J Intell Manuf (2006) 17:285–299
DOI 10.1007/s10845-005-0005-x

Data mining for improving the quality of manufacturing: a feature set decomposition approach

Lior Rokach · Oded Maimon

Received: September 2004 / Accepted: September 2005

Abstract Data mining tools can be very beneficial for discovering interesting and useful patterns in complicated manufacturing processes. These patterns can be used, for example, to improve manufacturing quality. However, data accumulated in manufacturing plants have unique characteristics, such as unbalanced distribution of the target attribute, and a small training set relative to the number of input features. Thus, conventional methods are inaccurate in quality improvement cases. Recent research shows, however, that a decomposition tactic may be appropriate here and this paper presents a new feature set decomposition methodology that is capable of dealing with the data characteristics associated with quality improvement. In order to examine the idea, a new algorithm called (Breadth-Oblivious-Wrapper) BOW has been developed. This algorithm performs a breadth first search while using a new F-measure splitting criterion for multiple oblivious trees. The new algorithm was tested on various real-world manufacturing datasets, specifically the food processing industry and integrated circuit fabrication. The obtained results have been compared to other methods, indicating the superiority of the proposed methodology.

Keywords Data mining · Quality engineering · Feature set-decomposition · Splitting criterion · F-measure

L. Rokach (✉)
Department of Information System Engineering,
Ben-Gurion University of the Negev, Israel
e-mail: liorrk@bgu.ac.il

O. Maimon
Department of Industrial Engineering, Tel-Aviv University,
Ramat Aviv, Tel Aviv 69978, Israel
e-mail: maimon@eng.tau.ac.il

Introduction

Data mining is a collection of tools that explore data in order to discover previously unknown patterns. The accessibility and abundance of information today makes data mining a matter of considerable importance and necessity.

One of the most practical techniques used in data mining is classification. The aim of classification is to build a classifier (also known as a classification model) by induction from a set of pre-classified instances. The classifier can be then used for classifying unlabelled instances. Given the long history and recent growth of the field, it is not surprising that several mature approaches to induction are now available to the practitioner. Decision tree induction is one of the most widely used approaches in data mining and machine learning for classification problems (see, for instance Quinlan, 1993). Decision Trees are considered to be self-explained models and easy to follow when compacted.

In many modern manufacturing plants, data that characterize the manufacturing process are electronically collected and stored in the organization's databases. Thus, data mining tools can be used for automatically discovering interesting and useful patterns in the manufacturing processes. These patterns can be subsequently exploited to enhance the whole manufacturing process in such areas as defect prevention and detection, reducing flow-time, increasing safety, etc. The literature presents several studies that examine the implementation of data mining tools in manufacturing (Fountain, Dietterich & Sudyka 2000; Gardner & Bieker 2000; Kusiak, 2001; Kusiak & Kurasek, 2001; Last & Kandel, 2001).

This paper focuses on mining quality-related data in manufacturing. Quality can be measured in many different ways. Usually the quality of batches of products is measured and not that of a single product. The quality measure can either have nominal values (such as "Passed"/"Not Passed")

or continuously numeric values (Such as the number of good chips obtained from silicon wafer or the pH level in a cream cheese). Even if the measure is numeric, it can still be reduced to a sufficiently discrete set of interesting ranges. In the cases that we examined, the goal was to find the relation between the quality measure (target attribute) and the input attributes (the manufacturing process data).

Classification methods can be used to improve the learning curve both in the learning rate, as well as in the target quality that is reached at the mature stage. The idea is to find a classifier that is capable of predicting the quality measure of a certain product or batch, based on its manufacturing parameters. Subsequently, the classifier can be used to set up the most appropriate parameters or to identify the reasons for the defects.

The distinction of data mining for quality improvement should be clarified. As opposed to classical methods such as design of experiment, data mining is considered as a "secondary data analysis of large databases" (Hand, 1998). The term "secondary" emphasizes the fact that the primary purpose of the database was not data analysis. That is to say, there is no control whatsoever on the data collected. Other classical methods, such as control charts, aim to monitor the process and not to infer the relationship between the target attribute and the input attributes.

The manufacturing parameters obviously include the characteristics of the production line (such as which machine has been used in each step, how each machine has been setup, etc.) and other parameters (if available) relating to the raw material that is used in the process; the environment (moistness, temperature, etc); the human resources that operate the production line (the experience level of the worker, which have been assigned on each machine in the line, the shift number) and other such significant factors.

Since the data accumulated in manufacturing plants has unique characteristics, conventional data mining methods are ineffective. More specifically, the following properties are considered problematic:

1. Imbalanced Distribution: The quality measure (target attribute) has imbalanced distribution. This happens as most of the batches pass the quality assurance examinations and only a few are considered invalid. On the other hand, the quality engineer is more interested in identifying the invalid cases (the less frequent class).
   Traditionally, the objective of the classification method is to minimize the misclassification rate, i.e. to maximize accuracy. However, for the unbalanced class distribution, accuracy is not an appropriate metric. A classifier working on a population where one class ("not passed QA") represents only 1% of the examples can achieve a significantly high accuracy of 99% by just predicting all the examples to be of the prevalent class ("passed QA"). Thus, the goal is to identify as many examples of the "not passed QA" class as possible (high recall) with as little false alarms (high precision). Traditional methods fail to obtain high values of recall and precision for the less frequent classes, as they are oriented toward finding global high accuracy (Joshi, 2002).
2. "Curse of Dimensionality": Usually in manufacturing plants there are many input attributes that may affect quality and the required number of labelled samples for supervised classification increases as a function of dimensionality (Jimenez & Landgrebe, 1998). The required number of training samples is linearly related to the dimensionality for a linear classifier and to the square of the dimensionality for a quadratic classifier.
   In terms of nonparametric classifiers, such as decision trees, the situation is even more severe. It has been estimated that as the number of dimensions increases, the sample size needs to increase exponentially in order to have an effective estimate of multivariate densities (Hwang, Lay, & Lippmann, 1994). In quality engineering mining problems, we would like to understand the quality patterns as soon as possible in order improve the learning curve. Thus, the training set is usually too small relative to the number of input features. Bellman (1961), working on complicated signal processing, was the first to define this phenomenon as the "curse of dimensionality". Techniques like decision trees that are efficient in low dimensions fail to provide meaningful results, when the number of dimensions increases beyond a "modest" size. Furthermore, humans generally more easily understand smaller classifiers, involving less features (probably less than 10). Smaller classifiers are also more appropriate for user-driven data mining techniques such as visualization.
3. "Mixed-Type": Usually the input attributes in manufacturing data are of mixed type, namely some of the attributes are numeric (such as temperature or duration) while other are categorical (such as the machine's model used in the process). A suitable solution should be capable of addressing both types in the same model.

There have been several attempts to address the class imbalance distribution problem. Most of these attempts are considered to be "external", namely the internal inducer (such as the decision tree algorithm) is not changed. Japkowicz and Stephen (2002) divide these "external" attempts into three categories:

1. "Re-sampling" methods in which the under-represented class gets over- sampled so as to match the size of the other class (Chawla, Bowyer, Hall, & Kegelmeyer, 2002; Estabrooks, Jo, & Japkowicz, 2004; Nickerson, Japokowicz, & Milios, 2001; Weiss & Provost, 2003);

2. "Downsizing" methods in which the over-represented class is downsized to match the size of the under-represented class (Kubat & Matwin, 1997);
3. "Learning by recognition" methods in which one of the two classes is ignored. With these methods, recognition-based, rather than discrimination-based schemas are used (Nugroho, Kuroyanagi, & Iwata, 2002).

Most methods for dealing with high dimensionality focus on feature selection techniques, i.e. selecting a single subset of features upon which the inducer (induction algorithm) will run, while ignoring the rest. The selection of the subset can be done manually using prior knowledge to identify irrelevant variables or feature selection algorithms. In the last decade, many researchers have shown increased interest in feature selection. Consequently many feature selection algorithms have been proposed, with some demonstrating remarkable improvements in accuracy. Since the subject is too wide to survey here, the reader is referred to Liu and Motoda (1998) for further reading.

Despite the popularity of feature selection methodologies, there are several drawbacks in using them in overcoming the "Curse of Dimensionality":

- The assumption that a large set of input features can be reduced to a small subset of relevant features is not always true; in some cases the target feature is actually affected by most of the input features and removing features will cause a significant loss of important information.
- The outcome (i.e. the subset) of many algorithms for feature selection (for example almost any of the algorithms that are based upon the wrapper methodology) is strongly dependent on the training set size. That is, if the training set is small, the size of the reduced subset will be small also. Consequently, relevant features might be lost. Accordingly, the induced classifiers might achieve lower accuracy compared to classifiers that have access to all relevant features.
- In some cases, even after eliminating a set of irrelevant features, the researcher is left with a relatively large number of relevant features.
- The backward elimination strategy, used by some methods, is extremely inefficient for working with large-scale databases, where the number of original features is more than 100.

A number of linear dimension reducers have been developed over the years including projection pursuit (Friedman & Tukey, 1973); factor analysis (Kim & Mueller, 1978); and principal components analysis (Dunteman, 1989). These methods are not aimed directly at eliminating irrelevant and redundant features, but are rather concerned with transforming the observed variables into a small number of "projections" or "dimensions". The underlying assumption in these methods is that the variables are numeric and the dimensions can be expressed as linear combinations of the observed variables (and vice-versa). Each discovered dimension is assumed to represent an unobserved factor and thus provides a new way of understanding the data (similar to the curve equation in the regression models).

The linear dimension reducers are enhanced constructive induction systems that use a set of existing features and a set of predefined constructive operators to derive new features (Pfahringer, 1994). These methods are effective for high dimensionality applications only if the original domain size of the input feature can be in fact decreased dramatically. There are several induction methods (such as support vector machines and neural networks) that deal directly with high-dimensional data. However, the resulting classifiers are usually incomprehensible to end-users.

Several researchers have shown that the decomposition methodology can be appropriate for mining manufacturing data (Kusiak, 2000; Maimon & Rokach, 2001). This paper focuses on feature set decomposition for generalizing the task of feature selection. Feature selection aims to provide a single representative set of features from which a classifier is constructed. On the other hand, feature set decomposition decomposes the original set of features into several subsets, and builds a classifier for each subset. Thus, a set of classifiers are trained such that each classifier employs a different subset of the original features set. Subsequently, an unlabelled instance is classified by combining the classifications of all classifiers. This method potentially facilitates the creation of a classifier for high dimensionality data sets without the above mentioned drawbacks of feature selection.

Although the feature set decomposition methodology can be useful in the problem discussed here, there is no literature that seeks for the best feature set decomposition structure. Moreover, there is no feature set decomposition algorithm to explicitly cope with the imbalanced distribution property. The main contribution of this paper is a novel algorithm that automatically seeks for the best mutually exclusive feature set decomposition by employing a new F-measure splitting criterion suited for oblivious decision trees. The superiority of the suggested algorithm over other methods is illustrated on various real-world manufacturing datasets.

Problem formulation

In a typical classification problem, a training set of labelled examples is given and the goal is to form a description that can be used to predict previously unseen examples. The training set can be described in a variety of languages, most frequently, as a collection of records that may contain duplicates. Each record is described by a vector of attribute values.

The notation A denotes the set of input attributes containing n attributes: $A = \{a_1, \ldots, a_i, \ldots, a_n\}$-and y-represents the class variable or the target attribute. Attributes (sometimes referred to as fields, variables or features) are typically one of two types: categorical (values are members of a given set), or numeric (values are real numbers). When the attribute a_i is categorical, it is useful to denote its domain values by $\mathrm{dom}(a_i) = \{v_{i,1}, v_{i,2}, \ldots, v_{i,|\mathrm{dom}(a_i)|}\}$, where $|\mathrm{dom}(a_i)|$ stands for its finite cardinality. In a similar way, $\mathrm{dom}(y) = \{c_1, \ldots, c_{|\mathrm{dom}(y)|}\}$, represents the domain of the target attribute. Numeric attributes have infinite cardinalities.

The instance space (the set of all possible examples) is defined as a Cartesian product of all the input attributes domains: $\mathrm{X} = \mathrm{dom}(a_1) \times \mathrm{dom}(a_2) \times \cdots \times \mathrm{dom}(a_n)$. The Universal Instance Space (or the Labelled Instance Space) U is defined as a Cartesian product of all input attribute domains and the target attribute domain, i.e.: $U = X \times \mathrm{dom}(y)$.

The training set consists of a set of m records and is denoted as $S = (\langle \mathbf{x}_1, y_1 \rangle, \ldots, \langle \mathbf{x}_m, y_m \rangle)$, where $\mathbf{x}_q \in X$ and $y_q \in \mathrm{dom}(y)$.

Usually, it is assumed that the training set records are generated randomly and independently according to some fixed and unknown joint probability distribution D over U. Note that this is a generalization of the deterministic case, when a supervisor classifies a record using a function $y = f(\mathbf{x})$.

Consider a set of examples labeled positive and negative, and a classifier predicting the label for each example (the choice as to which class is called positive is usually arbitrary. In this case the "not passed QA" class will be considered as positive). A positive (negative) example that is correctly classified by the classifier is called a true positive (true negative); a positive (negative) example that is incorrectly classified is called a false negative (false positive). These numbers can be organized in a confusion matrix shown in Table 1.

The accuracy measure that is usually employed for evaluating the performance of classifiers is defined as:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}} \quad (1)$$

However, when there are many negative examples (many examples that "passed QA") it is useful to measure the classification performance by ignoring the correctly predicted negative data. Well-known performance measures in this case

Table 1 Confusion matrix for binary classification problem

Actual \ Classification	Classified as positive	Classified as negative
Actually positive	True positive	False negative
Actually negative	False positive	True negative

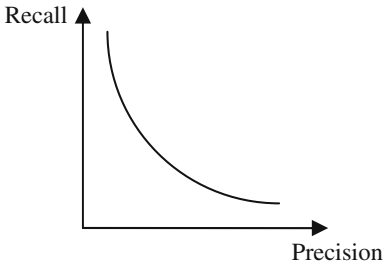


Fig. 1 A Typical precision-recall diagram

are precision (P) and recall (R). Precision measures how many examples classified as "not passed QA" class are indeed "not passed QA". Recall measures how many examples of "not passed QA" class are correctly classified. Mathematically these measures are defined as:

$$P = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}, \quad (2)$$

$$R = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}. \quad (3)$$

The notion of "precision" and "recall" are widely used in information retrieval (Van Rijsbergen, 1979) and data mining (Weiss & Zhang, 2003). Statistics uses complementary measures known as "type-I error" and "type-II error".

Usually there is a tradeoff between the precision and the recall. Trying to improve one measure often results in a deterioration of the second measure. Figure 1 shows a typical precision-recall graph. This two-dimensional graph is closely related to the well-known receiver operating characteristics (ROC) graphs in which the true positive rate (recall) is plotted on the Y-axis and the false positive rate is plotted on the X-axis (Ferri, Flach, & Hernández-Orallo, 2002). However unlike the precision-recall graph, the ROC diagram is always convex.

Given a probabilistic classifier, this trade-off graph may be obtained by setting different threshold values. In a binary classification problem, the classifier prefers the class "not pass" over the class "pass" if the probability for "not pass" is at least 0.5. However, by setting a different threshold value other than 0.5, the trade-off graph can be obtained.

The problem described here is in fact a multi-criteria decision-making (MCDM). The simplest and the most commonly used method to solve MCDM is the weighted sum model. This technique combines the criteria into a single value by using appropriate weighting. The basic principle behind this technique is the additive utility assumption. The criteria measures must be numerical, comparable and expressed in the same unit. Nevertheless, in the case discussed here, the arithmetic mean can mislead. Instead, the harmonic mean provides a better notion of "average". More specifically, this measure is defined as (Van Rijsbergen, 1979):

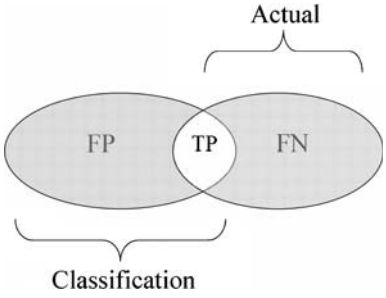


Fig. 2 A graphic explanation of the F-measure

$$F = \frac{2 \cdot P \cdot R}{P + R}. \tag{4}$$

The intuition behind the F-measure can be explained using Fig. 2. Figure 2 presents a diagram of a common situation in which the right ellipsoid represents the set of all defective batches and the left ellipsoid represents the set of all batches that were classified as defective by a certain classifier. The intersection of these sets represents the true positive (TP), while the remaining parts represent false negative (FN) and false positive (FP). An intuitive way of measuring the adequacy of a certain classifier is to measure to what extent the two sets match, namely to measure the size of the unshaded area. Since the absolute size is not meaningful, it should be normalized by calculating the proportional area. This value is in fact the F-measure:

$$\begin{aligned}&\text{Proportion of unshaded area}\\ &= \frac{2 \cdot (\text{True Positive})}{\text{False Positive} + \text{False Negative} + 2 \cdot (\text{True Positve})}\\ &= F.\end{aligned} \tag{5}$$

The F-measure can have values between 0 to 1. It obtains its highest value, when the two sets presented in Fig. 2 are identical and it obtains its lowest value when the two sets are mutually exclusive. Note that each point on the precision-recall curve may have a different F-measure. Furthermore, different classifiers have different precision-recall graphs.

The problem of decomposing the input feature set can be formally phrased as follows:

Given a training set S with input feature set $A = \{a_1, a_2, \ldots, a_n\}$, *and a target feature y from an unknown distribution D over the labelled instance space, the goal is to maximize the value of the F-measure by combining a set of* ω ($\omega \geq 1$) *classifiers, such that the feature subsets used by the classifiers are mutually exclusive.*

It should be noted that the optimal is not necessarily unique. Furthermore it is not obligatory that all input features will actually belong to one of the subsets.

This paper focuses on decision trees classifiers, which are combined using the Naive Bayes combination (Duda & Hart, 1973). The Naive Bayes combination assumes that each features subset used by a certain classifiers is independent of the rest given the value of the target attribute. Consequently an unlabeled instance $\boldsymbol{x}$ is assigned to class c_i with probability of:

$$\widehat{P}(c_i|x) = \alpha \cdot \hat{P}(c_i) \cdot \prod_{k=1}^{\omega} \frac{\hat{P}(c_i|I_k, x)}{\hat{P}(c_i)}, \tag{6}$$

where:

- $\hat{P}(c_i)$ is the a priori probability to be assigned to class c_i. $\hat{P}(c_i)$can be easily estimated by counting the frequency with which the target value c_i occurs in the training set.
- $\hat{P}(c_i|I_k, \boldsymbol{x})$ is the probability of class c_i given an instance $\boldsymbol{x}$ and a particular classifier I_k. In case of decision trees, it can be estimated by using the appropriate frequencies in the relevant leaf. However, using the frequency as is, will typically over-estimate the probability. In order to avoid this phenomenon, it is useful to perform the Laplace's correction (Niblett, 1987).
- α is a normalization factor that ensures that the conditional probabilities of all possible class labels sums up to 1. (In practice we do not need to explicitly evaluate this factor because it is constant for a given instance $\boldsymbol{x}$).

The BOW algorithm

Overview

In order to solve the problem defined in Section "Problem formulatin", we suggest using a hill-climbing, search procedure—breadth-oblivious-wrapper (BOW). This heuristic algorithm begins with a single empty subset. Each iteration of the algorithm considers changing the current decomposition structure by adding an unused feature to one of the existing subsets or to a newly created subset.

Classifier representation

Each subset is represented by an oblivious decision tree (ODT) in which all nodes at the same level test the same feature (Last, Maimon, & Minkov, 2002). Figure 3 demonstrates a typical ODT with three input features: the slicing machine model used in the manufacturing process; the rotation speed of the slicing machine and the shift (i.e. when the item was manufactured); and the Boolean target attribute representing whether that item passed the quality assurance test. The arcs that connect the hidden terminal nodes and the nodes of the target layer are labeled with the number of records that fit this path. For instance, there are twelve items in the training set, which were produced using the old slicing machine that was setup to rotate at a speed greater than 1000 rpm and that were classified as "good" items (i.e. passed the QA test).

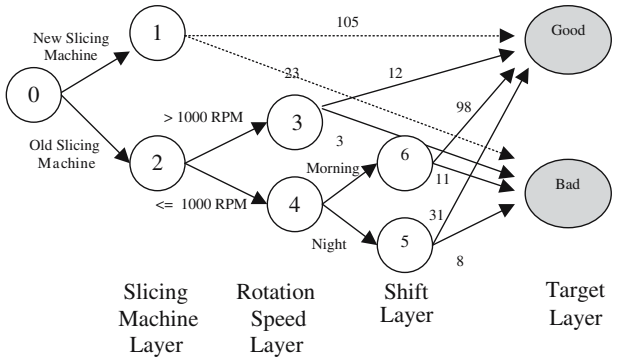


Fig. 3 Oblivious decision tree for quality assurance

Search space

The algorithm iteratively searches for the best decomposition structure. Moving forward to the next iteration is performed in two phases. In the first phase, the algorithm enumerates overall attributes. For each attribute, it checks the feasibility of adding it to one of the existing subsets or to a newly created a subset. Adding an attribute to an existing subset is performed by adding a new layer in the suitable ODT and connecting it to the nodes of the last layer. The nodes of a new layer are defined as the Cartesian product combinations of the previous layer's nodes with the values of the new added feature. In order to avoid unnecessary splitting, the algorithm splits a node only if it increases a certain performance measure. If no node has been split, then this attribute should not be added to that subset.

In the second phase the algorithm compares the performance of all feasible changes. For each change, it evaluates the global performance measure obtained by the entire decomposition structure. That is to say, it compares the contributing effect of each change on the Naïve Bayes combination of all ODTs. The change with the best performance measure is selected for the next iteration.

In order not to be trapped in a local optimum, the algorithm may select the best change and continue to the next iteration, even if the selected addition does not improve the global performance or even worsens it. The output of the algorithm is the structure with the highest performance found during the process and not necessarily the last structure. The search stops when there is no feature left. Nevertheless, if the computational resources are limited, then the search can be stopped when no improvement is achieved.

Figure 4 shows the search space of a 3-feature set decomposition. Note that each layer in this graph stands for a different iteration and that in each run, only one path starting from the root is examined.

The F-measure splitting criterion

As mentioned in the previouis section, the proposed algorithm begins with a single-node decision tree representing an empty set of input attributes. A node is split if it provides an increase in the F-measure. A new input attribute is selected to maximize the total increase of the F-measure { XE "Entropy"} as a result of splitting the nodes of the last layer. The nodes of the new layer are defined as a Cartesian product of split nodes of the previous layer and values of the new input attribute.

Based on Eq. 4 the maximum F-measure obtained by splitting the dataset S according the categorical values of the attribute a_i for any threshold value is:

$$\kappa(i, S, c_{\text{ref}}) = \max_t F\_\text{Measure}(i, S, c_{\text{ref}}, t) = \max_t \frac{2 \cdot P(i, S, c_{\text{ref}}, t) \cdot R(i, S, c_{\text{ref}}, t)}{P(i, S, c_{\text{ref}}, t) + R(i, S, c_{\text{ref}}, t)}, \quad (7)$$

where:

- S is the dataset to be split (the set of records belonging to the discussed node).
- i is the index of the input attribute according to which the split is performed.
- C_{ref} is the referred class. In the context of this paper it is the "Not Passed QA" class.
- $P(i, S, c_{\text{ref}}, t)$ and $R(i, S, c_{\text{ref}}, t)$ are the precision and recall obtained by splitting the training set according to values of $\text{dom}(a_i)$. These values may be calculated according to the following equations:

$$P(i, S, c_{\text{ref}}, t) = \frac{\sum_{v_{i,j} \in \text{dom}(a_i)} m_{c_{\text{ref}},i,j} \cdot \Gamma(i, j, S, c_{\text{ref}}, t)}{\sum_{v_{i,j} \in \text{dom}(a_i)} m_{.,i,j} \cdot \Gamma(i, j, S, c_{\text{ref}}, t)}, \quad (8)$$

$$R(i, S, c_{\text{ref}}, t) = \frac{\sum_{v_{i,j} \in \text{dom}(a_i)} m_{c_{\text{ref}},i,j} \cdot \Gamma(i, j, S, c_{\text{ref}}, t)}{m_{c_{\text{ref}}}}, \quad (9)$$

$$\Gamma(i, j, S, c, t) = \begin{cases} 1 & \text{if } m_{c,i,j} \geq t \cdot m_{.,i,j} \\ 0 & \text{otherwise,} \end{cases} \quad (10)$$

where:

- $m_{c_{\text{ref}}}$ is the number of records in S satisfying $y = c_{\text{ref}}$,
- $m_{.,i,j}$ is the number of records in S satisfying $a_i = v_{i,j}$,
- $m_{c_{\text{ref}},i,j}$ is the number of records in S satisfying $y = c_{\text{ref}}$ and $a_i = v_{i,j}$, and
- t is the threshold for favoring a certain class.

The denominator in Eq. 8 represents the number of records in S that were classified as

C_{ref} assuming threshold t and that S was split according to attribute a_i. The nominators in Eqs. 8 and 9 represent the number of records in S that belong to class C_{ref} and that were

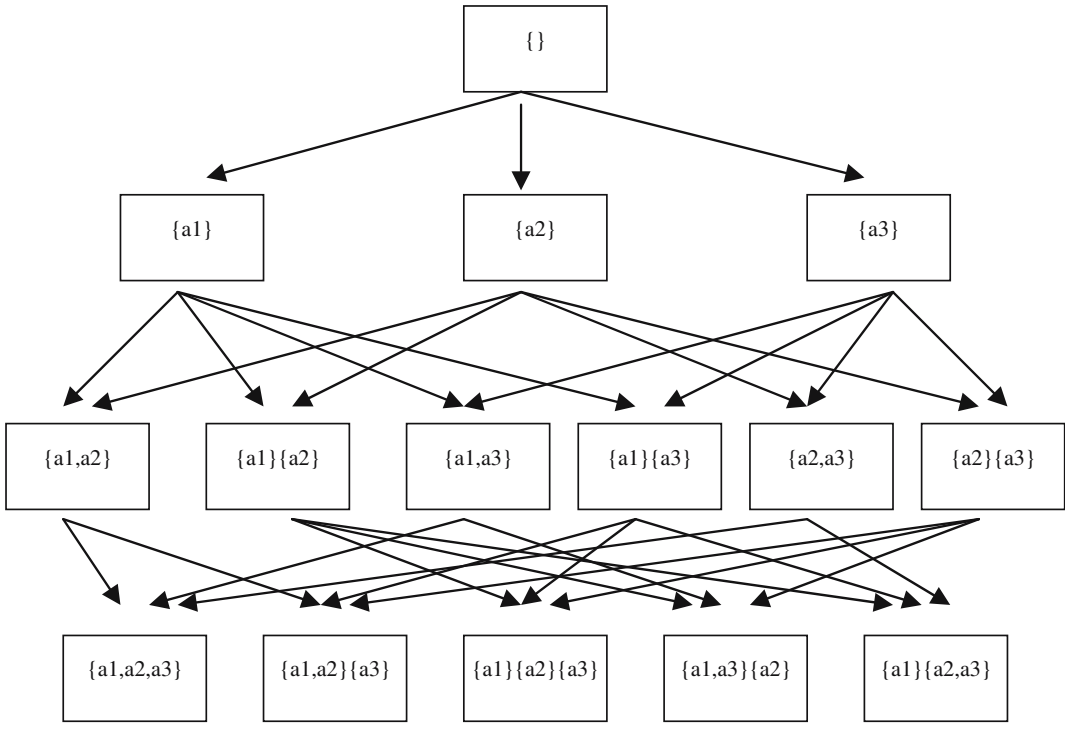


Fig. 4 Breadth first search for a 3-feature set decomposition

Table 2 Illustrative dataset with two categorical input attributes and a binary target attribute

a_1 Slicing machine	a_2 Shift	y Quality measure
New	Morning	Good
New	Night	Bad
Old	Night	Bad
New	Night	Good
Old	Night	Bad
Old	Morning	Good
New	Morning	Bad
Old	Night	Good
Old	Morning	Bad
New	Night	Good

classified correctly assuming threshold t and that S was split according to attribute a_i

An important issue arises when one of the above equations is not defined (the denominator is equal to zero). If the recall is not defined, then it means that there are no records that belong to the referred class. In other words, there is no use in splitting the node. If the precision is not defined (the threshold is too high), we simply set the F-measure to zero and by that the algorithm ignores this point.

Table 2 presents a small dataset S having two categorical input attributes: a_1 (slicing machine) and a_2 (shift) and a binary target attribute which represents the quality measure. Note that in this case dom$(a_1) = \{$"New","Old"$\}$, dom$(a_2) = \{$"Morning", "Night"$\}$, dom$(y) = \{$"Good", "Bad"$\}$. Figures 5(a) and 5(b) present the tree obtained by splitting S according to a_1 (slicing machine) and a_2 (shift) respectively.

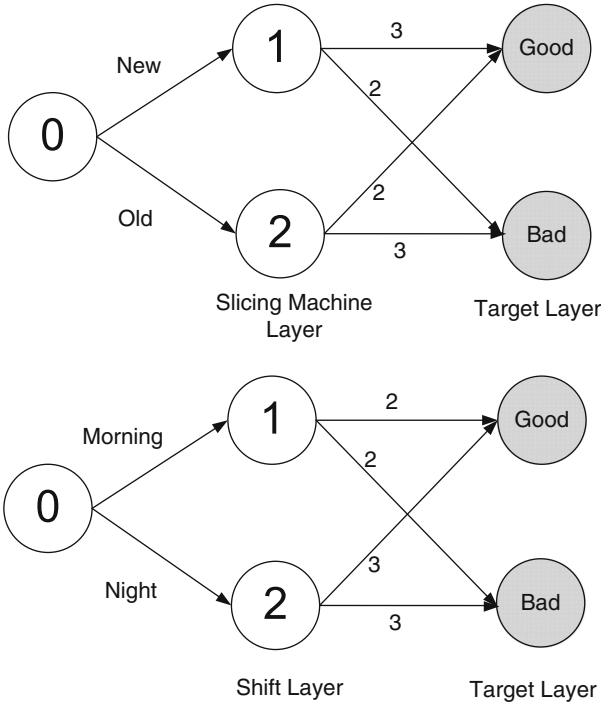


Fig. 5 Tree obtained by splitting dataset in Table 2 according to Slicing Machine and Shift

Assuming that $C_{\text{ref}} =$ "Bad" and the threshold $t = 0.5$, then the following values can be obtained for attribute a_1 (slicing machine):

$$P(1, S, \text{"Bad"}, 0.5) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.5)}{\sum_{j=1}^{2} m_{.,1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.5)} = \frac{2 \cdot 0 + 3 \cdot 1}{5 \cdot 0 + 5 \cdot 1} = 0.6,$$

$$R(1, S, \text{"Bad"}, 0.5) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.5)}{m_{\text{"Bad"}}} = \frac{2 \cdot 0 + 3 \cdot 1}{5} = 0.6,$$

$$F\_\text{Measure}(1, S, \text{"Bad"}, 0.5) = \frac{2 \cdot P(1, S, \text{"Bad"}, 0.5) \cdot R(1, S, \text{"Bad"}, 0.5)}{P(1, S, \text{"Bad"}, 0.5) + R(1, S, \text{"Bad"}, 0.5)} = \frac{2 \cdot 0.6 \cdot 0.6}{0.6 + 0.6} = 0.6.$$

Assuming that $C_{\text{ref}} =$ "Bad" and the threshold t=0.3 then the following values can be obtained for attribute a_1 (slicing machine):

$$P(1, S, \text{"Bad"}, 0.3) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.3)}{\sum_{j=1}^{2} m_{.,1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.3)} = \frac{2 \cdot 1 + 3 \cdot 1}{5 \cdot 1 + 5 \cdot 1} = 0.5,$$

$$R(1, S, \text{"Bad"}, 0.3) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.3)}{m_{\text{"Bad"}}} = \frac{2 \cdot 1 + 3 \cdot 1}{5} = 1,$$

$$F\_\text{Measure}(1, S, \text{"Bad"}, 0.3) = \frac{2 \cdot P(1, S, \text{"Bad"}, 0.3) \cdot R(1, S, \text{"Bad"}, 0.3)}{P(1, S, \text{"Bad"}, 0.3) + R(1, S, \text{"Bad"}, 0.3)} = \frac{2 \cdot 0.5 \cdot 1}{0.5 + 1} = \frac{2}{3}.$$

Assuming that C_{ref}="Bad" and the threshold t=0.1 then the following values can be obtained for attribute a_2 (shift):

$$P(2, S, \text{"Bad"}, 0.1) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},2,j} \cdot \Gamma(2, j, S, \text{"Bad"}, 0.1)}{\sum_{j=1}^{2} m_{.,2,j} \cdot \Gamma(2, j, S, \text{"Bad"}, 0.1)} = \frac{2 \cdot 1 + 3 \cdot 1}{5 \cdot 1 + 5 \cdot 1} = 0.5,$$

$$R(1, S, \text{"Bad"}, 0.1) = \frac{\sum_{j=1}^{2} m_{\text{"Bad"},1,j} \cdot \Gamma(1, j, S, \text{"Bad"}, 0.1)}{m_{\text{"Bad"}}} = \frac{2 \cdot 1 + 3 \cdot 1}{5} = 1,$$

$$F\_\text{Measure}(2, S, \text{"Bad"}, 0.1) = \frac{2 \cdot P(2, S, \text{"Bad"}, 0.1) \cdot R(2, S, \text{"Bad"}, 0.2)}{P(2, S, \text{"Bad"}, 0.1) + R(2, S, \text{"Bad"}, 0.1)} = \frac{2 \cdot 0.5 \cdot 1}{0.5 + 1} = \frac{2}{3}.$$

Actually there is no need to enumerate all possible thresholds. Each splitting branch (a certain value of the candidate input attribute) defines another possible threshold value. By sorting the thresholds, one can get the desired precision-recall diagram. For instance, if the dataset is divided according to a_1, then the threshold values that must be checked are: 0.6 = 3/5 and 0.4 = 2/5. In this case:

$$F\_\text{Measure}(1, S, \text{"Bad"}, 0.4) = \frac{2}{3},$$

$$F\_\text{Measure}(1, S, \text{"Bad"}, 0.6) = 0.6.$$

Resulting:

$$\kappa(1, S, \text{"Bad"}) = \max_t F\_\text{Measure}(1, S, \text{"Bad"}, t) = \max(0.6, \frac{2}{3}) = \frac{2}{3}.$$

The default maximum F-measure obtained before splitting is defined as:

$$\kappa^*(S, c_{\text{ref}}) = \frac{2 \cdot P^*(S, c_{\text{ref}})}{P^*(S, c_{\text{ref}}) + 1}, \tag{11}$$

where:

$$P^*(S, c_{\text{ref}}) = \frac{m_{c_{\text{ref}}}}{|S|}. \tag{12}$$

Note that the algorithm considers splitting the dataset S by the attribute a_i only if:

$$\kappa(a_i, S, c_{\text{ref}}) > \kappa^*(S, c_{\text{ref}}). \tag{13}$$

For instance taking the case described in Table 2, the default maximum F-measure is $\kappa^*(S, \text{“Bad”}) = 0.5$. Because in this case $\kappa(1, S, \text{“Bad”}) = 2/3$, then splitting according to a_1 is considered to be a legitimate split. Note that the default maximum F-measure is always equal or less than the maximum F-measure obtained after splitting.

The maximum F-measure presented in Eq. 7 for categorical attributes can be extended for numerical attributes as specified in Eqs. 14–18. The idea is to divide the continuous domain into two sub-domains: $a_i > v$ and $a_i \leq v$. Although finding the optimal value of v can be performed in many ways, it is preferable to use a one-dimensional golden section search method.

$$\kappa(a_i, S, c_{\text{ref}}) = \max_{t,v} \frac{2 \cdot P(a_i, S, c_{\text{ref}}, t, v) \cdot R(a_i, S, c_{\text{ref}}, t, v)}{P(a_i, S, c_{\text{ref}}, t, v) + R(a_i, S, c_{\text{ref}}, t, v)}, \tag{14}$$

where:

$$P(a_i, S, c_{\text{ref}}, t, v) = \frac{\mu_{c_{\text{ref}},i,v} \cdot \Gamma_1(a_i, S, c_{\text{ref}}, v, t) + (m_{c_{\text{ref}}} - \mu_{c_{\text{ref}},i,v}) \cdot \Gamma_2(a_i, S, c_{\text{ref}}, v, t)}{\mu_{.,i,v} \cdot \Gamma_1(a_i, S, c_{\text{ref}}, v, t) + (|S| - \mu_{.,i,v}) \cdot \Gamma_2(a_i, S, c_{\text{ref}}, v, t)}, \tag{15}$$

$$R(a_i, S, c_{\text{ref}}, t, v) = \frac{\mu_{c_{\text{ref}},i,v} \cdot \Gamma_1(a_i, S, c_{\text{ref}}, v, t) + (m_{c_{\text{ref}}} - \mu_{c_{\text{ref}},i,v}) \cdot \Gamma_2(a_i, S, c_{\text{ref}}, v, t)}{m_{c_{\text{ref}}}}, \tag{16}$$

$$\Gamma_1(a_i, S, c, v, t) = \begin{cases} 1 & \text{if } \mu_{c,i,v} \geq t \cdot \mu_{.,i,v} \\ 0 & \text{otherwise,} \end{cases} \tag{17}$$

$$\Gamma_2(a_i, S, c, v, t) = \begin{cases} 1 & \text{if } (m_c - \mu_{c,i,v}) \geq t \cdot (|S| - \mu_{.,i,v}) \\ 0 & \text{otherwise,} \end{cases} \tag{18}$$

where:

- $\mu_{c,i,v}$ is the number of records in S satisfying $y = c$ and $a_i > v$.
- $\mu_{.,i,v}$ is the number of records in S satisfying $a_i > v$.

Eqs. 7–18 assume that there is only a single ODT. However, in case of multiple ODTs as in the case of a feature set decomposition framework, it should be extended in order to evaluate its global effect on the entire constellation. In some cases, splitting a node by a certain attribute has a tremendous effect on the performance of the ODT in which the node is contained. It has no effect, however, on the entire decomposition performance. This situation occurs, for example, when there are redundant attributes. In such cases, two relevant attributes may explain the target attribute in the same way and each attribute can be used for describing the target attribute. However, using both of them in the same ODT or in two different ODTs does not improve performance.

Rewriting Eqs. 7–10, we obtain:

$$\kappa_{\text{Global}}(a_i, S, c_{\text{ref}}, I_1, \ldots, I_{\omega-1}) = \max_t \frac{2 \cdot P_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1}) \cdot R_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1})}{P_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1}) + R_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1})}, \tag{19}$$

where:

$$P_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1}) = \frac{\sum_{v_{i,j} \in \text{dom}(a_i)} \sum_{x \in \{S | a_i = v_{i,j},\, y = c_{\text{ref}}\}} \Gamma_{\text{Global}}(i, S, c_{\text{ref}}, j, t, I, I_1, \ldots, I_{\omega-1}, \boldsymbol{x})}{\sum_{v_{i,j} \in \text{dom}(a_i)} \sum_{x \in \{S | a_i = v_{i,j}\}} \Gamma_{\text{Global}}(i, S, c_{\text{ref}}, j, t, I, I_1, \ldots, I_{\omega-1}, \boldsymbol{x})}, \tag{20}$$

$$R_{\text{Global}}(a_i, S, c_{\text{ref}}, t, I_1, \ldots, I_{\omega-1}) = \frac{\sum_{v_{i,j} \in \text{dom}(a_i)} \sum_{x \in \{S | a_i = v_{i,j},\, y = c_{\text{ref}}\}} \Gamma_{\text{Global}}(i, S, c_{\text{ref}}, j, t, I, I_1, \ldots, I_{\omega-1}, \boldsymbol{x})}{m_{c_{\text{ref}}}}, \tag{21}$$

$$\Gamma_{\text{Global}}(i, S, c, j, t, I_1, \ldots, I_{\omega-1}, x) = \begin{cases} 1 & \text{if } \dfrac{\frac{m_{c,i,j}}{m_{.,i,j}} \cdot \prod_{k=1}^{\omega-1} \frac{\hat{P}(y=c|I_k,x)}{\hat{P}(y=c)}}{\sum_{c_j \in \text{dom}(y)} \frac{m_{c_j,i,j}}{m_{.,i,j}} \cdot \prod_{k=1}^{\omega-1} \frac{\hat{P}(y=c_j|I_k,x)}{\hat{P}(y=c_j)}} > t \\ 0 & \text{otherwise,} \end{cases} \tag{22}$$

where $I_1, \ldots, I_{\omega-1}$ represents the existing ODTs classifier (not including the ODT to which the investigated node belong). Equation 19 represents the maximum F-measure value obtained by splitting S according to a_i assuming that $I_1, \ldots, I_{\omega-1}$ exists.

Unlike classical splitting criteria (such as Information Gain, Gain Ratio, Gini Index, etc), the F-measure criterion does not perform a comparison between the impurity of the parent node with the weighted impurity of the children after splitting. Nevertheless, as in the case of Information Gain and Gini Index, the proposed F-measure never reports a worse performance after trying a split than before splitting. Consequently, this criterion may not be appropriate for deciding when to stop growing the tree.

The F-measure criterion can be also compared to the recently area under curve (AUC) splitting criterion proposed

in Ferri et al. (2002). The AUC criterion is based on the area generated under the ROC graph. The attribute that obtains the higher AUC criterion is selected as the splitting attribute. Note that for computing the area under the curve, it is sufficient to compute the area between two consecutive points, representing two different branches of the candidate attribute.

In order to avoid over-fitting, the F-measure is not evaluated over the training set as is. Instead we apply the wrapper approach (John, Kohavi, & Pfleger, 1994). With this approach, the decomposition structure is evaluated by repeatedly sampling the training set and measuring the F-measure of the inducers obtained for this decomposition on an unused portion of the training. After the best change is selected, the suitable ODT is updated, this time using the entire training set.

Making a classification

In order to classify an unlabelled instance, the following steps should be performed
A. For each ODT:

- Locate the appropriate leaf for the unseen instance.
- Extract the frequency vector (how many instances relate to each possible value of the target feature).
- Transform the frequency vector to a probability vector according to Laplace's law of succession.

B. Combine the probability vectors using the Naive Bayes combination.
C. Select the target value according to what maximizes the Naive Bayes combination.

Experimental study

Overview

To illustrate the potential of the feature set decomposition approach in data mining quality assurance problems and to evaluate the performance of the proposed algorithm, a comparative experiment was conducted on three real-life datasets obtained from two manufacturers with an average yearly income of more than one billion dollars. The first dataset was obtained from a manufacturer of dairy products. The last two datasets were obtained from a wafer manufacturer.

This experiment compares the BOW algorithm to the Naive Bayes and C4.5 (Quinlan, 1993) algorithms. The Naïve Bayes was chosen since it represents a specific point in the search space of the BOW algorithm. The C4.5 algorithm was selected because it is considered as a state-of-the-art decision tree algorithm, which is widely used in many other comparative studies. The following subsections describe each one of the case studies and the results.

Manufacturing Cottage Cheese

Objective

Manufacturing cottage cheese is one of the most complicated processes in producing dairy products. The process, which may take up to 20 h, usually involves many stages. In the first stage, milk is skimmed using a centrifuge. This phase also produces a cream, which will be used later as the "dressing" to be added to the skimmed cheese. Once the skimmed milk is pasteurized, the milk is cooled and then transferred to a cheese vat. Here, lactic acid culture, to initiate a partial acidification process, and a coagulating enzyme are added.

The milk is left for several hours to allow acidity development as well as curd formation. During this time, the pH level in the milk drops to the desired level and a coagulum (or curd) is formed. Longitudinal and vertical stainless still knives cut the curd to create small curd cubes. In the curd cutting, whey separates from the curd cubes.

Prior to the next step, a slow scalding process, by steam injection of the entire double-jacket vat, the cubes are held for a short period of time enabling them to become firmer. During the scalding process, the curd cubes become more grain-like and more whey is released. At the end of this stage, the contents of the vat are transferred to the curd drainer where the curd grains are mechanically separated from the whey. The curd grains are then transferred to the washer-cooler where they are washed from the acid that developed during the acidification process and cooled by the cold water. The cold grains are then further moved to the creamer where fresh cream ("dressing") and salt are added to form the final product—fresh cottage cheese. The cheese is then pumped to the filling machine and the final packages are palletized and transferred to cold storage ready for distribution. During this long process, a few hundred parameters can be measured or adjusted.

As in every dairy product, there is a chance that a specific batch will be found sour when consumed by the customer, prior to the end of the product's shelf-life. During its shelf-life, the product's pH value normally drops. When it reaches a certain value, the consumer reacts to it as a spoiled product; even though there is no any bacteriological problem. For every batch manufactured, the dairy department performs randomly tests for pH as well organoleptic (taste) at the end of the shelf-life. The samples are kept in the laboratory at a temperature of 7 °C, compared to 4 °C, which is the recommended storage temperature in a home refrigerator. The higher laboratory temperature simulates abuse handling of the product along the cooling chain. The product shelf-life is

determined by the dairy department (generally 12–14 days), assuming that the product retains its organoleptic properties to the end of its shelf-life.

The aim of our study of this plant's production processes was to identify batches (at the end of the manufacturing process) with a high probability of becoming sour (at the end of shelf-life) based on the process variables.

Data

The training data set includes 800 records. Each record has 70 input attributes representing various manufacturing variables. Most of the parameters fall into one of the following classes: temperature, duration, raw material quantities, and machines. For example, we have used the following attributes: average cooling temperature, scalding duration, calcium quantity, culture quantity, etc. The target attribute represents the pH value after two weeks. It can have two values: "Tasty" (pH 4.9–5.3) and "Sour" (below pH 4.9).

Results

The BOW obtained four ODTs using totally 24 different attributes. In order to compare the results of BOW to other algorithms, we used the 10-fold cross-validation procedure. According to this procedure, the training set was randomly partitioned into 10 disjoint instance groups. Each instance group was used once in a test set and nine times in a training set. Since the F-measure and the accuracy on the validation instances is a random variable, the confidence interval was estimated by using the normal approximation. Table 3 shows the mean of the F-measure regarding the "Sour" class and the overall accuracy obtained by using 10-fold cross-validation along with their confidence interval.

Table 3 also shows the number of features used by each algorithm. This table indicates that of the relatively low values that the three algorithms obtained for the F-measure, BOW obtained the highest. Although the proposed method does not mean to improve accuracy but rather the F-measure, it is interesting to note that it obtained the highest accuracy as well. This implies that using the F-measure criterion has not negatively affected overall accuracy. Moreover, BOW employed a much greater number of features than a single decision tree. This implies that the proposed decomposition method can address high-dimension problems by letting more relevant input features affect the classification model. The Naïve Bayes classifier uses all available input features including irrelevant features.

Table 3 The accuracy and the F-measure for the cottage-cheese dataset

Criterion	Naïve Bayes	C4.5	BOW
Accuracy	$77.81\% \pm 2.8\%$	$77.52\% \pm 2.8$	$85.92\% \pm 0.5\%$
F-measure	$31.32\% \pm 3.5\%$	$26.91\% \pm 2.6\%$	$39.4\% \pm 2.1\%$
No of features used	70	12 ± 3.1	24 ± 4.8

Figure 6 presents two ODTs obtained for the above problem. In the first ODT there are three layers representing the parameters: CW_WASH_DUR (Cold Water Wash Duration); FINAL_COOLING_TEMP; and AVG_COOLING _TEMP. Each leaf is specified with the most frequent class (Tasty/Sour) and the appropriate probability vector.

Figure 7 presents the precision-recall diagram. This figure indicates that BOW and Naïve Bayes are more efficient than the C4.5, with BOW more efficient than Naïve Bayes in most of the graph. There are a few points in which Naïve Bayes obtains better results. However, the superiority of Naïve Bayes in these points is negligible. By analysing the precision-recall tradeoff graph, one notices that the precision obtained for a recall value of 100% is identical for all methods. This precision value indicates that the dataset is seriously imbalanced.

It is worth mentioning that the theoretical complexity of the BOW algorithm is identical to the complexity of other oblivious decision algorithms (such as IFN). However, because the current implementation of the BOW algorithm continues until all input attributes are used (in order to avoid local optimum), the actual execution time of BOW is slightly longer than the time required to build a single decision tree.

Yield of IC manufacturing

Objective

An integrated circuit (IC) is a miniature electric circuit containing large numbers of electronic devices packaged on a single chip made of semiconductor material. Manufacturing an IC begins with the production of a semiconductor wafer. An area on the wafer containing a single discrete device or IC is called a chip. Depending on the dimensions of the wafer and the dies, several hundred chips are formed on a single wafer.

While the number and variety of process steps may change from manufacturer to manufacturer, fabricating a wafer usually contains more than 100 steps (Van Zant, 1997). The wafer manufacturing process is largely mechanical. Measurements (for instance flatness, surface quality verification, visual inspection) are taken at various stages of the process to identify defects induced by the manufacturing process, to eliminate unsatisfactory wafer materials and to sort the wafers into batches of uniform thickness to increase productivity.

After the wafer is manufactured, integrated circuits are fabricated on its surface with a single wafer bearing several integrated circuits, all produced at the same time. Each lot undergoes hundreds of individual processing steps, in which

Fig. 6 The first two ODT obtained by BOW for the Cottage Cheese Manufacturing

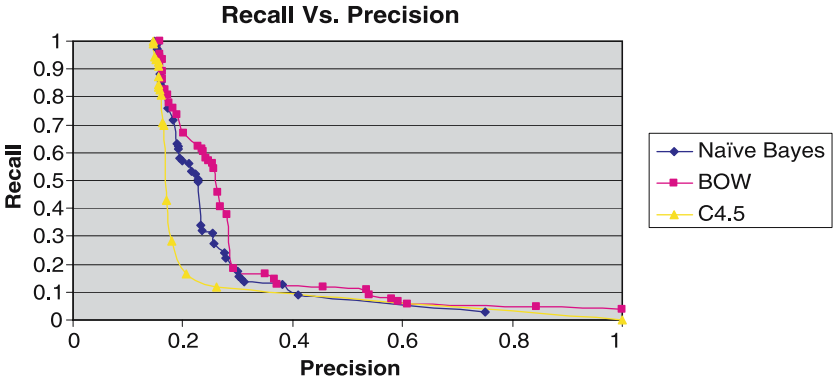
```
ODT # 1:
--------

CW_WASH_DUR <= 286
|   FINAL_COOLING_TEMP <= 5.9
|   |   AVG_COOLING_TEMP <= 10.1: Tasty (0.864,0.136)
|   |   AVG_COOLING_TEMP > 10.1: Sour (0.323,0.674)
|   FINAL_COOLING_TEMP > 5.9
|   |   AVG_COOLING_TEMP <= 12.3: Tasty (0.682,0.318)
|   |   AVG_COOLING_TEMP > 12.3: Sour (0.286,0.714)
CW_WASH_DUR > 286: Tasty (0.906,0.094)


ODT # 2:
--------

POOL_CODE = 501: 0 (124.35/17.0)
POOL_CODE = 502: 0 (121.35/18.0)
POOL_CODE = 503: 0 (120.34/16.0)
POOL_CODE = 504
|   DELAY_AFTER_COOK <= 49.3: 0 (55.18/5.0)
|   DELAY_AFTER_COOK > 49.3
|   |   ACIDIFICATION_DURATION<= 100: 0 (22.06/2.0)
|   |   ACIDIFICATION_DURATION > 100
|   |   |   CUT_END_TEMP <= 27.4: 0 (26.07/6.0)
|   |   |   CUT_END_TEMP > 27.4: 1 (25.06/9.06)
POOL_CODE = 505: 0 (93.27/14.0)
POOL_CODE = 506: 0 (115.33/9.0)
```

Fig. 7 The precision-recall diagram in Cottage Cheese manufacturing



different parts of the ICs are etched in thin layers of material grown or deposited on the working surface of the wafers. Each process step must be tightly controlled to ensure dimensional tolerances. After a high-precision diamond saw cuts the wafers into chips, they are mounted onto packages.

Fabrication of a single lot requires several months. The data are accumulated for each fabrication tool at both the wafer and lot level, using an information system known as "manufacturing execution system". IC manufacturing lines provide many data-mining opportunities. In IC manufacturing, data-mining could have tremendous economic impact, raising profitability by increasing throughput and reducing costs, consequently (Fountain et al., 2000).

For this paper we examined two different datasets obtained from a wafer manufacturer providing design support, manufacturing and turnkey services for integrated ICs on silicon wafers in geometries from 1.0 to 0.18 μm.

The main goal of data mining in IC manufacturing databases is to understand how different parameters in the process affect the line throughput. The throughput of IC manufacturing processes is measured by "yield," which is the number of good products (chips) obtained from a silicon wafer. Since the capability of very expensive microelectronics equipment usually limits the number of wafers processed per time unit, the yield is the most important criterion in determining the effectiveness of an IC process.

Table 4 The accuracy and the F-measure for the yield dataset

Criterion	Naive Bayes	C4.5	BOW
Accuracy	84.28% ± 2.1%	78.85% ± 3.6%	92.86%5.3%
F-measure	64.5% ± 4.9%	17.4% ± 7.2%	82.8% ± 4.3%
No of features used	257	4 ± 1.4	16 ± 7.9

Table 5 The accuracy and the F-measure for the IC-Test dataset

Criterion	Naïve Bayes	C4.5	BOW
Accuracy	92.82% ± 2.5%	89.24% ± 1.9%	96.81% ± 0.6%
F-measure	90.3% ± 3.2%	83.8% ± 1.6%	95.4% ± 0.9%
No of features used	220	9 ± 3.2.	26 ± 2.7

Data

The training dataset includes only 70 records. Each record represents a single wafer and has 257 input attributes labelled $p1, \ldots, p257$ that represent the setting of various parameters used in the manufacturing process of this wafer. The target attribute represents the yield, which the manufacturer's quality engineer has manually divided into two groups: High and Low. More than half of the attributes are numeric. The input attributes specify several machine parameters (for instance, the rotation speed of the slicing machine or the slicing machine model) that may affect the yield. A distinctive value (in case of categorical attributes) or the mean value (in case of numeric values) replace missing values. Due to the high-commercial confidentiality of the process data, we will not explain here the specific meaning of the measured parameters.

Results

Running the BOW on the dataset resulted in five subsets. The average subset size was 3.2 attributes. Table 4 presents the F-measure regarding the "Low" yield and the overall accuracy. The results indicate that the BOW achieved better results in both F-measure and accuracy compared to the C4.5 and Naïve Bayes. Moreover, as in the first case study, the BOW algorithm employed a greater number of features than C4.5. Setting the process parameters according to the classifier obtained by BOW can improve yields by up to 10%.

The IC test

Objectives

The fabricated ICs undergo two series of exhaustive electric tests that measure the operational quality. The first series of tests, which is used for reducing costs by avoiding packaging defective chips, is performed while ICs are still in wafer form. The second series of tests, which is used for quality assurance of the final chip, is carried out immediately after the wafers are cut into chips and mounted onto packages.

The electric tests are performed by feeding various combinations of input signals into the IC. The output signal is measured in each case and compared to the expected behavior. There are wafers that perform well on the first series but fail later in the second series. The goal is to check whether the results of the first series can be further analyzed in order to predict the outcome of the second series. This can be used to reduce the number of wafers that are unnecessarily sliced and packed, eliminating the need for a second series of exhaustive electric tests for most of the devices.

Data

The training data set includes 395 records. Each record has 220 input attributes labeled $p1,\ldots,p220$ representing the electric result values obtained in the first series of tests. Most of the input features represent voltage levels. The target attribute is binary representing "pass" and "not pass" devices according to the functionality of the device in the second testing series. Similar to the yield problem, a distinctive value or the average value replaces missing values depending on the data type.

Results

Running the BOW algorithm on the above data has created seven ODTs containing 26 electronic tests that can be used as indicators for the results of the second series of tests. Table 5 shows the mean of the F-measure regarding the "no pass" class and the overall accuracy obtained by using 10-fold cross-validation along with their confidence interval (with a confidence level of 95%). As in the case of the yield dataset, the BOW algorithm obtained the most encouraging results.

Figure 8 presents the precision-recall diagram for each algorithm. The graph shows that BOW and Naïve Bayes have similar performance and that both are much more efficient than C4.5. The graph also shows that a precision value of 100% can be obtained for a recall value of 67%. In other words, 67% of the defected devices can be identified in the first series of tests without having any false alarm (no high-quality devices will be lost).

Conclusion

Classification problems in quality assurance are characterized by many contributing features relative to the training set

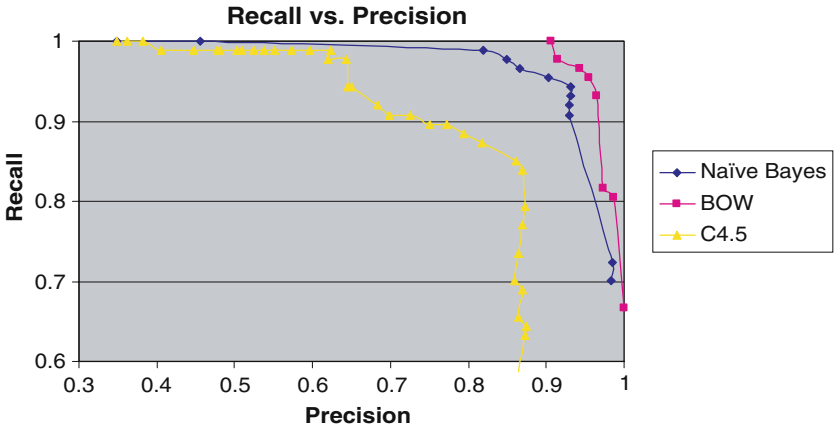


Fig. 8 The precision-recall diagram in IC Test problem

size and the imbalanced distribution of the target attribute. This paper presents a new, mutually exclusive feature set decomposition methodology designed specifically for these circumstances. The basic idea is to decompose the original set of features into several subsets, build a decision tree for each projection, and then combine them.

This paper proposes the BOW algorithm for discovering the appropriate decomposition structure. It was tested with over three real-life datasets. The results show that this framework tends to outperform other comparable methods in the accuracy and the F-measure. The above leads to the conclusion that feature set decomposition can be used for solving classification problems in quality assurance.

One limitation with the suggested algorithm is that it has no backtracking capabilities (for instance, removing a single feature from a subset or removing an entire subset). Furthermore, the search currently begins from an empty decomposition structure, which may be the reason why the number of features in each subset is relatively small. It might be useful to start the search from a better position.

Additional issues to be further studied include: examining how the feature set decomposition concept can be implemented using other inducers like support vector machines and by examining other techniques to combine the generated classifiers (like voting).

References

Bellman, R. (1961), *Adaptive control processes: a guided tour*. NJ: Princeton University Press.

Chawla, N.V., Bowyer, K.W., Hall, L.O., & Kegelmeyer, W.P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Duda, R., & Hart, P. (1973). *Pattern Classification and Scene Analysis*. New-York: Wiley.

Dunteman, G.H. (1989). *Principal Components Analysis*. CA, Beverley Hills: Sage Publications.

Estabrooks, A., Jo, T., & Japkowicz, N. (2004). A multiple resampling method for learning from imbalances data sets. *Computational Intelligence*, *20*(1), 18–36.

Ferri, C., Flach P., & Hernández-Orallo, J. (2002). Learning decision trees using the area under the ROC curve. In C. Sammut & A. Hoffmann, (Eds.), Proceedings of the 19th International Conference on Machine Learning. pp 139–146. CA: Morgan Kaufmann.

Fountain, T. Dietterich T., & Sudyka B. (2000). Mining IC test data to optimize VLSI testing. In J. Simoff & O. Zaiane, (Eds.), Proceedings 6th ACM SIGKDD Conference. Boston: MA, USA. pp 18–25.

Friedman, J.H., & Tukey, J.W. (1973). A Projection pursuit algorithm for exploratory data analysis. *IEEE Transactions on Computers*, *23*(9), 881–889.

Gardner, M., & Bieker, J. (2000). Data mining solves tough semiconductor manufacturing problems. In J. Simoff & O. Zaiane, (Eds.), Proceedings 6th ACM SIGKDD Conference. Boston: MA, USA. pp 376–383.

Hand, D. (1998), Data mining—reaching beyond statistics. *Research in Official Statistics*, *1*(2), 5–17.

Hwang, J., Lay S., & Lippman, A. (1994). Nonparametric multivariate density estimation: A comparative study. *IEEE Transaction on Signal Processing*, *42*(10), 2795–2810.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: a systematic study. *Intelligent Data Analysis Journal*, *6*(5), 429–449.

Jimenez, L.O., & Landgrebe D.A. (1998). Supervised classification in high- dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *IEEE Transaction on Systems Man, and Cybernetics — Part C: Applications and Reviews*, *28*, 39–54.

John, G.H., Kohavi R., & Pfleger, P. (1994). Irrelevant features and the subset selection problem. In W. Cohen, & H. Hirsh, (Eds.), *Proceedings of the Eleventh International Conference In Machine Learning*. New Brunswick: NJ, pp 121–129. USA, CA: Morgan Kaufmann.

Joshi, V.M. (2002). On evaluating performance of classifiers for rare classes. In H. Wang, S.P. Yu, & S. Stolfo (Eds.), *Proceedings Second IEEE International Conference on Data Mining*. IEEE Computer Society Press, pp 641–644. San Jose, California.

Kim, J.O., & Mueller, C.W. (1978). *Factor Analysis: Statistical Methods and Practical Issues*. CA: Sage Publications.

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced data sets: one-Sided sampling. *Proceedings of the Fourteenth International Conference on Machine Learning, Nashville*, TN, USA, pp. 179–186.

Kusiak, A. (2000). Decomposition in data mining: An industrial case study. *IEEE Transactions on Electronics Packaging Manufacturing*, *23*(4), 345–353.

Kusiak, A. (2001). Rough Set Theory: A Data Mining Tool for Semiconductor Manufacturing. *IEEE Transactions on Electronics Packaging Manufacturing*, *24*(1), 44–50.

Kusiak, A., & Kurasek, C. (2001). Data Mining of Printed-Circuit Board Defects. *IEEE Transactions on Robotics and Automation*, *17*(2), 191–196.

Last, M., & Kandel, A. (2001). Data mining for process and quality control in the semiconductor industry. In D. Braha (ed.), *Data Mining for Design and Manufacturing: Methods and Applications*. Dordrecht: Kluwer Academic Publishers, pp 207–234.

Last, M., Maimon, O., & Minkov, E. (2002). Improving stability of decision trees. *International Journal of Pattern Recognition and Artificial Intelligence*, *16*(2), 145–159.

Liu, H., & Motoda, H. (1998). *Feature Selection for Knowledge Discovery and Data Mining*. Dordrecht: Kluwer Academic Publishers.

Maimon, O., & Rokach, L. (2001). Data mining by attribute decomposition with semiconductors manufacturing case study. In D. Braha, (ed.), *Data Mining for Design and Manufacturing: Methods and Applications*. Dordrecht: Kluwer Academic Publishers, pp. 311–336.

Niblett, T. (1987). Constructing decision trees in noisy domains. In Bratko, I., & Lavrac, N. (Eds.), *Proceedings of the Second European Working Session on Learning*. Sigma Press, Wilmslow: England, pp 67–78.

Nickerson, A., Japkowicz, N., & Milios, E. (2001). Using unsupervised learning to guide resampling in imbalanced data sets. *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics*, pp 261–265.

Nugroho, A.S., Kuroyanagi, S., & Iwata, A. (2002). A Solution for Imbalanced Training Sets Problem by CombNET-II and Its Application on Fog Forcasting. *Transactions on Information and Systems, The Institute of Electronics, Information and Communication Engineers*, *85*(7), 1165–1174.

Pfahringer, B. (1994). Controlling constructive induction in CiPF. In F. Bergadano, & L. De Raedt (Eds.), *Proceedings of the seventh European Conference on Machine Learning*. pp 242–256. Springer-Verlag.

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

Van Rijsbergen, C.J. (1979). Information retrieval, butterworth ISBN 0-408-70929-4

Van Zant, P. (1997). *Microchip fabrication: a Practical Guide to semiconductor processing*. New York: McGraw-Hill.

Weiss, G.M., & Provost, F. (2003). Learning when training data are costly: the effect of class distribution on tree induction. *Journal of Artificial Intelligence Research*, *19*, 315–354.

Weiss, G.M., & Zhang, T. (2003). Performance analysis and evaluation. In Y. Nong, (ed.), *The Handbook of Data Mining*. Lawrence Erlbaum Associates Publishers, pp 425–439.