International Journal of Production Research,
Vol. 45, No. 13, 1 July 2007, 3059–3084

Taylor & Francis
Taylor & Francis Group

Theory of actionable data mining with application to semiconductor manufacturing control

D. BRAHA*†‡, Y. ELOVICI§ and M. LAST§

†Charlton College of Business, University of Massachusetts, Dartmouth, MA 02747, USA
‡New England Complex Systems Institute, Cambridge, MA 02138, USA
§Department of Information Systems Engineering, Ben-Gurion University, PO Box 653, Beer-Sheva 84105, Israel

(*Revision received January 2006*)

Accurate and timely prediction of a manufacturing process yield and flow times is often desired as a means of reducing overall production costs. To this end, this paper develops a new decision-theoretic classification framework and applies it to a real-world semiconductor wafer manufacturing line that suffers from constant variations in the characteristics of the chip-manufacturing process. The decision-theoretic framework is based on a model for evaluating classifiers in terms of their value in decision-making. Recognizing that in many practical applications the values of the class probabilities as well as payoffs are neither static nor known exactly, a precise condition under which one classifier 'dominates' another classifier (i.e. achieves higher payoff), regardless of payoff or class distribution information, is presented. Building on the decision-theoretic model, two robust ensemble classification methods are proposed that construct composite classifiers that are at least as good as any of the existing component classifiers for all possible payoff functions and class distributions. It is shown how these two robust ensemble classifiers are put into practice by developing decision rules for effectively monitoring and controlling the real-world semiconductor wafer fabrication line under study.

Keywords: Decision theory; Knowledge discovery; Actionable data mining; Cost-sensitive classification; Ensemble classification; Stochastic yield; Semiconductor manufacturing

1. Introduction

Various techniques that aim at improving the ability of meeting delivery schedules are employed by the semiconductor industry in order to increase their manufacturing profitability. Meeting delivery schedules is often considered difficult; particularly due to uncertainty associated with the material's quality and the state of the manufacturing equipment, resulting in unpredicted flow times of individual semiconductor batches. Consequently, it is highly desirable to develop techniques for predicting the quality of a production batch during the manufacturing process, thus having a satisfactory estimation of the overall production time.

*Corresponding author. Email: braha@necsi.org

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online © 2007 Taylor & Francis
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540600654475

Two basic performance measures are often utilized to assess profitability in the semiconductor industry: yield and flow times of production batches (Cunningham *et al*. 1995, Last and Kandel 2001). Here, 'yield' is defined as the ratio between the number of integrated circuits (chips) that emanate from the production process and pass quality control tests and the maximal theoretical number of integrated circuits that can be produced from the same number of wafers. Accurate estimation of the yield per batch is an important parameter that affects control and production planning (Barad and Braha 1996, Braha 1999). Imprecise estimation of the yield may result in delayed product delivery, or inventory costs due to excess manufacturing. The flow time measure refers to the problem of estimating the completion dates of individual batches in the production line. The flow times are not fixed due to unexpected equipment malfunctions along the production line and line balancing problems. Flow times and yield are closely related.

Managing the yield and flow times is a challenging engineering task. In the semiconductor industry, the problem appears in the form of yield variability between individual batches and even between specific wafers of the same production batch. This problem is associated with any production line regardless of the manufacturing technology. New methods that help to understand this variability could also suggest ways to improve the overall manufacturing process.

The manufacturing data collected by semiconductor companies is constantly growing. Still, it is very difficult to locate the important parameters that should be used to build a model, which would accurately estimate the yield. Tobin *et al*. (2000) found that the data collected does not enable manufacturing departments to effectively monitor and control the production process. Thus, there is a need for automated yield management systems, which will be able to explain and predict yield variations by using sophisticated data management and data mining techniques.

Last and Kandel (2001) applied the Info-Fuzzy Network (IFN) methodology of data mining and knowledge discovery to Work-in-Process (WIP) data obtained from a semiconductor plant. IFN is an advanced classification model having the form of an oblivious decision graph (Kohavi and Li 1995). In this methodology, the recorded features of each manufacturing batch include its design parameters, process tracking data, line yield, etc. The data are prepared for data mining by converting a sequential dataset into a relational table. Fuzzy-based techniques of automated perception are then used for producing a compact set of linguistic rules from the induced classification models. Other classifier algorithms have been utilized in a variety of semiconductor manufacturing processes as described in Braha (2001) and Braha and Shmilovici (2002, 2003). The present paper develops a model for evaluating classifiers in terms of their value in decision-making. Recognizing that in many practical applications the values of the class probabilities as well as payoffs (usually represented by misclassification costs) are neither static nor exactly known, we present a precise condition under which one classifier 'dominates' another classifier (i.e. achieves higher payoff), regardless of payoff or class distribution information. Building on the decision-theoretic model, we propose two robust ensemble classification methods that construct composite classifiers, which are at least as good as any of the component classifiers for all possible payoff functions and class distributions. We show how these two robust ensemble classification methods can be used to improve the prediction accuracy of yield and flow time of every batch in a real-world semiconductor manufacturing environment.

More specifically, the contribution of this paper is fourfold:

- We develop a new model for evaluating classifiers in terms of their value in decision-making. According to the new model, the predictions displayed by a classifier, or an ensemble of classifiers, lead to actions by the decision-maker. The actions, based on the derived information, are determined so as to maximize the expected payoff for the decision-maker. The proposed model is based on the belief that the question of evaluating classifiers can only be addressed in a microeconomic framework, and that the predictions displayed by a classifier are 'effective' only insofar as the derived information leads to actions that increase the expected payoff for the decision-maker (other decision-theoretic considerations may include dynamic allocation problems, e.g. Freund and Schapire 1997).
- Building on the decision-theoretic model, we derive a precise condition under which one classifier 'dominates' another (i.e. achieves higher expected payoff) for all possible cost and class distributions. This condition is based on the respective confusion matrices of the compared classifiers; thus, it decouples classification performance from class and cost distribution information. Similarly to the ROC convex hull method (Provost and Fawcett 2001), the method we propose for the comparison of classifier performance shares some of the goals of robustness to imprecise (changing) class distributions and misclassification costs. (The ROC convex hull method for evaluating classifiers is limited to two class domains. Extending this method to multiple dimensions is an open problem (Provost *et al.* 1998).)
- In the context of standard non-cost-sensitive learning, many authors have found that the accuracy of standard classifiers (e.g. decision trees) can be improved or exceeded through composite classifier architectures incorporating a set of component classifiers (Skalak 1995). When payoffs associated with the decisions are taken into account, our modelling approach enables us to address the goal of achieving a robust composite classifier, i.e. a composite classifier that produces a dominating model, which is at least as good (which here means achieving higher payoff) as any of the constituent classifiers for all possible payoff and class distributions. We propose two ways by which component classifiers can be combined. In both cases, we show that the resulting composite classifier produces dominating models.
- Finally, we apply the decision-theoretic classification framework to a real-world situation in the semiconductor industry showing how this research is useful beyond the purely theoretical arena. We show how two robust ensemble classification methods constructing composite classifiers can be used to reduce fabrication costs as a result of improving the prediction accuracy of the expected yield and flow time of every batch in the semiconductor manufacturing process.

The paper is organized as follows. Section 2 reviews knowledge discovery with classifiers that is one of the widely used tasks in data mining. Section 3 presents the basic decision-theoretic framework, and the notion of a 'dominating' classifier. We present a condition under which one classifier 'dominates' another based on their underlying confusion matrices. Section 4 presents two ways by which component classifiers can be combined. We show that, in both cases, the resulting composite

classifier is at least as good ('dominates') as each component classifier. In section 5, we describe an application of the proposed decision-theoretic framework to a real-world problem of production control in semiconductor manufacturing. Section 6 concludes the paper. For clarity, the proofs of the theorems are presented in Elovici and Braha (2003).

2. Knowledge discovery with classifiers

Classification is one of the most widely used tasks of machine learning. The most common method for evaluating the performance of a classifier is to assess its predictive accuracy on a test set or using various resampling techniques such as cross-validation and boot-strapping, which provide a very good indication of the predictive performance on future records (Kohavi 1995).

The classification accuracy performance measure tacitly assumes that misclassification costs are equal, which is rarely the case in real-world applications (Provost and Fawcett 1997, Provost *et al*. 1998). This realization has led to the development of studies that take the cost into consideration (Turney 1995). Considerable work has been devoted to classification accuracy (Provost *et al*. 1998). The most common method of achieving this cost-sensitive learning, which is concerned with the developing and the analysis of algorithms that produce minimum cost-classifiers and do not necessarily maximize the objective is to rebalance ('stratify') the training set given to the learning algorithm (Chan and Stolfo 1998, Domingos 1999, Elkan 2001).

Often in real-world applications, the misclassification costs and class distributions are imprecise or change from situation to situation (Friedman and Wyatt 1997, Zahavi and Levin 1997, Provost *et al*. 1998). In such cases, several proposals for comparing multiple classifiers have been made (Bradley 1997, Provost *et al*. 1998, Adams and Hand 1999, Margineantu and Dietterich 2000). Provost and Fawcett (2001) proposed a method, called the ROC convex hull, for evaluating classifiers in two class domains based on the true positive and false positive rates. The method specifies how classifiers can be compared under imprecise misclassification costs and class distributions. Domingos (1998) proposed a method for comparing two alternative classifiers by employing a cost model based on the notion of Net Present Value (NPV). (The proposed cost model extends the models used in Turney (1995) and Masand and Piatetsky-Shapiro (1996).) The comparison method uses as input variables the classifier's confusion matrix, the 'cash flow matrix', the cost per decision, the one-time cost of deploying the system, and the rate of return on investments.

While much research on cost-sensitive learning and classifier evaluation has focused on making optimal cost-sensitive classification decisions, there is virtually no rigorous and formal research related to the question of actionability — the ability of the classifier to suggest concrete and profitable action (or strategy) by the decision-makers. (The common approach employs Bayes' optimal prediction, which assigns each example to the class that minimizes the conditional risk, i.e. the expected cost of predicting that an example belongs to a particular class. The Bayes' optimal prediction is guaranteed to achieve the lowest possible overall expected cost.)

The present paper sets the question of actionability in the context of decision-making under uncertainty in general, and information economics in particular. The question of decision-making under uncertainty has been extensively studied in the fields of decision theory, organizational decision-making, and management science (e.g. Raiffa 1976); however, this line of research has not yet been explored by the data mining and machine learning community. The model, the results, and the case study presented in this paper aim at closing this gap.

3. Decision-theoretic classification model

3.1 *Basic terminology*

The paper considers induction problems for which machine-learning algorithms generate classification models that are used to classify future unlabeled examples (e.g. 'silicon wafers') into two or more predetermined classes (e.g. 'high wafer yield'). It views a classification algorithm as a tool used by the decision-maker that generates 'signals' (e.g. predicted 'wafer yield') about unobservable 'states' of the world (e.g. actual 'wafer yield'). The decision-maker then selects an action (e.g. 'scrapping a wafer') out of a set of available actions. The rule that assigns an action for each possible observed signal is called a decision rule. The decision-maker aims at implementing the 'optimal' decision rule, which maximizes the expected payoff derived from using the classification algorithm.

The proposed decision-theoretic framework is applied to the classification problem as follows. (The present modelling approach is based on the Information Structure model presented by Marschak (1971), McGuire and Radner (1986) and others (Demski 1972, Ahituv and Ronen 1988).) Certain examples (instances) are to be labelled as coming from a set of actual classes, say $S=\{s_1,\ldots,s_n\}$. Let $\pi=(\pi_1,\ldots,\pi_n\}$ denote the vector of prior probabilities of the actual classes in the population under study. Each example gives rise to certain measurements, which together form the feature vector X. The task of the classifier is to classify an example to one of n predicted classes in $Y=\{y_1,\ldots,y_n\}$ on the basis of the observed value $X=x$. We use the labels $Y=\{y_1,\ldots,y_n\}$ for the classifications produced by a model in order to distinguish between the actual class and the predicted class of an example. For simplicity and without loss of generality, we assume the predicted classes to have the same meaning as the actual classes (as in standard classification learning). In section 3, when considering multiple classifier combination, we allow for the set of predicted classes Y to be distinct from the set of actual classes S. The classifier measure of performance is defined in terms of the confusion matrix P (Domingos 1998, Kohavi and Provost 1998). (In practice, the confusion matrix can be estimated by dividing the database randomly into training and test instances, learning on the training instances, and counting the number of test instances for which the predicted class is j given the actual class is i, e.g. Turney (1995) and Domingos (1998).) The confusion matrix is a stochastic (i.e. Markovian) matrix of size $n\times n$ of conditional probabilities, where each of its elements p_{ij} defines the probability of predicting a class y_j given an example of actual class s_i.

Next, we extend the traditional classification problem by incorporating *decisions* associated with the classifier predictions. More specifically, a previously unseen

example is introduced to the classifier, the decision-maker observes the predicted class, and chooses actions accordingly. This is radically different from the classical formulation of the classification problem where the sets of classes and actions are assumed to be identical. Let $A = \{a_1, \ldots, a_m\}$ be a finite set of actions that can be taken by the decision-maker. The decision rule of the decision-maker is described by a stochastic matrix D of size $n \times m$, where an element $d_{i,j}$ denotes the probability that the decision-maker applies action a_j given the predicted class y_i.

After following the probabilistic decision rule, a particular action a_i is realized and employed by the decision-maker. Actions are payoff-sensitive. (The present model is formulated in terms of revenues or benefits instead of costs (hence, the use of 'payoff'). Costs can be treated as negative revenues (e.g. Elkan 2001).) If the decision-maker applies strategy a_i and the example's actual class turns out to be s_j, then a payoff $u_{i,j}$ is incurred. Let U be the cardinal payoff matrix of size $m \times n$, which associates payoffs with *pairs* of actions and actual classes. (Existing models of cost-sensitive learning (e.g. Provost and Fawcett 2001) consider only misclassification costs, i.e. costs associated with pairs of predicted and actual classes rather than costs (payoffs) resulting from concrete actions.)

3.2 *Evaluating classifier performance*

Given a particular decision matrix D, the expected payoff is:

$$EU = trace(P \cdot D \cdot U \cdot \Pi) \tag{1}$$

where *trace* is the matrix trace operator. The square matrix Π is obtained by placing the vector of prior probabilities π along the main diagonal and zeros elsewhere, and the *trace* operator denotes the sum of the elements along the main diagonal.

The decision-maker wishes to maximize the expected payoff as given by equation (1). This is achieved by choosing an optimal decision matrix $D^* \in \mathcal{D}$, where $\mathcal{D}$ is the set of all Markovian matrices. Since the expected payoff as given by equation (1) is linear in the elements $d_{i,j}$ of the decision matrix D, the optimal decision rule can be obtained by solving the following linear programming problem:

$$\max_{d_{ij}} \; trace(PDU\Pi)$$

subject to:

$$\sum_{j=1}^{m} d_{ij} = 1 \quad \text{for} \quad i = 1, 2, \ldots, n \tag{2}$$

$$d_{ij} \geq 0 \quad \text{for} \quad i = 1, 2, \ldots, n, \; j = 1, 2, \ldots, n$$

The constraints in (2) follow from the properties of a stochastic matrix. It can be shown that at least one of the optimal solutions is in the form of a decision matrix whose elements are 0 or 1 (a pure decision rule). This fact can be exploited to obtain an efficient algorithm for producing the optimal decision rule (we make use of this fact below).

Having established the basic model, we define the classifier performance as the maximum expected payoff that can be achieved by employing the classifier

as a means for delivering potential payoff. (Recall that a classifier affects the maximum expected payoff through its confusion matrix P.)

Example 1: The following simple example is provided to illustrate the above notation. Consider a loan-screening application in which applicants for a loan from a bank are classified as one of three classes: 'low', 'medium' or 'high' payment risks. The bank has applied a machine-learning algorithm to its database of previous loan applications and their payment results, and has induced a classifier that is used to classify future loan applications. The set of actions that can be taken by the bank (the decision-maker), based on the currently employed classifier's prediction, includes 'approve an application' or 'reject an application'. The consequence of rejecting a low payment-risk applicant carries a certain reputation cost; the cost of approving a loan for a high payment-risk applicant can be much higher. The above information is formalized as follows:

- Actual classes (S) of an applicant: {'low risk', 'medium risk', 'high risk'}.
- Prior probabilities (π): {0.8 for 'low risk', 0.15 for 'medium risk', 0.05 for 'high risk'}.
- Predicted classes (Y): {'low risk', 'medium risk', 'high risk'}.
- Actions (A): {'approve application, 'reject application}.
- Payoff matrix: $U =$

		Actual Class		
Action	low	medium	high
approve	200	100	−1000
reject	−30	−20	20

- Confusion matrix: $P =$

		Predicted Class		
Actual Class	low	medium	high
low	0.7	0.2	0.1
medium	0.1	0.8	0.1
high	0.05	0.15	0.8

The optimal decision rule D, which is obtained by maximizing the expected payoff via solving the linear programming problem as given by (2), is:

- Optimal decision rule: $D =$

	Action	
Predicted Class	approve	reject
low	1	0
medium	1	0
high	0	1

By plugging this optimal decision rule in the expected payoff EU as given by equation (1), we obtain the maximum expected payoff of 145.6.

Assume that the bank is considering modifying the existing classification procedure by employing a new classifier for which the confusion matrix is:

$Q =$

	Predicted Class		
Actual Class	low	medium	high
low	0.9	0.05	0.05
medium	0.1	0.8	0.1
high	0.1	0.15	0.75

The maximum expected payoff that can be achieved by this classifier is 152.25. Thus, the new classifier represented by the confusion matrix Q is preferred over the existing classification system. (In reality, the final decision whether to deploy the new classification system is an investment decision, which can be addressed using standard net present value (NPV) analysis (Domingos 1998).)

3.3 *Comparing classifiers*

Given a set of classifiers, a database of past instances, and a precise cost function, the classifier that achieves the largest maximum expected-payoff value is selected. Unfortunately, in many practical applications the values of the class probabilities or payoffs are neither static nor precisely known (Provost and Fawcett 2001). Without class distribution and payoff information, a method for comparing multiple classifiers that is robust to imprecise and changing environments is required. In the context of our decision-theoretic framework, we present a precise condition under which one classifier 'dominates' another (i.e. achieves higher payoff) for all possible cost and class distributions.

As mentioned above, we can identify classifiers in terms of their respective confusion matrices. That is, given two classifiers that are used to classify future unlabeled examples coming from the same set of actual classes, we can contrast their measures of performance by comparing their respective confusion matrices. The domination condition below (Theorem 1) is based on the respective confusion matrices of the compared classifiers; thus, it decouples classification performance from payoff and class distribution information.

Definition 1: Consider two classifiers C_P and C_Q with corresponding confusion matrices P and Q, respectively. The confusion matrix Q dominates the confusion matrix P if the maximal expected payoff yielded by P is not larger than that yielded by Q for all payoff matrices U and all prior probability matrices Π. We also say, in this case, that 'classifier C_Q dominates classifier C_P'.

The domination condition is given in terms of the following partial rank ordering of confusion matrices (known as Blackwell's theorem; Demski 1972).

Theorem 1: *The confusion matrix Q dominates the confusion matrix P if and only if there exists a Markovian matrix M with appropriate dimensions such that $Q \cdot M = P$.*

Explanation: Assume that the decision-maker must select one of several alternative classification systems before observing the predicted class. To this end, the decision-maker can apply the procedure described in section 3.2 to evaluate the maximum payoff that can be achieved from each classifier—given the decision-maker's prior probability of the actual classes, the corresponding confusion matrices, and the payoff matrices corresponding to the alternative classifiers. This process of determining the payoff maximizing decisions for each classifier will allow the decision-maker to rank order the classifiers according to the expected payoffs derived from employing them.

Can two classifiers be rank ordered directly without carrying out the process of comparing their expected payoff to the decision-maker? Theorem 1 shows that this can be done by looking for a direct representation of one classifier as a

transformation of the other. The condition stated in Theorem 1 can also be written as $P_{ik} = \sum_j Q_{ij} M_{jk}$; therefore, M_{jk} can be interpreted as the conditional probability that when the predicted class k is obtained by classifier C_P, and a predicted class j was actually obtained by classifier C_Q. In other words, the predictions of the classifier C_Q are garbled by the matrix M. Intuitively, the garbled classifier C_P is essentially of lesser utility than classifier C_Q. Put differently, classifier C_Q yields a more precise description of which of the actual classes will occur than the second classifier C_P (C_Q is finer than C_P). Since the finer classifier conveys as much and possibly more information about the actual classes, it is generally preferred to another.

Example 2: Consider two classifiers represented by the following confusion matrices:

$$P = \begin{bmatrix} 0.6 & 0.2 & 0.2 \\ 0.2 & 0.6 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}, \quad Q = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.1 & 0.9 & 0 \\ 0.25 & 0.25 & 0.5 \end{bmatrix}.$$

It can be checked that the Markovian matrix

$$M = \begin{bmatrix} 0.65 & 0.15 & 0.2 \\ 0.15 & 0.65 & 0.2 \\ 0 & 0 & 1 \end{bmatrix}$$

satisfies $Q \cdot M = P$; thus, according to Theorem 1, the classifier represented by Q dominates the classifier represented by P, regardless of payoff or class distribution information.

In reality, it would be hard to find evidence for which standard learning algorithms (e.g. C4.5 or naïve Bayes) produce dominating classifiers (models) for standard benchmark data sets (e.g. Provost *et al.* 1998). Thus, if the class probabilities and payoffs are unavailable, we cannot claim that one standard learning algorithm dominates the other. In this case, one has to look at ranges of class distributions and payoffs for which each classifier dominates. Despite that, we show (building on Theorem 1) in the remaining of the paper how to construct composite classifiers that dominate any of their component classifiers. (A composite classifier is obtained by learning several models and then 'combining' their predictions (Skalak 1995, Braha and Shmilovici 2002).)

4. Building 'dominating' classifiers

4.1 *Composite classification*

The past several years has witnessed a resurgence of research by the machine learning and pattern recognition communities to learn how to create and combine an ensemble of classifiers (Skalak 1995, Braha and Shmilovici 2002). There are many ways by which a composite classifier can be designed to perform supervised learning. For example, each component classifier can classify the observed feature vector to one of a number of classes. Then, the combining classifier incorporates the predictions of the component classifiers (e.g. by voting) to form the composite classifier prediction. There are various architectures for combining classification algorithms of which the primary architectures are: stacked generalization

(Wolpert 1992), boosting (Schapire 1990, Freund and Schapire 1997), bagging (Breiman 1996), and recursive partitioning (Brodley 1994).

When considering machine-learning algorithms for which the classification accuracy is the used performance metric, combining the predictions of a set of classifiers has been empirically documented to be an effective way to create composite classifiers that are generally more accurate than any of the component classifiers on disparate domains such as identification of satellite data (Benediktsson *et al.* 1993), hand-written character recognition (Ho *et al.* 1994), economic and weather predictions (Clemen 1989), and protein sequence identification (Zhang *et al.* 1992). (The interest in classifier combination has also been heightened by the possibility that by intelligently combining a set of 'simple' classifiers, we may be able to perform classification better than with the sophisticated algorithms currently available (Skalak 1995).)

Although the fact that classifiers can be combined to create a composite classifier with higher accuracy has not been formally shown, several strategies have been effective in achieving this goal. (For an excellent review of various composite classifiers design criteria and architectures, see Skalak (1995).) The first strategy is to construct component classifiers that are highly accurate as independent classifiers. The second strategy is to have composite classifiers that are diverse and behave very differently from one another. The third strategy is to avoid prohibitively expensive classifiers both in terms of number of component classifiers and the computational resources assigned for their training.

While much research on composite classification has focused on constructing architectures that achieve highly accurate composite classifiers, little attention has been given to composite classification when misclassification costs are taken into account. Domingos (1999) presented a cost-sensitive learning procedure called MetaCost. (For other cost-sensitive learning procedures that incorporate multiple classifiers include, see Chan and Stolfo (1998) and Ting and Zheng (1998).) MetaCost, which uses a variant of Bagging (Breiman 1996) as the ensemble method, is based on relabelling training examples with their estimated minimal-cost classes, and applying the error-based learner to the new training set. The method is shown empirically to yield lower costs compared to error-based classification and to stratification. Provost and Fawcett (2001) proposed a visualization method in two-class domains (called ROC convex hull), which given a set of classifiers produces a robust classifier that performs at least as the best classifier under any target cost and class distributions. The method requires exact knowledge of the target conditions (misclassification costs and class distribution) when used for classification in run time (Provost and Fawcett 2001, section 3.6). (More specifically, the ROC convex hull method needs to translate environmental conditions to the slope of the corresponding iso-performance line (m_{ec}). This, in turn, is used to locate the false-positive (FP) value of the point where the slope of the ROC convex hull is m_{ec}.) Despite its intuitive appeal, the ROC convex hull method is hard to generalize to multiple dimensions (Domingos 1998, Provost *et al.* 1998).

Both of the above methods focus on optimal cost-sensitive classification decisions. Building on our decision-theoretic model, when payoffs associated with the decisions are taken into account, the fundamental question is whether classifiers from any given model type can be combined to create dominating composite classifier, i.e. robust classifiers that achieve higher payoffs for the decision-maker for

all possible costs and class distributions. In the following, we formally show that the answer is affirmative by proposing two ways by which component classifiers can be combined.

4.2 *Cartesian composite classification*

This section presents an ensemble method (called Cartesian composite classification) that combines into one composite classifier two independent component classifiers that are used to classify examples coming from the same set of actual classes. The predictions displayed by the Cartesian composite classifier will then lead to actions by the decision-maker.

The ensemble method presented below assumes that component classifiers are probabilistically independent. Component classifiers are probabilistically independent if the probability of deciding by one classifier that an example of actual class s_i belongs to class y_j does not depend on the classification produced by the other classifier. For error-based learners, the independence assumption has been examined by a number of researchers who observed an improvement in predictive accuracy from combining classifiers that make independent errors or are not highly correlated (Hansen and Salamon 1990, Wolpert 1992, Skalak 1995, Ali and Pazzani 1996). Several methods for creating independent component classifiers have been identified (Skalak 1995): training the component classifiers on different samples of the training set; using classifiers from different model types (e.g. decision trees and neural networks); making the classifiers applicable to separate regions of the space of examples; selecting component classifiers that apply different features; and varying the dependent parameters of a classifier, thus creating different component classifiers from the same model type.

4.2.1 Defining the Cartesian composite classifier. Next, we define the Cartesian composite classifier. First, we describe the Cartesian product of two vectors x and y, $x \times y$, by taking all possible products between the elements of x and those of y. That is, if $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_m)$ then $x \times y$ is the 1 by nm vector:

$$x \times y = (x_1y_1, x_1y_2, \ldots, x_1y_m, x_2y_1, x_2y_2, \ldots, x_2y_m, \ldots, x_ny_1, x_ny_2, \ldots, x_ny_m).$$

The *Cartesian* product of two matrices P and Q of size n by m_1 and n by m_2, respectively, is defined as follows:

$$P \times Q = \begin{pmatrix} p^1 \times q^1 \\ p^2 \times q^2 \\ \vdots \\ p^n \times q^n \end{pmatrix}, \tag{3}$$

where p^i (respectively q^i) is the ith row in P (respectively Q).

Consider two independent classifiers represented by the confusion matrices P and Q of size $n \times m_1$ and $n \times m_2$, respectively. (Recall that the model allows for classifiers, where the set of predicted classes is not necessarily identical to the set of actual classes. This is especially relevant when considering multiple classifier combination.) The two independent classifiers are combined into one composite classifier, called the

Cartesian composite classifier, whose confusion matrix R of n rows and m_1m_2 columns is defined as follows:

$$R = P \times Q. \tag{4}$$

The Cartesian composite classifier is used to classify examples coming from the same set of actual classes as that of P and Q, and it classifies actual example classes to one of the predicted classes in the set $Y^P \times Y^Q$. (The Cartesian composite classifier can be defined over several component classifiers, e.g. $R = (P \times Q) \times S$, etc.)

Example 3: Consider two independent classifiers represented by the following confusion matrices:

$$P = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} y_1^P & y_2^P \end{array} \\ \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix} \end{array}, \quad Q = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cc} y_1^Q & y_2^Q \end{array} \\ \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix} \end{array}.$$

The Cartesian composite classifier is used to classify examples coming from the set of actual classes $S^{\text{Cartesian}} = \{s_1, s_2\}$. The task of the classifier is to classify an example to one of the four classes in $Y^{\text{Cartesian}} = \{y_1^P \& y_1^Q, y_1^P \& y_2^Q, y_2^P \& y_1^Q, y_2^P \& y_2^Q\}$on the basis of the observed value $X = x$. The confusion matrix associated with the Cartesian composite classifier is:

$$R = P \times Q = \begin{array}{c} \\ s_1 \\ s_2 \end{array} \begin{array}{c} \begin{array}{cccc} y_1^P \& y_1^Q & y_1^P \& y_2^Q & y_2^P \& y_1^Q & y_2^P \& y_2^Q \end{array} \\ \begin{pmatrix} 0.48 & 0.32 & 0.12 & 0.08 \\ 0.06 & 0.24 & 0.14 & 0.56 \end{pmatrix} \end{array}.$$

The Cartesian composite classifier is implemented as follows. First, an example (say with actual class S_2) is classified by both classifiers, and their predictions are concatenated to form one of the four predicted classes in $Y^{\text{Cartesian}}$. The decision-maker observes the predicted class (say $y_1^P \& y_2^Q$), and chooses the optimal action (say a_1) accordingly from within the set of available actions $A^{\text{Cartesian}} = \{a_1, a_2\}$. In other words, both classifiers are run, and a payoff $u_{1,2}$ is associated with the action a_1 and actual class s_2.

Table 1 shows an algorithm for producing the optimal decision rule given two independent classifiers P and Q combined with the Cartesian product. The algorithm in table 1 exploits the fact that at least one of the optimal solutions of the linear programming problem (2) is in the form of a decision matrix whose elements are 0 or 1 (a pure decision rule).

4.2.2 Domination of the Cartesian composite classifier. We now show that the Cartesian composite classifier produces a dominating model that is at least as good (in terms of maximization of payoffs) as any of the component classifiers for all possible payoff and class distributions.

Theorem 2: *If R is the Cartesian product of the confusion matrices P and Q, then R dominates both confusion matrices P and Q.*

Table 1. Algorithm for generating the optimal decision rule given a Cartesian composite classifier.

Input: π, P, Q, A, U
Output:
Decision rule of the Cartesian Classifier, D^*.
Maximum expected payoff achievable from the Cartesian Classifier, V^*.
begin
Compute confusion matrix $R = P \times Q$
Let $D = [d_{ij}]$ be an algebraic matrix representing the decision rule of the composite classifier. Compute the linear polynomial $trace(R \cdot D \cdot U \cdot \Pi) = \sum_i \sum_j c_{ij} d_{ij}$ (c_{ij} is the constant coefficient of the variable d_{ij})
for every predicted class i of the composite classifier, let $j^* = \arg\max_j \{c_{ij}\}$.
Set $d_{ij} = 1$ for $j = j^*$, and $d_{ij} = 0$ for $j \neq j^*$.
Set $D^* = [d_{ij}]$, $V^* = \sum_i \sum_j c_{ij} d_{ij}$.
Compute the maximum expected payoff, $V^* = trace(R \cdot D^* \cdot U \cdot \Pi)$
end

We conclude from Theorem 2 that the maximum expected payoff that is achieved by a standard classifier could be improved, regardless of payoff or class distribution information, through Cartesian composite classifier architectures that incorporate additional component classifiers. (The final decision whether to deploy the Cartesian composite classifier or not is an investment decision, which depends on the deployment costs of the additional component classifiers.)

Given a Cartesian composite classifier, the decision-maker may wish to improve one or more of the component classifiers. Theorem 3 below shows that improving the 'quality' of any one of the constituent classifiers improves the effectiveness (in terms of maximization of payoffs) of the overall composite classifier as well.

Theorem 3: *Let P, Q, and R be three confusion matrices. If R dominates Q, then $R \times P$ dominates $Q \times P$.*

4.3 *Doubly-Cartesian composite classification*

Section 3.1 showed how several independent component classifiers that are used to classify examples coming from the same set of actual classes are combined using the Cartesian operator into one composite classifier. This section addresses scenarios in which the component classifiers are independent, and are used to classify examples coming from disjoint sets of actual classes. We say that such component classifiers are disjoint. To illustrate, consider two classifiers: a decision tree classifier that is used to classify examples coming from one of two credit risk classes (e.g. good or bad), and a neural network classifier that is used to classify examples coming from one of two credit usage classes (e.g. heavy or light). For error-based learners, a few combining algorithms, which incorporate disjoint component classifiers have been suggested, such as the Error Correction Output Coding (ECOC) ensemble method (Dietterich and Bakiri 1991). Building on our decision-theoretic model, we next present an ensemble method (called Doubly-Cartesian composite classification), which incorporates a set of disjoint component classifiers.

4.3.1 Defining the Doubly-Cartesian composite classifier. First, we define the Doubly-Cartesian product of two matrices P and Q of size $n_1 \times m_1$ and $n_2 \times m_2$, respectively, as follows:

$$P \otimes Q = \begin{pmatrix} p^1 \times q^1 \\ p^1 \times q^2 \\ \vdots \\ p^1 \times q^{n2} \\ \vdots \\ p^{n1} \times q^1 \\ p^{n1} \times q^2 \\ \vdots \\ p^{n1} \times q^{n2} \end{pmatrix}, \tag{5}$$

where p^i (respectively q^i) is the ith row in P (respectively Q), and $\times$ is the Cartesian product defined in section 3.2.1.

Next we show how several independent component classifiers that are used to classify examples coming from disjoint sets of actual classes are combined into one composite classifier. Consider two independent classifiers represented by the confusion matrices P and Q of size $n_1 = m_1$ and $n_2 = m_2$, respectively. The two independent classifiers are combined into one composite classifier, called the Doubly-Cartesian composite classifier, whose confusion matrix R of $n_1 n_2$ rows and $m_1 m_2$ columns is defined as follows:

$$R = P \otimes Q. \tag{6}$$

The Doubly-Cartesian composite classifier is used to classify examples coming from the set of actual classes in $S^P \times S^Q$, and it classifies actual example classes to one of the predicted classes in the set $Y^P \times Y^Q$. Layered networks of classifiers can be easily defined, e.g. $(P \times Q) \otimes (R \times S)$, etc.

Example 4: Consider two independent classifiers represented by the following confusion matrices:

$$P = \begin{array}{c} \\ s_1^P \\ s_2^P \end{array} \begin{array}{c} \begin{array}{cc} y_1^P & y_2^P \end{array} \\ \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix} \end{array}, \quad Q = \begin{array}{c} \\ s_1^Q \\ s_2^Q \end{array} \begin{array}{c} \begin{array}{cc} y_1^Q & y_2^Q \end{array} \\ \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix} \end{array}.$$

The Doubly-Cartesian composite classifier is used to classify examples coming from the set of actual classes $S^{\mathrm{D-Cartesian}} = \{s_1^P \& s_1^Q, s_1^P \& s_2^Q, s_2^P \& s_1^Q, s_2^P \& s_2^Q\}$. The task of the classifier is to classify an example to one of the four classes in

$Y^{\text{D-Cartesian}} = \{y_1^P \& y_1^Q, y_1^P \& y_2^Q, y_2^P \& y_1^Q, y_2^P \& y_2^Q\}$ on the basis of the observed value $X = x$. The confusion matrix associated with the Doubly-Cartesian composite classifier is:

$$R = P \otimes Q = \begin{array}{c} \\ s_1^P \& s_1^Q \\ s_1^P \& s_2^Q \\ s_2^P \& s_1^Q \\ s_2^P \& s_2^Q \end{array} \begin{array}{c} \begin{array}{cccc} y_1^P \& y_1^Q & y_1^P \& y_2^Q & y_2^P \& y_1^Q & y_2^P \& y_2^Q \end{array} \\ \begin{pmatrix} 0.48 & 0.32 & 0.12 & 0.08 \\ 0.16 & 0.64 & 0.04 & 0.16 \\ 0.18 & 0.12 & 0.42 & 0.28 \\ 0.06 & 0.24 & 0.14 & 0.56 \end{pmatrix} \end{array}.$$

The Doubly-Cartesian composite classifier is implemented as follows. An example (say with actual classes s_2^P and s_2^Q) is classified by both classifiers, and their predictions are concatenated to form one of the four classes in $Y^{\text{D-Cartesian}}$. The decision-maker observes the predicted class (say $y_1^P \& y_2^Q$), and chooses the optimal action (say a_1) accordingly from within the set of available actions $A^{\text{D-Cartesian}} = \{a_1, a_2\}$. A payoff $u_{1,3}$ is associated with the action a_1 and actual class $s_2^P \& s_1^Q$.

Table 2 shows an algorithm for producing the optimal decision rule given two disjoint classifiers P and Q combined with the Doubly-Cartesian product.

4.3.2 Domination of the Cartesian composite classifier. Next, we prove that the Doubly-Cartesian composite classifier *dominates* any of the component classifiers from which it is formed, regardless of payoff or class distribution information.

Theorem 4: *If R is the Doubly-Cartesian product of the confusion matrices P and Q, then R dominates both confusion matrices P and Q.*

In addition, it is shown that improving one of the component classifiers provides a more effective composite classifier for the decision-maker, regardless of the quality of the other component classifiers.

Table 2. Algorithm for generating the optimal decision rule given a Doubly-Cartesian composite classifier.

Input: π^P, π^Q, P, Q, A, U
Output:
Decision rule of the Doubly-Cartesian Classifier, D^*.
Maximum expected payoff achievable from the Doubly-Cartesian Classifier, V^*.
begin
Compute confusion matrix $R = P \otimes Q$
Compute the vector of prior probabilities of the actual classes in $S^{\text{D-Cartesian}}$, $\pi^{\text{D-Cartesian}} = \pi^P \times \pi^Q$.
Same as Steps 2–5 in table 1.
end

Theorem 5: *Let Q, R be two confusion matrices that are used to classify examples coming from the same set of actual classes. Let P be a confusion matrix that is disjoint from both Q and R. If R dominates Q, then $R \otimes P$ dominates $Q \otimes P$.*

5. Case study

5.1 *Semiconductor production control problem*

This section applies the decision-theoretic classification framework to a real-world semiconductor wafer fabrication line that suffers from constant variations in the characteristics of the chip fabrication process.

Integrated circuits are built up from a number of patterned silicon, oxide, or metal layers with specific characteristics. Photolithography is the most complex and time-consuming process of the operations involved in the fabrication of a wafer. The goal of the lithographic process is to transfer accurately a set of opaque images from the masks, which represent the elements of the basic circuit design, to a substrate on the wafer, with virtually zero defects. A photoresistant chemical is used to transfer the desired pattern by masking specific regions of the device from etching or ion implantation processes. During the process of wafer fabrication, a series of loops are performed, each one adding another layer to the device. Each loop is comprised of some or all of the major steps of photolithography, etching, stripping, diffusion, ion implantation, deposition, and chemical/mechanical planarization. At each stage, various inspections and measurements are performed to monitor the process and equipment. Supporting the entire process is a complex infrastructure of material supply, waste treatment, support, logistics, and automation.

The two important parameters that represent the process variability are 'line yield' and 'flow time'. The 'line' or 'overall' yield of a semiconductor batch is defined as the ratio between the number of good microelectronic chips obtained from a completed batch and the maximum number of chips that can be obtained from the same batch if no chips are scrapped. The actual yield of a batch can be precisely measured only at the end of the manufacturing process, although the early steps in a batch routing mostly affect it. 'Flow time' is the number of days required to complete the production of a given batch. Although the processing time of each production step in the semiconductor industry is nearly fixed, the waiting times between the operations are subject to such variable factors as equipment downtimes, process failures, and line congestions.

The Production Planning and Control department at the case study company is interested in predicting accurately the expected yield and flow time of every batch in the manufacturing process as early as possible. Since the direct cost of each fabrication step is nearly fixed, only batches with a high yield should continue their production, while batches expected to have a low yield should be stopped and scrapped immediately as being non-profitable (Barad and Braha 1996, Braha 1999). The batch flow time also affects profitability — slow batches may lose their designated customers, and thus turn into 'dead' inventory.

The expected payoff per batch can be calculated by the following formula:

$$Payoff = p_u \cdot P_S(t) \cdot Y \cdot Q - C \tag{7}$$

where p_u is the unit price of a chip; $P_s(t)$ is the probability of selling a good chip (decreases with the flow time t); Y is the actual line yield (i.e. percentage of good chips in a batch); Q is the maximum number of chips in a batch; and C is the cost of continuing the production of one batch to the end of the manufacturing process.

It is worthwhile to reiterate here that traditional cost-sensitive learning and classifier evaluation techniques do not provide the conceptual framework necessary for incorporating actions stemming from predictions (central for the above real-world production control). Indeed, as mentioned in section 2, the novelty of our work is the disassociation of predictions and actions. This framework also broadens the scope of traditional ensemble classification techniques in two important ways:

- It allows the combination of models that classify the examples into different sets of classes to produce a single model for choosing actions.
- It can be used to compose classifiers based on disjoint object types.

Below we show how the above advantages are put into practice by developing decision rules for effectively monitoring and controlling the semiconductor wafer fabrication line under study.

5.2 *Applying the decision theoretic classification framework*

To address the underlying production control problem, the decision-theoretic classification framework has been applied, and is stated as follows:

- Examples (instances). Manufacturing batches in their early stages of production.
- Actual classes. Actual values of 'yield' and 'flow time/date of completion' (two disjoint sets of actual classes). Both values refer to the 'pieces' part of the manufacturing process (see below). While the classification algorithms we used can deal with continuous values for the different input variables (attributes), the class target function should have discrete values (Braha 2001, Braha and Shmilovici 2002, 2003). To that end, the data corresponding to the continuous output variables (i.e. 'yield' and 'flow time/date of completion') have been disretized prior to applying the algorithms. Two methods could be employed to transform a continuous dependent variable into discrete class (Braha and Shmilovici 2002). In the first method, human experts define discrete classes based on their experience and intuition. In the second method, a clustering algorithm (e.g. Kohonen's self-organizing maps) is applied to search for hidden clusters (i.e. 'natural classes') in the input vectors, which correspond to input patterns with similar characteristics. In our case, the first method has been used; particularly due to the repeated nature of the semiconductor wafer fabrication line and the knowledgeable people involved in eliciting the discrete classes.
- The number of discrete classes defined by the human experts has been determined based on the intrinsic trade-off between model accuracy and model precision. According to information theory, the entropy of an output variable is proportional to the number of classes. This implies that the accuracy of the classification algorithm is expected to decrease with increasing number of classes. On the other hand, when classes represent

discretization intervals, model predictions (e.g. of 'actual yield' or 'expected payoff per batch') will be more precise if the continuous output range is partitioned into more intervals. In our case, the experts suggested that ten discretization intervals would provide a sufficient precision in predicting the values of the two continuous output variables (and expected payoff per batch thereof). If the training set could increase significantly, a finer partition of the variables range (beyond ten) could lead to even more precise prediction of the continuous output values without impeding the model's accuracy. Here, we denote the actual 'yield' classes as {AY0, AY1, AY2, AY3, AY4, AY5, AY6, AY7, AY8, AY9}, where AY stands for the 'actual yield'. Similarly, the actual 'flow time/date of completion' classes are denoted as {AD0, AD1, AD2, AD3, AD4, AD5, AD6, AD7, AD8, AD9}, where AD stands for the 'actual date of completion.'

- Actions. 'Scrapping a batch' (since it is not expected to be profitable) versus 'continuing production of a batch'). It is assumed that the decision is taken immediately after the batch wafers have been cut ('diced') into microchips ('pieces'). The transition from wafers to individual microchips is considered a crucial point in the semiconductor manufacturing process. Most yield excursions are caused by operations preceding this point.
- Predicted classes. Predicted values of 'yield' and 'flow time/date of completion'. The predicted 'yield' classes are denoted as {PY0, PY1, PY2, PY3, PY4, PY5, PY6, PY7, PY8, PY9}, where PY stands for the 'predicted yield'. The predicted 'flow time/date of completion' classes are denoted as {PD0, PD1, PD2, PD3, PD4, PD5, PD6, PD7, PD8, PD9}, where PD stands for the 'predicted date of completion.'
- Payoff. The net profit as a result of applying a certain action ('scrapping a batch' or 'continuing production') to a given batch, as determined by equation (7). The payoff is equal to the difference between the batch selling price and the cost associated with manufacturing the batch after a decision is made. The action of 'scrapping a batch' always results in zero profit, since no further costs are incurred and no income is obtained. The action 'continuing a batch production' may result in either a net profit or a net loss, depending on the batch yield and its flow time. The payoff matrix associated with the action 'continuing production' is presented in table 3. The values in table 3 have been calculated by applying equation (7). This equation includes three constant terms, which do not depend on the output variables yield and flow time: unit price of a chip p_u, the maximum number of chips in a batch Q, and the cost of continuing production C. The payoff is directly proportional to the yield Y and the probability of selling a good chip $P_s(t)$, which is a decreasing function of the flow time t. Thus, we can see in table 3 that the payoff increases as we go down from the first yield class AY0 (representing the lowest yield) to the last yield class AY9 (representing the highest yield). On the other hand, going from left (the shortest flow time) to right (the longest flow time) decreases the payoff, eventually rendering it negative. The actual values of the parameters used in formula (7) cannot be revealed here due to their confidentiality. Due to the commercial confidentiality, the actual values of the parameters used in equation (7) cannot be revealed.

Table 3. Payoffs when taking the action 'continuing production'.

Payoffs	Actual 'date of completion' class									
Actual 'yield' class	AD0	AD1	AD2	AD3	AD4	AD5	AD6	AD7	AD8	AD9
AY0	36	(80)	(138)	(203)	(265)	(350)	(448)	(559)	(661)	(964)
AY1	322	174	100	18	(62)	(171)	(296)	(438)	(567)	(954)
AY2	438	277	196	107	20	(98)	(234)	(388)	(530)	(950)
AY3	504	336	251	158	67	(57)	(199)	(360)	(508)	(947)
AY4	557	383	295	198	104	(24)	(171)	(338)	(491)	(945)
AY5	604	425	334	235	138	6	(146)	(318)	(475)	(944)
AY6	652	467	374	272	172	36	(120)	(297)	(459)	(942)
AY7	700	510	414	308	206	66	(95)	(277)	(444)	(940)
AY8	751	555	456	347	242	98	(68)	(255)	(427)	(939)
AY9	867	658	553	437	324	171	(6)	(206)	(389)	(935)

Two classification algorithms of different character have been used. C4.5 builds a decision tree using a standard top-down induction procedure (Quinlan 1993). C4.5 post-prunes an induced tree by using the pessimistic error approach. IFN is an Oblivious Decision Graph, where each level (layer) is associated with a single input feature (Maimon and Last 2000). Unlike C4.5, the IFN induction algorithm is based on completely pre-pruning the model by ignoring statistically insignificant features and patterns. Maimon and Last (2000) showed that the IFN algorithm tends to produce compact models that use much less attributes than C4.5. The experiments in (Last *et al*. 2002) also suggest that these models are usually more stable than C4.5-based models. Both classification algorithms have been trained and tested on non-identical (randomly generated) samples. The strategy of using different types of component classifiers along with selecting different training sets is well known effectively to achieve independent component classifiers (Skalak 1995).

The manufacturing data sets include 1378 batches with yield data and 1635 batches with flow time data. A complete description of these data sets is available in Last and Kandel (2001). The partition into the training/testing sets was 945/433 and 1102/533 for the yield and flow time data, respectively.

The confusion matrices representing the IFN classifier for predicting the 'yield', the C4.5 classifier for predicting the 'yield', the IFN classifier for predicting the 'date of completion', and the C4.5 classifier for predicting the 'date of completion' are described, in tables 4a–d, respectively.

The payoffs associated with the action 'continuing production' (denoted by C) are calculated by using equation (7), and are shown in table 3. The payoffs associated with the action 'scrapping a batch' (denoted by S) are 0 for any pair of actual 'yield' and 'date of completion' classes.

5.3 *Constructing the classifiers*

This section examines a variety of classifiers by applying the algorithms presented in tables 1 and 2. Table 5 presents the characteristics of the various composite classifiers that have been constructed. For example, row 9 indicates a Doubly-Cartesian classifier that combines the classifiers described in rows 5 and 6.

Table 4a. Confusion matrix of the IFN classifier for predicting 'yield'.

P_Y^{IFN}	Predicted class									
Actual class	PY0	PY1	PY2	PY3	PY4	PY5	PY6	PY7	PY8	PY9
AY0	0.273	0.303	0.061	0.030	0.212	0.030	0.000	0.030	0.000	0.061
AY1	0.071	0.179	0.125	0.107	0.393	0.018	0.000	0.018	0.000	0.089
AY2	0.049	0.268	0.122	0.146	0.268	0.073	0.000	0.049	0.000	0.024
AY3	0.077	0.179	0.026	0.205	0.333	0.077	0.000	0.000	0.000	0.103
AY4	0.000	0.074	0.130	0.222	0.407	0.037	0.000	0.037	0.000	0.093
AY5	0.045	0.114	0.068	0.182	0.409	0.045	0.000	0.114	0.000	0.023
AY6	0.056	0.083	0.194	0.111	0.361	0.000	0.000	0.083	0.000	0.111
AY7	0.026	0.158	0.079	0.132	0.289	0.053	0.000	0.105	0.000	0.158
AY8	0.019	0.074	0.000	0.167	0.259	0.056	0.000	0.130	0.000	0.296
AY9	0.079	0.079	0.026	0.053	0.211	0.026	0.000	0.132	0.000	0.395

Table 4b. Confusion matrix of the C4.5 classifier for predicting 'yield'.

P_Y^{C45}	Predicted class									
Actual class	PY0	PY1	PY2	PY3	PY4	PY5	PY6	PY7	PY8	PY9
AY0	0.242	0.152	0.212	0.030	0.091	0.030	0.091	0.091	0.000	0.061
AY1	0.179	0.107	0.054	0.071	0.125	0.143	0.143	0.071	0.071	0.036
AY2	0.171	0.171	0.098	0.073	0.073	0.171	0.220	0.000	0.000	0.024
AY3	0.128	0.103	0.077	0.103	0.051	0.103	0.256	0.026	0.077	0.077
AY4	0.074	0.074	0.111	0.204	0.130	0.019	0.185	0.056	0.093	0.056
AY5	0.091	0.068	0.045	0.068	0.114	0.114	0.295	0.023	0.136	0.045
AY6	0.139	0.028	0.083	0.028	0.056	0.167	0.250	0.083	0.111	0.056
AY7	0.079	0.053	0.132	0.053	0.105	0.132	0.132	0.053	0.079	0.184
AY8	0.074	0.093	0.019	0.111	0.056	0.111	0.148	0.037	0.093	0.259
AY9	0.026	0.026	0.053	0.000	0.026	0.132	0.105	0.053	0.079	0.500

Table 4c. Confusion matrix of the IFN classifier for predicting 'date of completion'.

P_D^{IFN}	Predicted class									
Actual class	PY0	PY1	PY2	PY3	PY4	PY5	PY6	PY7	PY8	PY9
AD0	0.262	0.279	0.148	0.000	0.115	0.148	0.016	0.016	0.016	0.000
AD1	0.245	0.358	0.132	0.000	0.094	0.151	0.000	0.019	0.000	0.000
AD2	0.167	0.130	0.204	0.000	0.111	0.333	0.037	0.019	0.000	0.000
AD3	0.167	0.167	0.148	0.000	0.148	0.315	0.019	0.000	0.037	0.000
AD4	0.183	0.100	0.150	0.000	0.150	0.300	0.083	0.033	0.000	0.000
AD5	0.111	0.159	0.159	0.000	0.143	0.365	0.032	0.016	0.016	0.000
AD6	0.146	0.208	0.083	0.000	0.021	0.188	0.229	0.125	0.000	0.000
AD7	0.265	0.162	0.103	0.000	0.029	0.221	0.088	0.118	0.015	0.000
AD8	0.250	0.075	0.050	0.000	0.125	0.150	0.150	0.175	0.025	0.000
AD9	0.219	0.219	0.125	0.000	0.094	0.188	0.031	0.094	0.000	0.031

Table 4d. Confusion matrix of the C4.5 classifier for predicting 'date of completion'.

P_D^{C45}	Predicted class									
Actual class	PY0	PY1	PY2	PY3	PY4	PY5	PY6	PY7	PY8	PY9
AD0	0.361	0.164	0.148	0.066	0.131	0.049	0.016	0.033	0.000	0.033
AD1	0.415	0.170	0.151	0.057	0.038	0.075	0.038	0.038	0.000	0.019
AD2	0.259	0.148	0.204	0.056	0.130	0.037	0.130	0.037	0.000	0.000
AD3	0.167	0.241	0.148	0.148	0.111	0.130	0.019	0.037	0.000	0.000
AD4	0.250	0.233	0.117	0.050	0.133	0.067	0.083	0.067	0.000	0.000
AD5	0.190	0.079	0.111	0.048	0.159	0.302	0.048	0.032	0.016	0.016
AD6	0.083	0.146	0.104	0.083	0.042	0.083	0.292	0.146	0.000	0.021
AD7	0.162	0.074	0.118	0.044	0.059	0.132	0.074	0.265	0.029	0.044
AD8	0.100	0.025	0.050	0.075	0.100	0.050	0.175	0.275	0.100	0.050
AD9	0.188	0.156	0.156	0.063	0.063	0.063	0.031	0.156	0.000	0.125

Table 5. Variety of classifiers considered by the Production Planning and Control department.

	Classifiers			
Type of classes	Type	Number	Confusion matrix	Number of predicted classes
1 Yield	IFN	1	P_Y^{IFN}	10
2 Yield	C4.5	1	P_Y^{C45}	10
3 Date of completion	IFN	1	P_D^{IFN}	10
4 Date of completion	C4.5	1	P_Y^{C45}	10
5 Yield	IFN, C4.5	2	$P_Y^{IFN} \times P_Y^{C45}$	100
6 Date of completion	IFN, C4.5	2	$P_D^{IFN} \times P_D^{C45}$	100
7 Yield and date of completion	IFN	2	$P_Y^{IFN} \otimes P_D^{IFN}$	100
8 Yield and date of completion	C4.5	2	$P_Y^{C45} \otimes P_D^{C45}$	100
9 Yield and date of completion	IFN, C4.5	4	$(P_Y^{IFN} \times P_Y^{C45}) \otimes (P_D^{IFN} \times P_D^{C45})$	10,000

The decision whether to scrap a batch or continue production will be determined after observing one of the 10 000 potential predicted classes.

We have generated the optimal decision rules (for each of the classifiers in table 5) to be used by the Production Planning and Control department. For example, table 6 presents the decision rule associated with classifier 5 (i.e. 'Cartesian ensemble of IFN for yield and C4.5 for yield'). To illustrate, if IFN predicts the yield class PY3 and C4.5 predicts the yield class PY7, then the process engineers should scrap the batch.

The process engineers should choose the classifier to work with by considering the following three factors: the performance of each classifier (i.e. maximum expected payoff, the deployment cost of each classifier, and the usability of each classifier. The maximum expected payoff of each classifier may be estimated by applying Step 5 in tables 1 and 2. These evaluation results are provided in table 7.

Table 6. Optimal decision rule for $P_Y^{IFN} \times P_Y^{C45}$.

	C4.5									
IFN	PY0	PY1	PY2	PY3	PY4	PY5	PY6	PY7	PY8	PY9
PY0	S	S	S	S	S	S	C	S	C	S
PY1	S	S	S	S	S	S	C	S	C	S
PY2	S	S	S	S	S	S	C	S	C	S
PY3	S	S	S	S	S	S	C	S	C	S
PY4	S	S	S	S	S	S	C	S	C	S
PY5	S	S	S	S	S	S	C	C	C	C
PY6	S	S	S	S	S	S	C	C	C	C
PY7	S	S	S	S	S	S	C	C	C	C
PY8	S	S	S	S	S	S	C	C	C	C
PY9	S	S	S	C	C	C	C	C	C	C

Table 7. Evaluation results.

Classifier	Maximum expected payoff	Computation time (s)
1. IFN for yield, P_Y^{IFN}	4.79453	0.141
2. C4.5 for yield, P_Y^{C45}	6.19675	0.062
3. IFN for date of completion, P_D^{IFN}	5.201	0.063
4. C4.5 for date of completion, P_D^{C45}	23.5085	0.078
5. Cartesian ensemble of IFN for yield and C4.5 for yield, $P_Y^{IFN} \times P_Y^{C45}$	8.65311	2.438
6. Cartesian ensemble of IFN for date of completion and C4.5 for date of completion, $P_D^{IFN} \times P_D^{C45}$	38.2436	2.515
7. Doubly-Cartesian ensemble of IFN for yield and IFN for date of completion, $P_Y^{IFN} \otimes P_D^{IFN}$	14.9279	13.703
8. Doubly-Cartesian ensemble of C4.5 for yield and C4.5 for date of completion, $P_Y^{C45} \otimes P_D^{C45}$	30.3458	15.844
9. Doubly-Cartesian ensemble of classifiers 5 and 6, $\left(P_Y^{IFN} \times P_Y^{C45}\right) \otimes \left(P_D^{IFN} \times P_D^{C45}\right)$	46.6229	>1 h

The evaluation results of each component classifier are presented in rows 1–4 of table 7. The results suggest that C4.5 is expected to achieve higher payoffs than the IFN classifier. It is seen that the maximum expected payoff can be improved (with respect to the component classifiers) by using Cartesian ensembles as shown in rows 5 and 6 of table 7. Rows 7 and 8 of table 7 show that the Doubly-Cartesian classifier constructed from using C4.5 outperforms the Doubly-Cartesian classifier obtained from using IFN. In addition, each Doubly-Cartesian classifier outperforms any of its constituent classifiers. Finally, the last row of table 7 shows the improved performance of using a complex hierarchical classifier (using both the Cartesian and Doubly-Cartesian operations) over the other classifiers. (Intuitively, repeated composition of a finite repertoire of classifiers eventually leads to a maximal dominating composite classifier.)

6. Summary

Classification systems have been used as a means for extracting valuable decision-making information in various applications ranging from medical diagnosis (Núñez 1991) to industrial production processes (Braha 2001). The recognition that in most machine-learning applications non-uniform misclassification costs are the governing rule has led to a resurgence of interest in cost-sensitive classification. In this paper, we extend the scope of cost-sensitive classification by including the aspect of 'actions'. More specifically, the decision-maker observes classes as predicted by the classification system and chooses actions accordingly. The decision-maker wishes to maximize the expected payoff by choosing an optimal decision rule. The underlying assumption of this paper is that the predictions displayed by a classifier are 'effective' only insofar as the derived information leads to actions that increase the payoff for the decision-maker.

The decision-theoretic framework allows one to define and analyse rigorously the concept of classifier domination — the ability of one classifier to achieve an expected payoff at least as high as another one, regardless of the current payoff values and class distributions. We present a precise condition under which one classifier 'dominates' another.

Besides providing an economic perspective on the value of 'extracted information', the decision-theoretic model hints at the wide range of robust composite classifier architectures that might arise from this point of view. In this paper, two architectures for combining classifiers have been proposed. In both cases, the combining classifiers have been shown to dominate any of the constituent classifiers. An indication of the utility of the proposed framework for complex real-world problems (beyond the purely theoretical aspects) has been demonstrated on the semiconductor manufacturing domain.

This paper suggests that on-line monitoring of the manufacturing process using data mining may be highly effective supporting previous work (Braha and Shmilovici 2002, 2003). It also shows that composite classifier architectures are expected to yield higher performance than the performance of each individual classifier on its own. This is particularly important for semiconductor manufacturing environments where data are scarce and costly, and various physical and chemical parameters that affect the process exhibit highly complex interactions. The component classifiers can use data sets of a relatively small volume, and need only a reasonable amount of time and memory for training and application. Thus, a composite classifier may be highly effective when embedded in real-time monitoring of semiconductor manufacturing processes (also Braha and Shmilovici 2002).

Beyond the expected benefits of the suggested framework, in reality one has to consider additional deployment factors. The large amount of yield data generated during daily semiconductor manufacturing operations makes it almost impractical to analyse the data manually for valuable decision-making information. In semiconductor manufacturing environments, this situation calls for new techniques and tools that can intelligently and (semi-)automatically store and manage large amounts of yield data, which will be turned into high-level and useful knowledge. This knowledge extraction process should be supported by powerful data-acquisition systems such as computers, microprocessors,

transducers, and analogue-to-digital converters for collecting, analysing, and transferring data (Braha and Shmilovici 2002).

In summary, the integration of real-time data-mining methodologies with closed-loop process control and decision analysis may become a critical ingredient of future yield management, which will be integrated, agile, and capable of continuous prevention and improvement.

References

Adams, N.M. and Hand, D.J, Comparing classifiers when the misallocation costs are uncertain. *Patt. Recog.*, 1999, **32**(7), 1139–1147.

Ahituv, N. and Ronen, B., Orthogonal information structures — a model to evaluate the information provided by second opinion. *Dec. Sci.*, 1988, **19**, 255–268.

Ali, K.M. and Pazzani, M.J., *Error Reduction through Learning Multiple Descriptions*. Technical Report No. 95-39, 1996 (Department of Information and Computer Science, University of California, Irvine, CA).

Barad, M. and Braha, D., Control limits for multi-stage manufacturing processes with binomial yield: single and multiple production runs. *J. Oper. Res. Soc.*, 1996, **47**(1), 98–112.

Benediktsson, J.A., Sveinsson, J.A., Swain, P.H. and Ersoy, O.K., Parallel consensual neural networks, in *Proceeding of the 1993 IEEE International Conference on Neural Networks*, 1993 (IEEE Computer Society Press: New York), pp. 27–32.

Bradley, A.P, The use of the area under the roc curve in the evaluation of machine-learning algorithms. *Patt. Recog.*, 1997, **30**(7), 1145–1159.

Braha, D., Manufacturing control of a serial system with binomial yields, multiple production runs, and non-rigid demand: decomposition approach. *IIE Trans.*, 1999, **31**(1), 1–10.

Braha, D. (Ed.)., *Data Mining for Design and Manufacturing: Methods and Applications*, 2001 (Kluwer, Boston, MA).

Braha, D. and Shmilovici, A., Data mining for improving a cleaning process in the semiconductor industry. *IEEE Trans. Semiconduct. Manuf.*, 2002, **15**(1), 91–101.

Braha, D. and Shmilovici, A., On the use of decision tree induction for discovery of interactions in a photolithographic process. *IEEE Trans. Semiconduct. Manuf.*, 2003, **16**(4), 644–652.

Breiman, L., Bagging predictors. *Mach. Learn.*, 1996, **24**, 123–140.

Brodley, C.E., *Recursive Automatic Algorithm Selection for Inductive Learning*. Technical Report No. 96-61, 1994 (Department of Computer Science, University of Massachusetts, Amherst, MA).

Chan, P. and Stolfo, S., Toward scalable learning with non-uniform class and cost distributions, in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, 1998, pp. 164–168.

Clemen, R.T., Combining forecasts: a review and annotated bibliography. *Int. J. Forecast.*, 1989, **5**, 559–583.

Cunningham, S.P., Spanos, C.J. and Voros, K., Semiconductor yield improvement: results and best practices. *IEEE Trans. Semiconduct. Manufact.*, 1995, **8**, 103–109.

Demski, J.S., *Information Analysis*, 1972 (Addison-Wesley: Reading, MA).

Dietterich, T.G. and Bakiri, G., Error-correcting output codes: a general method for improving multiclass inductive learning programs, in *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, 1991 (AAAI Press, Madison, WI), pp. 572–577.

Domingos, P., How to get a free lunch: a simple cost model for machine-learning applications, in *Proceedings of the AAAI-98/ICML-98 Workshop on the Methodology of Applying Machine Learning*, 1998 (AAAI Press, Madison, WI), pp. 1–7.

Domingos, P., MetaCost: a general method for making classifiers cost-sensitive, in *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, 1999 (AAAI Press, Madison, WI), pp. 155–164.

Elkan, C., The Foundations of Cost-Sensitive Learning, in *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 2001 (Morgan Kaufmann).

Elovici, Y. and Braha, D., A decision-theoretic approach to data mining. *IEEE Trans. Sys. Man Cybernet. A: Sys. Humans*, 2003, **33**(1), 42–51.

Freund, Y. and Schapire, R.E., A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Sys. Sci.*, 1997, **55**(1), 119–139.

Friedman, C.P. and Wyatt, J.C., *Evaluation Methods in Medical Informatics*, 1997 (Springer: New York, NY).

Hansen, L.K. and Salamon, P., Neural network ensembles. *IEEE Trans. Patt. Anal. Mach. Intell.*, 1990, **12**, 993–1001.

Ho, T.K., Hull, J.J. and Srihari, S.N., Decision combination in multiple classifier systems. *IEEE Trans. Patt. Anal. Mach. Intell.*, 1994, **16**, 66–75.

Kohavi, R., A study of cross-validation and bootstrap for accuracy estimation and model selection, in: *Proceedings of IJCAI-95*, edited by C.S. Mellish, 1995 (Morgan Kaufmann), pp. 1137–1143.

Kohavi, R. and Li, C.-H., Oblivious decision trees, graphs, and top-down pruning, in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, pp. 1071–1077.

Kohavi, R. and Provost, F., Glossary of terms. *Mach. Learn.*, 1998, **30**(2/3), 271–274.

Last, M. and Kandel, A., Data mining for process and quality control in the semiconductor industry. In *Data Mining for Design and Manufacturing: Methods and Applications*, edited by D. Braha, pp. 207–234, 2001 (Kluwer: Boston, MA).

Last, M., Maimon, O. and Minkov, E., Improving stability of decision trees. *Int. J. Patt. Recog. Artif. Intell.*, 2002, **16**(2), 145–159.

Maimon, O. and Last, M., Knowledge Discovery and Data Mining. *The Info-Fuzzy Network (IFN) Methodology*, 2000 (Kluwer: Boston, MA).

Margineantu, D. and Dietterich, T., Bootstrap methods for the cost-sensitive evaluation of classifiers, in *Machine Learning: Proceedings of the Seventeenth International Conference*, pp. 583–590, 2000 (Morgan Kaufmann).

Marschak, J., Economics of information systems. *J. Am. Stat. Assoc.*, 1971, **66**, 192–219.

Masand, B.M. and Piatetsky-Shapiro, G., A comparison of approaches for maximizing business payoff of prediction payoff. *Proc. Knowledge Disc. Data Mining*, 1996, **2**, 195–201.

McGuire, C.B. and Radner, R., *Decision and Organization*, 1986 (University of Minnesota Press: Minneapolis, MN).

Núñez, M., The use of background knowledge in decision tree induction. *Mach. Learn.*, 1991, **6**, 231–250.

Provost, F. and Fawcett, T., Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions, in *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, 1997 (AAAI Press, Madison, WI), pp. 43–48.

Provost, F. and Fawcett, T., Robust classification for imprecise environments. *Mach. Learn.*, 2001, **42**(3), 203–231.

Provost, F., Fawcett, T. and Kohavi, R., The case against accuracy estimation for comparing classifiers, in *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 445–453, 1998 (Morgan Kaufmann).

Quinlan, J.R., *C4.5: Programs for Machine Learning*, 1993 (Morgan Kaufmann).

Raiffa, H., *Decision Analysis*, 1976 (McGraw Hill: New York, NY).

Schapire, R.E., The strength of weak learnability. *Mach. Learn.*, 1990, **5**, 197–227.

Skalak, D.B., *Prototype Selection for Composite Nearest Neighbor Classifiers*. Technical Report No. 95-74, 1995 (Department of Computer Science, University of Massachusetts, Amherst, MA).

Ting, K.M. and Zheng, Z., Boosting trees for cost-sensitive classifications, in *Proceedings of the 10th European Conference on Machine Learning*, Chemnitz, Germany, 1998, pp. 191–195.

Tobin, K.W., Karnowski, T.P. and Lakhani, F., A survey of semiconductor data management systems technology, in *Proceedings of SPIE's 25th Annual International Symposium on Microlithography*, Santa Clara, CA, USA, February 2000.

Turney, P., Cost-sensitive classification: empirical evaluation of a hybrid genetic decision tree induction algorithm. *J. Artif. Intell. Res.*, 1995, **2**, 369–409.

Wolpert, D., Stacked generalization. *Neur. Network.*, 1992, **5**, 241–259.

Zahavi, J. and Levin, N., Issues and problems in applying neural computing to target marketing. *J. Direct Market.*, 1997, **11**(4), 63–75.

Zhang, X., Mesirov, J.P. and Waltz, D.L., A hybrid system for protein secondary structure prediction. *J. Molec. Biol.*, 1992, **225**, 1049–1063.