

Automating the Quality Assurance of an On-line Knowledge-Based Classifier

By Fusing Multiple Off-line Classifiers

Piero P. Bonissone
General Electric Global Research
Schenectady, NY 12309, USA
bonissone@research.ge.com

Abstract

We address two problems in the lifecycle of a production classifier: the *monitoring* of its decisions quality and the *updating* of the classifier over time. The proposed architecture consists of four off-line classifiers and an associative fusion module. The fusion is a T-norm based outer-product of the classifiers' normalized outputs. By attaching a confidence measure to each output of the fusion, we generate a distribution of the production classifier's quality. The lower tail of this distribution identifies the least reliable cases, which become candidates for *auditing* and manual QA. The upper tail identifies the most reliable cases, which become candidates for *updating* the standard reference data set used to design and tune the production classifier. We illustrate this approach with an insurance underwriting problem.

Keywords: Classification, Fusion, T-norms, Fuzzy Constraints, Case based Reasoning.

1 Introduction

The automation of real-world decision-making processes requires addressing each step in the lifecycle of its underlying decision engine. The development and deployment of such engine represent the first stage of its lifecycle. Once an engine has been placed in production, it is equally important to monitor its performance and probe the quality of its decisions. In previous papers we have described the design and optimization of two decision engines for underwriting insurance applications. Both engines, based on fuzzy constraints and fuzzy case-based reasoning [1,3], used an evolutionary algorithm to optimize their underlying parameters and minimize the cost of misclassification [4]. The core of this optimization was the generation of a set of Standard Reference Decisions (SRD). In [5] we discussed the lifecycle of a classifier with an emphasis on its validation,

verification, and knowledge-base maintenance. In this paper we focus on the monitoring and quality assurance of the production classifier's decisions.

2. The Classification Problem

As presented in [5] we will use a representative, challenging classification problem: the process of underwriting insurance applications. Insurance underwriting (UW) is a complex decision-making task traditionally performed by trained individuals. An underwriter must evaluate each insurance application in terms of its potential risk for generating a claim, e.g. mortality in the case of term life insurance. An application is compared against standards adopted by the insurance company, which are derived from actuarial principles related to mortality. Based on this comparison, the application is classified into one of the risk categories available for the type of insurance requested by the applicant. The *accept/reject* decision is also part of this risk classification, since risks above a certain tolerance level will typically be rejected. The estimated risk, in conjunction with other factors such as gender, age, and policy face-value, will determine the appropriate price (premium) for the insurance policy. When all other factors are the same, to retain the fair value of expected return, higher risk entails higher premium.

We define an insurance application as an input vector \bar{X} containing discrete, continuous, and attribute variables. These variables represent the applicant's medical and demographic information that has been identified by actuarial studies to be pertinent to the estimation of the applicant's claim risk. Similarly, we define the output space \bar{Y} , e.g. the underwriting decision space, as an ordered list of rate-classes. Due to the intrinsic difficulty of representing risk as a real number on a scale, the output space \bar{Y} is subdivided into bins (rate-classes) containing similar risks. The underwriting process can be summarized as a discrete classifier that maps an input vector \bar{X} into a decision space \bar{Y} , where: $|\bar{X}| = n$ and $|\bar{Y}| = T$.

The underwriting (UW) problem is not straightforward due to several reasons:

- 1) UW is a *highly nonlinear* mapping, since small incremental changes in one input can cause large changes in the corresponding rate-class.
- 2) Most inputs require *interpretations*. Thus the underwriter's subjective judgment will often play a role in this process. Variations in underwriter training and experience will likely cause underwriters variability in their decisions.
- 3) These interpretations require an intrinsic *flexibility* to ensure a balance between *risk-tolerance*, necessary to maintain price competitiveness, and *risk-avoidance*, necessary to prevent overexposure to risk.
- 4) Legal and compliance regulations require that the models used to make the underwriting decisions be *transparent* and *interpretable*.

To address these requirements we extended traditional AI reasoning methodologies, such as rule-based and case-based reasoning, with soft computing techniques, such as fuzzy logic and evolutionary algorithms. With these hybrids systems, we provide flexibility and consistency, while maintaining interpretability and accuracy.

3. The Production Classifier

For this underwriting application we used a Fuzzy Logic Engine (FLE) based on rule-encoded underwriting standards. Each rule set is an intersection of fuzzy constraints defining the boundaries between rate classes. First, these constraints were determined from underwriting guidelines. Then, they were refined using knowledge engineering sessions with expert underwriters to identify factors such as blood pressure levels and cholesterol levels, which were critical in defining the applicant's risk and corresponding premium. The goal of the classifier is to assign an applicant to the most competitive rate class, providing that the applicant's vital data meet all of the constraints of that particular rate class to a minimum degree of satisfaction. The constraints for each rate class r are represented by n fuzzy sets: $A_i^r(x_i)$ $i=1, \dots, n$. Each value $A_i^r(x_i)$ is interpreted as the *degree of preference* induced by value x_i for satisfying constraint A_i^r . After evaluating all constraints, we compute two measures for each rate class r . The first one is the *degree of intersection* of all constraints and measures the *weakest* constraint satisfaction:

$I(r) = \bigcap_{i=1}^n A_i^r(x_i) = \text{Min}_{i=1}^n A_i^r(x_i)$ ¹. The second one is a *cumulative* measure of missing points (the complement of the average satisfaction of all constraints), and measures the *overall tolerance* allowed to each applicant i.e.:

$$MP(r) = \sum_{i=1}^n (1 - A_i^r(x_i)) = n \left(1 - \frac{1}{n} \sum_{i=1}^n A_i^r(x_i) \right) = n(1 - \bar{A}^r)$$

The final classification is obtained by comparing the two measures, $I(r)$ and $MP(r)$ against two lower bounds defined by thresholds τ_1 and τ_2 . The parametric definition of each fuzzy constraint $A_i^r(x_i)$ and the values of τ_1 and τ_2 are design parameters that were initialized with knowledge engineering sessions. Figure 1 illustrates an example of three constraints (trapezoidal membership functions) associated with rate class Z, the input data corresponding to an application, and the evaluation of the first measure, indicating the weakest degree of satisfaction of all constraints.

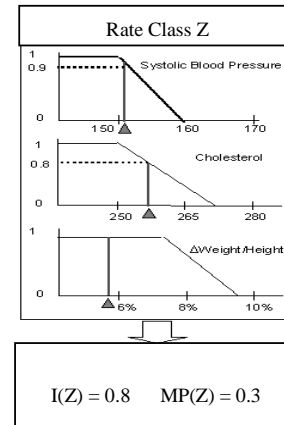


Figure 1. Example of Three Fuzzy Constraints for Rate Class Z

After a parametric optimization and a five-fold cross-validation, described in [4,5,9] the classifier was placed in production.

4. Lifecycle Stages after Deployment: Monitoring and Maintenance.

A serious challenge to the successful deployment of intelligent systems is their ability to remain valid and accurate over time, while compensating for drifts and accounting for contextual changes that might otherwise render their knowledge-base stale or obsolete. This issue has been an ongoing concern in deploying AI expert systems and continues to be

¹ This expression implies that each criterion has equal weight. If we want to attach a weight w_i to each criterion A_i we could use the weighted minimum operator:

$$I(r) = \bigcap_{i=1}^n W_i A_i^r(x_i) = \text{Min}_{i=1}^n \left(\text{Max}((1 - w_i), A_i^r(x_i)) \right) \quad \text{where } w_i \in [0, 1].$$

a critical issue in deploying knowledge-based classifiers. The maintenance of a classifier is essential to its long-term usefulness since, over time, the configuration of a FLE may become sub-optimal. It is necessary to modify the SRD over time to reflect these contextual changes. We developed specialized editors to achieve this objective. By modifying the SRD to incorporate the desired changes (by altering previous standard reference decisions), we create a new *target* for the classifier. Then, we use the same evolutionary optimization tools to find a new configuration for the classifier to *approximate the new target*.

It is equally important to monitor the classifier performance over time to identify those new, highly reliable cases that could be used to update the SRD. To address this objective, we have implemented an off-line quality assurance (QA) process, based on a fusion module, to test and monitor the production FLE that performs on-line rate classification. At periodic intervals, e.g., every week, the fusion module and its components will review the decisions made by the FLE during the previous week. The purpose of this fusion is to assess the quality of the FLE performance over that week. This fusion will identify the *best* cases, which could be used to tune the production engine, and *controversial* or *unusual* cases, which could be audited or reviewed by human underwriters.

In our approach we designed the classifiers around a set of *Standard Reference Decisions* (SRD), which embodies the results of the ideal behavior that we want to *reach* during development and that we want to *track* during production use. The SRD are the training and validation set for the classifier, and in general are a benchmark set that must be maintained over time. To this end we propose to use a fusion module for updating the SRD with new cases, without requiring expensive manual screenings [5].

5. Architecture for the Quality Assurance

To ensure the integrity of the cases used to update the SRD, we decided to develop a collection of independent classifiers specially tuned for accuracy. These classifiers, which do not need to meet the same interpretability requirements as the FLE, are part of the fusion process described in Figure 2.

This process is decomposed into four steps, which will be discussed in Section 7:

1. Input tagging, output generation, discounting and post-processing of classifiers' outputs.
2. Determination of a combined decision via the associative fusion of the classifiers' outputs.
3. Determination of confidence measure.

4. Identification of candidate cases for auditing and SRD updating.

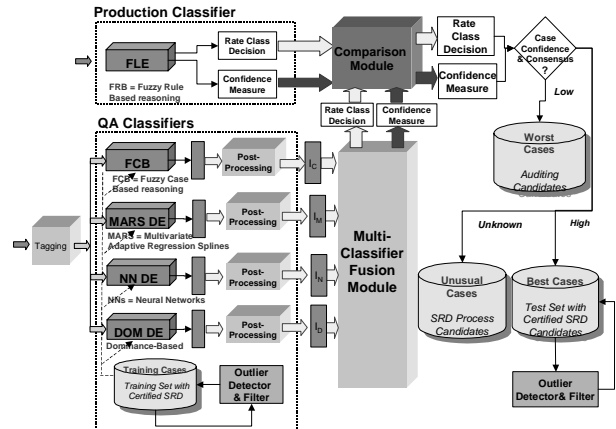


Figure 2: Top-level Fusion Architecture

6. Classifiers Used in the Fusion Module

6.1 Fuzzy Case-Based Engine

Rule-based engines are *generative* approaches to develop knowledge-based solutions, while Case-based (CB) engines are *analogical* approaches to solve the same problems. Knowledge engineering is used to determine the most efficient indices for representing cases, to define the fuzzy similarity metric that induces the best ranking upon the retrieved cases, and to adapt the solution of the closest cases to the application on hand. The parameters used by case-based systems to search for similar cases and to rank them are design choices that need to be determined and maintained over time for optimal performance.

The CB engine executes four basic steps [1,3]:

1. It compares a probe against the CB of past cases and retrieves cases that are similar to the probe.
2. It calculates a distance between each retrieved case and the probe to quantify the similarity between the two cases. In our case we used the Generalized Bell Functions to measure similarity.
3. It generates a weighted histogram of the rate-class distribution for the retrieved neighboring cases.
4. It determines a solution for the probe case using the median (or a similar aggregation operator) of the resulting histogram.

6.2 Feed-forward Neural Network Classifier

A neural network with multiple output nodes is a typical design for multiple-class classifiers where each of the NN output nodes corresponds to each rate class. However, neural networks with multiple output nodes have large number of weights and biases, and require large training sets and long training time for properly training the network.

Since in the UW problem the size of the training set is relatively small (in comparison with the number of features and classes) we have used multiple binary neural networks to perform the multi-class classification. This approach not only reduces the complexity of the network and its training time, but also improves the classification performance. The architecture of the neural network classifier for Non-nicotine users is shown in Figure 3.

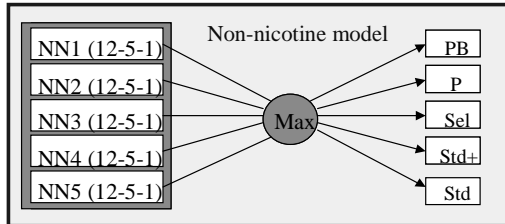


Figure 3: NN Architecture

Each binary network has the structure of 12-5-1, i.e., 12 input nodes, 5 hidden neurons, and 1 output node. Activation functions for both hidden and output neurons are logistic sigmoidal functions. The range of target values is scaled to [0.1 0.9] to prevent saturation during training process. The Levenberg-Marquardt numerical optimization technique is used as the back-propagation learning algorithm to achieve second-order training speed. Each binary network represents an individual rate class and is trained with the targets of one-vs-other. During classification for an unknown case, each network generates a partial membership value of the unknown case to the class represented by the network. The final rate-class assignment of the unknown case is determined by the MAX decision rule.

6.3 MARS-Based Classifier

MARS (multivariate regression splines) [8] is an adaptive nonparametric regression technique, able to capture main and interaction effects in a hierarchical manner. Being a piecewise-linear adaptive regression procedure, MARS can approximate very well any non-linear structure, if present. However, global models cannot easily incorporate jumps in decision boundaries of a large number of variables in an extremely small bounded range. We addressed this problem in two ways. First, we used a *Tag* encoding, described in section 7.1, which leverages domain knowledge to help the MARS search algorithm in initializing the “knots” in the right place. Then we developed a *Parallel Network* arrangement of models. We created a collection of MARS models, each of which solves a two-class problem, and we collated their outputs in a manner similar to the one used for the NN classifiers.

6.4 Dominance-Based Classifier

All input variables in a vector \bar{X} describing an application can be constructed such as smaller values represent more desirable values. We use the partial order induced by the “ \leq ” operator over the input vectors to indicate this preference. When vector \bar{X}_1 *dominates* \bar{X}_2 , then we know that its associated rate-class cannot be worse than the one associated with \bar{X}_2 . Using this dominance concept, we generated two subsets for each rate-class: the *Pareto-best* subset and the *Pareto-worst* subset, representing the *least risky* and the *most risky* candidates within the same rate-class. These subsets are samples from the two hypothetical risk surfaces in the feature space that bound the rate-class from above and below, respectively. The classifier will assign a rate-class to a new candidate by verifying if it lies within these two bounding risk surfaces. When this occurs, the decision is always correct. However, this situation is not very frequent (~20% of coverage in the UW problem). Thus this classifier produces very low coverage at maximum accuracy.

7. Fusion module

7.1 Input Tagging, Output Generation, Discounting, and Outputs Post-processing

Tagging. Typically, data-driven algorithms can improve their performance if we can augment them with domain-knowledge. In our case, we wanted to leverage existing knowledge about risk classification to initialize or influence the various classifiers. Using data-mining experiments, we identified four input variables, among the ones defined by actuarial studies, which had the highest impact on the rate-class decision. Since the same studies indicate the critical thresholds that govern such risk, there was no reason to re-learn those thresholds and guidelines. We defined a set of inequality constraints on these four variables and represented them as as crisp rules. With these rules, we created a *tag*, which was an ordinal categorical variable developed from a collection of indicators for the various decision boundaries, as defined by human experts. This utilization of this available domain knowledge boosted the classifiers’ performance, leading to an accuracy improvement of about 1-2% on average.

Output Generation. The output of each classifier is a distribution representing the degree to which a rate class is selected. The set of all possible rate classes represents the universe of all answers that can be considered by the classifiers. An assignment to the universe represents the classifier’s *ignorance*.

Specifically, each module generates an output vector $\mathbf{I} = [I(1), I(2), \dots, I(N+1)]$ where $I(i) \in [0, M]$ where M is a large real value and N is the number of rate-classes. We consider each entry $I(i)$, for $i=1, \dots, N$, as the un-normalized degree to which the case could be classified in rate class i . The last element, $I(N+1)$ indicates the degree to which the case cannot be decided and the entire universe of rate classes is selected.

Discounting. The reliability of each classifier can be represented by a static or dynamic discounting factor, which reflects the expected accuracy of the classifier. In a manner similar to Shafer's belief theory [11], a static discounting factor represents a *prior* expectation about the classifier's reliability, e.g., the average past accuracy of the classifier.

Post-processing. A dynamic discounting represents a *conditional* assessment of the classifier's reliability, e.g., if a classifier is basing its output on an insufficient number of points, then its decision is no longer reliable. We leverage this conditional knowledge in a post-processing stage that is applied to the output of each model. The post-process computes a set of features: cardinality, entropy, the difference between the distribution's mode and the second highest value. These features are compared with some experimentally determined thresholds to determine the validity of the model's output. If the output does not meet these quality constraints, the model's decision is replaced by a non-commitment assignment (i.e. full ignorance).

7.2 Generating a Combined Decision via an Associative Fusion of the Modules' Outputs

The different semantics of the classifiers described in section 6 pose a challenge to the fusion operation. While the outputs of some classifiers could be interpreted as probability densities (e.g., the rate-class frequencies displayed by the histograms of cases retrieved by the CB engine), other models, such as the NN or MARS, generate possibilistic distributions or scores. We can only interpret these distributions as membership functions with a normalized cardinality, without necessarily assigning any probabilistic semantics to them. Furthermore, all classifiers share a common preprocessing annotation (*Tagging*) that could potentially introduce a common bias. To combine the outputs of the classifiers we propose an aggregation method that does not require orthogonality among the classifiers. The proposed method is also associative.

Let us consider m classifiers, S_1, \dots, S_m , such that the output of classifier S_j is vector \mathbf{I}^j containing the

normalized weight assignment made to the N rate-classes. Let's recall the last $(N+1)^{\text{th}}$ element represents the classifier's lack of commitment, i.e., $\mathbf{I}^j = [I^j(1), I^j(2), \dots, I^j(N+1)]$, where:

$$I^j(i) \in [0,1] \text{ subject to the constraint: } \sum_{i=1}^{N+1} I^j(i) = 1$$

We define the un-normalized fusion of the outputs of two classifiers S_1 and S_2 as:

$$F(\mathbf{I}^1, \mathbf{I}^2) = \text{Outerproduct}(\mathbf{I}^1, \mathbf{I}^2, T) = A$$

where the outer-product takes as arguments the two N -dimensional vectors \mathbf{I}^1 and \mathbf{I}^2 and generates as output the $N \times N$ dimensional array A , in which each element $A(i,j)$ is the result of applying the operator T to the corresponding vector elements, namely $\mathbf{I}^1(i)$ and $\mathbf{I}^2(j)$, i.e.:

$$A(i,j) = T[\mathbf{I}^1(i), \mathbf{I}^2(j)]$$

The operator $T(x,y)$ is a Triangular Norm [10]. The selection of the best T-norm to be used as intersection operation in the fusion of the classifiers depends on the potential correlation among the classifiers to be fused [2]. Because the T-norms are associative, then so will be the fusion operator, i.e.,

$$F(\mathbf{I}^1, F(\mathbf{I}^2, \mathbf{I}^3)) = F(F(\mathbf{I}^1, \mathbf{I}^2), \mathbf{I}^3)$$

Each element $A(i,j)$ represents the fused assignment of the two classifiers to the intersection of rate classes r_i and r_j . Figure 4 illustrates our assumption, i.e., the fact that each rate-class is disjoint and that U is the universe of all rate-classes.

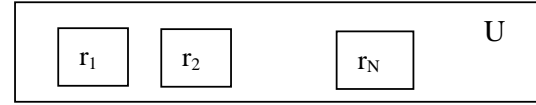


Figure 4: N disjoint (rate) classes defined in U

Given our premise that the rate-classes are disjoint, we will have four possible situations

- (a) When $i=j$ & $i < (N+1)$ then $r_i \cap r_j = r_j \cap r_i = r_i$
- (b) When $i=j$ & $i = (N+1)$ then $r_i \cap r_j = U$
- (c) When $i \neq j$ & $i < (N+1)$ and $j < (N+1)$
then $r_i \cap r_j = \phi$ (the empty set)
- (d) When $i \neq j$ & either $i = (N+1)$
or $j = (N+1)$ then $r_i \cap r_j = r_j$
or $r_i \cap r_j = r_i$

Intersection	r_1	r_2	...	r_{N-1}	r_N	U
r_1	r_1	ϕ	ϕ	ϕ	ϕ	r_1
r_2	ϕ	r_2	ϕ	ϕ	ϕ	r_2
...	ϕ	ϕ	...	ϕ	ϕ	...
r_{N-1}	ϕ	ϕ	ϕ	r_{N-1}	ϕ	r_{N-1}
r_N	ϕ	ϕ	ϕ	ϕ	r_N	r_N
U	r_1	r_2	...	r_{N-1}	r_N	U

Figure 5: Intersection of disjoint rate-classes and U

Figure 5 illustrates the result of intersecting the rate classes and the universe U . The decisions for each

rate-class can be gathered by adding up *all* the weights assigned to them. According to the four possible situations described above, weights can be assigned to a specific rate class only in situation a) and d), as illustrated in Figure 5:

$$\begin{aligned} \text{Weight}(r_i) &= A(i,i) + A(i,N+1) + A(N+1,i) \\ \text{Weight}(U) &= A(N+1,N+1) \end{aligned}$$

The final output is a rate-class distribution and a measure of conflict among all the classifiers. We normalize the final output (showing the degree of rate-class selection as a percentage) and identify the strongest selection of the fusion.

7.3 Determination of a Confidence Measure

We propose to use a measure of the scattering of the weights around the main diagonal as a confidence metric for the fusion. The more weights are assigned to elements outside the main diagonal, the larger is the amount of conflict among the classifiers. We can represent this concept by defining a penalty matrix $\mathbf{P}=[P(i, j)]$ of the form:

$$P(i, j) = \begin{cases} \max(0, (1 - W * |i - j|)^d) & \text{for } 1 \leq i \leq N \text{ and } 1 \leq j \leq N \\ 1 & \text{for } i = (N + 1) \text{ or } j = (N + 1) \end{cases}$$

This function rewards the presence of weights on the main diagonal, indicating agreement between the two classifiers, and penalizes the presence of weights off the main diagonal, indicating conflict. The conflict increases in magnitude as the distance from the main diagonal increases. Figure 6 illustrates the penalty matrix \mathbf{P} for $W=0.2$ and $d=5$.

\mathbf{P}	r_1	r_2			r_N	\mathbf{U}
r_1	1	0.3	0.1	0	0	1
r_2	0.3	1	0.3	0.1	0	1
	0.1	0.3	1	0.3	0.1	1
	0	0.1	0.3	1	0.3	1
r_N	0	0	0.1	0.3	1	1
\mathbf{U}	1	1	1	1	1	1

Figure 6: Penalty matrix \mathbf{P} for $W=0.2$ and $d=5$

Of course we could use other functions to penalize elements off the main diagonal. This type of penalty function indicates that *conflict is gradual*, since the rate-classes are ordered. We want to represent the notion that the difference between r_1 and r_2 is smaller than then the one between r_1 and r_3 . The shape of the penalty matrix \mathbf{P} captures this concept, as \mathbf{P} shows a confidence that decreases non-linearly with the distance from the main diagonal. A measure of the normalized confidence \hat{C} is the sum of element-wise products between \hat{A} and \mathbf{P} , e.g.:

$$\hat{C} = \text{Norm. Confidence}(\hat{A}, \mathbf{P}) = \sum_{i=1}^{N+1} \sum_{j=1}^{N+1} \hat{A}(i, j) * P(i, j)$$

and \hat{A} is the normalized matrix A : $\hat{A} = A / |A|$

7.4 Identification of Candidate Cases for Auditing, and for Updating the SRD

Finally, we use the confidence measure and the agreement between the fusion's and the production engine's decision to assess the quality of the latter. We label the cases in terms of the confidence in their decision, e.g. *low*, *medium*, *high*, or *unknown*. When the fusion produces a *low* degree of confidence, the case is selected for *auditing*. When the fusion produces a *high* degree of confidence and agrees with the decision of the production engine, the case becomes a candidate for *augmenting the SRD*. When the fusion is non-committal (i.e., it does not commit to any class), the case is a candidate for a *review by senior underwriters* who will generate a standard reference decision. These cases will be later used to retrain the offline classifiers. When the fusion produces a *medium* degree of confidence, we will not use the case, but we will monitor the frequency of its occurrence.

8. Experimental Results

8.1 Selecting the Lowest Quality Cases for QA.

The fusion module was tested with several data sets. In an analysis based on 1,875 applications of non-nicotine users, we found that, for a given confidence threshold ($TI=0.2$), we had an agreement between the fused decision and the engine in 92.75% of the cases, while in 2.35% the fusion did not make any decision. In 4.9% of the cases, the fusion disagreed with the production engine, identifying possible candidates for an audit. Table 1 shows the effect of changing the confidence threshold TI . Each column shows the number of cases whose confidence measure \hat{C} is greater or equal to TI . As this threshold increases, so does the number of *No Fused Decision*.

Table 1: Output of Fusion for Low Confidence Threshold TI (for non-nicotine users)

T=Product, W=0.2, d=7	Confidence Threshold TI										
	1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0
No Fused Decisions	1875	381	270	193	127	78	61	52	44	40	39
Complete agreement	0	1462	1558	1625	1666	1686	1688	1688	1690	1692	1692
False Positive	0	14	17	18	29	41	50	54	59	59	59
False Negatives	0	13	19	24	28	43	46	48	49	49	49
Corrections	0	4	10	14	22	24	27	28	28	28	28
Complete disagreement	0	1	1	1	3	3	3	5	5	7	8
Total	1875	1875	1875	1875	1875	1875	1875	1875	1875	1875	1875

Table 2 shows the results of our experiments on the set of 1,875 applications of non-nicotine users. Note that in a production environment we only have FLE and Fusion available. The use of the SRD is only possible during the training phase of the models and the fusion module. For a given confidence threshold of 0.20, we eliminated 44 cases for which the result of the fusion was deemed *too weak to be used*. Of

the remaining 1,831 cases in which the fusion module delivered a decision, it disagreed with the FLE production engine in (28+59+5=92) cases. These are the least reliable cases that we would like to select for manual auditing. This subset represents a **4.9%** of the entire case population. In 28 of the 92 cases, the FLE decision was incorrect (i.e. it disagreed with the SRD).

Table 2: Summary of Fusion of Non-nicotine users (for threshold $TI = 0.2$)

Non-Nicotine Users (Threshold = 0.2)	
No Fused Decisions	44
Agreements	1690
False Positive	59
False Negatives	49
Corrections	28
Complete disagreement	5
Total	1875

An auditing based on this information would have been very productive, finding ~30% of the sample to be incorrect decisions. Of course, there are still 49 cases of false negatives, in which the fusion agrees with the FLE and yet they are both wrong.

8.2 Selecting the Best Cases to Augment the SRD.

By pushing the threshold TI to the other extreme, we can select cases with the least amount of conflict among the classifiers. These cases are the best candidates to augment and update the SRD. Their final selection would be based on a rate-class stratification, to insure a balanced updating.

Table 3: Output of Fusion for High Confidence Threshold TI (for non-nicotine users)

	Confidence Threshold TI				
	0.987	0.985	0.980	0.975	0.970
T=Product, W=0.2, d=7	0.987	0.985	0.980	0.975	0.970
No Fused Decisions	1706	1678	1583	1567	1540
Agreements	168	195	286	302	329
False Positive	0	0	1	1	1
False Negatives	1	2	5	5	5
Corrections	0	0	0	0	0
Complete disagreement	0	0	0	0	0

Table 3 shows that 169 cases had a confidence measure \hat{C} greater or equal to 0.987. Except for one case, the remaining 168 cases are correct decisions. This set represents the most reliable **9%** of all cases.

8.3 Experiments Results using different T-Norms

The results in Tables 2 and 3 are computed using the *scalar product* as the outer-product operator. When this choice is combined with a strict determination of the conflict (e.g. for d =infinite in the computation of penalty matrix P), we have an aggregation

analogous to that of Dempster-Shafer's rule of combination. This aggregation relies on the sources being evidentially uncorrelated. However, in reality this is not an easy constraint to satisfy. In our case, we have used a pre-processing stage (*Tagging*) to annotate the input data with domain knowledge. Since this pre-processing stage is applied to a common set of inputs for *all* models, it is possible to inject a common bias; hence the models could exhibit partial positive correlation.

To account for this positive correlation we decided to use a t-norm operator different from the scalar product to generate the outer product. We used Schweizer and Sklar's parameterized family of T-norms [10], which has been studied by many researchers [2,7], and performed some experiments for positive values of parameter p . For the reader's convenience, we recall Schweizer and Sklar's parameterized family of T-norms:

$$T(a,b)_p = (a^{-p} + b^{-p} - 1)^{-1/p} \quad \text{when } p > 0$$

$$T(a,b)_p = (a * b) \quad \text{when } p \rightarrow 0$$

$$T(a,b)_p = (a^{-p} + b^{-p} - 1)^{-1/p} \begin{cases} \text{when } p < 0 \\ \text{and } (a^{-p} + b^{-p}) \geq 1 \end{cases}$$

$$T(a,b)_p = 0 \begin{cases} \text{when } p < 0 \\ \text{and } (a^{-p} + b^{-p}) \leq 1 \end{cases}$$

Table 4 was constructed using the same penalty matrix derived from $W=0.2$ and $d=7$ and using the same threshold confidence $TI=0.2$.

Table 4: Fusion as a function of T-norms (for non-nicotine users)

	T-Norm [p]			
	$T_{[p>0]}$	$T_{[p=0.5]}$	$T_{[p=1]}$	$T_{[p \rightarrow \infty]}$
W=0.2, d=7, $TI=0.2$				
No Fused Decisions	44	39	42	42
Agreements	1690	1702	1690	1690
False Positive	59	50	60	60
False Negatives	49	43	48	48
Corrections	28	36	28	29
Complete disagreement	5	5	7	6
Total	1875	1875	1875	1875

From this table we observe that a minor adjustment for positive correlation ($p=0.5$) produces the *Pareto-best* results for the fusion operator. The 39 cases for which the fusion module makes no decisions provide relevant information. In 35 cases, the FLE also avoids making any decision by sending the application to a human underwriter, while the SRD assigns it to the least-desirable rate class. In the remaining 4 cases, the FLE fully disagrees with the SRD, by selecting different classes. Hence these four cases can be easily identified for auditing.

8.4 Experiments with static discounting

We repeated the same experiments and discounted each classifier by its degree of accuracy exhibited during the training stage. However, all classifiers had similar accuracy (~90-94%) and discounting did not create any notably different result.

9. Conclusions and Future Work

We have focused our efforts on two problems: *monitoring* the quality of a knowledge-based classifier used in daily production, and *supporting* the classifier's lifecycle, by updating its Standard Reference Decisions. To address these problems, we have developed four off-line classifiers built on different technologies: case-based, neural networks, multiple adaptive splines, and Pareto dominance. We needed to fuse the classifiers' outputs and compare them with that of the production classifier. Given the heterogeneous semantics of the off-line classifiers, we could not use traditional probabilistic approaches to fusion, such as Dempster-Shafer [11].

We considered the output of each classifier as a weight assignment, representing the (un-normalized) degree to which a given rate-class was selected by the classifier. As in DS theory, the assignment of weights to the universe all rate-classes represented the lack of commitment to a specific decision. After a post-processing stage, in which weak outputs (exhibiting low cardinality or high entropy) were removed and replaced with non-committing statements, the weight distributions were normalized to have unit-valued cardinality. We combined the normalized outputs by using an associative fusion operator. The fusion mechanism was based on an outer-product of the outputs, using a Triangular norm as the operator. The output of the fusion was a rate-class distribution and a measure of conflict among all the classifiers. We normalized the final output, identified the strongest selection of the fusion, and qualified the decision with an associated confidence measure. Finally, we used the confidence measure to assess the quality of the production engine. When the degree of confidence of a fused decision was below a lower confidence bound, that case became a candidate for *auditing*. When the degree of confidence of a fused decision was above an upper confidence bound, that case became a candidate for *augmenting the SRD*. We tested the fusion module with data sets for nicotine and non-nicotine users. In an analysis of 1,875 non-nicotine applications, we generated a distribution of the quality of the production engine. By focusing on the two tails of such distribution, we identified

~9% of the most reliable decisions and 4.9% of the least reliable ones.

The proposed fusion module plays a key role in supporting the quality assurance of the knowledge-based classifier, by selecting the most questionable cases for auditing. At the same time, the fusion module supports the SRD lifecycle by identifying the most reliable cases used for updating it.

As part of our future work we plan to add a fifth classifier, based on Support Vector Machine technology [6]. A prototype of this classifier, tested on the same data set, has shown accuracy and coverage comparable to those of the MARS- and NN-based classifiers.

References

- [1] K. Aggour, M. Pavese, P. Bonissone and W. Cheetham (2003). SOFT-CBR: A Self-Optimizing Fuzzy Tool for Case-Based Reasoning, *5th Int. Conf. on Case-Based Reasoning (ICCBR)*, pp. 5-19, Springer-Verlag, Trondheim, Norway, 2003.
- [2] P. Bonissone and K.S. Decker (1986). Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-off Precision and Complexity, in Kanal and Lemmer (eds.) *Uncertainty in Artificial Intelligence*, pp. 217-247, North-Holland.
- [3] P. Bonissone, W. Cheetham (2001). Fuzzy Case-based Reasoning for Decision Making, *IEEE Int. Conf. on Fuzzy Systems*, Melbourne, Australia.
- [4] P. Bonissone, R. Subbu, K. Aggour (2002). Evolutionary Optimization of Fuzzy Decision Systems for Automated Insurance Underwriting, *IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE '02)*, pp 1003-1008, Honolulu, Hawaii, USA.
- [5] P. Bonissone (2003). The life cycle of a Fuzzy Knowledge-based Classifier, *North American Fuzzy Information Processing Society (NAFIPS 2003)*, pp. 488-494, Chicago, IL, Aug. 2003.
- [6] N. Cristianini and J. Shawe-Taylor (2000). *An introduction to support vector machines*, Cambridge University Press.
- [7] D. Dubois and H. Prade (1984). Criteria aggregation and ranking of alternatives in the framework of fuzzy set theory, *TIMS/Studies in the Management Science*, Zimmerman, Zadeh, Gaines (eds.), Vol. 20, pp. 209-240, Elsevier.
- [8] J.H. Friedman (1991). Multivariate Adaptive Regression Splines. *Annals of Statistics*, 19: 1-141.
- [9] A. Patterson, P. Bonissone, and M. Pavese (2003). Six Sigma Quality Applied Throughout the Lifecycle of an Automated Decision System, *Journal of Quality and Reliability International* (to appear).
- [10] B. Schweizer and A. Sklar (1983). *Probabilistic Metric Spaces*, North Holland, New York, NY, USA.
- [11] G. Shafer (1976). *A mathematical theory of evidence*, Princeton University Press, Princeton, NJ, USA.