# Scorecard Element in PMML 4.1 Provides Rich, Accurate Exchange of Predictive Models for Improved Business Decisions

Andrew Flint
FICO
200 Smith Ranch Road
San Rafael, CA 94903
+1 (415) 446 6632
AndrewFlint@fico.com

Alex Guazzelli
Zementis, Inc.
6125 Cornestone Court East, Suite 250
San Diego, CA 92121
+1 (858) 330 0780 x1011
Alex.Guazzelli@zementis.com

## ABSTRACT
This paper illustrates the dedicated Scorecard element introduced in the 4.1 specification of the PMML standard, including the various design and computational options available for returning reason codes alongside each computed score. The paper is intended to help both producers and consumers of scorecards as PMML documents.

## Categories and Subject Descriptors
H.2.8 [Database Management]: Data Mining; G.3 [Probability and Statistics]: Statistical Computing, Statistical Software; I.5.1 [Models]: Statistical Models, Neural Nets.

## General Terms
Algorithms, Management, Documentation, Standardization, Languages.

## Keywords
Scorecards, PMML, Predictive Model Markup Language, Predictive Analytics, Data Mining, Decision Management, Credit Scoring, Reason Codes, Adverse Action Reasons.

## 1. INTRODUCTION
Scorecards represent a potent and commercially significant formulation of predictive models, used extensively in retail banking to estimate and rank-order consumer credit risk. As a generalized additive model using binned representations of its input fields, the scorecard formulation is also frequently used to predict other types of binary and continuous outcomes, including marketing response, customer attrition, fraud detection, revenue potential, and insurance premium loss ratios.

In these business contexts, scorecards are valued for their predictive strength, simplicity and transparency, making the derived formula more immediately interpretable than other model types. Frequently, scorecard models are outfitted with reason codes, which provide a clear means to explain the calculated score, and may be offered to a consumer, or might better inform an operational decision under manual review.

Owing to their commercial importance, predictive analytics vendors have sought a better means to easily exchange scorecard models among software platforms, and until now, the scorecard recipe has been improvised in competing forms of the Predictive Model Markup Language (PMML), using combinations of standard transformations and custom extensions.

PMML is the de facto standard used to represent data mining models [1][2][3]. As a language, PMML is composed of elements and attributes that represent a host of predictive model formulations as well as data pre- and post-processing. However, up to version 4.0, PMML lacked a standard element to represent scorecards. That has been rectified in PMML 4.1 which now incorporates the `Scorecard` element.

This paper describes not only the new standard element to represent scorecards in PMML, but also the common techniques to assign and rank reason codes, which are particularly helpful to enterprises practicing decision management.

## 2. SCORECARD STRUCTURE AND LEARNING TECHNIQUES
The 4.1 specification of the PMML standard introduces the `Scorecard` general element devoted to describing predictive scorecards (Figure 1). This is the main element used to host all the scorecard features and computations, such as binning transformations, model weights, and reason codes. And as such, the new standard provides a complete recipe to compute scorecard-derived predictive scores as well as the ranked reason codes that accompany the score itself.

```
<Scorecard modelName="SimpleScorecard"
    functionName="regression"
    useReasonCodes="true"
    reasonCodeAlgorithm="pointsBelow"
    initialScore="0" baselineMethod="max">
    ...
</Scorecard>
```

**Figure 1. PMML's new Scorecard element and its attributes**

The predictive scorecard is a generalized additive model which uses discretized representations of the predictive inputs. Consistent with other models designed to return numeric estimates, a scorecard in PMML assigns its `functionName` attribute to value "regression".

Figure 1 also demonstrates the other top-level attributes of the `Scorecard` element. Attribute `useReasonCodes` is of type Boolean and indicates whether reason codes should be computed as part of the scorecard model (reason codes are further discussed in Section 3). As suggested by its name, the attributes

`initialScore` sets the initial score for the model, analogous to an intercept term in a regression model.

While they do not factor in to the score calculation itself, baseline scores provide a critical reference value when ranking reason codes, and are described in Section 4. The top-level attribute `baselineMethod` indicates the technique used to compute the baseline scores during model development. And not shown in Figure 1 is the optional attribute `baselineScore`, which may be used to define a single baseline score for the entire scorecard. Our examples in this paper instead assign unique baseline scores to each characteristic in the scorecard.

Also not shown in this example is the top-level attribute `algorithmName`, which is an optional string that can be populated by the PMML producer to simply document the algorithm used to derive the scorecard's model weights from the training dataset.

In a scorecard, numeric and categorical inputs are referred to as characteristics. Whereas, nominal numeric inputs are broken into ranges, categorical inputs may be taken at face value or further "grouped" into a reduced cardinality set. In all cases, these mutually exclusive and collectively exhaustive groups are generally referred to as attributes or bins, and the scorecard model weights (sometimes referred to as coefficients, partial scores, or points) are assigned to each bin, generally to maximize or minimize some data mining objective function over the training dataset.

This paper does not explore in depth the algorithms to discretize predictors into bins, nor to optimize bin-level model weights. However, several techniques (deciles, CHAID, entropy-minimization, and many others) are popular for determining bins of numeric predictors, both of supervised and unsupervised forms. In the case of supervised binning algorithms, techniques may vary with the type of outcome (e.g., binary, multinomial, or continuous dependent variables).

Similarly, many techniques exist to calculate model weights from the training dataset, such as divergence maximization, Bernoulli likelihood maximization, least squared error minimization, and others, depending on the nature of problem and the type of dependent variable.

Regardless of how the bins and model weights have been derived, the responsibilities of PMML producers and consumers are to faithfully describe and reproduce the *scoring* calculation. The scoring calculation is the application of the complete scorecard to a modeling case, which in the real world is often an account holder, a credit applicant, a purchase transaction, an insurance policy, a medical claim, or other business event demanding an informed decision.

Table 1 illustrates a simple scorecard model, similar to those that might be used to rank order credit accounts by repayment risk.

The tabular presentation makes the complete scorecard model easy to understand for all, and the score calculation itself is simple and straightforward.

For example, a fairly new credit user, with 1 year of credit history, with no missed payments, and with a checking account but no savings account, would receive a score of: 64 + 85 + 52 = 201. In this light, it becomes easy to see why model weights are also sometimes called points or partial scores, as the scorecard simply awards point values to different observable behaviors.

**Table 1. Illustration of simple scorecard model**

| Variable | Bin | Score Weight | Reason Code |
|---|---|---|---|
| Years of experience with credit | Missing, unknown | 53 | UK1 |
| | 0 | 53 | EX1 |
| | 1 – 4 | 64 | EX1 |
| | 5 – 9 | 75 | EX1 |
| | ≥ 10 | 81 | UND |
| | *Baseline score: 81* | | |
| Months since last late payment | Missing, unknown | 78 | UK1 |
| | No missed payments (-1) | 85 | UND |
| | 0 – 1 | 50 | DQ3 |
| | 2 – 5 | 63 | DQ3 |
| | 6 – 11 | 71 | DQ3 |
| | 12 – 23 | 76 | DQ3 |
| | ≥ 24 | 79 | DQ1 |
| | *Baseline score: 85* | | |
| Bank account references | Checking & savings | 56 | UND |
| | One account only | 52 | AS1 |
| | None | 41 | AS2 |
| | Missing, unknown | 43 | AS3 |
| | *Baseline score: 56* | | |

Figure 2 shows the corresponding PMML representation for the first set of computations, as applied to variable "Years of experience with credit", which we translated to input variable "experienceInYears" in the PMML code.

```
<Characteristics>
   <Characteristic name="experienceInYearsScore"
       baselineScore="81">
      <Attribute partialScore="53" reasonCode="UK1">
         <SimplePredicate field="experienceInYears"
             operator="isMissing"/>
      </Attribute>
      <Attribute partialScore="53" reasonCode="EX1">
         <SimplePredicate field="experienceInYears"
             operator="lessThan" value="1"/>
      </Attribute>
      <Attribute partialScore="64" reasonCode="EX1">
         <CompoundPredicate booleanOperator="and">
            <SimplePredicate field="experienceInYears"
                operator="greaterOrEqual" value="1"/>
            <SimplePredicate field="experienceInYears"
                operator="lessThan" value="5"/>
         </CompoundPredicate>
      </Attribute>
      <Attribute partialScore="75" reasonCode="EX1">
         <CompoundPredicate booleanOperator="and">
            <SimplePredicate field="experienceInYears"
                operator="greaterOrEqual" value="5"/>
            <SimplePredicate field="experienceInYears"
                operator="lessThan" value="10"/>
         </CompoundPredicate>
      </Attribute>
      <Attribute partialScore="81" reasonCode="UND">
         <SimplePredicate field="experienceInYears"
             operator="greaterOrEqual" value="10"/>
      </Attribute>
   </Characteristic>
   ...
</Characteristics>
```

**Figure 2. PMML code used to represent the computations involving a scorecard characteristic.**

In PMML, scorecard characteristics reside inside element `Characteristics` (note the plural form) which serves as an envelope for all of the scorecard's characteristics. These are each represented in turn by element `Characteristic`, which contains one or more `Attribute` elements. These `Attribute` elements carry most of the scorecard's information, as they define the binnings associated with each characteristic, the model weight associated with each bin, and the optional reason code assigned to each bin. PMML also allows for a single reason code to be assigned to an entire `Characteristic`. However, if a reason code is defined on the bin level, it takes precedence over the reason code defined on the characteristic level.

The scoring technique using a scorecard is simply to locate the bin which matches the entity to be scored, for each input variable, and to sum the assigned score weights of each matched bin across all characteristics. In PMML, the matching of input variables to each bin is represented by a series of predicate elements which evaluate to "true" or "false". These elements are used extensively in the PMML element `TreeModel`. `Scorecard` borrows this elegant representation to allow for the flexible definitions of each `Attribute` in each `Characteristic`, including open, closed, half-open, and unbounded numeric intervals, single discrete values, simple inequality, and compound Boolean logic.

According to the PMML code shown in Figure 2, if the input for variable "experienceInYears" is equal to "3", characteristic "experienceInYearsScore" is assigned score weight "64" (represented by attribute `partialScore`), since the `CompoundPredicate` element which checks for values greater than or equal to 1 and less than "5" evaluates to "true". The `partialScore` of 64 is further summed up with the score weights of all remaining characteristics, plus the `initialScore`, to arrive at the final scored value.

Per the PMML 4.1 specification, it is important for consumers and producers to agree that the *first* Attribute with a matching predicate shall be the scored attribute. The PMML specification does not demand the attributes be mutually exclusive, and the requirement to select the first attribute ensures a consistent behavior if and when overlapping predicates exist.

## 3. ASSIGNING REASON CODES WITHIN SCORECARD MODELS

Scores calculated from predictive scorecards help decision makers render sensible business decisions by generating estimates of future business performance given the known facts at hand. In some applications, however, the decision maker may also require an *explanation* as to why the score at hand rates high or low on its spectrum.

In credit-related decisions, these explanations accompanying the score are frequently referred to *adverse action reasons* or simply *reason codes*, and may be a legal requirement per the United States' Equal Credit Opportunity Act and analogous legislation in other countries.

In other situations, the return of reason codes might aid in customer service ("you are not offered a discounted auto insurance premium because you have had several recent moving violations"), or aid a decision maker in triaging problems ("the risk of fraud on this purchase is high, as indicated by an unusually large distance between billing and shipping addresses").

Once the variable selection, variable binning, and model weights have been determined for the scorecard, the model developer or business analyst may assign reason codes to individual input variables, or to the individual bins within each variable. (The latter choice is reflected in both Table 1 and Figure 2.)

Reason codes should of course be chosen to reflect the nature of the measured predictive variable, and if two or more in-model variables are strongly related, the analyst may elect to assign a single code to more than one variable (or to the individual bins of different variables). In regulated industries, this assignment may be followed by a formal in-house or external legal review.

While the scorecard in Table 1 and the PMML sample in Figure 2 both assign reason codes to the individual bins of each characteristic, the PMML specification allows an alternative assignment of precisely one reason code to each `Characteristic` as a whole.

The clear advantage of assigning potentially unique codes to individual bins is that the reason codes may be cited with directionality and greater specificity, thus improving the quality of the description. The disadvantage is simply that it entails more work and thought by the model designers and reviewers.

Naturally, a PMML producer could choose to only ever assign reason codes to the individual bins, as that is the more general case. The PMML consumer must be prepared to accept reason codes assigned at either level.

The assigned codes themselves are the data communicated in the PMML specification. External to the model, it is likely that an organization maintains a mapping of reason codes to explanatory text. Such an example is provided in Table 2, and these textual explanations may be used in a letter, as on-screen diagnostic, or as part of an oral communication, explaining the key factors that influence the individual score.

**Table 2. Sample reason code explanations**

| Reason Code | Textual explanation |
|---|---|
| AS1 | Assets limited to a single account |
| AS2 | No additional liquid assets |
| AS3 | No known existence of liquid assets |
| DQ1 | Prior history of delinquency |
| DQ3 | One or more delinquencies too recently |
| EX1 | Length of credit history is too short |
| UK1 | Unknown prior credit history |
| UND | Undefined (should not appear) |

Finally, it is worth noting that the assignment and review of reason codes to the bins or variables of a predictive scorecard is often an important step in assessing the suitability (and in some cases legality) of a decision-enhancing predictive scorecard.

# 4. REASON RANKING TECHNIQUES

It is most common for reason codes to be calculated (ranked) during the calculation of the score itself. Variations on how the reasons may be ranked come down to four considerations, which we explore in this section:

- Choice of baseline
- Choice of direction
- Need for uniqueness
- Number of codes

With the model's orientation and intended use in mind, the analyst shall determine the appropriate baseline values, the direction of comparison, and the number of reason codes to return.

The PMML producer will record the baseline values themselves (in `baselineScore` attributes), will optionally document the method used to arrive at them (in the `baselineMethod` attribute), and must communicate the intended direction of comparison (`reasonCodeAlgorithm` attribute). The PMML producer will also document the number of reason codes to return as part of the `Output` element.

And finally, the PMML consumer will recognize the baseline values, compare partial scores to baselines in the appropriate direction, and shall return a ranked set of unique (non-repeating) reason codes.

## 4.1 Choice of baseline

The impact of any individual reason on a final score's calculation is naturally derived from the weights of the scorecard model itself. Points gained or lost are easily associated to the reasons assigned to the variable or to the matching bin. But the notion of gained or lost demands a sense of *baseline*; gained or lost relative to what?

In the PMML representation of scorecards, each individual predictive variable is assigned a numeric baseline score to answer this question. Setting the baseline for each variable is the responsibility of analyst designing the scorecard model, and naturally this is captured and communicated by the PMML producer software. (As a convenience, a single baseline score to be used by all characteristics in the model can be set in the `Scorecard` element's top-level `baselineScore` attribute.)

Popular techniques for establishing baselines for each predictive variable include:

a. The maximum score weight in the characteristic

b. The minimum score weight in the characteristic

c. The expected (mean) partial score for the characteristic, typically derived from the model-training dataset

d. The neutral score value, or "no information" score, for the characteristic

These four choices are enumerated in the PMML standard as "max", "min", "mean", and "neutral", respectively, and can be optionally documented in the `baselineMethod` attribute at the top-level `Scorecard element`. A fifth option, "other", is also provided in the PMML standard. Our example in Table 1 uses a maximum score as the baseline.

Note that while the baseline values used in (a) and (b) may be found directly within the scorecard table, those values in techniques (c) and (d) are almost certain to fall somewhere in between the actual score weights of the scorecard table. In all five cases, it is the responsibility of the PMML producer to clearly state the baseline value in the `baselineScore` element of each `Characteristic` (or to state a single baseline value for all characteristics).

And while the minimum and maximum score weights per variable are trivially understood, the mean and neutral baselines deserve some attention.

### *4.1.1 Mean partial score as baseline*

The baseline score determined via the mean is simply the expected scored value for the variable, as determined from the training dataset. Table 3 provides an example for one of our scorecard's variables.

**Table 3. Sample calculation of mean baseline score**

| Variable | Bin | Score Weight | Training dataset distribution |
|---|---|---|---|
| Months since last late payment | Missing, unknown | 78 | 0% |
| | No missed payments (-1) | 85 | 48% |
| | 0 – 1 | 50 | 3% |
| | 2 – 5 | 63 | 7% |
| | 6 – 11 | 71 | 9% |
| | 12 – 23 | 76 | 12% |
| | ≥ 24 | 79 | 21% |
| Mean partial score | | **78.8** | |

Figure 3 shows the partial PMML code for variable "Months since last late payment" that would correspond to this use of a baseline value. Notice that the attribute `baselineScore` which is set to "78.8", the mean partial score for characteristic "monthsSinceLastLatePayment".

```
<Characteristic name="monthsSinceLastLatePaymentScore"
    baselineScore="78.8">
   <Attribute partialScore="50" reasonCode="DQ3">
      <SimplePredicate field="monthsSinceLastLatePayment"
          operator="isMissing"/>
   </Attribute>
    ...
 </Characteristic>
```

**Figure 3. PMML code including attribute "baselineScore".**

### *4.1.2 Neutral partial score as baseline*

The neutral or risk-neutral score weight is sometimes also referred to as the "no information" weight, as it is the partial score value corresponding to the population odds (or, if you prefer, prior probability or "bad rate") of the training population.

To be concrete, suppose we are modeling the future likelihood of a successfully repaid loan versus a default on the loan. As a whole, the entire credit portfolio may achieve some historical population odds, say, 24 good payers for every 1 defaulted loan, or a 4% prior probability of default.

When the modeler examines potential predictors using a training dataset of previously funded loans, she seeks variables which spread that risk across the bins of the variable. While some bins exhibit better-than-average risk, others exhibit worse-than-average risk. The degree to which a variable can separate better-from-worse risk, the stronger a candidate it is for inclusion in the final model.

Now, suppose a bin exists within the scorecard variable such that its own risk level matches perfectly the total population odds (24:1, in our example). Knowing that a credit applicant falls into this bin provides no *new* information, as that bin is no better and no worse than the total population average.

While no such bin is guaranteed to exist in any scorecard variable, on any training dataset, the concept provides analysts an effective reference point when designing and reviewing scorecards. And that hypothetical risk-neutral bin can also provide the baseline score in reason code rankings.

To recall, regardless of which technique the analyst uses to establish the baseline score of each characteristic, the PMML producer and consumer will always register and find, respectively, this value in the `baselineScore` attribute of each `Characteristic`. (In the unlikely event that a single baseline score is shared by all variables, it can be stated just once, in the top-level `baselineScore` attribute.)

The technique used by the analyst to arrive at the baseline scores may be documented by the PMML producer in the `baselineMethod` attribute, at the top level of the level `Scorecard`. The option is not required for accurate scoring or reason code ranking, but is helpful in documenting the design of the predictive model.

## 4.2 Choice of direction

In the PMML specification, the `reasonCodeAlgorithm` attribute carries a significant name, but its choice set is simple and clear. Reasons are ranked by comparing the awarded points to the baseline score, either from above or from below.

The model designer and the PMML producer software choose whether reasons should be ranked by their points above the baseline score or points below the baseline score. Naturally, this choice of direction is informed by the analyst's design of the model, specifically concerning the model's orientation to its training target, and how the model will be used to influence business decisions.

To be concrete, suppose a model designer's intent is to produce a scoring system that awards high scores to the best credit risks (highest probability of successful repayment), and low scores to the worst credit risks (highest probability of default), and to provide informative diagnostics as to why the candidate scored "too low" compared to a decision threshold (e.g., approve loan only if total score is above 210). In this case, the reason codes would likely be ranked by their differences *below* the baseline scores, providing the top reasons they failed to score higher, and hence failed to meet and exceed the decision threshold.

Regardless of how the baseline values have been established, the following calculations represent the appropriate directional differences, *d*, for a single reason code within a single characteristic.

For `reasonCodeAlgorithm` set to `pointsBelow`,

$$d_b = baselineScore - partialScore$$

And for `reasonCodeAlgorithm` set to `pointsAbove`,

$$d_a = partialScore - baselineScore$$

In its own score calculations, the PMML consumer must of course compute the difference from baseline in the correct direction, to produce the intended ranking of reason codes, during its score calculations. Note that these are signed calculations, and that for baselines other than min or max, negative differences must be expected. The translation of these per-characteristic differences into a final ranking of reasons is discussed next.

## 4.3 Need for uniqueness

It is possible in PMML, and often quite suitable in designing predictive scorecards for decision management applications, to cite a single code from different input predictors. In Table 1, the reason code "UK1" is indeed cited by two different characteristics: as the first attributes of years of experience, and months since delinquency, both relating to the absence of credit information. And allowing scorecard models to cite in-common codes across characteristics naturally raises the question of how the reasons should be ranked if and when codes occur multiple times.

If the same reason code is cited by two different characteristics, then it is clearly arguable that the model designer's intent was to suggest that one common cause lies behind the points gained or lost. And in this sense, it only makes sense to *aggregate* the points lost toward that reason.

In our example from Table 1, a single scoring case with an unknown length of credit history, and an unknown duration since delinquency, would "lose" 81–53=28 and 85–78=7 points from the baselines (maximum partial scores), respectively, on the first two variables of the scorecard.

Arguably then, the *total* points lost for reason UK1 is 28+7=35 points.

Supposing instead (and in contrast to Table 1) that the baselines were derived from the mean scores — and that those means were 67.2 and 78.8, respectively — the same scoring case would lose 67.2–53=14.2, 78.8–78=0.8, or a total of 15.0 points, for reason UK1.

When preparing the final vector of ordered reason codes to return with each calculated score, the total points deviating the baselines dictates the ranking. The reason code with the largest accumulated differences from baseline is the first reason, followed by the reasons with the next largest accumulated differences.

## 4.4 Common combinations

The authors of the PMML specification for scorecards sought to allow for the most popular combinations of reason code ranking techniques, but to also enable flexibility and creativity for scorecard model designers, and to make simple work of the implementation for PMML consumers.

Although more combinations are plausible, the most popular and business-sensible selections of baseline and direction are:

- Points below the maximum score as baseline
- Points above the minimum score as baseline
- Points above or below a mean-score baseline
- Points above or below a risk-neutral baseline

While there is no requirement in the PMML 4.1 specification to *disallow* a nonsensical choice such as "points above maximum" in the PMML document, it would be prudent for the PMML producer software to warn or disallow a user from requesting such a combination.

It is also worthwhile to also explore the relative merits of these popular techniques, and to consider *why* the scorecard designer might choose one over another.

Reason code rankings which use the extreme points as baselines have an advantage in that they will always (barring the exceptional case of a "perfect" score) have at least one reason to cite when explaining a score's calculation. For scores that will be used in multiple decision situations, or that have no single decision threshold, this may be a distinct advantage.

Reason code rankings drawn from mean or risk-neutral baselines might return more meaningful, actionable reasons. Rather than highlighting the elements that make the score far from perfect, it may be more meaningful to explain where and why the score is failing to at least be average. In decision areas where a yes/no threshold may hover in the same area as the population average, this may be more informative and appropriate.

This discussion also suggests ideas for still other baseline calculation techniques. If a decision threshold is known to occur nowhere near the mean score value, but indeed near one tail of the distribution — which may be the case for rare events, such as transaction fraud, claims fraud or security breaches — one might well prefer other forms of baseline calculation. And in such cases, the PMML producer's and consumer's tasks remain unchanged: simply assign or recognize the numeric baseline value associated with each characteristic.

## 4.5 Reason codes as outputs

In PMML 4.1, the element `Output` was modified so that reason codes could also be part of the model's output. Figure 4 shows the element `Output` for our sample scorecard model, configured to return four output fields. The first output field is to contain the predicted value, as indicated by the `feature` attribute. For a scorecard model, the predicted value refers to the overall score obtained from summing up all partial scores and initial score. The remaining three output fields related to reason codes as specified by the `feature` attribute's value of "reasonCode". Since three different reason codes are to be returned in this example, their importance or relevance in explaining the overall score is defined by attribute `rank`. The output field with `rank` equal to "1" is the most significant reason: the reason code with the largest accumulated differences from baselines. The reason with `rank` equal to "2" had the second largest accumulated differences, and so on.

```
<Output>
   <OutputField name="Final Score"
    feature="predictedValue"
    dataType="double" optype="continuous"/>
   <OutputField name="Reason Code 1" rank="1"
    feature="reasonCode"
    dataType="string" optype="categorical"/>
   <OutputField name="Reason Code 2" rank="2"
    feature="reasonCode"
    dataType="string" optype="categorical"/>
   <OutputField name="Reason Code 3" rank="3"
    feature="reasonCode"
    dataType="string" optype="categorical"/>
</Output>
```

**Figure 4. Output element for a typical scorecard. One output field is devoted to feature "predictedValue" while the three remaining output fields are devoted to feature "reasonCode".**

By setting a fixed number of output fields with `feature` equal to "reasonCode", the model designer and the PMML producer determine how many reasons should be returned with each score.

The PMML consumer should of course be prepared for the event when the *requested* number of reasons actually exceeds the number of reasons which indeed have positive accumulated differences from the baselines on any individual scored record. In this case, the recommended behavior is to simply return empty strings for these "missing" reason codes.

# 5. CONCLUSION

The scorecard is a highly interpretable and transparent form of predictive model, which lends itself well to decision management applications. Users of the score can easily appreciate how the score is calculated and how its returned values and reason codes can be used to empirically aid their decision making. The new 4.1 specification of PMML provides a complete recipe for accurately describing, exchanging and calculating scorecards and their associated explanations.

# 6. REFERENCES


[1] A. Guazzelli, W. Lin, T. Jena (2010). *PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics.* CreativeSpace (available on Amazon.com).

[2] A. Guazzelli, M. Zeller, W. Lin, G. Williams. PMML: An Open Standard for Sharing Models. The R Journal, Volume 1/1, May 2009.

[3] R. Pechter. What's PMML and What's New in PMML 4.0? ACM SIGKDD Explorations Newsletter, July 2009