On classification, ranking, and probability estimation

Peter Flach[1] and Edson Takashi Matsubara[2]

[1] Department of Computer Science, University of Bristol, United Kingdom
Peter.Flach@bristol.ac.uk
[2] Instituto de Ciências e Matemáticas e de Computação, Universidade de São Paulo, Brazil
edsontm@icmc.usp.br

Abstract. Given a binary classification task, a ranker is an algorithm that can sort a set of instances from highest to lowest expectation that the instance is positive. In contrast to a classifier, a ranker does not output class predictions – although it can be turned into a classifier with help of an additional procedure to split the ranked list into two. A straightforward way to compute rankings is to train a scoring classifier to assign numerical scores to instances, for example the predicted odds that an instance is positive. However, rankings can be computed without scores, as we demonstrate in this paper. We propose a lexicographic ranker, *LexRank*, whose rankings are derived not from scores, but from a simple ranking of attribute values obtained from the training data. Although various metrics can be used, we show that by using the odds ratio to rank the attribute values we obtain a ranker that is conceptually close to the naive Bayes classifier, in the sense that for every instance of *LexRank* there exists an instance of naive Bayes that achieves the same ranking. However, the reverse is not true, which means that *LexRank* is more biased than naive Bayes. We systematically develop the relationships and differences between classification, ranking, and probability estimation, which leads to a novel connection between the Brier score and ROC curves. Combining *LexRank* with isotonic regression, which derives probability estimates from the ROC convex hull, results in the lexicographic probability estimator *LexProb*.

Keywords. Ranking, probability estimation, ROC analysis, calibration

1 Introduction

ROC analysis is increasingly being employed in machine learning. It has brought with it a welcome shift in attention from classification to ranking. There are a number of reasons why it is desirable to have a good ranker, rather than a good classifier or a good probability estimator. One of the main reasons is that accuracy requires a fixed score threshold, whereas it may be desirable to change the threshold in response to changing class or cost distributions. Good accuracy obtained with one threshold does not imply good accuracy with another. Furthermore, good performance in both classification and probability estimation is easily and trivially obtained if one class is much more prevalent than the other, but this wouldn't be reflected in ranking performance.

In this paper we show that it is possible to have a good ranker *and* a good probability estimator. In fact, we show that, once we have a good ranker, there is a straightforward

Dagstuhl Seminar Proceedings 07161
Probabilistic, Logical and Relational Learning - A Further Synthesis
http://drops.dagstuhl.de/opus/volltexte/2008/1382

procedure to obtain calibrated probabilities from the ROC convex hull. Interestingly, the ranker we start from does not need to output probabilities or even scores. We propose a very simple non-scoring ranker, which is based on a linear preference ordering on attributes, which is then used lexicographically. In extensive experiments reported elsewhere [1] we demonstrate that both our lexicographic ranker and our lexicographic probability estimators perform comparably with much more sophisticated models.

The outline of the paper is as follows. In Section 2 we compare and contrast the notions of classification, ranking, and probability estimation. Section 3 is devoted to performance assessment in each of these cases. Section 4 defines lexicographic ranking and the *LexRank* algorithm. In Section 5 we uncover the fundamental relationship between ROC curves and the Brier score or mean squared error of the probability estimates. Section 6 defines the lexicographic probability estimator *LexProb* through a novel ROC-based calibration procedure. Section 7 concludes.

2 Classification, ranking, and probability estimation

Let $X = A_1 \times \ldots \times A_n$ be the instance space over the set of discrete attributes $A_1, \ldots, A_n$. A *classifier* is a mapping $\hat{c} : X \to C$, where C is a set of labels. It partitions the instance space into $c = |C|$ decision regions. For a binary classifier, $C = \{+, -\}$.

A *ranker* orders the instance space X, expressing an expectation that some instances are more likely to be positive than others. The ranking is a total order, possibly with ties. The latter are represented by an equivalence relation over X, so the total order is on those equivalence classes; we call them *segments* in this paper. For notational convenience we represent a ranker as a function $\hat{r} : X \times X \to \{>, =, <\}$, deciding for any pair of instances whether the first is more likely ($>$), equally likely ($=$), or less likely ($<$) to be positive than the second. (By a slight abuse of notation, we also use $>$ and $<$ for the total order on the segments of X). If $X_1, X_2 \subseteq X$ are segments such that $X_1 > X_2$, and there is no segment X_3 such that $X_1 > X_3 > X_2$, we say that X_1 and X_2 are *adjacent*.

We can turn a ranker into a binary classifier by selecting a pair of adjacent segments $X_1 > X_2$ and assigning any instance in X_1 or in any $X' > X_1$ to the positive class, and any instance in X_2 or in any $X'' < X_2$ to the negative class. Furthermore, given a ranker $\hat{r}$, we can construct another ranker $\hat{r}'$ by joining two adjacent segments X_1 and X_2, and removing $X_1 > X_2$ from the total order. We say that $\hat{r}'$ is *coarser* than $\hat{r}$, or equivalently, that the latter is *finer* than the former. This induces a partial order on the set of all rankers, with the trivial ranker (which maps all instances to the same segment, i.e., doesn't express any preferences) as coarsest element, and all rankers without ties as finest elements.

A *scoring classifier* is a mapping $\hat{s} : X \to \mathbb{R}$, assigning a numerical score $\hat{s}(x)$ to each instance x. We will use the convention that higher scores express more preference for the positive class. A *probability estimator* is a scoring classifier that assigns probabilities, i.e., a mapping $\hat{p} : X \to [0,1]$. $\hat{p}(x)$ is taken to be an estimate of the posterior $p(+|x)$, i.e., the true probability that a random instance with attribute-value vector x belongs to the positive class.

Clearly, given a scoring classifier $\hat{s}$ (or a probability estimator) we can construct a ranker $\hat{r}$ as follows:

$$\hat{r}(x_1,x_2) = \begin{cases} > \text{ if } \hat{s}(x_1) > \hat{s}(x_2) \\ = \text{ if } \hat{s}(x_1) = \hat{s}(x_2) \\ < \text{ if } \hat{s}(x_1) < \hat{s}(x_2) \end{cases}$$

Furthermore, we can turn a scoring classifier into a classifier by turning the associated ranker into a classifier as described above, or equivalently, by setting a threshold $t \in \mathbb{R}$ and assigning all instances x such that $\hat{s}(x) \geq t$ to the positive class and the remaining instances to the negative class.

Example 1. [2] and [3] showed that decision trees can be used as probability estimators and hence as rankers. For a given (unlabelled) tree, different rankers can be obtained by imposing an ordering on the leaves. Once an ordering is fixed, we obtain a classifier by labelling the first k^+ leaves in the ordering positive and the remaining k^- leaves negative. We obtain a probability estimator by considering the numbers of positive (n_i^+) and negative (n_i^-) training examples belonging to the i-th leaf. The estimated posterior odds in $leaf_i$ is then $\frac{P(+|leaf_i)}{P(-|leaf_i)} = \frac{n_i^+}{n_i^-}$ (or $\frac{n_i^+ + 1}{n_i^- + 1}$ if we apply Laplace correction, as recommended by [3]). Figure 1 shows a small example data set, and a decision tree induced from it. The ranking obtained from the training set is leaf 1 – leaf 4 – leaf 2 – leaf 3.

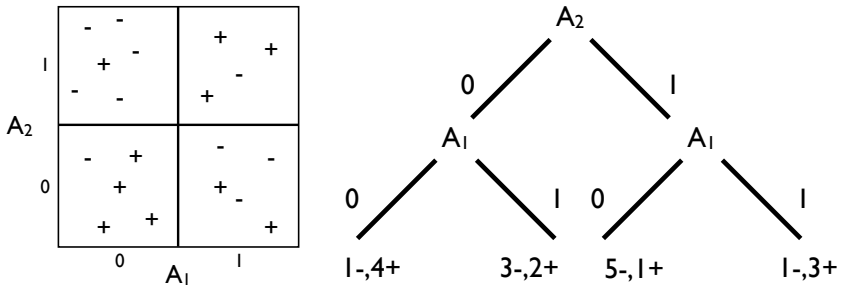

Fig. 1. A data set, and an induced decision tree. Instead of leaf labellings, the class distributions of training instances are indicated for each leaf.

3 Assessing classifiers, rankers, and probability estimators

The performance of a binary classifier can be assessed by tabulating its predictions on a test set with known labels in a *contingency table* or confusion matrix, with actual classes in rows and predicted classes in columns. The most important quantities assessing classification performance are *true positive rate*: the proportion of correctly labelled positives; *false positive rate:* the proportion of incorrectly labelled negatives; *accuracy*: the proportion of correctly labelled examples. An *ROC plot* plots true positive rate on

the Y-axis against false positive rate on the X-axis; a single contingency table corresponds to a single point in an ROC plot.

The performance of a ranker can be assessed by drawing a piecewise linear curve in an ROC plot, known as an *ROC curve*. The curve starts in $(0,0)$, finishes in $(1,1)$, and is monotonically non-decreasing in both axes. Each segment of the curve corresponds to one of the segments induced by the ranker; the order of the segments corresponds to the total ordering on the segments. If the i-th segment contains n_i^+ out of a total of n^+ positives and n_i^- out of n^- negatives, the segment's vertical length is n_i^+/n^+, its horizontal width is n_i^-/n^- and its slope is $l_i = \frac{n_i^+}{n_i^-}c^{-1}$, where $c = n^+/n^-$ is the prior odds. We will denote the proportion of positives in a segment as $p_i = \frac{n_i^+}{n_i^+ + n_i^-} = \frac{l_i}{l_i + 1/c}$. We will call these *empirical probabilities*; they allow us to turn a ranker into a probability estimator , as we will show later. Notice that, since c is constant, p_i increases monotonically with l_i.

Each point on an ROC curve connecting two segments corresponds to the true and false positive rates achieved on the same test set by the classifier obtained from the ranker by splitting the ranking between those two segments. Points on a segment correspond to a classifier that assigns a random part of the instances in that segment to the positive class and the rest to the negative class. The area under the ROC curve or *AUC* estimates the probability that a randomly selected positive is ranked before a randomly selected negative, and is a widely used measure of ranking performance, equivalent to the Wilcoxon-Mann-Whitney sum-of-ranks test to decide whether two samples are drawn from different distributions [4]. It can be calculated as $\frac{1}{n^+n^-}\sum_i n_i^-(n_i^+/2 + \sum_{j>i} n_j^+)$, which adds up, for each negative, all positives preceding it in the ranking (ties count for 1/2).

An ROC curve is *convex* if the slopes l_i are monotonically non-increasing when moving along the curve from $(0,0)$ to $(1,1)$. A concavity in an ROC curve, i.e., two or more adjacent segments with increasing slopes, indicates a locally worse than random ranking. In this case, we would get better ranking performance by joining the segments involved in the concavity, thus creating a coarser classifier.

Example 2. Evaluating the decision tree from Example 1 on the training set results in the ROC curve given in Figure 2. Notice that the curve has as many segments as the tree has leaves. The non-optimal labellings of the tree are indicated as well. Since the predicted probabilities of a decision tree are exactly the empirical probabilities, its training set ROC curve is always convex.

The performance of a scoring classifier can be assessed in the same way as a ranker. Alternatively, if we know the true scores $s(x)$ we can calculate a loss function such as *mean squared error* $\frac{1}{|T|}\sum_{x\in T}(\hat{s}(x) - s(x))^2$, where T is the test set. In particular, for a probabilistic classifier we may take $s(x) = 1$ for a positive instance and $s(x) = 0$ for a negative; in that case, mean squared error is also known as the *Brier score* [5]. Other loss functions are possible, but in this paper we use the Brier score as it allows an extremely useful connection with ROC curves through its decomposition into calibration and refinement, as we will now show. Note that the Brier score takes probability estimates into account but ignores the rankings (it does not require sorting the estimates).

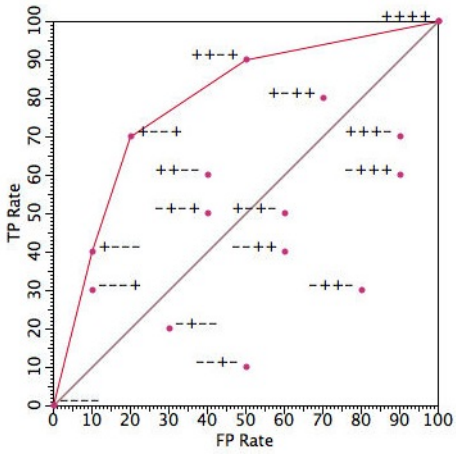

Fig. 2. Training set ROC curve of the decision tree from Figure 1.

Conversely, ROC curves take rankings into account but ignore the probability estimates. Brier score and AUC thus measure different things and are not directly comparable.

4 Lexicographic ranking with *LexRank*

While a ranker is commonly obtained by sorting the scores of a scoring classifier as indicated in Section 2, it is possible to define a ranker without scores. A very simple way to do so – probably the simplest – is to assume a preference order on attribute values, and to use that ordering to rank instances lexicographically. Experimental results indicate that such a simple ranker can perform competitively with more sophisticated models such as decision trees and naive Bayes. Moreover, it can be turned into an accurate and calibrated probability estimator by using a simple probability assignment derived from the ranker's ROC curve, as we show in Section 6.

For notational convenience we will assume that all attributes are binary; since a nominal attribute with k values can be converted into k binary attributes, this doesn't represent a loss of generality.

Definition 1 (Lexicographic ranking). *Let $A_1,\ldots,A_n$ be a set of boolean attributes, such that the index represents a preference order. Let v_{i+} denote the preferred value of attribute A_i. The* lexicographic ranker *corresponding to the preference order on attributes and attribute values is defined as follows:*

$$\hat{r}_{lex}(x_1,x_2) = \begin{cases} > \text{ if } A_j(x_1) = v_{j+} \\ < \text{ if } A_j(x_1) \neq v_{j+} \end{cases}$$

where j denotes the lowest attribute index for which x_1 and x_2 have different values (if no such index exists, the two instances are tied).

A lexicographic ranker can be represented as an unlabelled binary decision tree with the following properties: (1) the only attribute occurring at depth i is A_i – i.e., along each path from root to leaf the attributes occur in the preference order; (2) in each split, v_{i+} is the left branch. Consequently, the ranking order is represented by the left-to-right order of the leaves. We call such a tree a *lexicographic ranking tree*.

Example 3. Figure 3 shows the lexicographic ranking tree obtained if A_2 is preferred to A_1, 1 is the preferred value of A_1, and 0 the preferred value of A_2. .

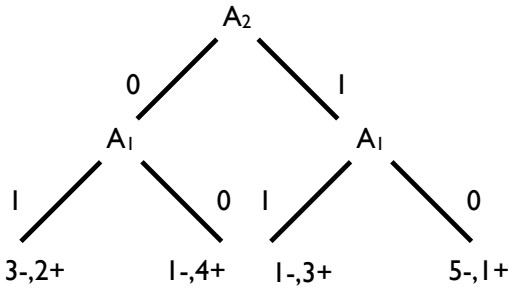

Fig. 3. A lexicographic ranker, represented as a decision tree. The class distributions in the leaves correspond to the data from Example 1, and are not part of the ranker.

Obviously the size of a lexicographic ranking tree is exponential in the number of attributes, and therefore not practical as a representation of a lexicographic ranker. We use it here to demonstrate that every lexicographic ranker corresponds to a decision tree ranker. The converse is, however, not true. This can be seen by noting that the ROC curve corresponding to the data from Example 1 is not convex (this is immediate from the leaf class distributions in Figure 3). For this data, none of the eight possible lexicographic rankers has a convex ROC curve.

A similar, but tighter connection can be made with the naive Bayes classifier, as we will now show.

Example 4. The naive Bayes ranker obtained from the data from Example 1 is as follows. We have $LR(A_1 = 0) = \frac{p(A_1=0|+)}{p(A_1=0|-)} = 5/6$, $LR(A_1 = 1) = \frac{p(A_1=1|+)}{p(A_1=1|-)} = 5/4$, $LR(A_2 = 0) = \frac{p(A_2=0|+)}{p(A_2=0|-)} = 6/4$, and $LR(A_2 = 1) = \frac{p(A_2=1|+)}{p(A_2=1|-)} = 4/6$. The prior odds doesn't affect the ranking, and so we can just use the products of these marginal likelihood ratios to determine the ranking: $LR(A_1 = 1)LR(A2 = 0) = 30/16 > LR(A_1 = 0)LR(A2 = 0) = 30/24 > LR(A_1 = 1)LR(A2 = 1) = 20/24 > LR(A_1 = 0)LR(A2 = 1) = 20/36$. This is a lexicographic ranking, which is equivalent to the lexicographic ranking tree in Figure 3.

We can further tighten this connection between lexicographic ranking and naive Bayes by using the odds ratio as the preference criterion for attributes.

Definition 2. *LexRank is the lexicographic ranker which uses the following preference criteria. The preferred value v_{i+} for attribute A_i is defined as the one which has $LR(A_i = v_{i+}) > 1$ (if there is no such value then the attribute doesn't express preference and can be discarded). The preference order on attributes is defined by sorting them on decreasing* odds ratio $OR(A_i) = \frac{LR(A_i=v_{i+})}{LR(A_i=v_{i-})}$*, where v_{i-} denotes the value of the non-preferred value.*[3]

The odds ratio can be interpreted as a decision tree splitting criterion. Its ROC isometrics [6] are shown in Figure 4.

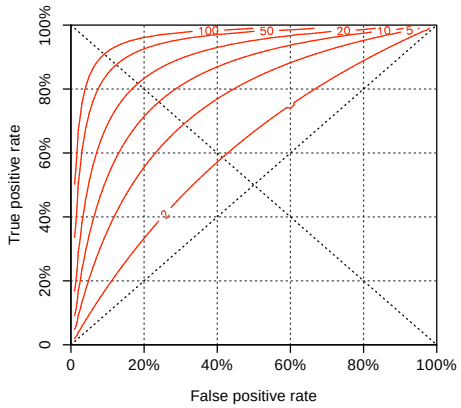

Fig. 4. ROC isometrics for the odds-ratio splitting criterion.

It is possible to show that for two binary attributes *LexRank* always produces the same ranker as naive Bayes. This is due to the fact that for binary attributes naive Bayes is a linear classifier [7], and the coefficients of the decision hyperplane are the logs of the odds ratio of each attribute. However, for more than two attributes there are naive Bayes models that cannot be represented by a preference order on attributes. We conclude that *LexRank* exhibits a stronger bias than naive Bayes, but experiments show this added bias does not result in a loss of ranking performance. The extreme simplicity of *LexRank*, which only involves a ranking of attributes and no probability calculations, makes it therefore very attractive.

[3] Strictly speaking, $OR(A_i)$ is not a ratio of odds but a ratio of likelihood ratios, but since the prior odds does not affect ranking the distinction between likelihood ratio and posterior odds can be conveniently ignored.

5 Brier score and ROC curves

In this section we demonstrate a fundamental and novel relationship between Brier score and ROC curves. We do this by means of a decomposition of the Brier score in terms of calibration loss and refinement loss. A very similar decomposition is well-known in forecasting theory (see, e.g., [8]), but requires a discretisation of the probability estimates and is therefore approximate. Our decomposition uses the segments induced by the ranking and is therefore exact. More importantly, our analysis provides a straightforward way to turn a ranker into a probability estimator using only the ranker's ROC curve.

Theorem 1. [4] *Given an ROC curve produced by a ranker on a test set T, let* n_i^+ *and* n_i^- *be the number of positives and negatives in the i-th segment of the ROC curve,* $n_i = n_i^+ + n_i^-$, $p_i = \frac{n_i^+}{n_i}$, *and* $\hat{p}_i$ *be the predicted probability in that segment. The Brier score is equal to*

$$BS = \frac{1}{|T|}\sum_i n_i(\hat{p}_i - p_i)^2 + \frac{1}{|T|}\sum_i n_i p_i(1-p_i)$$

Both of these terms are a weighted average over all segments of the ROC curve. The first term, the *calibration loss*, is a weighted average of the squared prediction error in each segment. It is important to note that the error is taken relative to p_i, which is the proportion of positives in the segment *and thus not necessarily 0 or 1*. In other words, the calibration loss as defined above relates the predicted probabilities to the empirical probabilities that can be obtained from the slopes of the segments of the ROC curve.

The second term in the Brier score decomposition is called *refinement loss*. This term is 0 if and only if all ROC segments are either horizontal or vertical, which is the case if all segments are singletons (or, more precisely, if all segments involve only examples of a single class). Consequently, refinement loss measures the coarseness of the ranker, hence its name. For instance, refinement loss is maximal (0.25) for the ranker which ties all test instances. Notice that refinement loss only takes empirical probabilities into account, not predicted probabilities. It is therefore a quantity that can be evaluated for any ranker, not just for probability estimators. Notice also that, while the Brier score itself does not require ranking the probability estimates, its decomposition into calibration loss and refinement loss does.

Example 5. The decision tree from Example 1 has 0 calibration loss on the training set (if Laplace correction is not used) and refinement loss $(5\cdot 4/5\cdot 1/5 + 4\cdot 3/4\cdot 1/4 + 5\cdot 2/5\cdot 3/5 + 6\cdot 1/6\cdot 5/6)/20 = 0.18$

We then have the following simple but important results.

Theorem 2. *The calibration loss is 0 only if the ROC curve is convex.*

A convex ROC curve is not a sufficient condition for perfectly calibrated probabilities. The reason is that the scores may be in the right order but some error terms may still be non-zero. However, simply setting the predicted probabilities equal to the empirical probabilities in each segment will always decrease the Brier score.

[4] Proofs of the theorems are given in [1].

Theorem 3. *Let $\hat{p}$ be a probability estimator with a convex ROC curve but a non-zero calibration loss. Let $\hat{p}'$ be derived from $\hat{p}$ by predicting p_i rather than $\hat{p}_i$ in each segment. Then $\hat{p}'$ has the same AUC as $\hat{p}$ but a lower Brier score.*

6 Probability estimation with *LexProb*

Even though *LexRank* doesn't obtain its ranks from scores, we can turn it into a calibrated probability estimator by employing the insights from the previous section. According to Theorem 3, given a convex ROC curve we can obtain calibrated probability estimates from the empirical probabilities derived from the slopes of the segments of the curves. But, unlike decision trees, lexicographic rankers cannot be guaranteed to have convex ROC curves on the training set. The answer is to use only part of the ranking, by constructing the *convex hull* of the ROC curve [9].

This can be understood as creating a coarser ranking, by joining adjacent segments that are in the wrong order[5]. Clearly, joining segments results in additional refinement loss (unless the two segments have the same slope), but this is compensated by setting the probability estimates equal to the empirical probabilities, hence obtaining zero calibration loss (in practice, we don't achieve zero calibration loss because we apply the Laplace correction in order to avoid overfitting).

Definition 3. *Given a training set, the* lexicographic probability estimator *LexProb is defined as follows:*

1. *let $\hat{r}_{lex}$ be the ranker constructed by LexRank from the training set; ;*
2. *construct $\hat{r}_{lex}$'s ROC curve on the training set;*
3. *construct the convex hull of the ROC curve;*
4. *let $\hat{r}'_{lex}$ be the corresponding coarser version of $\hat{r}_{lex}$;*
5. *let $\hat{p}_{lex}$ be the probability estimator that assigns to each segment of $\hat{r}'_{lex}$ the empirical probability obtained from the slope of the ROC convex hull, smoothed with Laplace correction.*

Notice that this procedure can be applied (from step 2) with any ranker, not just with a lexicographic one. As a calibration procedure it can be shown to be equivalent to isotonic regression [10]; a proof can be found in [11].

7 Conclusions

In this paper we have investigated the relationship and differences between classification, ranking and probability estimation. Machine learning has for (too) long concentrated on classification alone, and the relationships between the three are not yet widely understood, but we believe our analysis is a step in that direction. We have furthermore introduced the notion of lexicographic ranking, which simply employs a linear preference order on attributes. To the best of our knowledge, this is the first ranker that doesn't base its ranking on numerical scores. Using the odds ratio for ranking attributes results

[5] It can also be understood as pruning the lexicographic ranking tree.

in a lexicographic ranker, called *LexRank*, which is close to naive Bayes. Finally, we have demonstrated a close and fundamental connection between ROC curves and the Brier score, linking in particular the calibration of probability estimates to the convexity of the ROC curve. This immediately suggests a simple calibration procedure that first constructs the ROC convex hull, which is in effect a discretisation of the scores, followed by assigning the empirical probability derived from the ROC segment to each bin. The fact that the ROC convex hull can be used for probability calibration was discovered independently in [11], who show that the procedure is in fact equivalent to isotonic regression. Here, we used it to obtain a calibrated lexicographic probability estimator called *LexProb*.

The advantage of lexicographic ranking and probability estimation is its extreme simplicity and robustness. It is remarkable that such a simple models holds its own against much more sophisticated models.

References

1. Flach, P., Matsubara, E.T.: A simple lexicographic ranker and probability estimator. In: Proceedings of the 18th European Conference on Machine Learning (ECML 2007), Springer (2007)
2. Ferri, C., Flach, P.A., Hernández-Orallo, J.: Learning decision trees using the area under the ROC curve. In Sammut, C., Hoffmann, A.G., eds.: Proceedings of the Nineteenth International Conference (ICML 2002), Morgan Kaufmann (2002) 139–146
3. Provost, F., Domingos, P.: Tree induction for probability-based ranking. Machine Learning **52**(3) (2003) 199–215
4. Hand, D.J., Till, R.J.: A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning **45**(2) (2001) 171–186
5. Brier, G.: Verification of forecasts expressed in terms of probabilities. Monthly Weather Review **78** (1950) 1–3
6. Flach, P.: The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In: Proceedings of the Twentieth International Conference on Machine Learning (ICML 2003). (2003) 194–201
7. Ling, C.X., Zhang, H.: The representational power of discrete Bayesian networks. Journal of Machine Learning Research **3** (2002) 709–721
8. Cohen, I., Goldszmidt, M.: Properties and benefits of calibrated classifiers. In Boulicaut, J.F., Esposito, F., Giannotti, F., Pedreschi, D., eds.: Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'04). Volume 3202 of Lecture Notes in Computer Science., Springer (2004) 125–136
9. Provost, F., Fawcett, T.: Robust classification for imprecise environments. Machine Learning **42**(3) (2001) 203–231
10. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In Brodley, C.E., Danyluk, A.P., eds.: Proceedings of the Eigteenth International Conference on Machine Learning (ICML 2001), Morgan Kaufmann (2001) 609–616
11. Fawcett, T., Niculescu-Mizil, A.: PAV and the ROC convex hull. Machine Learning **68**(1) (2007) 97–106