Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación
Master de Ingeniería de Software, Métodos Formales y Sistemas de Información

Master Thesis

An Evaluation of Calibration Methods for Data Mining Models in Simulation Problems

Candidate:
Antonio Bella Sanjuán

Supervisors:
José Hernández Orallo, María José Ramírez Quintana and César Ferri Ramírez

– December 2008 –

This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN under grant TIN 2007-68093-C02-02, and the Spanish project "Agreement Technologies" (Consolider Ingenio CSD2007-00022)

Author's address:

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia
España

Contents

Abstract

Data mining can help us in decision making by using data to obtain accurate models upon which evaluate several options. In general, data mining is used for a specific area in isolation, and models from different areas of a business/organisation are not usually related, so optimal local decisions are good. The problem is when we have to make several decision and they are interwoven with each other. In this case, in general, the best local decisions do not make the best global result. Therefore, we propose using simulation to merge the local data mining models and to obtain a good global result. In this work, we apply simulation to face a customer relationship management (CRM) problem, i.e., a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers. Usually, the decisions make by the data mining models are accompanied with probabilities. If we merge these local probabilistic models, we will need accurate probabilities. If we have not had realistic probabilities, it probably will not affect to the local model, but it will be a disaster when the local models were combined. To assure accurate probabilities, we calibrate the probabilities of the models. So, in order to improve model combination, we also analyse the main calibration techniques and propose a new one inspired in binning-based methods in which the calibrated probabilities are obtained from k instances. Our calibration method use all the problem attributes to calculate the calibrated probabilities, while the classical calibration methods only use the estimated probability. Moreover, our proposed method has the advantage that it can be applied to multiclass problems directly, something that other calibration methods cannot. In general, calibrated models are necessary in complex scenarios where several models have to be combined if we want to obtain good overall results, since a single uncalibrated local model can make the overall model diverge.

1

Introduction

Decision making is something that we are constantly doing, most of the times without being aware about it. For example, if we go to the work by car, we will choose a path. Probably, we always choose the same path, because we know that this is the shortest and/or the fastest one. But, one day, if there is a traffic jam, we can choose another path. In this example, if we eventually make a bad decision, at most, we will spend more time to arrive to work. A different situation is when our decision can cost a lot of money to our organisation.

Fortunately, if we have data of the problem, data mining can help us to make a suitable decision. In fact, the real problem comes when we have to make several related decisions at a time. In this situation, the best local decisions does not always give the best global result. In this work, we propose to combine local models and then, to use simulation to obtain a good global result. We illustrate it with a customer relationship management (CRM) problem, i.e., a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers.

Each local model is a data mining model. Frequently, each model accompanies their predictions with probabilities (probabilistic models), i.e., they give a reliability of the predictions. If we have a single problem, it probably will not matter if the probabilities are not so realistic. For example, if we want to choose a new supplier for our company, we will select the supplier with the maximum probability according to our probabilistic model. The problem is when we have several related problems. If we want to select a fast and cheap supplier, we will have two related problems and, consequently, will need two different probabilistic models. One of the model will be related with how fast suppliers are and the other model will be related with how cheap they are. In this kind of scenarios, we need accurate probabilities, because we merge the models, and bad estimations of the probabilities in a local model may distort the global result.

One solution is calibration. In this work, we present an exhaustive study of the most important calibration techniques, as well as the most common calibration measures used to evaluate the quality of the models. Additionally, we present a new calibration method: "Similarity-Binning Averaging". We explain the advantages of our method and compare it experimentally, with the classical calibration methods.

This work is organised as follows. Chapter 2 is about calibration: Section 2.1 is an introduction to calibration, Section 2.2 explains the classical calibration measures, Section 2.3 explains the most common calibration methods, in Section 2.4 we present our calibration method and in Sections 2.5 and 2.6 the experiments comparing the calibration methods are presented. In Chapter 3 a CRM problem is solved combining several data mining problems by means of simulation: Section 3.1 is an introduction to multi-decision data mining problems, Section 3.2 shows how to face a campaign design with one product and Section 3.3 explain the problems when there is more than one product and a way to solve them using simulation, in Section 3.4 some experimental results are shown. In Chapter 4 we apply our "Similarity-Binning" calibration method to a campaign design with N products: in Section 4.1 some experiments under different settings are shown. Finally, Chapter 5 is the conclusion and Chapter 6 is the future work.

2

Calibration of Machine Learning Models

The evaluation of machine learning models is a crucial step before their application because it is essential to assess how well a model will behave for every single case. In many real applications, not only is it important to know the "total" or the "average" error of the model, it is also important to know how this error is distributed and how well confidence or probability estimations are made. Many current machine learning techniques are good in overall results but have a bad distribution/assessment of the error.

For these cases, calibration techniques have been developed as postprocessing techniques in order to improve the probability estimation or the error distribution of an existing model.

This chapter presents the most common calibration techniques and calibration measures. Both classification and regression are covered, and a taxonomy of calibration techniques is established. Special attention is given to probabilistic classifier calibration.

2.1 Introduction to Calibration

One of the main goals of machine learning methods is to build a model or hypothesis from a set of data (also called evidence). After this learning process, the quality of the hypothesis must be evaluated as precisely as possible. For instance, if prediction errors have negative consequences in a certain application domain of a model (for example, detection of carcinogenic cells), it is important to know the exact accuracy of the model. Therefore, the model evaluation stage is crucial for the real application of machine learning techniques. Generally, the quality of predictive models is evaluated by using a training set and a test set (which are usually obtained by partitioning the evidence into two disjoint sets) or by using some kind of cross-validation or bootstrap if more reliable estimations are desired. These evaluation methods work for any kind of estimation measure. It is important to note that different measures can be used depending on the model. For classification models, the most common measures are accuracy (the inverse of error), f-measure, or macro-average.

In probabilistic classification, besides the percentage of correctly classified instances, other measures such as logloss, mean squared error (MSE) (or Brier's score) or area under the ROC curve (AUC) are used. For regression models, the most common measures are MSE, the mean absolute error (MAE), or the correlation coefficient.

With the same result for a quality metric (e.g. MAE), two different models might have a different error distribution. For instance, a regression model R_1 that always predicts the true value plus 1 has a MAE of 1. However, it is different to a model R_2 that predicts the true value for $n-1$ examples and has an error of n for one example. Model R_1 seems to be more reliable or stable, i.e., its error is more predictable. Similarly, two different models might have a different error assessment with the same result for a quality metric (e.g. accuracy). For instance, a classification model C_1 which is correct 90% of the cases with a confidence of 0.91 for every prediction is preferable to model C_2 which is correct 90% of the cases with a confidence of 0.99 for every prediction. The error self-assessment, i.e., the purported confidence, is more accurate in C_1 than in C_2.

In both cases (classification and regression), an overall picture of the empirical results is helpful in order to improve the reliability or confidence of the models. In the case of regression, the model R_1, which always predicts the true value plus 1, is clearly uncalibrated, since predictions are usually 1 unit above the real value. By subtracting 1 unit from all the predictions, R_1 could be calibrated and interestingly, R_2 can be calibrated in the same way. In the case of classification, a global calibration requires the confidence estimation to be around 0.9 since the models are right 90% of the time.

Thus, calibration can be understood in many ways, but it is usually built around two related issues: how error is distributed and how self-assessment (confidence or probability estimation) is performed. Even though both ideas can be applied to both regression and classification, this chapter focuses on error distribution for regression and self-assessment for classification.

Estimating probabilities or confidence values is crucial in many real applications. For example, if probabilities are accurated, decisions with a good assessment of risks and costs can be made using utility models or other techniques from decision making. Additionally, the integration of these techniques with other models (e.g. multiclassifiers) or with previous knowledge becomes more robust. In classification, probabilities can be understood as degrees of confidence, especially in binary classification, thus accompanying every prediction with a reliability score [8]. In regression, predictions might be accompanied by confidence intervals or by probability density functions.

Therefore, instead of redesigning existing methods to directly obtain good probabilities or better error distribution, several calibration techniques have recently been developed. A calibration technique is any postprocessing technique that attemps to improve the probability estimation or to improve the error distribution of a given predictive model. A general calibration technique can be used to improve any existing machine learning method: decision trees, neural networks, kernel

methods, instance-based methods, Bayesian methods, etc. It can also be applied to hand-made models, expert systems or combined models

Depending on the task, different calibration techniques can be applied and the definition of calibration can be stated more precisely. The most common calibration techniques are divided into four groups and each group has a type code to identify it:

- TYPE CD. Calibration techniques for discrete classification ("(class) distribution calibration in classification" or simply "class calibration"): a typical decalibration arises when the model predicts examples of one or more classes in a proportion that does not fit the original class distribution. In the binary case (two classes), it can be expressed as a mismatch between the expected value of the proportion of classes and the actual one. For instance, if a problem has a proportion of 95% of class a and 5% of class b, a model predicting 99% of class a and 1% of class b is uncalibrated, although it could have a low error (ranging from 4% to 5%) . This error distribution can be clearly shown on a confusion or contingency table. Therefore, class calibration is defined as the degree of approximation of the true or empirical class distribution with the estimated class distribution. The standard calibration model procedure is performed by changing the threshold that determines when the model predicts a or b, making this threshold stricter with class a and milder with class b to balance the proportion. Note that, in principle, this type of calibration might produce more error. In fact, this is usually the case when a useful model for problems with very imbalanced class distribution must be obtained, i.e., the minority class has very few examples. Note that this calibration can be used to match global proportions as well as local proportions. This is related to the problem of "repairing concavities" [17].

- TYPE CP. Calibration techniques for probabilistic classification ("probabilistic calibration for classification"): a probabilistic classifier is a decision system that accompanies each prediction with a probability estimation. If the classification model predicts that it is 99% sure, it should expect to be right 99% of the time. If it is only right 50% of the time, it is not calibrated because its estimation was too optimistic. Similarly, if it predicts that it is only 60% sure, it should be right 60% of the time. If the classifier is right 80% of the time, it is not calibrated because its estimation was too pessimistic. In both cases, the expected value of the number or proportion of correct guesses (in this case, the probability or the confidence assessment) does not match the actual value. Calibration is thus defined as the degree of approximation of the predicted probabilities to the actual probabilities. More precisely, a classifier is perfectly calibrated if, for a sample of examples with predicted probability p, the expected proportion of positives is close to p. Note that even though, accuracy and calibration are dependent on each other, they are very different things. For instance, a random classifier (a coin tosser), which always

assigns 0.5 probability to its predictions, is perfectly calibrated. On the other hand, a very good classifier can be uncalibrated if correct positive (respectively negative) predictions are accompanied by relatively low (respectively high) probabilities. Also note that good calibration usually implies that estimated probabilities are different for each example (except for the random coin tosser). For some examples, confidence will be high, and for other more difficult ones, confidence will be low. This implies that measures to evaluate this type of calibration must evaluate agreement between the expected value and the real value in a local way by using partitions or bins of the data.

- TYPE RD. Calibration techniques to fix error distribution for regression ("distribution calibration in regression"): the errors in this case are not regularly distributed along the output values. The error is concentrated in the high values, or the average error (either positive or negative) is not close to zero. The expected value, which should be close to the actual value, can be defined in several ways. For instance, the expected value of the estimated value (y_{est}) should be equal (or close) to the real value (y), i.e., $E(y_{est}) = E(y)$ or, equivalently, $E(y_{est} - y) = 0$. In the example R_1 above, $E(y_{est}) = E(y) + 1$. The mean error (its expected value) would be 1 and not 0. Another equation that shows that a model might be uncalibrated is the expected value of the quotient between the estimated value and the real value, $E(y_{est}/y)$, which should be equal or close to 1. If this quotient is greater than one, the error is usually positive for high values and negative for low values. Typically, these problems are detected and penalised by classical measures for evaluating regression models [23]. Also, many techniques (e.g. linear regression) generate calibrated models (at least for error distribution and self-assessment). Other kinds of more sophisticated techniques especially hand-made models might be uncalibrated and may require a calibration.

- TYPE RP. Calibration techniques to improve probability estimation for regression ("probabilistic calibration for regression"): this is a relatively new area [4] and is applicable when continuous predictions are accompanied or substituted by a probability density function (or, more restrictively, confidence intervals). Regression models of this kind are usually referred to as "density forecasting" models. Instead of saying that the temperature is going to be 23.2º Celsius, a probability density function can be given assigning a probability of 0.9 indicating that the temperature is between 21º and 25º is 0.9, or a probability of 0.99 indicating that the temperature is between 15º and 31º. If the predictions are very accurate, the density functions (and, hence, confidence intervals) should be narrower. If the predictions are bad, the density functions should be broader in order to approximate the estimated probabilities to the real probabilities. As in the type CP, in general, a good calibration requires that these density functions be specific to each prediction, i.e., for some cases where the

confidence is high, confidence intervals will be narrower. For difficult cases, confidence intervals will be broader. As in the type CP, measures to evaluate this type of calibration must evaluate agreement between the expected value and the real value in a local way by using partitions or bins of the data.

Table 2.1 summarises these four types of calibration.

TYPE	Task	Problem	Global/Local	What is calibrated?
CD	Classification	Expected class distribution is different from real class distribution	Global or local	Predictions
CP	Classification	Expected/estimated probability of correct guesses is different from the real proportion	Local	Probability/confidence
RD	Regression	Expected output is different from the real average output	Global or local	Predictions
RP	Regression	Expected/estimated error confidence intervals or probability density functions are too narrow or too broad	Local	Probability/confidence

Table 2.1: A taxonomy of calibration problems.

Note that types CD and RD must necessarily modify predictions in order to calibrate the results. In type CD, if the class threshold is moved, some predictions change. In RD if an attempt is made to reduce high values and increase low values, predictions also change. In contrast, for types CP and RP, calibration can be made without having to modify predictions: only confidence assessments or probabilities need to be adjusted. For CP, in particular, these kinds of calibrations are known as *isotonic*. Consequently, some measures such as average error will not be affected by these two types of calibrations.

Additionally, if calibration is to be improved, measures are also needed to evaluate this improvement. A calibration measure is any measure that is able to quantify the degree of calibration of a predictive model. For each type of calibration model, some specific measures are useful to evaluate the degree of calibration, while others are only partially sensitive or completely useless. For instance, for CP, the most common measure, accuracy (or % of errors), is completely useless. For RP, the most common measure, MSE, is completely useless. Some of these calibration measures are reviewed in the following section.

Of all the types shown in Table 2.1, type CP is the one that has recently received the most attention. In fact, for many researchers in machine learning, the term "calibration" usually refers to this type, without having to specify that there are other types of calibration. Additionally, this is the type that has developed more techniques and more specific measures. Furthermore, regression techniques and measures have been traditionally developed to obtain calibrated models, so less

improvement is expected from calibration techniques. For this reason, this chapter focuses mainly on classification problems, and specifically on type CP.

This chapter provides a general overview of calibration and a review of some of the most-well-known calibration evaluation measures and calibration methods which have been proposed for classification and regression. This chapter also analyses some open questions and challenges that affect future research on calibration.

2.2 Calibration Evaluation Measures

As mentioned in the introduction, a calibration measure is any measure that can quantify the degree of calibration of a predictive model. As Table 2.2 shows, many classical quality metrics are not useful for evaluating calibration techniques. In fact, new and specific measures have been derived or adapted to evaluate calibration, especially for types CP and RP.

TYPE	Calibration measures	Partially sensitive measures	Insensitive measures
CD	Macro-averaged accuracy, proportion of classes	Accuracy, mean F-measure, ...	Area Under the ROC Curve (AUC), MSE^p, LogLoss, ...
CP	MSE^p, LogLoss, CalBin, CalLoss		AUC, Accuracy, mean F-measure, ...
RD	Average error, Relative error	MSE^r, MAE, ...	
RP	Anderson-Darling (A2) test		Average error, relative error, MSE^r, MAE, ...

Table 2.2: The second column shows the calibration measures for each type of calibration problem. References and definitions for all the measures can be found in [15] and [4]. The third and fourth columns show measures which are partially sensitive (but not very useful in general) or completely insensitive to each type of calibration.

The second column in Table 2.2 shows the calibration measures. However, does not mean that these measures *only* measure calibration. For instance, for type CP, CalBin and CalLoss only evaluate calibration, while MSE or Logloss evaluate calibration as well as other components. These two types of measures are referred to as *pure* and *impure* calibration measures, respectively. However, pure calibration measures may make a classifier which always predicts the positive prior probability look perfectly calibrated according to these measures. Other impure metrics are insensitive to calibration: for example, qualitative measures (accuracy, mean F-measure, etc.), where the calibration function is applied to the threshold; or ranking measures (such as AUC), where the calibration modifies the value of the probabilities but not their order. Therefore, calibration has emerged as an important issue, since many traditional quality metrics completely disregard it. Hence, many machine learning techniques generate uncalibrated models.

Note that we use two different terms for Mean Square Error, MSE^p and MSE^r.

The reason for this is that MSE^p is used for classification and compares the estimated probabilities with the actual probability (0 or 1), while MSE^r is used for regression and compares two continuous values.

Most of the measures shown in the second column of Table 2.2 are very well-known and do not require any further definition. Nevertheless, it is important to remark the following: macro-averaged accuracy is the average of the partial accuracies for each class; the proportion of classes is computed for the predictions on a dataset and can be compared with the real proportion; and average error and relative error are well-known in regression. The rest of this section is devoted to explaining MSE^p, LogLoss, CalBin, CalLoss for type CP and Anderson-Darling (A2) test for RP.

2.2.1 Calibration Measures for Type CP

In this chapter, the following notation is used. Given a dataset T, n denotes the number of examples, and C denotes the number of classes. $f(i,j)$ represents the actual probability of example i to be of class j. It is assumed that $f(i,j)$ always takes values in 0,1 and is strictly not a probability but an indicator function. The number of examples of class j is denoted as $n_j = \sum_{i=1}^{n} f(i,j)$. $p(j)$ denotes the prior probability of class j, i.e., $p(j) = n_j/n$. Given a classifier, $p(i,j)$ represents the estimated probability of example i to be of class j taking values in [0,1].

Mean Squared Error

Mean Squared Error (MSE) is a measure of the deviation from the true probability. In classification, MSE^p is also known as Brier Score.

MSE is defined as:

$$MSE = \frac{\sum_{j=1}^{C}\sum_{i=1}^{n}(f(i,j) - p(i,j))^2}{n \cdot C} \tag{2.1}$$

Although MSE was originally not a calibration measure, it was decomposed in [25] in terms of calibration loss and refinement loss. An important idea of decomposition is that data is organised into bins, and the observed probability in each bin is compared to the predicted probability or to the global probability. Some kind of binning is present in many calibration methods and measures. For decomposition, T is segmented in k bins.

$$MSE = \frac{\sum_{j=1}^{C}\sum_{l=1}^{k}\sum_{i=1,i\in l}^{n_l} n_l \cdot (p(i,j) - \bar{f}(i,j))^2 - \sum_{l=1}^{k} n_l \cdot (\bar{f}_l(i,j) - \bar{f}(j)) + \bar{f}(j) \cdot (1 - \bar{f}(j))}{n \cdot C} \tag{2.2}$$

where $\bar{f}_l(i,j) = \sum_{i=1,i\in l}^{n_l} \frac{f(i,j)}{n_l}$ and $\bar{f}(j) = \sum_{i=1}^{n} \frac{f(i,j)}{n}$. The first term measures the calibration of the classifier while the rest of the expression measures other components, which are usually grouped under the term "refinement". The calibration component is the only one that is affected by isotonic calibrations. The other components, discrimination and uncertainty, are not modified if probabilities are calibrated in such a way that the order is not modified (i.e., isotonic), since bins will not be altered.

LogLoss

Logloss is a similar measure and is also known as probabilistic cost or entropy. It is related to the Kullback-Leibler distance between the real model and the inferred model [20], [21] and [10]. It is defined as follows:

$$LogLoss = \sum_{j=1}^{C} \sum_{i=1}^{n} \frac{(f(i,j) \cdot \log p(i,j))}{n} \tag{2.3}$$

Calibration by Overlapping Bins

One typical way of measuring classifier calibration consists of splitting the test set into several segments or bins, as the MSE decomposition shows (even though MSE does not need to use bins to be computed). The problem of using bins is that if too few bins are defined, the real probabilities are not properly detailed to give an accurate evaluation. Also, if too many bins are defined, the real probabilities are not properly estimated. A partial solution to this problem is to make the bins overlap.

A calibration measure based on overlapping binning is CalBin [6]. This is defined as follows: for each class, all cases must be ordered by predicted $p(i,j)$, giving new indices i^*. Then, the first 100 elements (i^* from 1 to 100) are taken as the first bin. Then, the percentage of cases of class j in this bin is calculated as the actual probability, $\hat{f}_j$. The error for this bin is $\sum_{i^*\in 1..100} |p(i^*,j) - \hat{f}_j|$. The second bin with elements (2 to 101) is taken and its error is computed in the same way. Finally, the errors are averaged. The problem of using 100 as the size of each bin (as Caruana and Niculescu-Mizil suggest in [6]) is that it might be too large of a bin for small datasets. Instead of 100, a different bin length, $s = n/10$ could be set in order to make it more size-independent. Formally:

$$CalBin(j) = \frac{1}{n-s} \sum_{b=1}^{n-s} \sum_{i^*=b}^{b+s-1} \left| p(i^*,j) - \frac{\sum_{i^*=b}^{b+s-1} f(i^*,j)}{s} \right| \tag{2.4}$$

Calibration Loss

Calibration can clarify the relationship between the AUC-based measures and ROC analysis [13] and [16]. The receiver operating characteristic curve (ROC curve) is a graphical depiction of classifiers based on their performance. It is generally applied to binary classifiers. The ROC space is a two-dimensional graph in which the True positive rate (the fraction of positives correctly classified or $tprate$) is plotted on the Y axis, and the False positive rate (the fraction of negatives incorrectly classified or $fprate$) is plotted on the X axis. Each discrete classifier produces an ($fprate$, $tprate$) pair that corresponds to a single point in the ROC space. Probabilistic classifiers provide a value (probability or score) that represents the degree to which an instance belongs to a class. In combination with a threshold, the classifier can behave as a binary classifier by assigning a class (for instance, positive) if the produced score is above the threshold or by assigning the other class (negative) otherwise. Each threshold produces a point in the ROC space, and a ROC curve is generated by drawing a line crossing all the points. The area under a ROC curve is abbreviated as AUC. "The AUC has an important statistical property: the AUC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. This is equivalent to the Wilcoxon test of ranks" [12]. Therefore, the AUC is a class separability or instance ranking measure because it evaluates how well a classifier ranks its predictions.

A perfectly calibrated classifier always gives a convex ROC curve. However even though, a classifier can produce very good rankings (high AUC), the estimated probabilities might differ from the actual probabilities.

One method for calibrating a classifier is to compute the convex hull or, equivalently, to use isotonic regression. From the decomposition of the Brier Score, Flach and Matsubara derive calibration loss and refinement loss, where calibration loss is defined as the mean squared deviation from empirical probabilities that are derived from the slope of ROC segments [16].

$$CalLoss(j) = \sum_{b=1}^{r_j} \sum_{i \in s_{j,b}} (p(i,j) - \sum_{i \in s_{j,b}} \frac{f(i,j)}{|s_{j,b}|}) \tag{2.5}$$

where r_j is the number of segments in the ROC curve for class j, i.e. the number of different estimated probabilities for class j : $|p(i,j)|$. Each ROC segment is denoted by $s_{j,b}$, with $b \in 1 \ldots r_j$, and formally defined as:

$$s_{j,b} = \{i \in 1 \ldots n | \forall k \in 1 \ldots n : p(i,j) \geqslant p(k,j) \wedge i \notin s_{j,b}, \forall d < b\} \tag{2.6}$$

2.2.2 Calibration Measures for Type RP

Anderson-Darling (A^2) test

For type RP, where the task is usually referred to as density forecasting, instead of predicting a continuous value, the prediction is a probability density function. Evaluating this probability density function in terms of calibration cannot be done with classical measures such as MSE^r, relative quadratic error, or other classical measures in regression. In [4] Carney and Cunningham adapt a well-known normality test, the Anderson-Darling (A^2) test over the probability integral transformation, as a measure of pure calibration. This measure is used to evaluate whether the probability density functions estimated by the regression model are accurate. For the specific definition see [4].

2.3 Calibration Methods

2.3.1 Calibration Methods for Type CD

In the case of discrete classification, the best way to know whether a model is uncalibrated according to the number of instances per class (type CD) is to analyse the confusion matrix. The confusion matrix is a visual way of showing the recount of cases of the predicted classes and their actual values. Each column of the matrix represents the instances in a predicted class, while each row represents the instances in an actual class. One benefit of a confusion matrix is that it is easy to see if the system is confusing two classes (i.e., commonly mislabelling one as another). For example, if there are 100 test examples and a classifier, an example of a confusion matrix with three classes a, b and c could be as in Table 2.3.

		Real		
		a	b	c
Predicted	a	20	2	3
	b	0	30	3
	c	0	2	40

Table 2.3: Confusion matrix with three classes.

In this matrix from 100 examples: 20 were from class a and all of them were well classified; 34 were from class b, and 30 of them were well classified as b, 2 were misclassified as a, and 2 were misclassified as c; finally, 46 of the examples were from class c, 40 of them were well classified as c, 3 were misclassified as a and 3 were misclassified as b. If they are grouped by class, there is a proportion of 20, 34, 46 for the real data, and a proportion of 25, 33, 42 for the predicted data. These proportions are quite similar and, therefore, the classifier is calibrated with regard to the original class distribution.

Another matrix can also be considered (Table 2.4).

		Real	
		a	b
Predicted	a	60	2
	b	40	23

Table 2.4: Confusion matrix with two classes.

In this matrix, the proportion of real data for classes a and b are (100, 25), while the proportion of predicted data are (62, 63). Thus, in this case, the model is uncalibrated. This type of disproportion is quite common and usually favours the majority classes. There are techniques to solve the problem once the model is obtained. To do so, the predictions of the models must be accompanied by probabilities or reliability values (or simply, scores). Then, the threshold that splits into the classes can be changed.

The technique presented by [22] can be applied. Even though is specialised in Bayesian classifiers, it can be applied to any classification model that accompanies its predictions by probabilities or reliabilities. A naïve Bayes classifier estimates the probabilities for each class independently. For example, for the following probabilities for each class: $p(a|x) = 0.2$ and $p(b|x) = 0.05$, there are no more classes. Therefore, the sum of the probabilities is not 1. In general, the naïve Bayes classifiers assign very low probabilities because the probability is the product of several factors that, in the end, reduce the absolute values of the probabilities. The decision rule used to apply the model is:

If $p(a|x) > p(b|x)$ then a else b

The consequence of this rule is that the result is not calibrated in most of the cases. It may be that this rule produces many more examples of the class a (or b) than there were in the original distribution. The solution to this problem is to estimate a threshold that is fitted to the original distribution.

If there are only two classes, the solution is very easy. A ratio of the two probabilities can be calculated: $r = p(a|x)/p(b|x)$. This ratio is 0 to infinite. It can also be normalised between 0 and 1 with a sigmoid, if desired. The aim is to obtain a threshold u with the test set where the distribution of the model is similar to the original distribution. Thus, the rule changes to:

If $r > u$ then a else b

In [22] Lachiche and Flach show that the results of the models can be improved significantly with only a small adjustment (in that work the threshold adjustment is based on the ROC analysis and it is extended to multiclass). In particular, from 25 analysed datasets, this simple optimisation significantly improved the accuracy in 10 cases and was reduced in only 4 of them.

Apart from that simple approximation, there are other works that calculate the

optimum threshold fitted to the original distribution, such as [26].

2.3.2 Calibration Methods for Type CP

Another case is the calibration of probabilistic classification models (type CP), which requires more sophisticated techniques. In this case, when the model predicts that the probability of the class a is 0.99, this means that the model is more confident that the class is a than when the probability is 0.6. Determining the reliability of a prediction is fundamental in many applications such as: diagnosis, instance selection, and model combination.

In addition to the measures introduced above (MSE, LogLoss, CalBin and CalLoss), the fundamental tool for analysing the calibration of models of this type is the reliability diagram [8]. In this diagram, the prediction space is discretised into 10 intervals (from 0 to 0.1, from 0.1 to 0.2, etc.). The examples whose probability is between 0 and 0.1 go into the first interval, the examples between 0.1 and 0.2 go into the second, etc. For each interval, the mean predicted value (in other words, the mean predicted probability) is plotted (X axis) versus the fraction of positive real cases (Y axis). If the model is calibrated, the points will approach the diagonal.

Figure 2.1 shows an example of an uncalibrated model and a calibrated model.

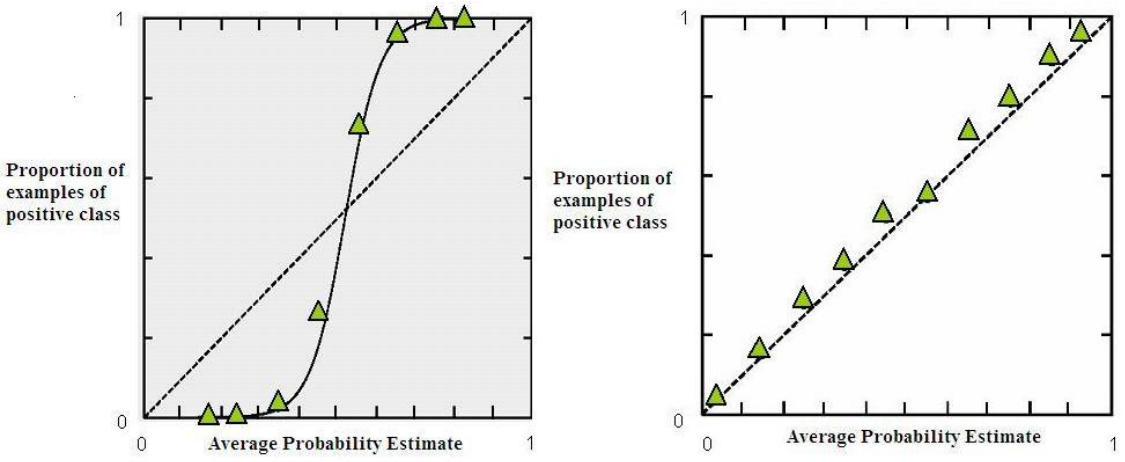

Figure 2.1: Left: an example of a reliability diagram of an uncalibrated model. Right: an example of a reliability diagram of a calibrated model.

For instance, in the model on the left, there are no examples with a predicted probability lower than 0.1. The next interval, where the examples have an assigned probability between 0.1 and 0.2 for the positive class (with a mean of 0.17), there are no examples of the positive class. Thus, these predictions have an estimated probability that is too high. It should be closer to 0 instead of to 0.17. The values at the end of the curve show that the examples with assigned probabilities between

0.7 and 0.8 are all from the positive class. They should probably have a higher probability because they are cases with greater certainty.

The model on the right shows that the correspondence is more corect: there are probabilities distributed from 0 to 1 and, moreover, they are usually the same as the percentage of examples.

There are several techniques that can calibrate a model like the one on the left and transform it into a model like the one on the right. The most common are: binning averaging, isotonic regression and Platt's method. The objective of these methods (as postprocessing) is to transform the original estimated probabilities[1] $p(i, j)$ into calibrated probability estimates $p^*(i, j)$. It is important to remark that all of these general calibration methods can only be used (directly, without approaches) in binary problems because they all use the sorted estimated probability to calculate the calibrated probability.

The calibration function is monotonically non-decreasing (also called isotonic). Most calibration methods presented in the literature are isotonic. This makes it reasonable to use MSE or LogLoss as measures to evaluate calibration methods since the "separability components" are not affected. This is clearly seen in the so-called "decompositions of the Brier score" [29] and [25], which is described above.

Binning Averaging

The first calibration method is called binning averaging [31] and consists of sorting the examples in decreasing order by their estimated probabilities and dividing the set into k bins (i.e., subsets of equal size). Then, for each bin b, $1 \leq b \leq k$, the corrected probability estimate for a case i belonging to class j, $p^*(i, j)$, is the proportion of instances in b of class j. The number of bins must be small in order to reduce the variance of the estimates. In their paper, Zadrozny and Elkan fixed $k = 10$ in the experimental evaluation of the method.

An illustrative example for explaining how this method works is shown here. Consider the following training set sorted by its probability of membership to the positive class grouped in 5 bins (Table 2.5).

Then, if a new example is assigned a score of 0.68, it belongs to bin 3 and its calibrated probability is $\frac{0.70+0.66+0.62+0.62}{4} = 0.65$

Isotonic Regression (Pair-Adjacent Violator Algorithm, PAV)

Another slightly more sophisticated technique also for two-class problems is isotonic regression. [1] presented a pair-adjacent violator algorithm (PAV) for calculating isotonic regression. The idea is that calibrated probability estimates must be a monotone decreasing sequence, i.e., $p_1 \geq p_2 \geq \ldots \geq p_n$. If the sequence is not satisfied each time that a pair of consecutive probabilities, $p(i, j)$ and $p(i + 1, j)$,

[1] Scores can also be used [32]

Bin	Instance	Score
1	e_1	0.95
	e_2	0.94
	e_3	0.91
	e_4	0.90
2	e_5	0.87
	e_6	0.85
	e_7	0.80
	e_8	0.76
3	e_9	0.70
	e_{10}	0.66
	e_{11}	0.62
	e_{12}	0.62
4	e_{13}	0.51
	e_{14}	0.49
	e_{15}	0.48
	e_{16}	0.48
5	e_{17}	0.45
	e_{18}	0.44
	e_{19}	0.44
	e_{20}	0.42

Table 2.5: Example of the binning averaging method.

does not satisfy the above property $p(i,j) < p(i+1,j)$, the PAV algorithm replaces both of them by their probability average, that is:

$$p^*(i,j) = p^*(i+1,j) = \frac{p(i,j) + p(i+1,j)}{2} \tag{2.7}$$

This process is repeated (using the new values) until an isotonic set is reached.

In the Table 2.6, an example of the PAV algorithm, extracted from [13] is shown. There are 15 instances, 6 negatives and 9 positives. The PAV algorithm begins by sorting the instances in decreasing order by score and assigning probability estimates of 1 for each positive example and 0 for each negative example. The algorithm iteratively looks for adjacent violators: a local non-monotonicity in the sequence. Initially, adjacent violators (a zero followed by a one) exist at instance pairs 2-3, 6-7, 9-10 and 12-13.

The algorithm operates from the bottom of the instance sequence to the top. First, in step a1, the violation generated by instances 12 and 13 is removed by pooling the two instances together and assigning them a probability estimate of 1/2 (see column a1). This introduces a new violation between instance 11 and the adjacent group 12-13. To remove this new violation, in step a2, instance 11 and the group 12-13 are pooled together, forming a pool of three instances (two negatives and one positive) whose probability estimate is 1/3. The result is shown in column a2.

Next, instances 9-10 (one negative and one positive) are pooled, assigning a probability of 1/2 to each instance. The result is shown in column b.

In steps c1 and c2, the violations between instances 6-8 (one negative and two positives) are removed in two steps. Similarly, instances 2-5 (one negative and three positives) are pooled into a group of probability 3/4 (the intermediate steps

♯	Score	Probabilities						
		Initial	a1	a2	b	c1	c2	d
0	0.9	1	1	1	1	1	1	1
1	0.8	1	1	1	1	1	1	1
2	0.7	0	0	0	0	0	0	0.75
3	0.6	1	1	1	1	1	1	
4	0.55	1	1	1	1	1	1	
5	0.5	1	1	1	1	1	1	
6	0.45	0	0	0	**0**	**0.5**	0.67	0.67
7	0.4	1	1	1	**1**			
8	0.35	1	1	1	1	**1**		
9	0.3	0	0	**0**	0.5	0.5	0.5	0.5
10	0.27	1	1	**1**				
11	0.2	0	**0**	0.33	0.33	0.33	0.33	0.33
12	0.18	**0**	**0.5**					
13	0.1	**1**						
14	0.02	0	0	0	0	0	0	0

Table 2.6: Example of the PAV algorithm.

are omitted). The final result is shown in column d. The sequence of probability estimates is now monotonically decreasing and no violators remain. This sequence can now be used as the basis for a function that maps classifier scores into probability estimates.

Platt's Method

Platt presents in [27] a parametric approach for fitting a sigmoid that maps estimated probabilities into calibrated ones. This method was developed to transform the outputs of a support vector machine (SVM) from the original values $[-\infty, \infty]$ to probabilities, but it can be extended to other types of models or probability variations. The idea consists of applying a sigmoid function to the values of the form:

$$p^*(i,j) = \frac{1}{1 + e^{A \times p(i,j) + B}} \tag{2.8}$$

The parameters A and B are determined so that they minimise the negative log-likelihood of the data.

Platt's method is most effective when the distortion in the predicted probabilities has a sigmoid form (as in the above example). Isotonic regression is more flexible and can be applied to any monotonic distortion. Nevertheless, isotonic regression is used to present overfitting problems in some cases. Also, all the above methods can use the training set or an additional validation set for calibrating the model. The quality of the calibration may depend on this possibility and the size of the dataset. This is a recurrent issue in calibration, and it has been shown that some methods are better than others for small calibration sets (i.e., Platt's scaling is more effective than isotonic regression when the calibration set is small [6].

Other Related Calibration Methods

Apart from the methods for obtaining calibrated probabilities, there are other calibration techniques that are only applicable to specific learning methods. For instance, smoothing by m-estimate [7] and Laplace [28] are other alternative ways of improving the probability estimates given by an unpruned decision tree. Probabilities are generated from decision trees as follows. Let T be a decision tree and let l be a leaf that contains n training instances. If k of these instances are of one class (for instance, of the positive class), then when T is applied to classify new examples, it assigns a probability of $p = \frac{k}{n}$ that each example i in l belongs to the positive class. However, using the frequencies derived from the count of instances of each class in a leaf might not give reliable probability estimates (for instance, if there are few instances in a leaf). Therefore, for a two-class problem, the Laplace correction method replaces the probability estimate by $p' = \frac{k+1}{n+2}$. For a more general multiclass problem with C classes, the Laplace correction is calculated as $p' = \frac{k+1}{n+C}$. As Zadrozny and Elkan state "the Laplace correction method adjusts probability estimates to be closer to 1/2, which is not reasonable when the two classes are far from equiprobable, as is the case in many real-world applications. In general, one should consider the overall average probability of the positive class should be considered, i.e., the base rate, when smoothing probability estimates" [31] (p. 610). Thus, smoothing by m-estimate consists of replacing the above-mentioned probability estimate by $p' = \frac{k+b \cdot m}{n+m}$, where b is the base rate and m is the parameter for controlling the shift towards b. Given a base rate b, Zadrozny and Elkan suggest using m such that $b \cdot m = 10$.

Another related technique that is also applicable to decision trees is curtailment [31]. The idea is to replace the score of a small leaf (i.e., a leaf with few training instances) with the estimate of its parent node, if it contains enough examples. If the parent node still has few examples, the same process is repeated with its parent node and so on until either a node is sufficiently populated or the tree root is reached.

2.3.3 Calibration Methods for Type RD

Most regression techniques to date have implicitly or explicitly considered the case when the goal is to have the expected output to be equal (or close) to the real average output (type RD). This is because there are two numeric outputs (the predicted value and the real value), so there is greater variety of corrective functions to apply. Figure 2.2 depicts a comparison of the behaviour of the test data, denoted by "real", and the model that is to be calibrated ("predicted").

As stated in section 2.1, the characteristic of a calibrated model for type RD is that the errors are equally distributed for the different output values. In other words, the expected value from distinct functions between the predicted value and the real value must be the accurate one according to the function. In one case, for example, the expected value of the difference between the estimated value (y_{est}) and

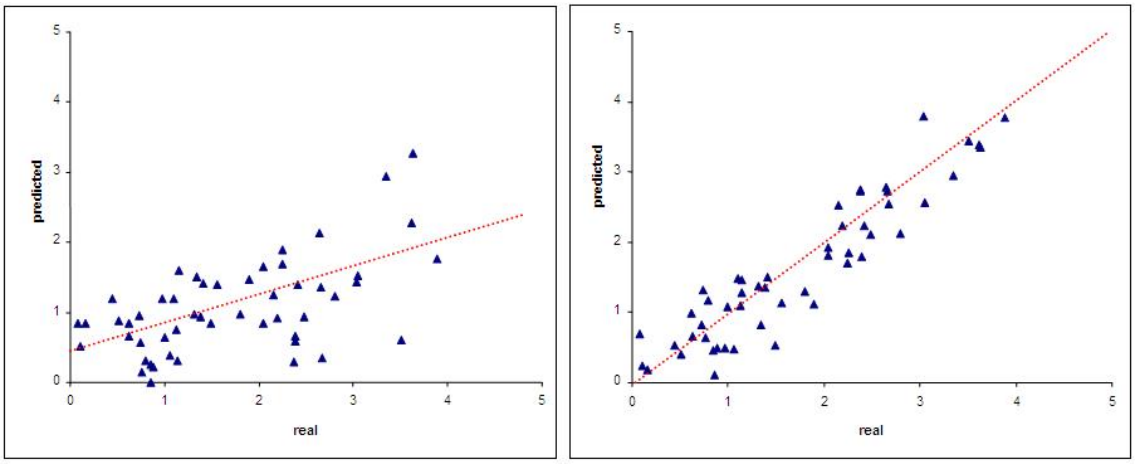

Figure 2.2: Calibration of regression models. Left: an uncalibrated model. Right: a calibrated model.

the real value (y) must be near to zero, so $E(y_{est} - y) = 0$. If this value is less than 0, then, on average, the real values are a little higher than the estimated ones; and if this value is greater than 0, on average, the real values are a little lower than the estimated ones. Most regression models usually have this difference quite well calibrated. However, in another case, the expected value of the quotient between the estimated value and the real value should be near to one, so $E(y_{est}/y) = 1$. If this quotient is greater than one, the error is usually positive for high values and negative for low values. Techniques such as linear regression usually give calibrated models, but others (nonlinear regression, local regression, neural networks, decision trees, etc.) can give uncalibrated models.

Logically, in both cases, the errors can be corrected by decreasing the high values and increasing the low values. In general, the solution comes from obtaining some type of estimation of the decalibration function between the real values and the predicted values. One of the most common approximations consists of calculating a linear regression (as in the plots shown in Figure 2.2) and applying it to the model in order to fit the calibration. These calibrations usually increase the mean squared error $\frac{(y-y_{est})^2}{n}$, but can reduce the relative mean squared error $\frac{(y-y_{est})^2}{y-mean(y_{est}))\times n}$ or the error tolerance. When the decalibration function is nonlinear (but it has a pattern), the problem of calibration becomes more complex, and some kind of nonlinear or local regression is needed to calibrate the model. In these cases, it is not considered to be a calibration process, as such, but rather a meta-learner with several stages (stacking, cascading, etc.).

2.3.4 Calibration Methods for Type RP

For type RP, the prediction is a probability density function, which is what must be improved. This is a much more complex problem since improving this type of calibration can be done by mangling the prediction estimates (i.e., the MSE can be increased as the result of calibrating). Consequently, a trade-off must be found.

In [4], they approach the problem by formulating it as a multi-objective optimisation problem. In fact, the two objectives of density forecasting are sharpness (a classical quality criterion based on negative log-likelihood (NLL) that tries to maximise the probability density at the observation) and calibration (using the Anderson-Darling (A^2) test over the probability integral transform in order to achieve empirical consistency in the probability estimates). The authors proposed an approach which consists of applying an "a posteriori" multi-objective evolutionary algorithm (MOEA). "A posteriori" means that it simultaneously optimises the two above-mentioned factors by finding a set of non-dominant solutions (called the Pareto-optimal front) from which the user can select the model which best adapts to its objectives. In this evolutionary approach, the population contains a set of models which are represented by means of a vector of parameters. Hence, "any density forecasting model that can be represented as a vector of values can be optimised using this framework" [5] (p.15). Also, at each step of the algorithm, the objective functions (NLL and A^2) are computed in order to determine if the model must be included in the Pareto set. Finally, evolution of the population is achieved by applying a mutation function. The approach can be generalised to use any other objective function as well as any evolutionary algorithm. The authors have applied their approach to two classes of models and MOEA's: Gaussian Mixture Model (GMM) using a Pareto Mixture Density Network (MDN) and Generalised Autoregressive Conditional Heteroscedasticity (GARCH) models.

2.4 Calibration by Multivariate Similarity-Binning

Most calibration methods are based on a univariate transformation function over the original estimated class probability. In binning averaging, isotonic regression or Platt's method, this function is just obtained through very particular mapping methods, using $p(i, j)$ (the estimated probability) as the only input variable. Leaving out the rest of the information of each instance (e.g. their original attributes) is a great waste of information for the calibration process. In the case of binning methods, the bins are exclusively constructed by sorting the estimated probability of the elements. Binning can be modified in such way that bins overlap or bins move as windows, but it still only depends on one variable (the estimated probability).

The core of our approach is to change the idea of "sorting" for creating bins, into the idea of using similarity to create bins which are specific for each instance. The rationale for this idea is as follows. If bins are created by using only the estimated

probability, calibrated probabilities will be computed from possibly different examples with similar probabilities. The effect of calibration is small, since we average similar probabilities. On the contrary, if we construct the bins using similar examples according to other features, probabilities can be more diverse and calibration will have more effect. Additionally, it will be sensitive to strong probability deviation given by small changes in one or more original features. This means that if noise on a variable can dramatically affect the output, probabilities will be smooth and, hence, more noise-tolerant. For instance, if $(3, 2, a)$ has class $true$ and $(2, 2, a)$ has class $false$, the estimated probability for $(3, 2, a)$ should not be too close to 1.

Based on this reasoning, we have implemented a new calibration method that we call Similarity-Binnig averaging (SB). In these method the original attributes and the estimated probability are used to calculate the calibrated one.

In Figure 3.2 we can observe the schema of the Similarity-Binning averaging method. The first step is to add the calculated scores or estimated probabilities of the $c-1$ classes, given by the classifier, as new attributes to the training set. In the second step the calibrated probability is the predicted class probability calculated by the k-NN algorithm.

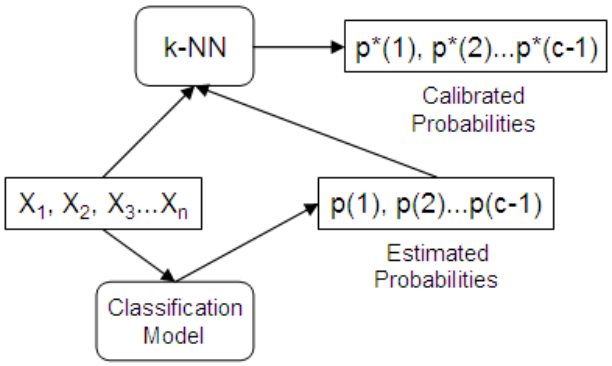

Figure 2.3: Schema of the Similarity-Binning averaging method.

Our method can be seen, in some sense, as a new application of "Cascading" [19] in the field of calibration, where it has never been used before.

Furthermore, the proposed method has the advantage that it can be applied to multiclass problems directly, something that other calibration methods cannot.

2.5 Calibration Experimental Results

For the experimental evaluation, we compare our calibration method explained at Section 2.4 with the classical calibration methods explained at Section 2.3, by means of the most common evaluation measures (Section 2.2). We have used machine learning algorithms implemented in the data mining suite WEKA [30].

We have selected 20 (small and medium-sized) binary datasets (Table 2.7) from the UCI repository [3]. The restriction to binary datasets is motivated to allow comparisons with traditional calibration methods. Randomly, each dataset is split into two different subsets: the training and the test sets (75% and 25% of the instances, respectively). Four different methods for classification have been used (with their default parameters): NaiveBayes, J48, IBk and Logistic. A total of 400 repetitions have been performed for each dataset (100 with each classifier). In each repetition the training set is used to train a classifier and calibrate the probabilities of the model, and the test set is used to test the calibration of the model. Furthermore, in each repetition the same training and test sets are used for all methods.

#	Datasets	Size	Nom.	Num.
1	Breast Cancer	286	9	0
2	Wisconsin Breast Cancer	699	0	9
3	Chess	3196	36	0
4	Horse Colic	368	15	7
5	Credit Rating	690	9	6
6	German Credit	1000	13	7
7	Pima Diabetes	768	0	8
8	Haberman BreastW	306	0	3
9	Heart Statlog	270	0	13
10	Hepatitis	155	13	6
11	House Voting	435	16	0
12	Ionosphere	351	0	34
13	Labor	57	8	8
14	Monks1	556	6	0
15	Mushroom	8124	22	0
16	Sick	3772	22	7
17	Sonar	208	0	60
18	Spam	4601	0	57
19	Spect	80	0	44
20	Tic-tac	958	8	0

Table 2.7: Datasets used in the experiments.

The calibration methods used in the experiments are: binning averaging (with 10 bins), PAV algorithm, Platt's method, and Similarity-Binning (SB) and Similarity-Rank-Binning (SRB) methods (with $k = 10$). All of them have been evaluated for the CalBin and MSE calibration measures. Apart from comparing the results of the calibration methods, we also compare them with two reference methods:

- Base: is the value obtained with the classification techniques without calibration.
- 10-NN[2]: is the value of using the 10 most similar instances (with the original attributes) to estimate the calibrated probability. In other words, it is equal to our method, but only uses the original attributes of the problem to make the bin of the 10 elements more similar and to obtain the calibrated probability.

[2]Implemented by an IBk with $k = 10$ in WEKA

In the Tables 2.8 and 2.9 we show the results with respect to CalBin and MSE for each method. These values are the average of the 400 repetitions for each dataset.

	CalBin					
	Base	10-NN	Bin	PAV	Platt	SB
1	0.1953	0.1431	0.2280	0.2321	0.1856	0.1827
2	0.0494	0.0374	0.0647	0.0447	0.0623	0.0408
3	0.0698	0.1472	0.0501	0.0397	0.0434	0.0491
4	0.1517	0.1216	0.1533	0.1535	0.1421	0.1164
5	0.1220	0.0882	0.1060	0.1035	0.1132	0.0848
6	0.1250	0.1340	0.1263	0.1393	0.1268	0.1233
7	0.1192	0.1049	0.1220	0.1351	0.1205	0.1105
8	0.1984	0.2028	0.2316	0.2400	0.1998	0.1994
9	0.1476	0.1412	0.1690	0.1587	0.1529	0.1443
10	0.1632	0.1359	0.1643	0.1673	0.1727	0.1332
11	0.0665	0.0625	0.0777	0.0542	0.0791	0.0516
12	0.1380	0.1588	0.1179	0.0990	0.1358	0.1064
13	0.1876	0.2996	0.1984	0.1464	0.2110	0.1820
14	0.1442	0.2794	0.1618	0.1355	0.1046	0.1443
15	0.0395	0.0366	0.0418	0.0358	0.0468	0.0368
16	0.0296	0.0158	0.0270	0.0236	0.0250	0.0194
17	0.2606	0.1916	0.2343	0.2376	0.2374	0.2007
18	0.0945	0.0471	0.0636	0.0568	0.0951	0.0466
19	0.3138	0.3497	0.2995	0.2911	0.3110	0.3110
20	0.1240	0.2094	0.1260	0.1198	0.0906	0.0934
AVG.	0.1370	0.1453	0.1382	0.1307	0.1328	**0.1188**

Table 2.8: Results by dataset. CalBin measure.

As we can see in the last row of Tables 2.8 and 2.9, with both calibration measures our method Similarity-Binning has obtained the best results. To confirm that, as suggested in [9], we have calculated a Friedman test and obtained that the six methods do not have identical effects, so we have calculated the Nemenyi post-hoc test to compare all methods with each other (with a probability of 99.5%). In Tables 2.10 and 2.11 the number of wins, ties and losses are shown, for each pair of methods, in terms of CalBin and MSE respectively. There are 80 cases for each combination, i.e. 20 datasets multiplied by 4 different classifiers, with 100 repetitions.

There are some differences between the results when calibration methods are evaluated with each measure (CalBin and MSE) (Tables 2.8 and 2.9). These differences come from the different nature of the measures. While CalBin is a measure that only evaluates calibration, MSE also evaluates other components.

It is important to remark that we are making general comparisons between methods in equal conditions. First of all we are comparing with classification methods without calibration (Base). Logically, calibration would not have had any sense if we had not improved it. The second comparison is with the 10-NN method, which is equal to our method, but only uses the original attributes of the problem to make the bin of the 10 elements more similar and to obtain the calibrated probability. The other 3 methods with we compare only use the estimated probability to calculate the calibrated probability. The most interesting comparison is with the binning averaging method, because our method is also based in the idea of binning.

As we can see in Tables 2.10 and 2.11, with both measures (CalBin and MSE),

	MSE					
	Base	10-NN	Bin	PAV	Platt	SB
1	0.2086	0.1912	0.2086	0.2095	0.2016	0.1998
2	0.0353	0.0262	0.0510	0.0343	0.0362	0.0306
3	0.0465	0.0648	0.0467	0.0387	0.0391	0.0227
4	0.1506	0.1351	0.1503	0.1449	0.1442	0.1336
5	0.1347	0.1121	0.1244	0.1203	0.1262	0.1160
6	0.1889	0.1795	0.1888	0.1883	0.1877	0.1814
7	0.1790	0.1749	0.1795	0.1786	0.1777	0.1821
8	0.1926	0.1992	0.1924	0.1936	0.1906	0.2005
9	0.1491	0.1435	0.1580	0.1503	0.1470	0.1469
10	0.1473	0.1294	0.1460	0.1483	0.1397	0.1305
11	0.0554	0.0568	0.0646	0.0482	0.0538	0.0456
12	0.1266	0.1297	0.1118	0.0981	0.1094	0.0996
13	0.1120	0.1233	0.1582	0.1128	0.1044	0.0907
14	0.1214	0.1047	0.1172	0.1030	0.1065	0.0517
15	0.0083	0.0006	0.0174	0.0040	0.0079	0.0001
16	0.0310	0.0311	0.0355	0.0266	0.0307	0.0241
17	0.2545	0.1760	0.2343	0.2286	0.2285	0.2080
18	0.1027	0.0814	0.0765	0.0721	0.0878	0.0690
19	0.2829	0.2459	0.2776	0.2637	0.2437	0.2692
20	0.1579	0.1141	0.1480	0.1448	0.1468	0.0817
AVG.	0.1343	0.1210	0.1343	0.1254	0.1255	**0.1142**

Table 2.9: Results by dataset. MSE measure.

10-NN	Bin	PAV	Platt	SB	
(28,27,25)	(17,24,39)	(23,29,28)	(17,24,39)	(**45**,15,20)	Base
	(21,19,40)	(31,14,35)	(29,12,39)	(**34**,34,12)	10-NN
		(30,43,7)	(30,29,21)	(**58**,14,8)	Bin
			(21,21,38)	(**45**,24,11)	PAV
				(**50**,14,16)	Platt

Table 2.10: Column vs. Row Nemenyi test: (wins,ties,losses). CalBin measure.

our method Similarity-Binning outperforms the others.

The results grouped by the classification method are presented in Tables 2.12, 2.13 and 2.14. There, we can observe that our calibration method works better for some classification methods than others. It is remarkable to see that no method is able to improve the calibration of Logistic Regression with CalBin. We also show, in that table, the results in terms of AUC. We can see there how the monotonic methods do not modify so much the AUC (the differences come from the ties) specially in the Platt's method, while there is a marked difference with our non-monotonic methods.

2.6 Multiclass Calibration Experiments

As we have already commented, there is not any direct multiclass calibration method to compare with, and we have not implemented another multiclass calibration methods to obtain good approximations of the results. So, we can only compare with the reference methods that we explained in Section 2.5, i.e., "Base" and "10-NN" methods.

In the Table 2.15 the datasets used for the experiments are shown.

10-NN	Bin	PAV	Platt	SB	
(31,28,21)	(18,25,37)	(25,42,13)	(34,28,18)	(**49**,18,13)	Base
	(16,22,42)	(33,18,29)	(33,13,34)	(**36**,31,13)	10-NN
		(42,36,2)	(43,30,7)	(**61**,13,6)	Bin
			(19,40,21)	(**43**,24,13)	PAV
				(**44**,17,19)	Platt

Table 2.11: Column vs. Row Nemenyi test: (wins,ties,losses). MSE measure.

	CalBin				
	Base	Bin	PAV	Platt	SB
10-NN	0.145	0.122	0.116	0.122	0.126
J48	0.153	0.157	0.152	0.161	0.117
NB	0.140	0.137	0.127	0.131	0.115
Log.	0.110	0.136	0.128	0.118	0.118
AVG.	0.137	0.138	0.131	0.133	**0.119**

Table 2.12: Results by Classification Method. CalBin measure.

The results obtained for the datasets of 3 classes, with the CalBin and MSE measures, are shown in the Table 2.16, and the results for the datasets of more than 3 classes are shown in the Table 2.17.

In the Tables 2.18 and 2.19 we show the results group by the classification method, for the datasets of 3 classes and more than 3 classes respectively.

As we can see in the results our calibration method, Similarity-Binning Averaging, outperforms the reference methods. But, as future work, it will be interesting to compare our method with other multiclass calibration methods.

	MSE				
	Base	Bin	PAV	Platt	SB
10-NN	0.121	0.119	0.110	0.109	0.113
J48	0.132	0.146	0.133	0.130	0.118
NB	0.152	0.131	0.124	0.133	0.108
Log.	0.132	0.141	0.135	0.130	0.117
AVG.	0.134	0.134	0.125	0.125	**0.114**

Table 2.13: Results by Classification Method. MSE measure.

	AUC				
	Base	Bin	PAV	Platt	SB
10-NN	0.875	0.866	0.871	0.875	0.873
J48	0.822	0.815	0.821	0.822	0.865
NB	0.860	0.849	0.854	0.859	0.880
Log.	0.852	0.841	0.843	0.851	0.870
AVG.	0.852	0.843	0.847	0.852	**0.872**

Table 2.14: Results by Classification Method. AUC measures.

#	Datasets	Size	Nom.	Num.	Classes
21	Balance Scale	625	0	4	3
22	Cmc	1473	7	2	3
23	Iris	150	0	4	3
24	New ThyroidW	215	0	5	3
25	Splice	3190	60	0	3
26	Tae	151	2	3	3
27	Waveform	5000	40	0	3
28	Wine	178	0	13	3
29	Lymphography	148	15	3	4
30	Vehicle	846	0	18	4
31	Anneal	898	32	6	5
32	Autos	205	10	15	6
33	Dermatology	366	33	1	6
34	Glass	214	0	9	6
35	Segment	2310	0	19	7

Table 2.15: Datasets used in the experiments.

	CalBin			MSE		
	Base	10-NN	SB	Base	10-NN	SB
21	0.1026	0.0947	0.0550	0.0777	0.0607	0.0565
22	0.1140	0.1498	0.1609	0.2110	0.2140	0.2158
23	0.0463	0.0436	0.0441	0.0270	0.0190	0.0275
24	0.0502	0.0584	0.0459	0.0334	0.0397	0.0318
25	0.0782	0.1556	0.0891	0.0552	0.0891	0.0527
26	0.2482	0.2513	0.2546	0.2218	0.2119	0.2200
27	0.0893	0.0576	0.0649	0.1054	0.0880	0.0976
28	0.0497	0.0492	0.0445	0.0296	0.0211	0.0291
AVG.	0.0973	0.1075	**0.0949**	0.0951	0.0930	**0.0914**

Table 2.16: Results by dataset. Meassures: CalBin and MSE. 3 classes

	CalBin			MSE		
	Base	10-NN	SB	Base	10-NN	SB
29	0.0962	0.0940	0.0861	0.0870	0.0710	0.0732
30	0.1087	0.0808	0.0720	0.1251	0.0993	0.0937
31	0.0287	0.0197	0.0190	0.0152	0.0127	0.0075
32	0.0879	0.0880	0.0715	0.0839	0.0824	0.0705
33	0.0376	0.0398	0.0325	0.0122	0.0135	0.0084
34	0.0891	0.0726	0.0767	0.0895	0.0715	0.0770
35	0.0441	0.0329	0.0366	0.0209	0.0117	0.0110
AVG.	0.0703	0.0611	**0.0563**	0.0620	0.0517	**0.0487**

Table 2.17: Results by dataset. Measures: CalBin and MSE. More than 3 classes

	CalBin		MSE	
	Base	SB	Base	SB
10-NN	0.1075	0.0929	0.0930	0.0893
J48	0.1108	0.1105	0.1168	0.1125
NaiveBayes	0.0973	0.0898	0.0872	0.0813
Logistic	0.0737	0.0863	0.0837	0.0824
AVG.	0.0973	**0.0949**	0.0951	**0.0914**

Table 2.18: Results by Classification Method. Measures: CalBin and MSE. 3 classes

	CalBin		MSE	
	Base	SB	Base	SB
10-NN	0.0611	0.0559	0.0517	0.0501
J48	0.0668	0.0604	0.0559	0.0504
NaiveBayes	0.0932	0.0564	0.0868	0.0495
Logistic	0.0601	0.0526	0.0535	0.0450
AVG.	0.0703	**0.0563**	0.0620	**0.0487**

Table 2.19: Results by Classification Method. Measures: CalBin and MSE. More than 3 classes

3

Using Simulation in Multi-Decision Data Mining Problems

Frequently, organisations have to face complex situations where decision making is difficult. In these scenarios, several related decisions must be made at a time, which are also bounded by constraints (e.g. inventory/stock limitations, costs, limited resources, time schedules, etc). We present a new method to make a good global decision when we have such a complex environment with several local interwoven data mining models. In these situations, the best local cutoff for each model is not usually the best cutoff in global terms. We use simulation with Petri nets to obtain better cutoffs for the data mining models. We apply our approach to a frequent problem in customer relationship management (CRM), more specifically, a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers and with usual inventory limitations. We experimentally compare two different methods to obtain the cutoff for the models (one based on merging the prospective customer lists and using the local cutoffs, and the other based on simulation), illustrating that methods which use simulation to adjust model cutoff obtain better results than a more classical analytical method.

3.1 Introduction to Multi-Decision Data Mining Problems

Data mining is becoming more and more useful and popular for decision making. Single decisions can be assisted by data mining models, which are previously learned from data. Data records previous decisions proved good or bad either by an expert or with time. This is the general picture for predictive data mining. The effort (both in research and industry) is then focussed on obtaining the best possible model given the data and the target task. In the end, if the model is accurate, the decisions based on the model will be accurate as well.

However, in real situations, organisations and individuals must make several decisions for several given problems. Frequently, these decisions/problems are interwoven with the rest, have to be made in a short period of time, and are accompanied

with a series of constraints which are also just an estimation of the real constraints. In this typical scenario, making the best local decision for every problem does not give the best global result. This is well-known in engineering and decision making, but only recently acknowledged in data mining. Examples can be found everywhere: we cannot assign the best surgeon to each operation in a hospital, we cannot keep a fruit cargo until their optimal consumption point altogether, we cannot assign the best delivering date for each supplier, or we cannot use the best players for three matches in the same week.

In this context, some recent works have tried to find optimal global solutions where the local solutions given by local models are not good. These works address specific situations: rank aggregation [11] and cost-sensitive learning are examples of this, a more general "utility-based data mining"[1] also addresses this issue, but also some other new data mining tasks, such as quantification [18], are in this line. Data mining applied to CRM (Customer-Relationship Management) [2] is also one of the areas where several efforts have also been done.

Although all these approaches can be of great help in specific situations, most of the scenarios we face in real data mining applications do not fit many of the assumptions or settings of these previous works. In fact, many real scenarios are so complex that the "optimal" decision cannot be found analytically. Approximate, heuristic or simplified global models must be used instead. One appropriate non-analytic way to find good solutions to complex problems where many decisions have to be made is through simulation.

In this work, we connect inputs and outputs of several data mining models and simulate the global outcome under different possibilities. Through the power of repeating simulations after simulations, we can gauge a global cutoff point in order to make better decisions for the global profit. It is important to highlight that this approach does not need that local models take the constraints into account during training (i.e. models can be trained and tested as usual). Additionally, we can use data which has been gathered independently for training each model. The only (mild) condition is that model predictions must be accompanied by probabilities (see e.g. [14]) or certainty values, something that almost any family of data mining algorithms can provide. Finally, probabilities and constraints will be used at the simulation stage for estimating the cutoff.

In order to do this, we use the basic Petri Nets formalism [24], with additional data structures, as a simple (but powerful) simulation framework and we use probabilistic estimation trees (classical decision trees accompanied with probabilities [14]). We illustrate this with a very frequent problem in CRM: we apply our approach to a direct-marketing campaign design where several alternative products have to be offered to the same house list of customers. The scenario is accompanied, as usual, by inventory/stock limitations. Even though this problem seems simple at the first sight, there is no simple good analytic solution for it. In fact, we will see that a

[1] (http://storm.cis.fordham.edu/~gweiss/ubdm-kdd05.html)

reasonable analytic approach to set different cutoffs for each product leads to suboptimal overall profits. In contrast, using a joint cutoff probabilistic estimation, which can be obtained through simulation, we get better results.

Section 3.2 sets the problem framework, some notation and illustrates the analytical (classical) approach. Section 3.3 addresses the problem with more than one product and presents two methods to solve it.

3.2 Campaign Design with One Product

Traditionally, data mining has been widely applied to improve the design of mailing campaigns in Customer Relationship Management (CRM). The idea is simple: discover the most promising customers using data mining techniques, and in this way, increase the benefits of a selling campaign.

The process begins by randomly selecting a sample of customers from the company database (house list). Next, all these customers receive an advertisement of the target product. After a reasonable time, a minable view is constructed with all these customers. In this table, every row represents a different customer and the columns contain information about customers; the predictive attribute (the target class) is a Boolean value that informs whether the corresponding customer has purchased or not the target product. Using this view as a training set, a probability estimation model is learned (for instance a probability estimation tree). This model is then used to rank the rest of customers of the database according to the probability of buying the target product. The last step is to select the optimal cutoff that maximises the overall benefits of the campaign, i.e. the best cutoff of the customer list ranked by customer buying probability.

The optimal cutoff can be computed using some additional information about some associated costs: the promotion material cost (edition costs and sending cost)($Icost$), the benefit from selling one product (b) and the cost to send an advertisement to a customer ($cost$). Given all this information, the *accumulated expected benefit* for a set of customers is computed as follows. Given a list C of customers, sorted by the expected benefit (for $c_k \in C, E\_benefit(c_k) = b \times p(c_k) - cost$), we calculate the *accumulated expected benefit* as $-Icost + \sum_{k=1}^{j} b \times p(c_k) - cost$, where $p(c_k)$ is the estimated probability that customer c_k buys the product and j is the size of the sample of customers to which a pre-campaign has offered the product. The optimal cutoff is determined by the value $k, 1 \leq k \leq j$ for which the greatest accumulated expected benefit is obtained.

The concordance between the real benefits with respect to the expected benefits is very dependent on the quality of the probability estimations of the model. Therefore, it is extremely important to train models that estimate accurate probabilities (e.g. see [14]). A more reliable estimation of the cutoff can be obtained by employing different datasets of customers (or by spliting the existing dataset): a training dataset for learning the probability estimation models, and a validation dataset to

compute the optimal cutoff. With this validation dataset the latter estimation of the *accumulated expected benefit* turns into a real calculation of the *accumulated benefit*, where $p(c_k)$ is changed by $f(c_k)$ in the formula, being $f(c_k)$ the response of c_k wrt. the product, such that $f(c_k) = 0$ if customer c_k does not buy the product and $f(c_k) = 1$ if c_k buys it. Then, the cutoff is determined by the greatest accumulated benefit.

Let us see an example where the benefit for the product is 200 monetary units (m.u.), the sending cost is 20 m.u. and the investment cost is 250 m.u. In Table 3.1 we compare the results obtained with each method. According to the *accumulated expected benefit* we will set the cutoff at 90% of the customers, which clearly differs from the maximum *accumulated benefit* (located at 70%).

Customer	Buys	Probability	E(Benefit)	Acc. Exp. Benefit	Acc. Benefit
				-250	-250
3	YES	0.8098	141.96	-108.04	-70
10	YES	0.7963	139.26	31.22	110
8	YES	0.6605	112.10	143.31	290
1	YES	0.6299	105.98	249.30	470
4	NO	0.5743	94.86	344.15	450
6	NO	0.5343	86.85	431.00	430
5	YES	0.4497	69.94	500.94	**610**
7	NO	0.2675	33.50	534.44	590
9	NO	0.2262	25.24	**559.68**	570
2	NO	0.0786	-4.29	555.39	550

Table 3.1: Accumulated expected benefit vs. Accumulated benefit.

3.3 Using Simulation for a Campaign Design with More than One Product

The approach shown at the previous section has been successfully applied to very simple cases (i.e. one single product for each campaign), but computing optimal cutoffs by analytic methods is impossible for more complex scenarios (more than one product, constraints for the products, etc.). Therefore, in this section we develop two different approaches: one is an extension of the analytic method, and the other is a more novel and original method based on simulation.

Back on our marketing problem, the objective now is to design a mailing campaign offering N products to a customer list, but taking the following constraints into consideration: there are stock limits (as usual), each product has a different benefit, and the products are alternative, which means that each customer would only buy one of them (or none). As we have seen at Section 3.2, a good solution, at least apriori, could be to determine a cutoff point defining the segment of customers we have to focus on. But now, since there are several products, it is not clear how

this cutoff can be defined/determined. Based on the idea of sorting the customers by their expected benefit, one possibility (what we call the *single approach*) is to combine (in some way, like adding or averaging) the optimal cutoffs which are analytically calculated for each product, in order to obtain a unique cutoff for the global problem.

An alternative method, that we call *joint simulation approach*, is to determine in a dynamic way the global cutoff. We use a validation set to simulate what will happen in a real situation if the customer receives the advertisement (of any of the N products).

Considering that all products have the same sending cost (*cost*), we define the following two alternative ways for obtaining a global cutoff using a validation set C:

1. **Single Approach**: For each product i, we downwardly sort C by the expected benefit of the customers, obtaining N ordered validation sets C_i (one for each product i). Now, for each C_i, $1 \leq i \leq N$, we determine its local cutoff point as we have explained in Section 3.2. Then, the global cutoff T is obtained by averaging the local cutoffs. In order to apply it, we now jointly sort the customers by their expected benefit considering all products at the same time (that is, just one ranked list obtained by merging the sets C_i). That produces as a result a single list SC where each customer appears N times. Finally, the cutoff T is applied over SC. Then, the real benefit obtained by this method will be the *accumulated benefit* for the segment of customers that will receive the advertisement for the total house list, which will be determined by this cutoff T.

2. **Joint Simulation Approach**: Here, from the beginning, we jointly sort the customers downwarded by their expected benefit of all the products, i.e. we merge the N sets C_i. However, we do not use local cutoffs to derive the global cutoff, but we calculate the cutoff by simulating $N \times |C|$ *accumulated benefits* considering all the possible cutoffs $T_j, 1 \leq j \leq N \times |C|$, where T_1 is the cutoff that only considers the first element of SC, T_2 is the cutoff that considers the two first elements of SC, and so on. Then, the best *accumulated benefit* gives the global cutoff.

To illustrate these two approaches consider a simple example consisting of 10 customers, 2 products (p_1 and p_2) and the parameters $Icost_{p_1} = 150$, $Icost_{p_2} = 250$, $b_1 = 100$, $b_2 = 200$, and $cost = 20$. Table 3.2 Left shows for each product the list of customers sorted by its expected benefit as well as the local cutoffs marked as horizontal lines. As we can observe, the cutoffs for products p_1 and p_2 are 90% and 70% respectively. Table 3.2 Right shows the global set and the global cutoff, which is marked by an horizontal line, computed by each approach. Note that the cutoff computed by the single and joint simulation methods is different. For the *single approach*, the global cutoff is 80% (the average of 90% and 70%), whereas the cutoff computed by the *joint simulation approach* is 90%.

Product p_1			
Customer	**E(Benefit)**	f_{p_1}	**Acc. Benefit**
			-150
2	76.61	1	-70
8	75.71	1	10
9	60.37	0	-10
5	48.19	1	70
1	44.96	1	150
7	30.96	0	130
10	24.58	1	210
3	23.04	0	190
6	7.81	1	270
4	-4.36	0	250

Product p_2			
Customer	**E(Benefit)**	f_{p_2}	**Acc. Benefit**
			-250
3	141.96	1	-70
10	139.26	1	110
8	112.10	1	290
1	105.98	1	470
4	94.86	0	450
6	86.85	0	430
5	69.94	1	610
7	33.50	0	590
9	25.24	0	570
2	-4.29	0	550

	Single & Joint Approaches			
	Customer	**Product**	**Acc. Benefit**	
			-400	
	3	p_2	-220	
	10	p_2	-40	
	8	p_2	140	
	1	p_2	320	
	4	p_2	300	
	6	p_2	280	
	2	p_1	360	
	8	p_1	440	
	5	p_2	620	
	9	p_1	600	
	5	p_1	680	
	1	p_1	760	
	7	p_2	740	
	7	p_1	720	
	9	p_2	700	
Single	10	p_1	780	
	3	p_1	760	
	6	p_1	840	Joint
	2	p_2	820	
	4	p_1	800	

Table 3.2: Left: Customers sorted by their expected benefit for the case of two products. Right: Customers and cutoff for the Single and Joint Simulation Approaches.

We have adopted Petri nets [24] as the framework to formalise the simulation. Petri nets are well-known, easy to understand, and flexible. Nonetheless, it is important to highlight that the method we propose can be implemented with any other discrete simulation formalism. We used a unique Petri net to simulate the behaviour of all the customers, but we also implemented additional data structures to maintain information about customers and products (e.g. remaining stock for each product, remaining purchases for each customer). The Petri net can work with as many products and customers as we need with no change in the Petri net structure. Other similar problems, as mailing campaigns with non-alternative products, can also be handled without changes. Figure 3.1 shows our Petri net which has 24 places and 10 transitions. Each customer arrives to the Petri net and, thanks to the additional data structures created, the suitable number of tokens are put in

each place to allow for the suitable transitions to be enabled/disabled and fired or not. E.g. if the remaining stock of the product is not zero a place *P12* is updated with as many tokens as the current stock is, and place *P11* is put to zero. The first place enables the transition *T4* that can be fired if the rest of conditions are fulfilled (place *P14* has a token), while the second place disables the transition *T5* that cannot be fired. Only two arcs have a weight not equal to one, the arc with the *product benefit* and the arc with the *sending cost*. The first arc finishes in the place *P1* (*Total gross benefit*) and the second one finishes in the place *P15* (*Total loss*). The total (or net) benefit for each cutoff is calculated subtracting the number of tokens accumulated in the places *P1* and *P15* (that is, *Total gross benefit* - *Total loss*).

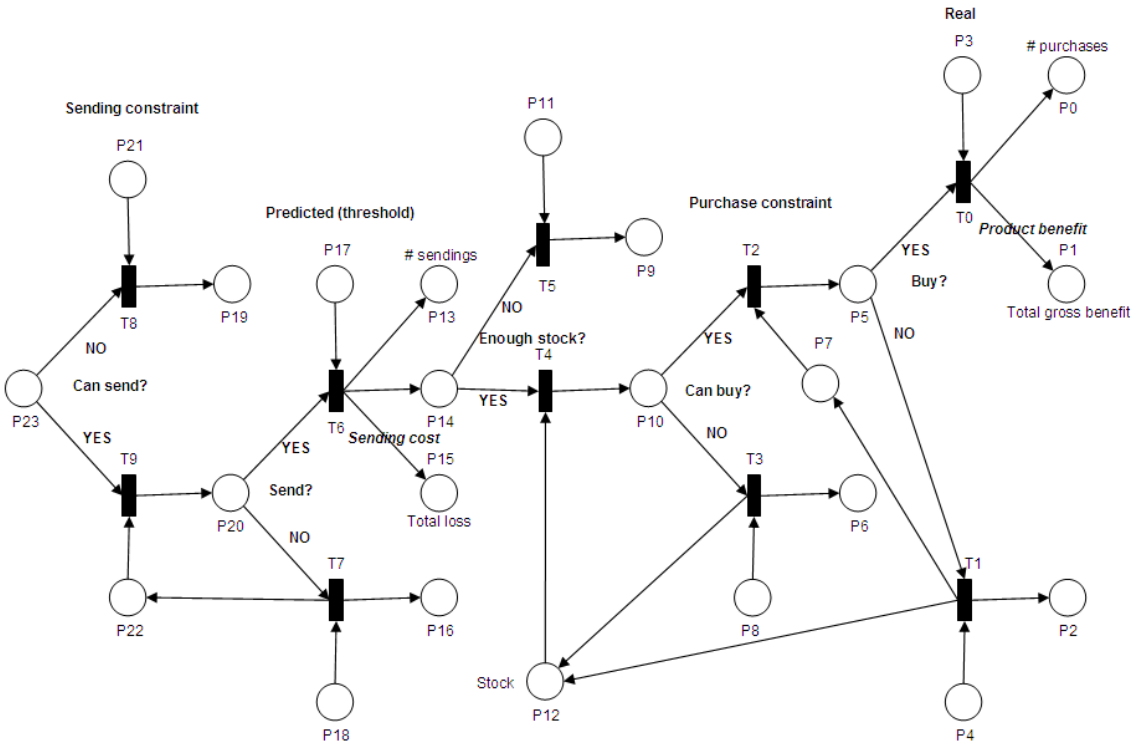

Figure 3.1: Petri net for our mailing campaign.

In this scenario, we consider that, at the most, only one of the N products can be bought since they are alternative products (e.g. several cars or several houses or different brands for the same product). This constraint suggests to offer to each customer only the product with the higher probability of being bought. If we impose this condition then we say that the approach is *with discarding*. In an approach with discarding, only the first appearance of each customer is taken into account. For instance, in the *single approach*, only the first occurrence of each customer in the customer segment determined by the global cutoff is preserved. Analogously, in the

joint simulation approach, the simulation process does not consider customers that have been already processed. However, since a prospective customer who receives an offer might finally not buy the product, we consider an alternative option which allows several offers to the same customer. This approach is called *without discarding*. The combination of the two approaches and the two options for considering customer repetitions give four scenarios that will be experimentally analysed in the following section. The notation used for referring to these four different methods is: *Single WO* (Single approach without discarding), *Single WI* (Single approach with discarding), *Joint WO* (Joint simulation approach without discarding), and *Joint WI* (Joint simulation approach with discarding).

3.4 Experiments of a Campaign Design with N Products

For this experimental evaluation, we have implemented the four methods explained at Section 3.3 and the Petri net in Java, and have used machine learning algorithms implemented in the data mining suite WEKA [30].

3.4.1 Experimental Settings

For these experiments we have taken a customers file (*newcustomersN.db*) from the SPSS Clementine[2] samples, as a reference. This file has information about only 200 customers, with 8 attributes for each one, 6 of them are nominal and the rest are numeric. The nominal attributes are the *sex* of the customers (male or female), *region* where they live (inner city, rural, town, suburban), whether they are *married*, whether they have *children*, whether they have a *car* and whether they have a *mortgage*. The numeric attributes are the *age* of the customers and their annual *income*.

Since 200 customers are too few for a realistic scenario, we have implemented a random generator of customers. It creates customers keeping the attribute distributions of the example file, i.e. for numeric attributes it generates a random number following a normal distribution with the same mean and deviation as in the example file, and for nominal attributes it generates a random number keeping the original frequency for each value of the attributes in the example file.

Also, to assign a class for each customer (wether s/he buys the product or not), we implemented a model generator. This model generator is based on a random decision tree generator, using the attributes and values randomly to construct the different levels of the tree. We have two parameters which gauge the average depth of the tree and most importantly, the probability of buying each product. We will use these latter parameters in the experiments below.

[2](http://www.spss.com/clementine/)

Therefore, the full process to generate a customer file for our experiments consists of generating the customer data with our random generator of customers and to assign the suitable class with a model obtained by our model generator.

Finally, these are the parameters we will consider and their possible values:

- Number of customers: 10000 (60% training, 20% validation and 20% testing)
- Number of products: 2, 3 and 4
- Probability of buying each product: 0.01, 0.05, 0.2, 0.5, 0.8, 0.95 or 0.99
- Benefits for each product: 100 monetary units (m.u.) for the product 1 and 100, 200, 500 or 1000 m.u. for the other products
- Sending cost (the same for all products): 10, 20, 50 or 90 m.u.
- Stock for each product: 0.1, 0.2, 0.5 or 1 (multiplied by number of customers)
- Investment cost for each product: benefits of the product multiplied by stock of the product and divided by 20
- Correlation (how similar the products are): 0.25, 0.5, 0.75 or 1

3.4.2 Experimental Results

The three main experiments consist in testing 100 times the four approaches for 2, 3 and 4 products, where all the parameters are selected randomly for the cases where there are several possible values.

If we look at overall results, i.e. averaging all the 100 experiments, as shown in Tables 3.3, 3.4 and 3.5, the results for 2, 3 and 4 products are consistent. As suggested in [9] we calculate a Friedman test and obtain that the four treatments do not have identical effects, so we calculate a post-hoc test (with a probability of 99.5%) This overall difference is clearly significant, as the significant analysis shown in Tables 3.3, 3.4 and 3.5, illustrates that the joint simulation approaches are better than the single ones. About the differences between with or without discarding methods, in the case of 2 products there are no significant differences. For 3 products the Single WI method wins the Single WO method, and the Joint WO method wins the Joint WI method. In the case of 4 products the approaches with discarding win the approaches without them. Moreover, in the case of 3 products, the Joint WO method clearly outperforms the other 3 methods and, in the case of 4 products is the Joint WI method which wins the rest of methods.

However, it is important to highlight that these values average many different situations and parameters, including some extreme cases where all the methods behave almost equally. This means that in the operating situations which are more frequent in real applications, the difference may be higher than the one reported by these overall results.

Moreover, in the case of 2 products, from the results of the 100 iterations we create three groups taking into account the probability of buying each product (probability of buying the product 1 is greater, equal or less than probability of buying the product 2) and 3 groups taking into account the stocks for the products (stock for the product 1 is greater, equal or less than stock for the product 2). The results obtained are shown in Figure 3.2. On one hand, the maximum benefit is obtained for all the methods and results are quite similar when the popularity (probability of buying) of both products is the same. On the other hand, the maximum benefit is obtained for all the methods and results are quite similar too when both products have the same stock. The results differ between the four methods especially when probabilities or stocks are different.

	2 products			
	S.WO	**S.WI**	**J.WO**	**J.WI**
Benefits	165626	164568	171225	169485
S.WO	-	=	√	√
S.WI	=	-	=	√
J.WO	X	=	-	=
J.WI	X	X	=	-

Table 3.3: Friedman test: wins (√) /loses (X)/draws(=). 2 products

	3 products			
	S.WO	**S.WI**	**J.WO**	**J.WI**
Benefits	182444	184077	186205	185694
S.WO	-	√	√	√
S.WI	X	-	√	=
J.WO	X	X	-	X
J.WI	X	=	√	-

Table 3.4: Friedman test: wins (√) /loses (X)/draws(=). 3 products.

	4 products			
	S.WO	**S.WI**	**J.WO**	**J.WI**
Benefits	220264	228483	231771	233724
S.WO	-	√	√	√
S.WI	X	-	=	√
J.WO	X	=	-	√
J.WI	X	X	X	-

Table 3.5: Friedman test: wins (√) /loses (X)/draws(=). 4 products

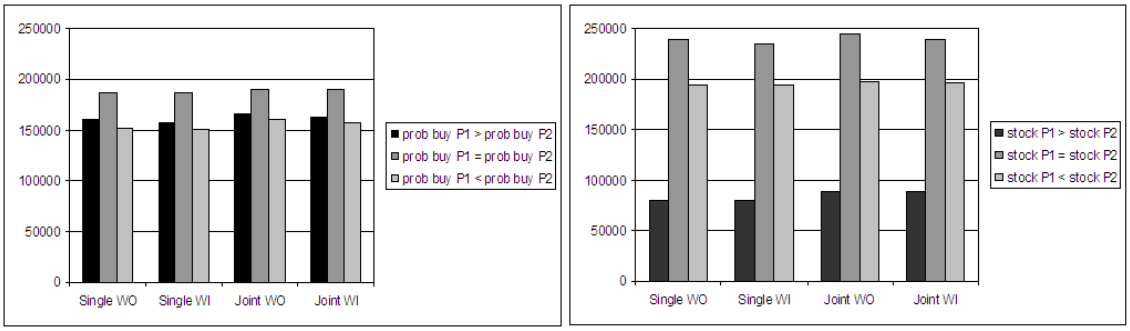

Figure 3.2: Left: Variations in probability of buying. Right: Variations in stocks.

4

Similarity-Binning Calibration Applied to Campaign Design with N Products

The main goal of this chapter is to apply our Similarity-Binning calibration method, described in Section 2.4, to a real problem. More concretely, we apply it to the campaign design with N products described in Section 3.3.

In Section 2.5 we have compared our Similarity-Binning calibration method with some well-known calibration methods, by means of standard data sets from the UCI repository [3], and we have seen how our calibration method outperforms the classical calibration methods. Next, we are going to apply it to the CRM problem explained before.

4.1 Experimental Results

The experimental settings used in these experiments are the same used in Section 3.4.1. The differences are that we have done some changes in some parameters to reduce a little the variability of the experiments. In particular, in Tables 4.1, 4.2 and 4.3 all the products have the same benefit (100 u.m.); in Tables 4.4, 4.5 and 4.6 each product has different benefit; and in Tables 4.7, 4.8 and 4.9 all the products have the same benefit and the probability of buying each product is also the same.

As suggested in [9], we have calculated the Wilcoxon Signed-Ranks test to compare the results of the pairs of methods without and within applying simulation (with a probability of 99.5%). The second column of the Tables 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 and 4.9 shows the benefit of the methods without applying simulation, while the fourth column shows the benefit of the methods applying our "Similarity-Binning" calibration method. The third column is the result of the Wilcoxon Signed-Ranks test, if there are no significant differences between the calibrated and non-calibrated methods the symbol = is used. If there are significant differences between them, the method with the highest benefit, logically, outperforms the another method. The symbol > is used if the benefit of the non-calibrated

method outperforms the calibrated one, and the symbol $<$ is used if the benefit of the calibrated method outperforms the non-calibrated one.

2 products The same benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	4181.7	=	5230.2
S.WI	9576	=	9973.4
J.WO	10074.1	>	9246.8
J.WI	10924.2	>	10796.1

Table 4.1: 2 products and the same benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

3 products The same benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	4881	=	8150.4
S.WI	24448.3	=	22817
J.WO	22562.3	=	21300.6
J.WI	28821.9	>	25754.7

Table 4.2: 3 products and the same benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

4 products The same benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	-5986.8	<	-5229.5
S.WI	10720.2	=	12731
J.WO	8445	<	9112.6
J.WI	13292.5	<	14959.5

Table 4.3: 4 products and the same benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

May be this CRM example is not the best scenario where to test if our calibration method works properly, because the experiments are very general and there is a lot of variance in the parameters, so, an exhaustive study of all the combinations is impossible. This situation implies that the results are not as better as they could be. But, apart from that, if we observe the results in detail we can extract some important conclusions. In the results with 2 and 3 products there are no significant differences between the non-calibrated and the calibrated methods, or the non-calibrated methods improve the calibrated ones; but, on the contrary, in

2 products Different benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	72557.9	>	67540.9
S.WI	75414	>	71403.2
J.WO	79998.4	>	76778.2
J.WI	79157.9	>	75399.2

Table 4.4: 2 products and different benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

3 products Different benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	239612.8	=	244149.2
S.WI	246399.3	=	249256.2
J.WO	275053.8	>	272517.6
J.WI	261307.3	=	260754.6

Table 4.5: 3 products and different benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

the results with 4 products there are no significant differences between the non-calibrated and the calibrated methods, or the calibrated methods improve the non-calibrated ones. So, in this scenario calibration achieves an improvement in the results when the number of products increase, i.e., when there is more related data mining models.

4 products Different benefit for each product			
	Benefit without calibration		Benefit with calibration
S.WO	296962.8	=	295840.6
S.WI	296129.5	=	299028.4
J.WO	331084.6	=	329987.1
J.WI	314403.4	=	314359.2

Table 4.6: 4 products and different benefit for each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

2 products The same benefit for each product and the same probability of buying			
	Benefit without calibration		Benefit with calibration
S.WO	16233	>	11854.9
S.WI	35356.7	>	29323.5
J.WO	43749.4	=	43914.7
J.WI	47903.3	=	46995.2

Table 4.7: 2 products, the same benefit for each product and the same probability of buying each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

3 products The same benefit for each product and the same probability of buying			
	Benefit without calibration		Benefit with calibration
S.WO	-2772.3	=	-5476.2
S.WI	35438	>	28303.3
J.WO	39639.6	=	40152
J.WI	46975.5	=	43925.7

Table 4.8: 3 products, the same benefit for each product and the same probability of buying each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

4 products The same benefit for each product and the same probability of buying			
	Benefit without calibration		Benefit with calibration
S.WO	-19083.5	=	-18511.5
S.WI	50273.3	=	50493
J.WO	40974.2	<	44858.3
J.WI	59281.1	<	63002.8

Table 4.9: 4 products, the same benefit for each product and the same probability of buying each product. Wilcoxon Signed-Ranks Test: There are no significant differences (=) / There are significant differences (> or <).

5

Conclusion

In this work, we have presented a new framework to address decision making problems where several data mining models have to be applied under several constraints and taking their mutual influence into account. The method is based on the conjunction of simulation with data mining models that estimate probabilities, and the adjustment of model cutoffs as a result of the simulation with a validation dataset. We have applied this framework to a direct marketing problem, and we have seen that simulation-based methods are better than classical analytical ones.

This specific direct marketing problem is just an example where our framework can be used. Almost any variation of a mailing campaign design problem could be solved (without stocks, with other constraints, non-alternative products, time delays, joint replies, etc.) in some cases with no changes in the presented Petri net and, in the worst case, by just modifying the Petri net that models the constraints and the relations between models. If not only the cutoff is to be determined but also the optimal stock or other important variables, then other techniques, such as evolutionary computation might be used to avoid a combinatorial explosion of the simulation cases. In our example, though, the combinations are not so huge to allow for an exhaustive analysis of all of them.

A general issue that arises in all of these global optimisation problems using data mining models is that a good combination highly depends on the models to be well calibrated. Although in the previous case, given the scenario, and also because of the use of unpruned decision trees with smoothed probabilities usually are initially well calibrated and do not require further calibration, in many other cases calibration will be neede to get good results from the integration.

Because of this, we have presented the most well-known calibration techniques and measures, it is important to remark that we have not presented only a state of the art, but we have established a taxonomy and a clarification of the calibration concept. Moreover, we have presented our new calibration method called Similarity-Binning averaging. We have generalised the idea of binning by constructing the bins using similarity to select the k-most similar instances. In this way, we have a different bin for each example and, of course, bins can overlap. Similarity is not computed by only using the estimated probabilities but our hypothesis was that calibration would be more effective the more information we are able to provide for computing this

similarity. Leaving the problem attributes out (as traditional calibration methods do) is like making the problem harder than it is.

The calibration experimental results we have presented in Section 2.5 confirm the previous hypothesis and show a significant increase in calibration for the two calibration measures considered, over three well-known and baseline calibration techniques: non-overlapping binning averaging, Platt's method and PAV. It is true that this calibration is mostly obtained because of the increase of AUC and it is, consequently, not monotonic, but in many applications where calibration is necessary the restriction of being monotonic is not only applicable, but it is an inconvenience. In fact, when calibrating a model, the original model and class assignments can be preserved, while the only thing that has to be modified is the new probabilities. In other words, the predictions of a comprehensible model composed of a dozen rules can be annotated by the estimated probabilities while preserving the comprehensibility of the original model.

Finally, we have applied our new calibration method to a campaign design with N products and overall in the case of 4 products, we have seen how the calibrated methods outperforms the non-calibrated methods, showing the importance of calibration when several local data mining problems are combined to obtain a good global result.

6

Future work

First of all, as "current work", we want to redo the experiments simplifying the parameters of the CRM problem or use a synthetic example to confirm, more clearly, the benefits of the calibration when we combined several data mining problems.

Next, following with the marketing problem, we are working in new scenarios, where the price of the products are not fix and the customer and the seller can negotiate. This situation is more complex, because the price of the product depends on the features of the customer.

Also, as future work, on one hand, out from marketing, we see prospective applicability in many other domains where we can apply simulation to solve multi-decision data mining problems. In particular, the ideas presented here were originated after a real problem we addressed recently in colaboration with a hospital, where resources and data mining models from different services were highly interwoven. Other domains which we are particular familiar with and we plan to use these ideas are the academic world (e.g. university), where we are using data mining models to predict the number of registered students per course each year, but until now we were not able to model the interdependencies between several courses.

On the other hand, to try to improve our Similarity-Binning calibration method, we are working on attribute-weighted k-NN to form the bins in order to gauge the importance of attributes for cases when there is a great number of attributes or a great number of classes. Similarly, we want to use locally-weighted k-NN, where closer examples have more weight, in order to make the method more independent from k. Another future work is the analysis of the method for multiclass problems. We have to find some other calibration methods to compare with, since binning, Platt's and PAV do not work with multiclass problems. A quite different future work is the use of repairing concavities techniques in ROC analysis [17] to solve conflicts between the original class ranking and the new estimated probabilities.

Bibliography

[1] M. Ayer, H.D. Brunk, G.M. Ewing, W.T. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.

[2] M. Berry and G. Linoff. *Mastering Data Mining: The Art and Science of Customer Relationship Management*. John Wiley & Sons, Inc., 1999.

[3] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[4] M. Carney and P. Cunningham. Making good probability estimates for regression. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou, editors, *ECML*, volume 4212 of *Lecture Notes in Computer Science*, pages 582–589. Springer, 2006.

[5] M. Carney, P. Cunningham, and B. M. Lucey. *Making Density Forecasting Models Statistically Consistent*. IIIS Discussion Paper Series, 2006.

[6] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proc. of the 10th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 69–78, 2004.

[7] B. Cestnik. Estimating probabilities: A crucial task in machine learning. pages 147–149, 1990.

[8] M. DeGroot and S. Fienberg. The comparison and evaluation of forecasters. *Statistician*, 31(1):12–22, 1982.

[9] J. Demsar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, January 2006.

[10] D. L. Dowe, G. E. Farr, A. J. Hurst, and K. L. Lentin. Information-theoretic football tipping. In *3rd Conf. on Maths and Computers in Sport*, volume 14, pages 233–241, 1996.

[11] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, and E. Vee. Comparing and aggregating rankings with ties. In *PODS '04: Proceedings of the 32nd symp. on Principles of database systems*, pages 47–58. ACM Press, 2004.

[12] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[13] T. Fawcett and A. Niculescu-Mizil. PAV and the ROC convex hull. *Machine Learning*, 68(1):97–106, 2007.

[14] C. Ferri, P. A. Flach, and J. Hernández-Orallo. Improving the AUC of probabilistic estimation trees. In *Machine Learning: ECML 2003, 14th European Conference on Machine Learning, Proceedings*, Lecture Notes in Computer Science, pages 121–132. Springer, 2003.

[15] C. Ferri, J. Hernández-Orallo, and R. Modroiu. An experimental comparison of performance measures for classification. *Pattern Recogn. Lett.*, 30(1):27–38, 2009.

[16] P. A. Flach and E. Takashi Matsubara. A simple lexicographic ranker and probability estimator. In *18th European Conference on Machine Learning*, pages 575–582. Springer, 2007.

[17] P.A. Flach and S. Wu. Repairing concavities in roc curves. In *Proc. 2003 UK Workshop on Computational Intelligence*, pages 38–44. University of Bristol, August 2003.

[18] G. Forman. Counting positives accurately despite inaccurate classification. In *ECML*, volume 3720 of *LNCS*, pages 564–575. Springer, 2005.

[19] J. Gama and P. Brazdil. Cascade generalization. *Machine Learning*, 41(3):315–343, 2000.

[20] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society, Series B*, 14:107–114, 1952.

[21] I. J. Good. Corroboration, explanation, evolving probability, simplicity, and a sharpened razor. *British Journal of the Philosophy of Science*, 19:123–143, 1968.

[22] N. Lachiche and P. A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using roc curves. In Tom Fawcett and Nina Mishra, editors, *ICML*, pages 416–423. AAAI Press, 2003.

[23] D. C. Montgomery, E. A. Peck, and G. G. Vining. *Introduction to Linear Regression Analysis*. Wiley-Interscience, 2007.

[24] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[25] A. H. Murphy. Scalar and vector partitions of the probability score: Part ii. n-state situation. *Journal of Applied Meteorology*, 11:1182–1192, 1972.

[26] D. B. O'Brien, M. R. Gupta, and R. M. Gray. Cost-sensitive multi-class classification from probability estimates. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 712–719, New York, NY, USA, 2008. ACM.

[27] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In P.J. Bartlett, B. Schölkopf, D. Schuurmans, and A.J. Smola, editors, *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, Boston, 1999.

[28] F. Provost and P. Domingos. *Well-trained PETs: Improving probability estimation trees.* (Technical Report CDER ♯00-04-IS). Stern School of Business, New York University, 2000.

[29] F. Sanders. On subjective probability forecasting. *Journal of Applied Meteorology*, 2(191-201), 1963.

[30] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations.* Elsevier, 2005.

[31] B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Proc. of the 18th Intl. Conference on Machine Learning*, pages 609–616, 2001.

[32] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *The 8th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 694–699. ACM, 2002.