ROC-tree: A Novel Decision Tree Induction Algorithm Based on Receiver Operating Characteristics to Classify Gene Expression Data

M. Maruf Hossain*† Md. Rafiul Hassan† James Bailey†

Abstract

Gene expression information from microarray experiments is a primary form of data for biological analysis and can offer insights into disease processes and cellular behaviour. Such datasets are particularly challenging to build classifiers for, due to their very high dimensional nature and small sample size. Decision trees are a seemingly attractive technique for this domain, due to their easily interpretable white box nature and noise resistance. However, existing decision tree methods tend to perform rather poorly for classifying gene expression data. To address this gap, we introduce a new technique for building decision trees that is better suited to this scenario. Our method is based on consideration of the area under the Receiver Operating Characteristics (ROC) curve, to help determine decision tree characteristics, such as node selection and stopping criteria. We experimentally compare our algorithm, called *ROC-tree*, against other well known decision tree techniques, on a number of gene expression datasets. The experimental results clearly demonstrate that *ROC-tree* can deliver better classification accuracy in a range of challenging situations.

Key words: Receiver Operating Characteristics (ROC), Decision Tree, Gene Expression, DNA Microarray, Classification, High Dimensional Dataset

1 Introduction

Gene expression measurements from DNA microarray [18] experiments provide information gleaned from tissue and cell samples that may be crucial to disease diagnoses and distinguish specific tumour types. Clinical decision support [39] is an emerging medical application domain for microarray gene expression technology; and is used to diagnose diseases, predict clinical outcomes and responses to treatment. However, gene expression data is particularly challenging to build classifiers for, since it typically has thousands of features, but only tens or at most hundreds of instances. There is a pressing need for simple and easily interpretable classification tools that can operate on high dimensional gene expression datasets with good accuracy.

Decision trees are a well known and classic classification technique [6], [31]. Moreover, they are self-explanatory, easily understood, interpreted and inferred from [6], and have been successfully used for classification in diverse areas such as radar signal classification, character recognition, remote sensing, medical diagnosis, expert systems, and speech recognition. Decision trees require little data preparation in terms of parameter settings, and are well suited to exploratory knowledge discovery. They can handle both numerical and categorical data. Also, their representation allows the illustration of any discrete value classifier [35]. Their "white box" nature means that it is possible to validate a model with statistical tests and for a human expert to interpret it. Consequently, the reliability of the model can be expressed and accounted for. Decision trees can handle datasets that may have errors or missing values. The model is typically robust and can handle large amounts of data in a relatively short time [16]. The most important characteristics of decision tree classifiers are their efficacy to divide a complex decision-making process into a number of simpler decisions, and thus provide a solution which is easily interpretable.

However, although decision tree classifiers have a lot of benefits, they tend to suffer from lower predictive power for gene expression microarray datasets. Therefore, even though several works applying them exist in the microarray literature (e.g., [3] – [41]), this technique is currently not so popular in the gene expression microarray domain [24], because there is a noticeable gap between their prediction performance, compared to that of other techniques such as Support Vector Machines (SVMs) [30] or even k-Nearest Neighbour (k-NN) [1].

EXAMPLE 1.1. *In the ALL-AML leukaemia gene expression dataset, C4.5 [33] has an average accuracy of only* $79.17 \pm 4.87\%$ *using* 10×10*-fold cross validation, whereas SVM and k-NN yield an average accuracy of* $98.61 \pm 1.26\%$ *and* $83.33 \pm 3.49\%$*, respectively. This lower accuracy of the decision tree classifier can discourage biologists to use it, even though the results are easy to interpret and can provide a clearer understanding of which genes are involved in the classification process and which combinations of genes can separate the classes.*

Such an example highlights the prediction accuracy gap that exists when using a decision tree classifier technique for gene expression microarray data. *Our aim in this paper is to improve the classification performance*

*Corresponding author {hossain@csse.unimelb.edu.au}
†Department of Computer Science and Software Engineering, The University of Melbourne, Australia.

of decision trees in this gene expression microarray domain. Although we will use comparison to the classification performance of support vector machines as one benchmark, our expectation is not to outperform this technique. *Rather, our belief is that a decision tree classifier which is only marginally less accurate than a support vector is in fact preferable for biologists to use, due to its superior interpretability.*

In this paper, we introduce a novel binary decision tree induction algorithm to classify patients from gene expression datasets. This has substantially better accuracy than existing decision tree induction techniques for such data. The proposed tree is based on consideration of the area under the Receiver Operating Characteristics (ROC) curve [20]. This value is equivalent of the Mann-Whitney U statistic [2] normalized by the number of possible pairings of positive and negative values, also known as the two sample Wilcoxon rank-sum statistic [21]. The area under the ROC curve (AUC) actually represents the probability that a randomly chosen positive example is correctly rated (ranked) with greater suspicion than a randomly chosen negative example. Moreover, this probability of correct ranking is the same quantity estimated by the non-parametric Wilcoxon statistic [4].

Deng *et al.* [10] demonstrated that the assumption that the gene expression data follows the Gaussian (normal) distribution often does not hold true in real world data. Therefore, a non-parametric test like Wilcoxon rank sum is statistically a better way to identify discriminative genes than parametric tests like t-test. Apart from establishing the suitability of non-parametric tests, they also showed that using the significance level threshold, instead of an arbitrary number of discriminative genes as the parameter carries the quality specification in statistics.

Traditionally, in data mining and machine learning, the ROC curve is used to test the quality of binary classifiers [12], [40]. Mamitsuka [26] used the AUC to rank genes by treating each gene as a single feature classifier. In this study, we generalise and adapt Mamitsuka's ranking technique for use in decision tree node selection. In short, we develop an algorithm, *ROC-tree*, to induce a binary decision tree to classify patients from gene expression data.

The intuitive outline of the technique is as follows:

REMARK 1.1. *For a given gene expression dataset $\mathcal{D}$ of n examples comprising m attributes: $x_1, x_2, x_3, \ldots, x_m$, each gene x_i (where $1 \leqslant i \leqslant m$) has some discriminative power, i.e., the influence of each gene on the classification accuracy can be measured. The ROC curve is plotted for each of the pairs formed by the "classifier" gene x_i and the corresponding class label Y_i. A node of the decision tree is determined by selecting the attribute with the highest AUC. It should be noted that the ROC-tree can be relatively robust to the small number of samples found in gene expression datasets, since in statistical terms, it does not employ any distributional assumptions.*

Challenges: For most classification techniques examined in the literature, the accuracy greatly suffers if the number of examples used for training is much lower than the number of attributes. Furthermore, having a large number of attributes, as is the case in most gene expression datasets, poses a great challenge for decision trees, since the data can be highly noisy and traditional measures such as information gain [33] or gini impurity [6] often lead to erroneous decisions.

Issues Related to Induction of Decision Trees: Inducing a decision tree using a top-down approach requires dealing with three other issues apart from selecting the best attribute to use at each node in the tree. Firstly, one has to choose a splitting threshold to form the two children for each node. This choice of splitting criterion affects the quality of the decision tree and has been the subject of considerable research. However, it is extremely important to split the dataset in a way so that uncertainty can be reduced maximally, i.e., most of the similar classes fall under the same branch [28]. Secondly, one needs a criterion to determine when to stop growing the tree. To avoid difficulties in choosing a stopping rule, most decision tree induction algorithms grow the tree to its maximum size where the terminal nodes are pure or almost pure, and then selectively prune the tree. Thirdly, the final issue is how to decide what class label to assign for the terminal (leaf) nodes.

Related Work: There exist numerous algorithms to construct decision trees. There have been several approaches to deal with large datasets. Catlett [9] explored two methods that efficiently induce decision trees from high dimensional datasets by reducing the computational complexity required for induction. Fifield [14] proposed a parallel implementation of the ID3 algorithm.

Ferri *et al.* [13] used area under the ROC curve (AUC) as a quality metric to choose nodes in a decision tree. They proposed a method that selects a *feature and split point* based on the AUC corresponding to a classifier for every potential class labelling for the induced child nodes. This is rather different to the technique we will present, which uses an AUC measure to select a node based on its classification performance and then uses the misclassification rate to choose a split point. Also, their technique grows the tree to its maximum size until the terminal nodes are pure or almost pure,

and then prunes the tree using "Pessimistic Error Pruning" [32], whereas, we use a stopping criterion based on AUC. We will later make a comparison between this technique and ours in the experiments.

Ma and Huang [25] proposed using the binormal AUC as the objective function for two-sample classification, and the threshold gradient directed regularization method for regularized estimation and biomarker selection. They also developed Monte Carlo based methods for evaluating the stability and prediction performance of the proposed estimator and individual biomarkers (i.e., genes).

Statnikov *et al.* [37] aimed to develop a reliable and powerful cancer diagnostic model creation system based on microarray data. Their focus on multi-category diagnoses attempted to keep the approach realistic. Their results suggested that the Multiclass Support Vector Machines (MC-SVMs) were the most accurate classifiers for cancer diagnoses from gene expression data, since they outperformed all the other methods considered. They also noted that gene selection techniques significantly improve the classification performance of both MC-SVMs and other classifiers used in their study. They concluded that the ensemble models did not always show improvements over non-ensemble models.

Our Contributions: Our main contributions in this paper are as follows:

- Development of a decision tree induction technique, *ROC-tree*, which in a novel way uses the area under the ROC curve to select nodes of the tree. This aims to address the current gap in classification performance of standard decision tree classifiers used for gene expression datasets.
- Determination of an AUC-based criterion to stop growing the tree.
- An experimental investigation which demonstrates that *ROC-tree* outperforms well known techniques like C5.0 [34], its predecessor C4.5 [33], ADTree [15], Random Forest [5], and Ferri *et al.*'s [13] AUCsplit in terms of accuracy as well as overall AUC value. Furthermore, the predictive power of *ROC-tree* for gene expression is comparable to the black box support vector machine approach, making it an attractive tool for biologists.

Organization: The remainder of the paper is organized as follows. In Section 2, we briefly discuss the fundamental concepts behind the algorithm developed. Our algorithm is formally described in Section 3. Section 4 presents the design of the experimental investigation. We present and discuss the results in Section 5.

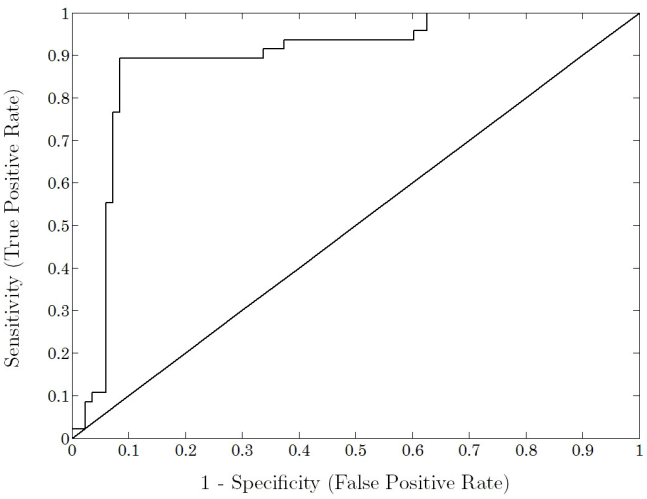

Figure 1: *A typical ROC curve*

Finally in Section 6, we suggest future improvements and conclude the paper.

2 Preliminaries

2.1 The Receiver Operating Characteristic Curve A Receiver Operating Characteristic (ROC) curve was first used in signal detection theory [20]. In machine learning, the ROC curve is used to evaluate the discriminative performance of binary classifiers. This is obtained by plotting the curve of the true positive rate (*Sensitivity*) versus the false positive rate (1 – *Specificity*) for a binary classifier by varying the discrimination threshold. Figure 1 shows a typical ROC curve.

Prior to plotting the ROC curve the *sensitivity* and *specificity* need to be calculated as follows:

- True Positive (TP) = The number of predicted positive cases that are actually positive.
- True Negative (TN) = The number of predicted negative cases that are actually negative.
- False Positive (FP) = The number of predicted positive cases that are actually negative.
- False Negative (FN) = The number of predicted negative cases that are actually positive.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2.1}$$

$$\text{Specificity} = \frac{TN}{FP + TN} \tag{2.2}$$

It is to be noticed that, all the calculations above are attained when using a particular classifier threshold. By varying the threshold, a set of values for these

Algorithm 1 AUROC (adapted from [12])

Input(s): $\mathcal{X}$: The set of examples
Output: Δ: The area under the ROC curve
Require: $\phi(i)$: The probabilistic classifier's estimate that example i is positive, P and N: The number of positive and negative examples, $P > 0$ and $N > 0$

```
Set X_sorted ← X sorted decreasing by φ scores
Set FP ← TP ← 0
Set FP_prev ← TP_prev ← 0
Set Δ ← 0
Set φ_prev ← −∞
Set i ← 1
while i ⩽ |X_sorted| do
    if φ(i) ≠ φ_prev then
        Δ ← Δ + TRAP_AREA(FP, FP_prev, TP, TP_prev)
        Set φ_prev ← φ(i)
        Set FP_prev ← FP
        Set TP_prev ← TP
    end if
    if i is a positive example then
        TP ← TP + 1
    else                          /* i is a negative example */
        FP ← FP + 1
    end if
end while
Δ ← Δ + TRAP_AREA(1, FP_prev, 1, TP_prev)
Δ ← Δ/(P × N)        /* scale from P × N onto the unit square */
return Δ
end
function TRAP_AREA(X_1, X_2, Y_1, Y_2)
Base ← |X_1 − X_2|
Height_avg ← (Y_1 + Y_2)/2
return Base × Height_avg
end function
```

measurements is obtained. This set of values is plotted in a two-dimensional Cartesian graph to yield the ROC curve. The ROC curve indicates the performance of the binary classifier, as it takes into account all the possible solutions by varying the discriminative threshold. The best performance would be produced, if the ROC curve matches with the upper left corner of the ROC space (which yields 100% *sensitivity* and 100% *specificity*). The closer the ROC curve is to the upper part of the ROC space, the better the performance of the classifier.

An ROC curve is a two dimensional illustration of classifier performance. Reducing ROC performance to a single scalar value to represent expected performance helps compare classifiers. An oft used method is to calculate the area under the ROC curve (AUC) [21], [4]. There are several ways to calculate AUC. Of these, one

Algorithm 2 CALCULATEROC

Input(s): $\mathcal{D}$: The matrix of training examples with the last column being the class label
Output: $\mathcal{A}$: Attribute with the highest AUC

```
Set R ← ∅        /* R is an array of the AUC of all attributes of D */
for i ← 1 to (number of column of D − 1) do
    R ← R ∪ AUROC(D_i)
end for
Sort R in descending order based on AUC
Set A ← R_1      /* In the case of a tie take the first attribute */
return A
end
```

way is to calculate using trapezoidal integration shown in Eq. (2.3) [29].

$$\begin{aligned} \text{AUC} &= \int_a^b f(\alpha)d\alpha \\ &\approx \sum_{i=1}^{n} \frac{h}{2}(f(a+(i-1)h)+f(a+ih)) \end{aligned} \tag{2.3}$$

where $\alpha = (1 - \text{Specificity})$, $a = 0$, $b = 1$,
$n =$ size to increase, and $h = \frac{b-a}{n}$.

Prior to calculating the AUC, the set of the values of *sensitivity* and *specificity* is normalized to the range [0, 1]. The AUC, being a part of the area of the unit square, has a value between 0 and 1. Since random guessing could produce the diagonal line between $(0, 0)$ and $(1, 1)$ with an area of 0.5, a classifier with an AUC less than 0.5 is undesirable [12]. A value of AUC equal to 1.0 represents that the performance of the binary classifier is 100% perfect, i.e., the classifier can discriminate the dataset accurately and with 100% *sensitivity* and 100% *specificity*. An AUC value close to 1 indicates better performance for a binary classifier [11].

3 ROC-tree

We now describe in more detail, the steps in our algorithm for building the decision tree using the ROC measure.

3.1 Selecting Nodes of the Tree Previous work has established the use of an ROC curve for feature selection, to identify the discriminative genes in the dataset [26]. First, the ROC curve is plotted for each of the pairs formed by each of the genes and the class label. This means treating a single gene as a classifier and calculating the classification in terms of the *sensitivity* and *specificity* by varying the operating point. We adapt this kind of idea as a selection method for nodes in

building the decision tree. For each feature (gene), the AUC is calculated and the gene with the highest AUC is selected as the node of the tree. Algorithm 1 shows how to generate ROC points for a given set and calculate the AUC from it; and Alg. 2 presents pseudocode for calculating the AUC for each gene.

Based on selection of a suitable splitting threshold for the selected attribute (to be discussed shortly), the dataset is then divided into two subsets. Each of the subsets is then used to further induce the tree in a similar way.

EXAMPLE 3.1. *Let us consider a dataset $\mathcal{D}$ of n examples, where each example comprises m attributes: $x_1, x_2, x_3, \ldots, x_m$. Each of the m attributes has a differing discriminative power reflected by its respective AUC. To calculate the discriminative power that is expressed in terms of AUC, we plot the ROC curve for each attribute paired with the class label, (i.e., $\{x_i, Y_i\}$, where $1 \leqslant i \leqslant m$ and Y is the vector of class labels) and calculate the AUC of the ROC curve. Suppose the attribute x_α, where $1 \leqslant \alpha \leqslant m$ has the highest AUC. So, for the ROC-tree, x_α is to be selected as the node. A suitable threshold is then obtained for this attribute and the dataset $\mathcal{D}$ is divided into two subsets: $\mathcal{D}_{left}$ and $\mathcal{D}_{right}$. Then, for each subset $\mathcal{D}_{left}$ and $\mathcal{D}_{right}$, we recursively use the similar process by excluding the attributes used at the parent nodes and thus, induce the decision tree.*

3.2 Splitting Threshold Selecting the best value to discriminate the classes based on the selected attribute is important. The splitting threshold is chosen by taking each value of the selected gene from the dataset, and then attempting to classify based on a chosen value and calculating the misclassification rate for that value. The value with the minimum misclassification rate is finally selected as the splitting threshold.

EXAMPLE 3.2. *Let us consider the dataset $\mathcal{D}$ of n instances, where each instance has m attributes: $x_1, x_2, x_3, \ldots, x_m$. Suppose, x_α, where $1 \leqslant \alpha \leqslant m$ has the highest AUC and is selected to be the node in the tree. Then for each value $x_{\alpha_1}, x_{\alpha_2}, x_{\alpha_3}, \ldots, x_{\alpha_n}$ of x_α of the dataset, we form the rule:*

$$\textbf{If } x_\alpha \geqslant x_{\alpha_j} \textbf{ then } Class = \ +ve$$
$$\textbf{else } Class = -ve,$$

where $1 \leqslant j \leqslant n$. We then attempt to classify the dataset with this rule, and note the misclassification rate. Then the x_{α_j} with the minimum misclassification rate is selected as the splitting threshold for attribute x_α.

When choosing the splitting threshold the classification error (misclassification rate) is used instead of the *sensitivity* and *specificity* measure. This is done because the *sensitivity* and *specificity* measure only gives us some idea on true and false positives individually, but does not give a complete picture of classification accuracy. On the other hand, the misclassification rate gives an easily interpretable picture of the performance of the classification. Moreover, splitting the dataset based on misclassification rate can reduce the uncertainty maximally, i.e., most of the similar classes fall under the same branch. Pseudocode for calculating the splitting threshold is presented in Alg. 3.

Algorithm 3 CALCULATESPLITTHRESHOLD

Input(s): $\mathcal{D}$: The matrix of training examples with the last column being the class label,
$\mathcal{A}$: Selected attribute
Output: θ: Splitting threshold for the node
Require: Y: The target attribute (i.e., the last column of $\mathcal{D}$), Γ: The rule to test the split

```
1:  Set θ ← −∞
2:  Set error ← 0                /* error is the counter for misclassification */
3:  Set thresError ← ∅   /* thresError is an array of error for each example */
4:  // Select each value of the attribute to choose the respective splitting threshold
5:  for each example i ∈ D do
6:     Formulate Γ ⇐
       IF A ⩾ A_{value_i} THEN Class = Y+
       ELSE Class = Y−   /* A_{value_i} is the value of the A in D for each example i; Y+ and Y− are the positive and negative class, respectively */
7:     Set error ← 0
8:     // Calculate misclassification
9:     for each example i ∈ D do
10:       if Γ does not satisfy the i^th example then
11:          error ← error + 1
12:       end if
13:    end for
14:    thresError ← thresError ∪ {error}
15: end for
16: Find index of the minimum member of thresError
17: θ ← A_index
18: return θ
19: end
```

3.3 Stopping Criterion To decide when to stop growing the tree, the AUC of the selected attribute is tested. A value of AUC equal to 1 represents that the selected attribute can classify the training dataset accurately with 100% *sensitivity* and 100% *specificity*. Therefore, a value of AUC close to 1 indicates that

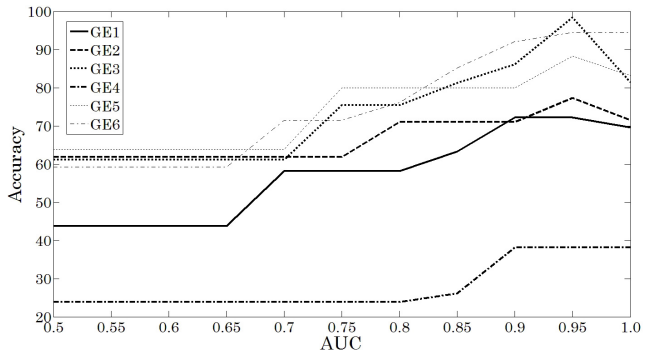

Figure 2: *The effect of the overall tree performance for different stopping threshold on six gene expression datasets*

the current node is pure and the tree need not grow. However, to avoid overfitting, growing the tree for a node is stopped, when the AUC for an attribute is found to be 0.95. This is because we do not want to grow the tree for a smaller subset of the training set. We have chosen to use the value of 0.95 for the stopping threshold, based on consideration of varying behaviour of this parameter for the gene expression datasets in our experimental evaluation. Figure 2 shows how the accuracy of the overall tree is affected by varying the value of this parameter, for a number of datasets (to be described later). A noticeable peak in the graph can be seen around the value of 0.95 for the stopping threshold (AUC).

Another situation that we take into account is that if the AUC of a chosen attribute is equal to or less than 0.5, it indicates the predictive power is worse than that of a random classifier [12]. So, when a selected attribute is found to have an AUC equal to or less than 0.5, the growing of the tree is stopped for that node.

3.4 Labelling the Leaf Nodes Once the stop condition is satisfied, further growing of the tree will be ceased. Nonetheless, a class labelling is needed for each of the two children induced by the split point. In this circumstance, we label by the positive class, the child node having values greater than or equal to the threshold value and label the other child node by the negative class. This is done to maintain consistency with the method that we used for selection of the splitting threshold.

Algorithm 4 presents the pseudocode for inducing the *ROC-tree* using the functions presented in Alg. 2 and Alg. 3.

3.5 Complexity of the Algorithm Let us consider a dataset $\mathcal{D}$ of N examples, where each example comprises M attributes: $x_1, x_2, x_3, \ldots, x_M$. Then the algorithm would need an $O(N \log N)$ sort followed by an $O(N)$ scan down the list, and the result would have a total complexity of $O(N \log N)$ for calculating the AUC of a single attribute or gene. Therefore, for all the genes, it will have a complexity of $O(MN \log N)$. To calculate the splitting threshold it will have another complexity of $O(N^2)$. Thus, the algorithm has a total complexity of $O(MN \log N + N^2)$ for choosing a single node and its splitting threshold.

Algorithm 4 ROCtree

Input(s): $\mathcal{D}$: The matrix of training examples with the last column being the class label
Output: $\mathcal{T}$: A decision tree

1: **if** $\mathcal{D} = \emptyset$ **then**
2: **return** a single node with $\emptyset$
3: **end if**
4: **if** $\mathcal{D}$ consists of records all with the same value for the class label **then**
5: **return** a single leaf node with that value
6: **end if**
7: $\mathcal{A} \leftarrow$ CalculateROC($\mathcal{D}$) /* $\mathcal{A}$ be the attribute with the largest AUC among all attributes in $\mathcal{D}$ */
8: $\theta \leftarrow$ CalculateSplitThreshold($\mathcal{D}, \mathcal{A}$) /* θ be the split point for attribute $\mathcal{A}$ */
9: Set $\mathcal{D}_{left}$ and $\mathcal{D}_{right}$ as the subsets of $\mathcal{D}$ consisting of records respectively with the value greater than or equal to and less than
10: Recursively apply *ROCtree* to subsets $\mathcal{D}_{left}$ and $\mathcal{D}_{right}$ until they are empty or the stopping criteria are met
11: **return** a tree $\mathcal{T}$ with root or node labelled $\mathcal{A}$ and arcs labelled a_1 and a_2, going respectively to the trees $ROCtree(\mathcal{D}_{left})$ and $ROCtree(\mathcal{D}_{right})$
12: **end**

4 Experiment Design and Datasets

For the experimental analysis, we compare against ten other techniques. These are: a number of well known simple decision tree induction techniques: C5.0 [34], its predecessor C4.5 [33], Ferri *et al.*'s [13] AUCsplit technique for decision trees, ADTree [15], Random Forest [5], REPTree and Random Tree. We also compared against several non decision tree classifiers: Naïve Bayes, k-NN and SVMs using linear kernel (SVM). The linear kernel is chosen because Deng *et al.* [10] showed that on gene expression datasets, SVMs with linear kernel perform better than SVMs with polynomial or RBM kernel when no explicit feature selection is performed.

Each of the techniques is applied on 12 datasets, of which 6 are GE datasets and 6 are non-GE datasets with having rather different characteristics. To evaluate

Table 1: *Properties of the datasets used in this study*

Dataset	No. of Attributes	No. of Instances
GE1	24,481	97
GE2	3,226	22
GE3	12,533	181
GE4	12,600	21
GE5	12,600	136
GE6	7,129	72
Hepatitis	19	155
Ionosphere	34	351
WBC	9	699
WDBC	30	569
WPBC	33	198
Pima Indians	8	768

the performance of the *ROC-tree* classifier, a 10-fold cross validation (CV) scheme is used 10 times for all 12 datasets.

4.1 Datasets Several gene expression datasets from previous studies have been obtained to validate the performance of the proposed algorithm. Apart from these, 6 non-GE datasets discussed widely in the literature have also been considered, to evaluate the performance of the proposed algorithm in different circumstances. They are: Hepatitis Domain Database (Hepatitis), Johns Hopkins University Ionosphere Database (Ionosphere), Pima Indians Diabetes Database (Pima), Wisconsin Breast Cancer Database (WBC), Wisconsin Diagnostic Breast Cancer dataset (WDBC) and Wisconsin Prognostic Breast Cancer dataset (WPBC); and are collected from the UCI Machine Learning Repository [27]. The properties of the datasets are illustrated in Tab. 1. The considered gene expression datasets are briefly described next.

Gene Expression Dataset 1 (GE1): This gene expression dataset is collected from the Integrated Tumor Transcriptome Array and Clinical data Analysis database [23] which was first used by van 't Veer *et al.* [39]. In this dataset, every sample is described by the expression levels of 24,481 genes. The dataset comprises 97 lymph-node-negative breast cancer patients, of which 46 positive and 51 negative cases were present.

Gene Expression Dataset 2 (GE2): Collected from the Microarray Project of the National Human Genome Research Institute, USA, this gene expression dataset was first used by Hedenfalk *et al.* [22]. All samples in this dataset are described by the expression levels of 3,226 genes. The subset used in this experiment comprised 14 patients with BRCA1 gene mutations and 8 patients with BRCA2 gene mutations.

Gene Expression Dataset 3 (GE3): This dataset is obtained from the Division of Thoracic Surgery [38] Brigham and Women's Hospital, Boston, MA, USA. It contained classification information between malignant pleural mesothelioma (MPM) and adenocarcinoma (ADCA) of the lung. There are 181 tissue samples (31 MPM and 150 ADCA) and each sample is described by the expression of 12,533 genes. This dataset was first used by Gordon *et al.* [19].

Gene Expression Dataset 4 (GE4): Collected from the Cancer Program [8], Broad Institute of MIT and Harvard, MA, USA, this gene expression dataset was first used by Singh *et al.* [36]. In this dataset, 21 prostate cancer patients were evaluable with respect to recurrence following surgery with 8 patients having relapsed and 13 patients having remained relapse free for at least 4 years. Each sample, in this dataset, is described by the expression of 12,600 genes.

Gene Expression Dataset 5 (GE5): This dataset is actually a variant of the gene expression dataset 4, and comprises tissue sample of 77 prostate cancer patients and 59 controlled volunteer.

Gene Expression Dataset 6 (GE6): This gene expression dataset is collected from the Cancer Program [7], Broad Institute of MIT and Harvard, MA, USA; and was first used by Golub *et al.* [18]. This dataset contains 47 ALL type and 25 AML type bone marrow samples, over 7,129 probes from 6,817 human genes.

5 Results and Discussion

The classification results for all 11 techniques on the considered datasets are presented in Tab. 2 and Tab. 3. The best performances among that of the reported classifiers are marked in bold. We begin our discussion by first focusing on classifiers other than the support vector machine.

The classification performance of the *ROC-tree* on the gene expression datasets clearly outperforms that of all the other reported decision trees. We conjecture that one of the reasons for this significantly better performance is due to the better computation of discriminative power of attributes used when building the tree. It is worth noting that for the datasets GE1 and GE6, the performance improvement of the *ROC-tree* is at least 8% and 13% better than all the other decision tree techniques. Similarly, for GE2, GE3, GE4 and GE5, there are consistent increases in accuracy. Results of statistical significance tests we conducted (Bonferroni-corrected t-tests [17]) confirmed that the improvement in accuracy is significant. For example, when comparing the best results of the *ROC-tree* with that of the C4.5, on 5 of the 6 gene expression datasets (GE2 to GE6) the p-value is less than 0.05 (0.042463, 0.012897, 0.030047, 0.036691 and 0.04914, respectively). GE1

Table 2: *Comparison of accuracy results from* 10×10*-fold cross validaion on six gene expression datasets*

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-tree	$\mathbf{72.16 \pm 4.32}$	$\mathbf{77.27 \pm 2.45}$	98.34 ± 0.89	38.10 ± 5.95	88.24 ± 2.33	94.44 ± 2.96
AUCsplit	63.58 ± 4.59	74.39 ± 1.63	96.14 ± 1.36	34.01 ± 2.87	82.47 ± 3.96	81.61 ± 3.28
C5.0	64.95 ± 6.21	59.09 ± 4.52	92.82 ± 1.21	23.81 ± 4.65	81.62 ± 4.12	80.55 ± 3.74
C4.5	62.89 ± 3.11	72.73 ± 1.36	95.03 ± 1.05	33.33 ± 4.59	79.42 ± 5.45	79.17 ± 4.87
ADTree	61.86 ± 4.28	68.18 ± 5.68	92.82 ± 2.19	32.86 ± 3.44	86.76 ± 2.63	86.11 ± 3.77
REPTree	52.18 ± 5.45	59.09 ± 3.92	95.03 ± 1.28	32.86 ± 3.46	80.88 ± 3.33	81.94 ± 4.26
Random Tree	55.67 ± 3.54	63.64 ± 2.58	79.56 ± 2.69	32.86 ± 3.12	62.50 ± 5.23	75.00 ± 3.90
Random Forest	62.89 ± 6.43	50.00 ± 5.33	93.92 ± 1.22	38.10 ± 5.27	80.88 ± 2.56	79.17 ± 2.36
Naïve Bayes	54.64 ± 3.38	59.09 ± 4.58	98.34 ± 0.03	33.33 ± 0.78	55.88 ± 4.76	98.61 ± 1.03
k-NN	62.89 ± 5.63	63.64 ± 4.32	93.92 ± 2.01	$\mathbf{57.14 \pm 5.97}$	78.68 ± 4.78	83.33 ± 3.49
SVM	68.04 ± 2.14	59.09 ± 2.98	$\mathbf{99.45 \pm 0.11}$	47.62 ± 5.63	$\mathbf{91.18 \pm 3.12}$	$\mathbf{98.61 \pm 1.26}$

Table 3: *Comparison of accuracy results from* 10×10*-fold cross validation on six non-gene expression datasets*

Method	Hepatitis	Ionosphere	WBC	WDBC	WPBC	Pima
ROC-tree	78.71 ± 7.65	84.05 ± 9.87	92.56 ± 5.43	90.69 ± 6.78	69.67 ± 8.33	63.54 ± 8.65
AUCsplit	82.10 ± 3.43	86.00 ± 7.31	95.88 ± 1.94	93.75 ± 3.39	70.53 ± 9.67	73.82 ± 5.35
C5.0	76.13 ± 2.35	89.46 ± 1.23	93.64 ± 1.65	93.29 ± 2.23	70.70 ± 4.12	73.94 ± 2.76
C4.5	80.00 ± 4.45	91.45 ± 3.36	93.84 ± 2.63	93.15 ± 1.26	75.25 ± 3.32	73.83 ± 2.89
ADTree	76.13 ± 2.96	$\mathbf{93.16 \pm 1.65}$	95.14 ± 1.77	94.02 ± 1.06	77.78 ± 5.42	72.92 ± 3.23
REPTree	78.71 ± 4.23	89.46 ± 1.46	93.99 ± 2.14	92.44 ± 2.33	73.74 ± 4.85	75.39 ± 4.55
Random Tree	72.91 ± 9.21	87.75 ± 3.64	94.13 ± 2.85	89.46 ± 3.67	68.18 ± 5.45	67.97 ± 6.49
Random Forest	81.94 ± 1.26	92.59 ± 3.26	95.99 ± 1.45	95.25 ± 1.37	$\mathbf{78.28 \pm 3.47}$	73.70 ± 4.98
Naïve Bayes	$\mathbf{83.87 \pm 1.71}$	82.62 ± 3.48	95.99 ± 0.74	92.97 ± 2.58	67.68 ± 5.08	76.30 ± 3.49
k-NN	81.94 ± 3.66	84.90 ± 3.74	$\mathbf{97.00 \pm 1.08}$	97.01 ± 1.12	74.24 ± 4.18	73.18 ± 3.23
SVM	76.77 ± 4.23	88.60 ± 2.43	96.85 ± 1.07	$\mathbf{97.72 \pm 1.04}$	76.26 ± 4.78	$\mathbf{77.34 \pm 5.01}$

Table 4: *Comparison of AUC value from* 10×10*-fold cross validaion on six gene expression datasets*

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-tree	$\mathbf{0.7386 \pm 0.01}$	$\mathbf{0.7933 \pm 0.03}$	$\mathbf{0.9393 \pm 0.04}$	0.2857 ± 0.05	0.8900 ± 0.33	$\mathbf{0.9504 \pm 0.01}$
AUCsplit	0.6421 ± 0.10	0.7360 ± 0.07	0.9221 ± 0.02	0.3012 ± 0.06	0.8130 ± 0.04	0.8241 ± 0.08
C4.5	0.5887 ± 0.03	0.6607 ± 0.02	0.9028 ± 0.01	0.2212 ± 0.09	0.7819 ± 0.04	0.6885 ± 0.03
SVM	0.6415 ± 0.02	0.4911 ± 0.08	0.9235 ± 0.00	$\mathbf{0.3077 \pm 0.23}$	$\mathbf{0.9216 \pm 0.07}$	0.92 ± 0.04

showed milder improvement (p-value is 0.093591).

The results of the AUCsplit technique were as good as C4.5 or better and slightly lower than our proposed algorithm on these gene expression datasets. This is interesting, since both techniques consider AUC values, yet use it to build the tree in different ways.

The improved performance of the *ROC-tree* over other decision trees may be due to the fact that the AUC is a better measure to select nodes for the tree in the context of gene expression data, compared to other uncertainty measurements such as entropy and information gain. A key factor to remember here is that gene expression data contains few instances, many attributes and tends to be noisy. It has also been shown that it is not safe to assume that gene expression data satisfies a normal distribution [10]. Hence the use of non parametric statistical measures, to which AUC calculation is closely connected via the Mann-Whitney test (as mentioned in the introduction), are likely to be appropriate for selecting good discriminating attributes.

We also computed the overall AUC value of selected classifiers, shown in Tab. 4, resulting from the 10×10 cross validation over the gene expression datasets. The AUC values of the four considered classifiers: *ROC-tree*, AUCsplit, C4.5 and SVM, indicate that AUCsplit improves over C4.5 more for AUC value than for accuracy value. For five of the six datasets, we see the *ROC-tree* has better AUC than either C4.5 or AUCSplit. In-

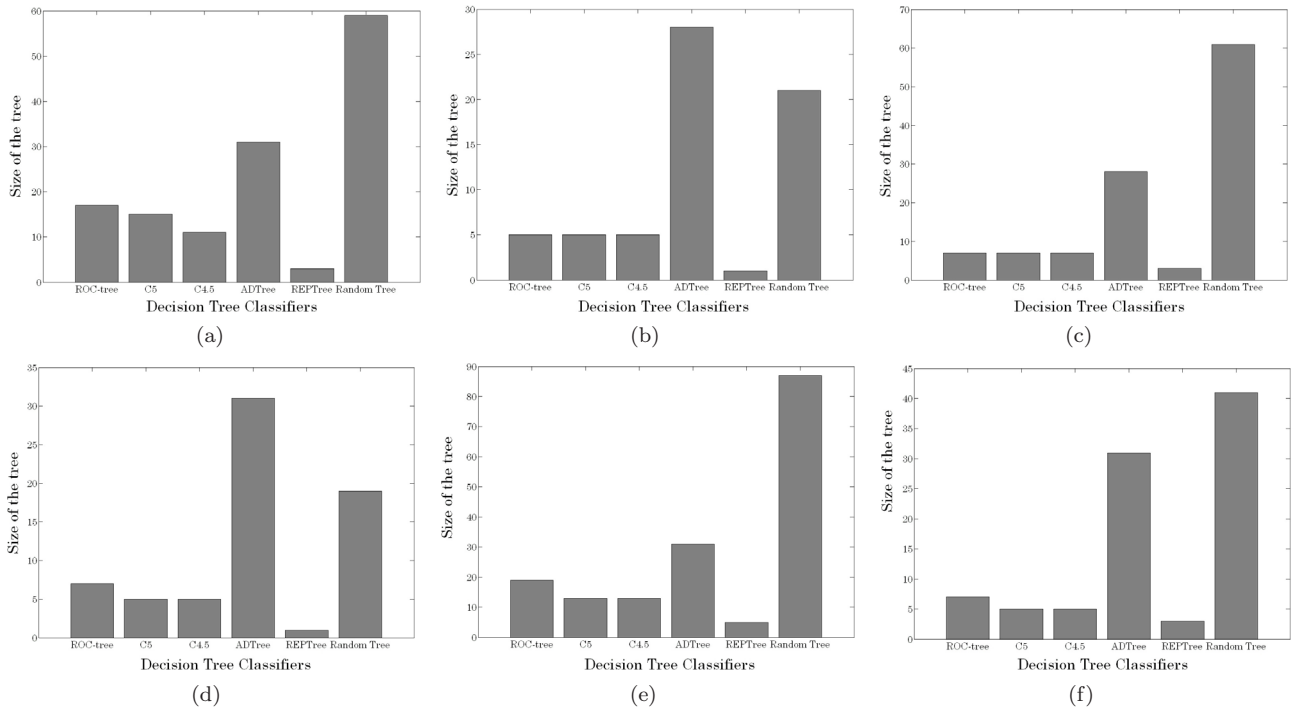

Figure 3: *Comparison of the sizes of the trees: ROC-tree, C5, C4.5, ADTree, REPTree and Random Tree on 6 gene expression datasets used in this study: (a) GE1, (b) GE2, (c) GE3, (d) GE4, (e) GE5, (f) GE6*

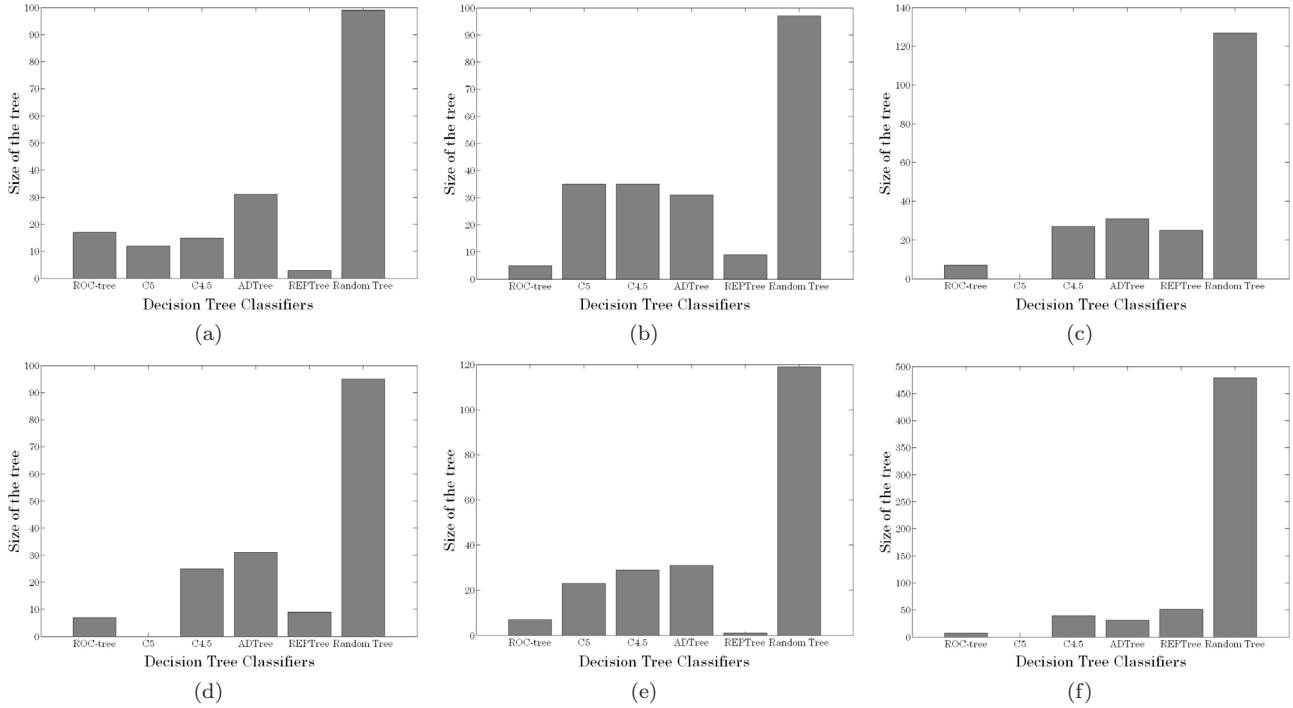

Figure 4: *Comparison of the sizes of the trees: ROC-tree, C5, C4.5, ADTree, REPTree and Random Tree on 6 non-GE datasets used in this study: (a) Hepatitis, (b) Ionosphere, (c) Pima Indians, (d) WBC, (e) WDBC, and (f) WPBC.*

terestingly, although the SVM technique showed excellent accuracy across 4 gene expression datasets out of 6 datasets, in terms of AUC, its performance was the best on only 2 datasets.

ROC-tree also performed impressively when compared to SVM. In particular, when compared to the SVM considering the accuracy metric, *ROC-tree* managed two wins (for GE1 and GE2) and was not far behind for GE3, GE5 and GE6. Interestingly, compared to SVM for the AUC metric, *ROC-tree* managed four wins (GE1, GE2, GE3, GE6) and was not far behind for GE4 and GE5. We believe this is an important result, since it establishes that the *ROC-tree* is highly competitive compared to the SVM, yet has the significant extra advantage of being interpretable to biologists.

Non-GE Datasets: It is interesting to notice that compared to the success in classifying the high dimensional gene expression datasets, the *ROC-tree* method yielded accuracy results for the non-GE datasets (see Tab. 3) which were rather less powerful compared to the other classifiers. The underlying reason for this is an open question, but intuitively, we believe it may be because in the presence of fewer attributes, it is difficult to identify ones with substantial discriminative power. This leads to sub-optimal choices being made when building the tree.

Another interesting trend in the results is that hybridized decision tree induction algorithms, like ADTree or Random Forest that performed quite well on the non gene datasets compared to simple decision tree induction techniques like C4.5, or its successor C5.0; did not perform impressively on the gene expression datasets.

Tree Size: The size of each tree built using the *ROC-tree* method is rather smaller compared to trees generated by ADTree or Random Tree. For almost all gene expression datasets, *ROC-tree* is about the same size as a C5 or C4.5 pruned tree (see Fig. 3). However, for the non-GE datasets, *ROC-tree* yielded the smallest tree on four datasets (Fig. 4(b), 4(c), 4(d), 4(f)). Though REPTree generated small trees in most cases, the accuracy is never impressive. Random Tree, on the other hand, always yielded the largest trees.

Running Time: Execution time for the considered algorithms is presented in Tab. 5. It can be seen that the times are relatively fast for all the gene expression datasets (at most tens of seconds). The table indicates that the *ROC-tree* takes slightly more time to build the tree than C5 does. The execution times of both these algorithms are somewhat higher than that of the other algorithms, including C4.5, which is the predecessor of C5. However, the execution time is still lower than the complex decision trees (i.e., ADTree and Random Forest) or the AUCsplit technique. Apart from tree size, another factor influencing the execution time may be the choice of implementation language used in a particular algorithm. For example, *ROC-tree* is implemented in MATLAB 2007a, C5 and AUCsplit are implemented in C/C++; and the rest are implemented in Java.

Table 5: *Comparison of execution time (in seconds) taken to build the model using all data on Gene Expression Datasets*

Method	GE1	GE2	GE3	GE4	GE5	GE6
ROC-tree	35.11	0.52	12.32	2.03	15.22	3.27
AUCsplit	36.34	1.58	42.76	41.32	43.09	4.38
C5	31.5	0.5	11.4	1.8	11.3	2.6
C4.5	9.92	0.11	4.89	0.41	4.3	1.05
ADTree	49.61	0.34	19.56	1.06	27.14	2.83
REPTree	3.81	0.08	1.67	0.34	2.05	0.5
Random Tree	13.88	0.13	12.06	0.42	3.45	0.59
Random Forest	42.69	0.47	23.14	1.78	17.61	3.7
Naïve Bayes	1.94	0.03	1.56	0.13	1.08	0.3
k-NN	0.11	0.001	0.09	0.03	0.08	0.03
SVM	5.16	0.09	3.39	0.83	3.28	0.61

6 Conclusion

This paper has focused on presenting an effective decision tree algorithm for classifying gene expression datasets. We have shown how the well known ROC measure can be used to construct decision trees with very high accuracy for such datasets. Indeed our classifier is able to substantially improve over the decision tree state of the art for this problem. Decision trees are particularly attractive for biologists due to their intepretability, being able to highlight which genes are actually influencing the classification.

For future work, we intend to consider extending our approach to situations where there are three or more classes in the data.

References

[1] D. Aha and D. Kibler. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[2] D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematics and Psychology*, 12:387–415, 1975.

[3] M. Beibel. Selection of informative genes in gene expression based diagnosis: a nonparametric approach. *Lecture Notes in Computer Sciences. Proc. ISMDA'00*, 1933:300–307, 2000.

[4] A. P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[5] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, October 2001.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classfication and Regression Trees*. Wadsworth, Belmont, 1984.

[7] http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=43.

[8] http://www.broad.mit.edu/cgi-bin/cancer/publications/pub_paper.cgi?mode=view&paper_id=75.

[9] J. Catlett. *Megainduction: Machine learning on very large databases*. PhD thesis, University of Sydney, Australia, 1991.

[10] L. Deng, J. Pei, J. Ma, and D. L. Lee. A Rank Sum Test Method for Informative Gene Discovery. In *Proc. KDD'04*, pages 410–419, Seattle, WA, USA, 2004.

[11] J. P. Egan. *Signal Detection Theory and ROC Analysis*. Academic Press Series in Cognition and Perception. Academic Press, London, UK, 1975.

[12] T. Fawcett. ROC Graphs: Notes and Practical Considerations for Researchers. Technical Report MS 1143, HP Laboratories, 2004.

[13] C. Ferri, P. Flach, and J. Hernández-Orallo. Learning decision trees using the area under the ROC curve. In *Proc. ICML 2002*, pages 139–146. Morgan Kaufmann, 2002.

[14] D. J. Fifield. Distributed tree construction from large datasets. B.S. Honours thesis, Australian National University, Canberra, Australia, 1992.

[15] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proc. ICML '99*, pages 124–133, Bled, Slovenia, 1999.

[16] J. Gehrke, R. Ramakrishnan, and V. Ganti. RainForest – A Framework for Fast Decision Tree Construction of Large Datasets. In *Proc. VLDB '98*, pages 416–427, San Francisco, 1998. Morgan Kaufmann.

[17] S. A. Glantz. *Primer of BioStatistics*. McGraw-Hill, NY, USA, 1992. 309–310.

[18] T. R. Golub, D. K. Slonim, P. Tamayo *et al.* Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science*, 286:531–537, 1999.

[19] G. J. Gordon, R. V. Jensen, Li-Li Hsiao *et al.* Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer And Mesothelioma. *Cancer Research*, 62:4963–4967, 2002.

[20] D. M. Green and J. M. Swets. *Signal detection theory and psychophysics*. John Wiley & Sons Inc., New York, USA, 1966.

[21] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143:29–36, 1982.

[22] I. Hedenfalk, D. Duggan, Y. Chen *et al.* Gene-expression profiles in hereditary breast cancer. *The New England Journal of Medicine*, 344(8):539–548, 2001.

[23] http://bioinfo-out.curie.fr/ittaca.

[24] I. Inza, P. Larrañaga, R. Blanco, and A. J. Cerrolaza. Filter versus wrapper gene selection approaches in DNA microarray domains. *Artificial Intelligence in Medicine*, 31:91–103, 2004.

[25] S. Ma and J. Huang. Regularized ROC method for disease classification and biomarker selection with microarray data. *Bioinformatics*, 21:4356–4362, 2005.

[26] H. Mamitsuka. Selecting features in microarray classification using ROC curves. *Pattern Recognition*, 39(12):2393–2404, 2006.

[27] http://www.ics.uci.edu/~mlearn/MLRepository.html.

[28] N. J. Nilsson. *Introduction to Machine Learning*. An early draft of a proposed textbook, 1996.

[29] M. Nishikawa, Y. Takakura, F. Yamashita, and M. Hashida. Basic pharmacokinetics of oligonucleotides and genes. In R. I. Mahato and S. W. Kim, editors, *Pharmaceutical Perspectives of Nucleic Acid-Based Therapeutics*, chapter 19, pages 409–433. CRC Press, London, UK, 2002.

[30] J. Platt. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[31] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1:81–106, 1986.

[32] J. R. Quinlan. Simplifying Decision Trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[33] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, CA, USA, 1993.

[34] http://www.rulequest.com/see5-info.html.

[35] L. Rokach and O. Maimon. Top-Down Induction of Decision Trees Classifiers–A Survey. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, 35(4):476–487, 2005.

[36] D. Singh, P. G. Febbo, K. Ross *et al.* Gene Expression Correlates of Clinical Prostate Cancer Behavior. *Cancer Cell*, 1:203–209, 2002.

[37] A. Statnikov, C. F. Aliferis, I. Tsamardinos, D. Hardin, and S. Levy. A comprehensive evaluation of multicategory classification methods for microarray gene expression cancer diagnosis. *Bioinformatics*, 25(5):631–643, 2005.

[38] http://www.chestsurg.org/publications/2002-microarray.aspx.

[39] L. J. van 't Veer, H. Dai, M. J. van de Vijver *et al.* Gene expression profiling predicts clinical outcome of breast cancer. *Nature*, 415:530–535, 2002.

[40] G. I. Webb and K. M. Ting. On the application of ROC analysis to predict classification performance under varying class distributions. *Machine Learning*, 58(1):25–32, 2005.

[41] H. Zhang, C.-Y. Yu, B. Singer, and M. Xiong. Recursive partitioning for tumor classification with gene expression microarray data. *PNAS USA*, 98(12):6730–6735, 2001.