings of the SAS Global 2016 Conference.* Cary, NC: SAS Institute Inc. Available at http://support.sas.com/resources/papers/proceedings16/SAS6367-2016.pdf

Siddiqi, Naeem. 2006. *Credit Risk Scorecards: Developing and Implementing Intelligent Credit Scoring.* Hoboken, N.J. John Wiley and Sons, Inc.

Talbot, Marc. 2016. Using Machine Learning to Derive Business Rules. Requirements Blog. Available at http://www.seilevel.com/requirements/using-machine-learning-derive-business-rules

Xinli Bao, Ph.D. 2007. "Mining Transaction/Order Data using SAS® Enterprise Miner™ Association Node." *Proceedings of the SAS Global 2007 Conference*. Available at http://www2.sas.com/proceedings/forum2007/132-2007.pdf

ACKNOWLEDGMENTS

The author wishes to thank Ming-Long Lam, Yan Gao, and Jianping Xue of SAS Institute Inc.

RECOMMENDED READING

- *SAS® Business Rules Manager 2.2 Users Guide*
- *SAS® Enterprise Miner™ 14.2: Reference Help*
- *Developing Credit Scorecards Using Credit Scoring for SAS® Enterprise Miner™*

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

David R. Duling
SAS Institute Inc.
David.Duling@SAS.com
https://www.sas.com/en_us/software/decision-management/decision-manager.html