In this work, a new algorithm of tree-structured classification for multivariate binary responses, the majority-vote method, is proposed. The majority-vote method is a variation of the original work of Breiman et al. (1984) on Classification And Regression Trees (CART® is a trademark of The Salford Systems.). The majority-vote method is similar to CART in that both methods use node impurity as the basis of the splitting rules. The majority-vote method differs from CART in that it determines tree size by choosing an optimal threshold value so that the cross-validated hit rate is maximized, whereas CART uses cost-complexity pruning to determine the optimal tree size. The original motivation of this work is to handle incomplete data, missing and censoring, in a Quantitative Structure Activity Relationship (QSAR) context, where the responses are continuous measurements of activity levels. We proceed by discretizing the responses into binary variables and using the majority-vote method to analyze the resulting binary responses. The performance of the majority-vote method is compared to its continuous response counterpart, MultiSCAM, a tree-structured algorithm for analyzing multivariate continuous responses. Multivariate analysis of variance (MANOVA) is used to evaluate the relative information loss due to discretization. The predictivity of the majority-vote method is evaluated by hit rate, a commonly used criterion in drug discovery. Simulation studies show that the majority-vote method outperforms MultiSCAM for censored data in that it yields higher hit rates.
Tree-structured Classification for Multivariate Binary Responses

by

JIUZHOU WANG

A dissertation submitted to the Graduate Faculty of North Carolina State University in partial satisfaction of the requirements for the Degree of Doctor of Philosophy

STATISTICS

Raleigh

2002

Approved By:

Professor Leonard A. Stefanski
Chair of Advisory Committee

Professor Marc G. Genton
Co-Chair of Advisory Committee

Professor S. Stanley Young

Professor John F. Monahan
To Meimei

and

Fengxiang Wang & Chunping Tan
Biography

Jiuzhou Wang was born to Fengxiang Wang and Chunping Tan on April 10, 1973 in Qiqihar, China.

Jiuzhou completed the BS degree in Mathematical Statistics and the MS degree in Probability and Statistics from Nankai University, Tianjin in 1992 and 1995, respectively. Before joining the Doctoral program in the department of Statistics at North Carolina State University in 1999, he was a lecturer in the department of Accounting at Nankai University.

Jiuzhou married Meimei on July 28, 1999 in Chengdu, China.
Acknowledgements

The author thanks GlaxoSmithKline for supporting his internship in the Cheminformatics group from March 2000 to December 2002, providing data and softwares and assisting with valuable scientific consultancy. Special thanks to Dr. Chris Keefer and Dr. Lei Zhu, who are always very instrumental. Dr. Leonard Stefanski and Dr. Marc Genton have been advising the author throughout this work. They are always great resources of valuable ideas and comments. This work is impossible without their supervising. The author is also grateful to Dr. Stanley Young. It is he that introduced me to the world of tree-structured methods, and in particular, the application of trees to QSAR. He also supplied many helpful comments to the author on both methodology and the organization of this article. The author was fortunate in having an expert in statistical computing, Dr. John Monahan, who made many valuable suggestions in the simulation study of this work, as a member of the dissertation committee.
## Contents

**List of Figures** vii  
**List of Tables** viii

### 1 Introduction
1.1 A motivating problem ........................................ 1  
1.2 Available approaches ...................................... 6  
1.2.1 Tree-structured methods for univariate response ........ 7  
1.2.2 Tree-structured methods for multivariate responses .... 7

### 2 The Majority-vote Method
2.1 Notations and definitions .................................. 12  
2.2 Growing a tree ............................................. 16  
2.3 Discretizing the response ................................. 20  
2.4 Evaluating the relative information loss due to discretization .... 22  
2.4.1 Multivariate Analysis of Variance (MANOVA) .......... 23  
2.4.2 Example: an analysis on a $576 \times 10$ data ........... 25  
2.5 Evaluating the predictivity of a tree: hit rate .......... 30  
2.6 Determining the tree size .................................. 31

### 3 Simulation Study Design
3.1 The underlying assumption of MultiSCAM .................. 35  
3.2 Simulating complete data .................................. 36  
3.3 Simulating censored data ................................... 37  
3.4 Evaluation rules ............................................ 40  
3.5 Evaluating the predictivity of a tree: hit rate .......... 46

### 4 Results of Simulation Study
4.1 Hit rates comparison: resubstitution estimates .......... 51  
4.1.1 The design of experiment ................................. 52  
4.1.2 The results .............................................. 53  
4.2 Hit rates comparison: test-sample estimates ............. 55  
4.2.1 The design of experiment ................................. 55
| 4.2.2  | The results on complete data                      | 56  |
| 4.2.3  | The results on censored data                      | 59  |
| 4.3    | Summary of the simulation study                   | 62  |
| 5      | Discussions, Conclusions and Future Work          | 65  |
| 5.1    | Tree-structured methods: univariate to multivariate| 65  |
| 5.2    | Conclusions                                      | 67  |
| 5.3    | Future work                                      | 68  |
| A      | Summary of Algorithms                             | 70  |
| A.1    | Pseudo code of MultiSCAM                          | 70  |
| A.2    | The tree-growing algorithm of the majority-vote method | 71  |
| A.3    | The complete-data generating algorithm            | 72  |
| A.4    | The censored-data generating algorithm            | 73  |
| A.5    | The tree-size determination algorithm             | 73  |
| B      | Proofs of Propositions                            | 75  |
| B.1    | Proof of Proposition 2.1                          | 75  |
| B.2    | Proof of Proposition 2.2                          | 77  |
| Bibliography                                 | 80  |
List of Figures

1.1 A hypothetical tree. The root node (N), which contains 576 observations, is split on \( X_{37} \); the left subnode of N (N0), which corresponds to \( X_{37} = 0 \), is further split on \( X_{16} \). .................................................. 6

1.2 Histograms of 10 response variables in a real QSAR data set. The skewness in response variables is obvious. The pole at 4.0 in the plot for variable \( Y_4 \), for example, is a result of single imputation (4.0 at censored values) .... 10

2.1 Node profiles of N00, N01 and N1 in Figure 1.1. The height of each bar represents the proportion of active compounds. ........................................ 19

2.2 \( \hat{\rho}_j(c) \) as function of c for a 576 \( \times \) 5 response data. ................. 21

2.3 Two-way scatter plot of the 10 response variables. There is an apparent linear relationship between \( Y_2 \) and \( Y_3 \). The dots on a straight line in other plots correspond to the imputed values (e.g. 4.0). ................. 26

2.4 Part of the tree generated by MultiSCAM on the 576 \( \times \) 10 data set ....... 27

2.5 Resubstitution hit rate (\( H_{RS} \)) and cross-validation hit rate (\( H_{CV} \)) as functions of \( \Delta_{\text{min}} \). The triangles are values of \( H_{RS} \) and the circles are values of \( H_{CV} \) .................................................. 34

3.1 A simple MultiSCAM tree ........................................ 36

3.2 Two-way scatter plots of a simulated censored data ................. 45

3.3 True and predicted binary activity ........................................ 49

4.1 \( H_{CV} \) and \( H_{TS} \) as functions of \( \Delta_{\text{min}} \) for the 20 complete data sets. Each data set corresponds to a panel. The solid lines are \( H_{TS} \) and the dotted lines are \( H_{CV} \) .................................................. 58

4.2 \( H_{CV} \) and \( H_{TS} \) as functions of \( \Delta_{\text{min}} \) for the 20 censored data sets. Each data set corresponds to a panel. The solid lines are \( H_{TS} \) and the dotted lines are \( H_{CV} \) .................................................. 62

4.3 \( H_{CV} \) and \( H_{TS} \) as functions of \( \Delta_{\text{min}} \). left panels = complete data, right panels = censored data. \( H_{TS} \) = solid lines (first row), \( H_{CV} \) = dotted lines (second row). The average curves are overlayed in the 3rd row. ................. 64
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Tree-structured Methods for Univariate Response</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Node profiles of nodes N00, N01 and N1 in Figure 1.1</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Mean, median, $c_{\text{max}}$ and $\hat{\rho}(c_{\text{max}})$. The relationship Median $&lt;$ Mean $&lt;$ $c_{\text{max}}$ verifies Proposition 2.2.</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>The relative information loss due to discretization. The numbers listed are the ratios of the $F$-values for MultiSCAM and the majority-vote method.</td>
<td>29</td>
</tr>
<tr>
<td>3.1</td>
<td>Two artificial data sets: The left table is the “true” data and the right table is the “observed” data. A star (*) indicates the cell is censored.</td>
<td>38</td>
</tr>
<tr>
<td>3.2</td>
<td>An example of the real response matrix $Z$ and its censoring indicator matrix $D$.</td>
<td>43</td>
</tr>
<tr>
<td>3.3</td>
<td>An example of the simulated complete response matrix $Y^{(\text{com})}$ and its censoring indicator matrix $C$.</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>An example of the simulated response matrix $Y^{(\text{cen})}$. A number in bold indicates that the corresponding cell is censored.</td>
<td>44</td>
</tr>
<tr>
<td>3.5</td>
<td>$2 \times 2$ table for computing hit rate.</td>
<td>48</td>
</tr>
<tr>
<td>4.1</td>
<td>Resubstitution hit rates of MultiSCAM and the majority-vote method for 20 complete and censored data sets. The 95% CI of $H^{(\text{com,MV})} - H^{(\text{com,MSCAM})}$ is $(0.007, 0.019)$; the 95% CI of $H^{(\text{cen,MV})} - H^{(\text{cen,MSCAM})}$ is $(0.021, 0.033)$.</td>
<td>53</td>
</tr>
<tr>
<td>4.2</td>
<td>Test-sample hit rates comparison for the 20 complete data sets. The 95% CI of $H^{(\text{com,MV})} - H^{(\text{com,MSCAM})}$ is $(-0.002, 0.012)$.</td>
<td>56</td>
</tr>
<tr>
<td>4.3</td>
<td>The effect of $\Delta_{\text{min}}$ on hit rate for the 20 complete data sets. The trees are built at 5 different values of $\Delta_{\text{min}}$: $\Delta_{\text{opt}}, \Delta_{\text{opt}} \pm 1, 3$ and 48.</td>
<td>57</td>
</tr>
<tr>
<td>4.4</td>
<td>Test-sample hit rates comparison for the 20 censored data sets. The 95% confidence interval of $H^{(\text{cen,MV})} - H^{(\text{cen,MSCAM})}$ is $(0.018, 0.039)$.</td>
<td>60</td>
</tr>
<tr>
<td>4.5</td>
<td>The effect of $\Delta_{\text{min}}$ on hit rate for the 20 censored data sets. The trees are built at five different values of $\Delta_{\text{min}}$: $\Delta_{\text{opt}}, \Delta_{\text{opt}} \pm 1, 3$ and 48.</td>
<td>61</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

In many applications, multivariate responses are of great interest. When a small number of covariates are associated with these responses and the responses are continuous measurements, both parametric and nonparametric models are available in the literature for modeling the relationship between the responses and the covariates. With a large number of covariates, many of the parametric models do not work well due to dimensionality problems. Keefer (2001) presented a recursive partitioning algorithm for analyzing multivariate continuous responses that is intended to handle a large number of binary covariates. The objective of our work is to investigate the use of a recursive partitioning algorithm for multivariate binary responses.

1.1 A motivating problem

The relationship between molecular structure and biological activity of compounds is of great interest in quantitative structure activity relationship (QSAR) studies. In a
QSAR study, the biological activity of a number of compounds on a few target proteins is measured. The measurement of biological activity is quantitative and continuous, usually in the range of $(3, 10)$ after log transformation. These activity measurements can be arranged in a matrix for which each column corresponds to a target protein and each row corresponds to a compound. Let us call it the response matrix and denote it by $Y$. The molecule structure of the compounds is recorded in terms of atom-pair descriptors. The structure information is placed into a matrix where each row describes a compound and each column represents a particular atom pair molecular feature. The entry at the $i$th row and the $j$th column is 1 if the $j$th atom pair is present in the $i$th compound and 0 otherwise. Therefore, this descriptor matrix (denoted by $X$) is a binary matrix. Note that computational chemists also make continuous molecular descriptors available, but we will not discuss the use of continuous descriptors in this work. To summarize, we have a response matrix $Y \in \mathbb{R}^{n \times q}$ and a descriptor matrix $X \in \{0, 1\}^{n \times p}$, where

- $n$ is the number of compounds involved in the study (typically $500 \leq n \leq 5000$),
- $p$ is the number of atom-pair descriptors (typically $3000 \leq p \leq 8000$),
- $q$ is the number of target proteins (typically $5 \leq q \leq 15$).
Let $X_i$ and $Y_i$ denote the $i$th row of $X$ and $Y$, respectively ($i = 1, \ldots, n$). The following is a typical data layout with $n = 576$, $q = 10$ and $p = 3270$.

\[
Y = \begin{pmatrix}
4.0 & 5.6 & \cdots & 6.1 \\
9.0 & 9.2 & \cdots & 7.9 \\
\vdots & \vdots & \ddots & \vdots \\
8.3 & 4.4 & \cdots & 4.2
\end{pmatrix}_{n \times q}, \quad X = \begin{pmatrix}
1 & 0 & \cdots & 1 \\
0 & 0 & \cdots & 1 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 1 & \cdots & 0
\end{pmatrix}_{n \times p}.
\]

A model for the regression of $Y_i$ on $X_i$ can be used to describe the relationship between structure and activity,

\[E(Y_i|X_i) = f(X_i), \quad i = 1, \ldots, n.\]

Possible candidates of $f(\cdot) \in \mathbb{R}^q$ include

- linear function of $X_i$: $f(X_i) = \beta_0 + \beta_1 X_i$, $\beta_0 \in \mathbb{R}^q, \beta_1 \in \mathbb{R}^{q \times p}$, $i = 1, \ldots, n$.

- piecewise function of $X_i$: $f(X_i) = \alpha_k$ if $X_i \in G_k$ for $k = 1, \ldots, K$, where $\alpha_k \in \mathbb{R}^q$ and $G_1, \ldots, G_K \subset \{0, 1\}^p$ is a partition of the row space of $X$, $i = 1, \ldots, n$.

In QSAR studies, traditional linear models have not received much attention for the following reasons:

(i) The curse of dimensionality (Bellman 1961): the number of descriptors $p$ is usually much greater than the number of compounds $n$.

(ii) The underlying relationships may involve nonlinearities, thresholds and interactions. Although any tree can be written as a linear model with a large number of interaction terms, it is very difficult to start with linear models as we don’t have prior knowledge
about which interactions to include. For example, in the typical data shown in Equation (1.1), there are 3270 descriptors and if up to four-way interactions are initially considered, we need \(3270^4\) interaction terms.

(iii) There may be a mixture of mechanisms where the activity of some compounds follows one model and the activity of other compounds follows a different model. And in this case, it is difficult to interpret the activity model in terms of molecular structure with linear regression models.

Tree-structured approaches have been shown to be successful in overcoming these difficulties. Hawkins et al. (1997) analyzed a large structure-activity data set using recursive partitioning where the response is univariate (i.e. \(q = 1\)). Keefer (2001) investigated the use of recursive partitioning in analyzing structure-activity data with multivariate continuous responses.

In real-world applications, the activity data are usually incomplete. Two main reasons of data incompleteness are: (1) left-censoring due to detection limits; and (2) missing due to difficulty of getting the actual activity (e.g. cost, unavailability of compounds, etc.). The data incompleteness directly affects the accuracy of estimation for population parameters, and leads to invalid statistical inference. Segal (1992) proposed an EM algorithm (under strong assumptions about correlation structure of the response variables) to handle the missing response problem in tree-structured methods. In Segal’s approach, the missing response problem is handled by assuming (1) a first-order autoregressive (AR1) covariance structure of the response variables, (2) a consecutive sequence of missing values, and (3) a missing completely at random (MCAR) mechanism (Laird 1988). However, the
assumptions under which the EM algorithm is carried out are unrealistic in our problem. In an attempt to lessen the effect of data incompleteness, we proceed as follows.

1. Discretize the original response into binary data (1 for active, 0 for inactive),

2. Use a tree-structured method to analyze the resulting binary data.

An explanation of the intuition for doing the binary coding follows. In the data sets we are interested in, about 30% are left-censored and another 10% are missing. We recode the response $Y$ to 0 if $Y < c$ (inactive) and 1 otherwise (active), where $c$ is a constant which is usually greater than the detection limit. Thus, the binary coding is expected to be accurate for the censored portion. As for the missing part, it is doubtful that the “missing at random” assumption is reasonable. In contrast, the missingness mechanism is most likely related to low activity, because less active compounds are more likely censored. Thus it is likely that most of the discretized data are coded correctly. An obvious advantage of doing the discretization is its robustness against censored and missing data. Because interest lies primarily in identifying active compounds, the information lost in the binary coding may not be substantial. However, discretizing the response necessarily entails some information loss and the following questions naturally arise: (1) how much information is lost from complete data, and (2) how well our method handles the data incompleteness problem. We will answer these questions by developing a tree-structured algorithm for analyzing multivariate binary responses and investigating the performance of the algorithm on real and simulated data.
1.2 Available approaches

In this section, we give a brief review of the literature on tree-structured methods for univariate responses, and introduce two currently available approaches to growing trees for multivariate responses. Tree-structured methods recursively partition the feature space, i.e. the row space of $X$, into disjoint regions. The resulting partition can be represented by a *dendrogram* or a *tree* as shown in Figure 1.1. Tree-based methods differ according to the following factors:

1. Response type (continuous/categorical, univariate/multivariate);

2. Covariate type (continuous/categorical);

3. Splitting criteria;

4. Tree size determination (forward/backward selection or stopping/pruning criteria).

![Figure 1.1: A hypothetical tree. The root node (N), which contains 576 observations, is split on $X_{37}$; the left subnode of N (N0), which corresponds to $X_{37} = 0$, is further split on $X_{16}$.](image-url)
1.2.1 Tree-structured methods for univariate response

An early tree-structured method, Automatic Interaction Detection (AID) introduced by Morgan and Sonquist (1963), splits a node on a predictor maximizing the between-node sum-of-squares. Kass (1975) introduced CHi-squared Automatic Interaction Detection (CHAID) which was later developed into Formal Inference-based Recursive Modelling (FIRM) by Hawkins (1982). The splitting criteria of both CHAID and FIRM rely on statistical tests and multiplicity adjustments that can greatly reduce AID’s tendency to overfit the data. Breiman et al. (1984) introduced Classification And Regression Trees (CART), that bases splits on a measure of node impurity. CART uses backward selection to determine the size of tree. In CART, a large tree is pruned using cross-validated, minimal cost-complexity criterion after growing the tree in a forward direction which is usually too large. The following outline of the pruning method in CART is cited from Segal (1988): (1) initially growing a very large tree; (2) iteratively pruning this tree all the way back up (to the root node), thereby creating a nested sequence of trees; (3) selecting the best tree from this sequence using test-sample or cross-validation estimates of error.

The differences and similarities between FIRM and CART for univariate response are summarized in Table 1.1.

1.2.2 Tree-structured methods for multivariate responses

Gillo and Shelly (1974) is probably the earliest work on tree-structured methods for multivariate responses (with an emphasis on survey studies). The authors called it MAID-M, for a multivariate version of the monitored automatic interaction detection (Gillo
Table 1.1: Tree-structured Methods for Univariate Response

<table>
<thead>
<tr>
<th></th>
<th>FIRM</th>
<th>CART</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response</td>
<td>continuous/categorical</td>
<td>continuous/categorical</td>
</tr>
<tr>
<td>Covariate</td>
<td>continuous/categorical</td>
<td>continuous/categorical</td>
</tr>
<tr>
<td>Splitting criteria</td>
<td>$t$-test, $F$-test, $\chi^2$-test</td>
<td>node impurity</td>
</tr>
<tr>
<td>Tree size determination</td>
<td>forward selection, multiplicity adjustment</td>
<td>backward selection, cost-complexity pruning</td>
</tr>
</tbody>
</table>

1972). It is a generalization of Morgan and Sonquist’s original work on AID. Recently, researchers have given attention to tree-structured methods for multivariate responses. Segal (1992) extended the tree-structured regression paradigm to multiple response settings and in particular to longitudinal data. Segal’s method is closely related to CART in that it uses the minimal cost-complexity pruning to determine the tree size. Keefer (2001) discussed an algorithm called MultiSCAM for recursive partitioning of multivariate continuous responses in a QSAR context. By assuming multivariate normality of the response variables, MultiSCAM judges splits by the $p$-value from a two-sample Hotelling’s $T^2$ test, and adjusts the $p$-value for multiple comparisons. The covariate that minimizes the adjusted $p$-value is chosen to split the data. Basically, MultiSCAM evolved from FIRM. The use of multiplicity adjustments in MultiSCAM reduces the tendency of overfitting which was not well handled in MAID-M. For reference, the pseudo-code of MultiSCAM algorithm is shown in Appendix A.1. The algorithm starts with the root node which comprises all the observations. Because the procedures for splitting nodes are identical, we present the algorithm for a generic node $t = \{y_1, \ldots, y_n\}$ with size $n$ in Appendix A.1.

Zhang (1998) generalized the methodology of CART to the case in which multivari-
ate binary responses are of interest. We will call Zhang’s method “the generalized entropy approach.” A distribution from the exponential family is used to fit the multivariate binary data in this approach. The generalized entropy criterion is defined as the maximum of the log-likelihood derived from this distribution. Given this definition of node homogeneity (or node impurity), the rest of the approach is identical to CART.

Though conceptually simple and statistically meaningful, both MultiSCAM and the generalized entropy approach have limitations. For MultiSCAM, if the normality assumption is violated (e.g. skewness in responses), it is questionable that the Hotelling’s $T^2$ test has the right size. Figure 1.2 shows the histograms of 10 response variables in a real QSAR data set. Skewness in responses is obvious in this case. Although in the presence of non-normality, one can take remedial steps to transform the data to normality in the univariate case, it is sometimes difficult to treat multivariate non-normality by transformation methods. Mardia (1975) pointed out that the Hotelling’s $T^2$ test is more sensitive to skewness than to kurtosis. In addition, data incompleteness cannot be easily handled in MultiSCAM. Zhang’s generalized entropy approach involves computing the maximum likelihood estimates of the parameters in the multivariate binary distribution model for every possible split. The procedure for finding the MLE has to be carried out through iteration, because there is no closed-form solution. When thousands of covariates are involved, the computational burden is enormous.

Segal (1988) investigated regression trees for censored responses in the univariate case. The splitting rules are based on the rank statistics for censored data. The reader is referred to Segal’s original proposal for a detailed discussion. The difficulty of extending
Figure 1.2: Histograms of 10 response variables in a real QSAR data set. The skewness in response variables is obvious. The pole at 4.0 in the plot for variable $Y_4$, for example, is a result of single imputation (4.0 at censored values).

Segal’s approach to the multivariate situation partly results from the lack of a natural way to order multivariate data.

In an attempt to solve the censoring and missing data problem in the data sets for which MultiSCAM may lead to unreliable conclusions, we propose a recursive partitioning
algorithm for analyzing multivariate binary responses that are obtained by discretizing the continuous responses. Here, it is necessary to emphasize that discretizing does not completely solve the incomplete data problem. Rather, it alleviates the problem as discussed in Section 1.1. We expect that the price we pay on information loss (due to discretization) will be rewarded by a more valid and robust inference.

The majority-vote algorithm is described in Chapter 2. Chapter 3 discusses algorithms for simulating complete and censored response data and evaluations rules for simulation study. Results of a simulation study are presented in Chapter 4. Chapter 5 concludes and discusses directions of future work. The summaries of algorithms and proofs of propositions appear in Appendices.
Chapter 2

The Majority-vote Method

2.1 Notations and definitions

We denote the covariate matrix by $X \in \{0, 1\}^{n \times p}$ where $p$ covariate variables (columns) are measured on $n$ individuals (rows). The response matrix is denoted by $Y \in \{0, 1\}^{n \times q}$ where $q$ response variables (columns) are observed on $n$ individuals (rows). For the $i$th individual ($i = 1, \ldots, n$), the covariates and responses are $X_i = (X_{i1}, \ldots, X_{ip})$ and $Y_i = (Y_{i1}, \ldots, Y_{iq})$, respectively. A typical set of data looks like

\[
\begin{pmatrix}
Y_{11} & Y_{12} & \cdots & Y_{1q} \\
Y_{21} & Y_{22} & \cdots & Y_{2q} \\
\vdots & \vdots & \ddots & \vdots \\
Y_{n1} & Y_{n2} & \cdots & Y_{nq}
\end{pmatrix},
\begin{pmatrix}
X_{11} & X_{12} & \cdots & X_{1p} \\
X_{21} & X_{22} & \cdots & X_{2p} \\
\vdots & \vdots & \ddots & \vdots \\
X_{n1} & X_{n2} & \cdots & X_{np}
\end{pmatrix},
\]

where the $X_{ij}$ and $Y_{ik}$ are either 0 or 1. For convenience, the column vectors of $X$ and $Y$ are denoted by $X_{(j)}$ ($j = 1, \ldots, p$) and $Y_{(k)}$ ($k = 1, \ldots, q$), respectively. For the types of problems we are interested in, $n$ is usually between 500 and 5000, $p$ is between 3000 and
Definition 2.1 A node $t$ is a subset of the data. A node that comprises all the response data is called the root node.

$$\text{root node} = \{Y_i : i = 1, \ldots, n\}$$

Definition 2.2 The size of node $t$, $n(t)$, is the number of response vectors in node $t$.

Definition 2.3 The profile of node $t$ is the mean response vector of node $t$,

$$(p_1(t), p_2(t), \ldots, p_q(t)),$$

where

$$p_k(t) = \frac{1}{n(t)} \sum_{Y_i \in \text{node } t} Y_{ik}.$$

Definition 2.4 Majority-vote prediction: Each response vector in a node $t$ is predicted by a binary $q$-vector $\hat{Y}(t) = (\hat{Y}_1(t), \ldots, \hat{Y}_q(t))$, where

$$\hat{Y}_k(t) = I(p_k(t) \geq 1/2)$$

is the mode of the $k$th response variable in node $t$.

Note. $\hat{Y}(t)$ in Definition 2.4 is indeed a multivariate mode defined by univariate modes.

Example 1. Suppose node $t$ comprises five observations of $q=4$ response variables:

$$\begin{pmatrix}
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 \\
0 & 1 & 0 & 0
\end{pmatrix}$$
then the size of $t$ is $n(t) = 5$ and $\hat{Y}(t) = (0, 1, 0, 1)$.

In the univariate response case (i.e. $q = 1$, each response is a scalar), CART uses the following measures of node impurity.

1. Misclassification Rate: $\{n(t)\}^{-1} \sum_{Y_i \in t} I \left(Y_i \neq \hat{Y}_1(t)\right)$,

2. Gini Index: $2p_1(t)(1 - p_1(t))$,

3. Cross Entropy: $-p_1(t) \log p_1(t) - (1 - p_1(t)) \log(1 - p_1(t))$,

Typically, either Gini index or cross entropy is used as the basis of the splitting criteria, while misclassification rate is used for evaluating the quality of a tree. Theoretically, distribution-based generalizations of these measures to the case of multivariate responses require modeling the multivariate binary vector. Three approaches for directly modeling multivariate binary distributions are discussed in Cox (1972):

(i) **Independent variables.** The component binary responses are treated as independent variables.

(ii) **Arbitrary multinomial distributions.** The sample is treated as a multinomial one with $2^q - 1$ independent parameters corresponding to the $2^q$ distinct combinations.

(iii) **Logistic models.** This kind of models is intermediate between (i) and (ii), allowing the presence of special kinds of dependence. Let $Y_i$ be the $i$th response variable and $Z_i = 2Y_i - 1$, so that the $Z$’s take values $\pm 1$ ($i = 1, \ldots, q$). Suppose that

$$\log P(Z_1 = z_1, \ldots, Z_q = z_q) = \sum_i \alpha_i z_i + \sum_{i<j} \alpha_{ij} z_i z_j + \cdots + \alpha_{12\ldots q} z_1 \cdots z_q - \Lambda, \quad (2.1)$$
where $\Lambda$ is a normalizing constant. If only the first degree terms are included, we have the independence model (i), whereas if all terms up to $z_1 \cdots z_q$ are taken we have the general multinomial model (ii).

Zhang (1998) generalized the cross entropy criterion of CART using a special form of Equation (2.1):

$$\log P(Z_1 = z_1, \ldots, Z_q = z_q) = \sum_i \alpha_i z_i + \beta \sum_{i<j} z_i z_j - \Lambda,$$

where only one parameter $\beta$ is used for two-way cross products. In this case, the procedure for estimating the parameters has to be carried out through iteration for each possible split. Thus, when thousands of covariates are involved, the computation task is formidable. The independence model (i) should not be appropriate in our case, because correlations among response variables are expected due to structural similarities among the proteins. The general multinomial model (ii) is only applicable when $n$, the number of observations, is fairly large and $q$ is fairly small, so that each cell contains a reasonable number of observations. In a typical data in our study, e.g., the data given by Equation (1.1), $n = 576$, $q = 10$, and $2^q = 1024$. Thus many cells contain no observations.

While there are always compromises between model complexity and accuracy, a simpler model would be more computationally feasible in large scale data mining problem. For this purpose, we investigate the use of simple generalizations of the misclassification rate and Gini index in tree-structured methods for multivariate responses. A simple generalization of the misclassification rate follows.
Definition 2.5  The Generalized Misclassification Rate of node $t$ is defined as

$$ R(t) = \frac{1}{qn(t)} \sum_{k=1}^{q} \sum_{Y_i \in t} I(Y_{ik} \neq \hat{Y}_k(t)) = \frac{1}{q} \sum_{k=1}^{q} R_k(t) \quad (2.3) $$

where

$$ R_k(t) = \frac{1}{n(t)} \sum_{Y_i \in t} I(Y_{ik} \neq \hat{Y}_k(t)) $$

Remarks. $R_k(t) = \min(p_k(t), 1 - p_k(t))$. If $p_k(t) \geq 1/2$, then $\hat{Y}_k(t) = 1$ and $Y_{ik} \neq \hat{Y}_k(t)$ is equivalent to $Y_{ik} = 0$, hence $R_k(t) = 1 - p_k(t)$. Otherwise, if $p_k(t) < 1/2$, then $\hat{Y}_k(t) = 0$ and $Y_{ik} \neq \hat{Y}_k(t)$ is equivalent to $Y_{ik} = 1$, hence $R_k(t) = p_k(t)$. Therefore,

$$ R_k(t) = \min(p_k(t), 1 - p_k(t)) \quad (2.4) $$

Example 2. For the node given in Example 1, $R(t) = \frac{1}{4(5)}(2 + 1 + 1 + 2) = 0.3$. □

The generalized misclassification rate is just the average of (univariate) misclassification rates of each $Y_{(k)}$ under the majority-vote criterion. Similarly, a simple generalization of Gini index is given by

Definition 2.6  The Generalized Gini Index of node $t$:

$$ G(t) = \frac{1}{q} \sum_{k=1}^{q} 2p_k(t)(1 - p_k(t)), $$

where $p_k(t) = \{n(t)\}^{-1} \sum_{Y_i \in t} Y_{ik}$.

2.2 Growing a tree

In this section, we describe a splitting rule which is based on the generalized misclassification rate $R(t)$. This is essentially identical to the splitting rules used in CART and Zhang’s generalized entropy criterion, except that we use a different node impurity
measure. Because the split procedures for every node are identical, we present the algorithm for a generic node $t$. Recall that we restrict ourselves to the case in which all covariate variables are binary.

Because the covariates are binary, each covariate corresponds to a unique split of node $t$, according to its value: a split of node $t$ on variable $X_{(j)}$ results in two subnodes of $t$, the left subnode $t_L(j)$ and the right subnode $t_R(j)$, where

$$t_L(j) = \{Y_i \in t : X_{ij} = 0\} \text{ and } t_R(j) = \{Y_i \in t : X_{ij} = 1\}.$$  

**Proposition 2.1** $n(t)R(t) \geq n(t_L(j))R(t_L(j)) + n(t_R(j))R(t_R(j)), \forall j \in \{1, \ldots, p\}.$

The proof of Proposition 2.1 appears in Appendix B.1. Note that $n(t)R(t)$ measures the total node impurity of node $t$, and the right-hand side is the total node impurity of two resulting subnodes of $t$. Proposition 2.1 indicates that the total node impurity never increases after a split. The following example illustrates this idea.

**Example 3.** Refer to Example 1. Suppose we want to split the node on covariate $X_{(1)} = (0, 1, 0, 0, 1)^T$, then the resulting subnodes will be $t_L(1) = \{Y_1, Y_3, Y_4\}$ and $t_R(1) = \{Y_2, Y_5\}$, i.e.,

$$t_L(1) = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}, \quad t_R(1) = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}. $$

We have $R(t_L(1)) = \frac{1}{(4)(3)}(1+1+1+0) = 0.25$ and $R(t_R(1)) = \frac{1}{(4)(2)}(0+0+0+0) = 0$. Note that $n(t)R(t) = 5 \times 0.3 = 1.5$, $n(t_L(1))R(t_L(1)) = 3 \times 0.25 = 0.75$ and $n(t_R(1))R(t_R(1)) = 2 \times 0 = 0$. Here,

$$1.5 = n(t)R(t) > n(t_L(1))R(t_L(1)) + n(t_R(1))R(t_R(1)) = 0.75.$$
That is, every possible split results in “purer” subnodes.

The “best” split of node $t$ is defined as the variable $x_{(j_0)}$ for which the decrease in node impurity is maximized, or,

$$j_0 = \arg \max_j \{ n(t) R(t) - [n(t_L(j)) R(t_L(j)) + n(t_R(j)) R(t_R(j))].$$

Let us denote the maximum decrease by $\Delta R(t)$. i.e.

$$\Delta R(t) = n(t) R(t) - [n(t_L(j_0)) R(t_L(j_0)) + n(t_R(j_0)) R(t_R(j_0))].$$ (2.5)

Having found the best split $j_0$ for the node $t$, we repeat this procedure for each resulting subnode $t_L$ and $t_R$ until some prespecified threshold is reached (e.g. the size of a node is less than 5, or $\Delta R(t) < 10$). By this algorithm, recursively partitioning the root node will eventually result in a tree $T$, the size of which (i.e. number of terminal nodes) can be controlled by adjusting the threshold values of node size and/or $\Delta R(t)$. The tree built can then be used to interpret the relationship between $Y$ and $X$. We illustrate this in the following example.

**Example 4.** Consider the tree shown in Figure 1.1, where we have $q = 5$ response variables.

Assume the profiles of node N00, N01 and N1 are as shown in Table 2.1. Note that $p_k(t)$ is the proportion of 1s in node $t$ for the $k$th response variable $Y_k$. Recall that the value 1 corresponds to active response. For example, the profile of node N1 is (0.9, 0.2, 0.3, 0.1, 0.1),

<table>
<thead>
<tr>
<th>Node</th>
<th>$p_1(t)$</th>
<th>$p_2(t)$</th>
<th>$p_3(t)$</th>
<th>$p_4(t)$</th>
<th>$p_5(t)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>N00</td>
<td>0.1</td>
<td>0.8</td>
<td>0.2</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>N01</td>
<td>0.3</td>
<td>0.4</td>
<td>0.9</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>N1</td>
<td>0.9</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>

1Strictly speaking, the total impurity after a split is less than or equal to the total impurity before a split.
i.e., of all the 248 compounds in node N1 (refer to Figure 1.1), \(248 \times 0.9 \approx 223\) are active against the first protein (P1), \(248 \times 0.2 \approx 49\) the second, and so on. Thus, we could say that node N1 is highly selective against P1, i.e., most compounds in node N1 are active on P1, but not active on other proteins. On the other hand, all the compounds in node N1 have a common feature: \(X_{37} = 1\). One can interpret the selectivity of node N1 as a result of having chemical feature \(X_{37}\). Other nodes can be explained in the same way. The node profiles in Table 2.1 are visualized in Figure 2.1.

![Figure 2.1: Node profiles of N00, N01 and N1 in Figure 1.1. The height of each bar represents the proportion of active compounds.](image)

For reference, pseudo code of the majority-vote method for growing a tree is given in
2.3 Discretizing the response

This section discusses how to choose the cutoff for discretizing the response. In general, discretizing the response will reduce information. Thus, the quality of the resulting binary data will affect the analysis. Hence, choosing the “best” cutoff is important. So that the discretized data preserves as much information as possible, we choose the cutoff for which the correlation between the continuous response and the corresponding binary response is maximized.

Let us first consider discretizing a continuous random variable $Y$. Let

$$Z_c = I(Y \geq c)$$

denote the discretized binary random variable, where $c$ is the cutoff point. The correlation coefficient between $Y$ and $Z_c$

$$\rho(c) = \text{corr}(Y, I(Y \geq c)) = \frac{E((Y - E(Y))Z_c)}{\sqrt{\text{Var}(Y)\text{Var}(Z_c)}}$$

(2.6)

is a measure of the information of $Y$ preserved in $Z_c$. Define $c_{\text{max}}$ as the value of $c$ that maximizes $\rho(c)$, i.e.

$$c_{\text{max}} = \text{arg max}_c \rho(c) = \text{arg max}_c \text{corr}(Y, I(Y \geq c)).$$

Although it is generally difficult to find a closed-form solution of $c_{\text{max}}$ for a given $Y$, under fairly general conditions, $c_{\text{max}}$, $E(Y)$ and $\text{Median}(Y)$ satisfy the following relationship.

**Proposition 2.2** If $\rho(c)$ has a unique local maximum, i.e., $\rho'(c) = 0$ has a unique solution, then
1. if $E(Y) > \text{Median}(Y)$ then $c_{\text{max}} > E(Y)$,

2. if $E(Y) < \text{Median}(Y)$ then $c_{\text{max}} < E(Y)$,

3. if $E(Y) = \text{Median}(Y)$ then $c_{\text{max}} = E(Y)$.

Hence, $c_{\text{max}}$ is related to the skewness in $Y$. The proof appears in Appendix B.2.

**Example.** To illustrate how $c_{\text{max}}$ is related to the mean and median of the response variable, let us choose part of the real data set given in Equation (1.1), where only the
first five response variables $Y_1, \ldots, Y_5$ are considered. The response matrix consists of activity data of 576 compounds against 5 proteins. For the $j$th ($j = 1, \ldots, 5$) response, we discretize $Y_{ij}$ into $I(Y_{ij} \geq c)$ ($i = 1, \ldots, 576$). The best cutoff $c_{j, \text{max}}$ for the $j$th response can be obtained by maximizing the sample correlation $\hat{\rho}_j(c)$ between $Y_{ij}$ and $I(Y_{ij} \geq c)$ ($i = 1, \ldots, 576$). Figure 2.2 shows plots of functions $\hat{\rho}_j(c)$ vs $c$ for this $576 \times 5$ response data. The corresponding sample mean, median, $c_{\text{max}}$ and $\rho(c_{\text{max}})$ are listed in Table 2.2.

Table 2.2: Mean, median, $c_{\text{max}}$ and $\hat{\rho}(c_{\text{max}})$. The relationship Median $<$ Mean $<$ $c_{\text{max}}$ verifies Proposition 2.2.

<table>
<thead>
<tr>
<th>Variable</th>
<th>$Y_1$</th>
<th>$Y_2$</th>
<th>$Y_3$</th>
<th>$Y_4$</th>
<th>$Y_5$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c_{\text{max}}$</td>
<td>4.86</td>
<td>5.51</td>
<td>6.09</td>
<td>5.85</td>
<td>4.79</td>
</tr>
<tr>
<td>Mean</td>
<td>4.58</td>
<td>5.32</td>
<td>5.97</td>
<td>4.98</td>
<td>4.56</td>
</tr>
<tr>
<td>Median</td>
<td>4.23</td>
<td>5.17</td>
<td>5.75</td>
<td>4.94</td>
<td>4.35</td>
</tr>
<tr>
<td>$\rho(c_{\text{max}})$</td>
<td>0.84</td>
<td>0.86</td>
<td>0.89</td>
<td>0.82</td>
<td>0.82</td>
</tr>
</tbody>
</table>

The ordering of the means, medians and $c_{\text{max}}$ values in Table 2.2 are consistent with the ordering in Proposition 2.2.

2.4 Evaluating the relative information loss due to discretization

Although the algorithm for discretizing the response discussed in Section 2.3 possesses some optimization properties, information loss is inevitable in our approach. To give a preliminary assessment of the relative information loss due to discretization, the majority-vote method is compared with MultiSCAM using multivariate analysis of variance (MANOVA). Here, MultiSCAM is used as a benchmark, because it is designed to handle
complete data and it uses all the information contained in the response data. We first discuss how MANOVA can be used to compare two different partition rules: MultiSCAM and the majority-vote method, then illustrate the idea with an example of real data analysis.

2.4.1 Multivariate Analysis of Variance (MANOVA)

Both MultiSCAM and the majority-vote method partition the sample into distinct classes identified by the terminal nodes. Therefore, each of the two methods can be viewed as a partitioning rule. A multivariate analysis of variance (MANOVA) on the resulting partition provides an overall assessment of the quality of the partition. This suggests a way of comparing MultiSCAM versus majority-vote method. First, let us give a brief review of MANOVA.

Suppose $n$ observations $\{y_1, \ldots, y_n\} \subset \mathbb{R}^q$ are partitioned into $K$ classes, $(t_1, \ldots, t_K)$, by a classifier. The $k$th class, $t_k$, consists of $n_k \left( \sum_{k=1}^{K} n_k = n \right)$ observations: $\{y_{i}^{(k)} \}_{i=1}^{n_k}$. Let us assume $\{y_{i}^{(k)} \}_{i=1}^{n_k}$ are identically independently distributed within class $t_k$ with $E \left( y_{i}^{(k)} \right) = \mu_k$ and are independent of observations in other classes. Now, let us consider testing the hypothesis that there are no node effects, i.e.,

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_K.$$

Under the homoscedasticity assumption, the classical test statistics for $H_0$ include Wilk’s $\Lambda$, Pillai’s trace (PT), Hotelling-Lawley’s trace (HLT) and Roy’s greatest root (RGR) (?). These test statistics, or the $p$-values associated with them, are measures of between-node heterogeneity with respect to within-node homogeneity. Hence they can be used to evaluate the overall quality of a partition. Because each of the four statistics can be transformed
to an $F$-statistic, it is expected that a better partition would yield a larger $F$-value (or a smaller $p$-value).

Now, we analyze the real data (continuous response) using two different methods:

1. Use MultiSCAM to build a tree with $K$ terminal nodes, $(t_1, \ldots, t_K)$, and compute the corresponding MANOVA test statistics, i.e., $\Lambda$, PT, HLT and RGR;

2. Discretize the response into binary variables using the algorithm described in Section 2.3 and then build a tree using the algorithm described in Section 2.2. We control the tree size by adjusting the stopping rules so that exactly $K$ terminal nodes are produced. This is to eliminate the effect of model complexity. Let us denote these nodes by $(t'_1, \ldots, t'_K)$ and compute the corresponding MANOVA statistics, i.e., $\Lambda$, PT, HLT and RGR.

Note that the two tree sizes are exactly the same ($K$). This in turn ensures the equivalence of the degrees of freedom of the corresponding $F$-statistics, for example, the Pillai’s trace (PT):

$$
\text{numerator degrees of freedom of PT on } (t_1, \ldots, t_K) = \text{numerator degrees of freedom of PT on } (t'_1, \ldots, t'_K)
$$

and

$$
\text{denominator degrees of freedom of PT on } (t_1, \ldots, t_K) = \text{denominator degrees of freedom of PT on } (t'_1, \ldots, t'_K),
$$

where PT can be replaced by any of the four MANOVA statistics, $\Lambda$, PT, HLT and RGR. Given the equivalence of degrees of freedom, $F$-statistics for two trees are comparable. In-
tuitively, a larger $F$-value means a better partitioning. Because MultiSCAM is presumably better than majority-vote for continuous response (without information loss), it is expected that MultiSCAM will yield larger $F$-values. The ratio of the corresponding $F$-values, e.g.,

$$\frac{F\text{-value for } \Lambda \text{ on } (t'_1, \ldots, t'_K)}{F\text{-value for } \Lambda \text{ on } (t_1, \ldots, t_K)}$$

is a measure of the relative information loss due to discretization.

### 2.4.2 Example: an analysis on a $576 \times 10$ data

**A description of the data set.** The activities of 576 candidate compounds against 10 target proteins were measured by pIC$_{50}$, where IC$_{50}$ is the 50 percent inhibition concentration, a commonly used activity measure, and pIC$_{50} = -\log_{10}(IC_{50})$. The pIC$_{50}$ values range roughly from 3 to 10. About 30% of the data are known to be left-censored due to detection limits, while another 10% are missing due to various reasons (e.g. cost, unavailability of compounds, etc.). In practice, most of the incomplete data cells (both censored and missing) are imputed by 4.0, which is usually the value for the detection limit. Figure 2.3 displays a two-way scatter plots of the 10 response variables.

Recall that the molecular descriptor matrix is fully observed, that is, the structure of every compound in terms of atom pair descriptors is known.

**An analysis with MultiSCAM.** Part of the tree generated by MultiSCAM on the data set described in Figure 2.3 is shown in Figure 2.4.

In Figure 2.4, each rectangular box corresponds to a node. The node on the top is the root node. The following information can be read from each non-terminal node (we take the root node as an example):
Figure 2.3: Two-way scatter plot of the 10 response variables. There is an apparent linear relationship between $Y_2$ and $Y_5$. The dots on a straight line in other plots correspond to the imputed values (e.g. 4.0).

1. $n = 576$ on top of the bar chart indicates that there are 576 observations in this node.

2. rP and aP below the bar chart are the raw $p$-value and Bonferroni-adjusted $p$-value for the two-sample Hotelling’s $T^2$ test, respectively. (see Appendix A.1 for details)
Figure 2.4: **Part of the tree generated by MultiSCAM on the 576×10 data set**

3. N at the bottom of the box is the name of the node. The left subnode of N is N0, the right subnode of N is N1, the left subnode of N0 is N00, the right subnode of N0 is N01, and so on. Thus, every node has a unique name and the position of a node in the tree can be identified by its name.

4. AP:O(1,1)-5-N(2,0) below the box indicates the splitting variable (atom pair). That is, all compounds that have this atom pair in their structure will go to the right subnode, while all the others will go to the left subnode.

Due to space limitations, the picture shown is only part of the actual tree, i.e. there are further splits of these nodes. The actual tree has 19 terminal nodes. Using SAS/PROC GLM, the multivariate analysis of variance of node effects outputs the following test statistics and their p-values.

```sas
proc glm data=tree.scamnode; title "Dataset=MSCAM"; class node;
```
model y1-y10=node/nouni; manova h=node; run;
Dataset=MSCAM

The GLM Procedure
Multivariate Analysis of Variance

MANOVA Test Criteria and F Approximations for
the Hypothesis of No Overall node Effect
H = Type III SSCP Matrix for node
E = Error SSCP Matrix

S=10 M=3.5 N=273

Statistic Value F Value Num DF Den DF Pr > F
Wilks' Lambda 0.01072617 18.15 180 4839.5 <.0001
Pillai's Trace 2.85942858 12.39 180 5570 <.0001
Hotelling-Lawley Trace 9.34754886 28.37 180 3385.5 <.0001
Roy's Greatest Root 5.67091745 175.48 18 557 <.0001

NOTE: F Statistic for Roy's Greatest Root is an upper bound.

This result is to be compared with the result of the majority-vote method (Appendix A.2).

An analysis with the majority-vote method. In order to make a reasonable comparison, we generated a tree using the tree-growing algorithm A.2, where the threshold values are chosen so that the tree size is 19, the same as the tree generated with MultiSCAM. Again, using the PROC GLM, the multivariate analysis of variance of node effects now gives the following result.

proc glm data=tree.mbnode19; title "Dataset=BINARY"; class node;
model y1-y10=node/nouni; manova h=node; run;
Dataset=BINARY

The GLM Procedure
Multivariate Analysis of Variance

MANOVA Test Criteria and F Approximations for
the Hypothesis of No Overall node Effect
H = Type III SSCP Matrix for node
E = Error SSCP Matrix

S=10  M=3.5  N=273

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Value</th>
<th>F Value</th>
<th>Num DF</th>
<th>Den DF</th>
<th>Pr &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wilks’ Lambda</td>
<td>0.03845452</td>
<td>12.06</td>
<td>180</td>
<td>4839.5</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Pillai’s Trace</td>
<td>2.08412728</td>
<td>8.15</td>
<td>180</td>
<td>5570</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Hotelling-Lawley Trace</td>
<td>6.78873099</td>
<td>20.60</td>
<td>180</td>
<td>3385.5</td>
<td>&lt;.0001</td>
</tr>
<tr>
<td>Roy’s Greatest Root</td>
<td>5.06524878</td>
<td>156.74</td>
<td>18</td>
<td>557</td>
<td>&lt;.0001</td>
</tr>
</tbody>
</table>

NOTE: F Statistic for Roy’s Greatest Root is an upper bound.

The ratios of the corresponding F-statistics are listed in Table 2.3.

Table 2.3: The relative information loss due to discretization. The numbers listed are the ratios of the F-values for MultiSCAM and the majority-vote method.

<table>
<thead>
<tr>
<th></th>
<th>MSCAM</th>
<th>MV</th>
<th>MV/MSCAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wilk’s Λ</td>
<td>18.15</td>
<td>12.06</td>
<td>0.6645</td>
</tr>
<tr>
<td>Pillai’s Trace</td>
<td>12.39</td>
<td>8.15</td>
<td>0.6578</td>
</tr>
<tr>
<td>Hotelling-Lawley Trace</td>
<td>28.37</td>
<td>20.60</td>
<td>0.7261</td>
</tr>
<tr>
<td>Roy’s Greatest Root</td>
<td>175.48</td>
<td>156.74</td>
<td>0.8932</td>
</tr>
</tbody>
</table>

The information loss due to discretization as measured by the ratio of F-statistics is roughly about 1/3.

Again, the results shown in this section are preliminary and only provide a rough assessment of the majority-vote method. An important issue, tree size determination, has not been considered so far. Instead, the threshold values (Δ_min and ν_min) have been chosen so that the tree size of the majority-voted method is equal to the tree size of MultiSCAM, in hope of eliminating the effect of model complexity.
2.5 Evaluating the predictivity of a tree: hit rate

One of the main purposes of modeling the relationship between chemical structure and biological activity of compounds is to predict the activity of new compounds, whose structures are known. Hand (1997) gave a good review of evaluation methods for classification rules. Though the misclassification rate is among the primary criteria, we focus mainly on the so-called “hit rate” which is a standard and popular criterion in discovery chemistry. When used as a predictive model, tree-structured methods predict the responses by a binary $q$-vector $\hat{Y}(t)$ (Definition 2.4). The hit rate for a tree is defined as the proportion of truly active responses in the group of predicted active responses. This is illustrated in the following $2 \times 2$ table.

<table>
<thead>
<tr>
<th>Observed</th>
<th>Predicted</th>
<th>$Y_{ik} = 0$</th>
<th>$Y_{ik} = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y_{ik} = 0$</td>
<td>$N_{00}$</td>
<td>$N_{01}$</td>
<td></td>
</tr>
<tr>
<td>$Y_{ik} = 1$</td>
<td>$N_{10}$</td>
<td>$N_{11}$</td>
<td></td>
</tr>
</tbody>
</table>

The hit rate is defined as

$$H = \frac{N_{11}}{N_{01} + N_{11}}.$$ 

Note that

$$N_{01} + N_{11} = \sum_{i=1}^{n} \sum_{k=1}^{q} I(\hat{Y}_{ik} = 1)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{q} \hat{Y}_{ik},$$

and

$$N_{11} = \sum_{i=1}^{n} \sum_{k=1}^{q} I(\hat{Y}_{ik} = 1)I(Y_{ik} = 1)$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{q} \hat{Y}_{ik} Y_{ik},$$
and thus the hit rate $H$ can be written as

$$H = \frac{\sum_{i=1}^{n} \sum_{k=1}^{q} \hat{Y}_{ik} Y_{ik}}{\sum_{i=1}^{n} \sum_{k=1}^{q} \hat{Y}_{ik}}.$$  

(2.7)

Note that Equation (2.7) is a general definition of hit rate, where $\hat{Y}_{ik}$ can be obtained by any predictive models. More specific definitions of hit rate based on simulated data are given in Chapter 3. The materials presented in this section just provide background knowledge for the discussions in Section 2.6.

### 2.6 Determining the tree size

The tree-growing algorithm discussed in Section 2.2 recursively partitions the sample data into disjoint subsets, until some stopping criteria are met. These stopping criteria include the minimal node size (i.e. the number of observations in a node, denoted by $\nu_{\text{min}}$) and the minimal decrease in total node impurity (denoted by $\Delta_{\text{min}}$). Specifically, we stop splitting a node if either its size does not exceed $\nu_{\text{min}}$ or after every possible split, the decrease in total node impurity does not exceed $\Delta_{\text{min}}$. As a result, the size of the resulting tree (i.e. total number of terminal nodes) is determined by $\nu_{\text{min}}$ and $\Delta_{\text{min}}$ in the following way: smaller values of $\nu_{\text{min}}$ and $\Delta_{\text{min}}$ result in larger trees.

As pointed out in Breiman et al. (1984), too large a tree will suffer from overfitting and too small a tree will not use some of the classification information available in the sample and hence suffers from underfitting. Therefore, determining tree size is important. Because hit rate is the major concern, the optimal model is one that maximizes the true hit rate. However, as with many measures of goodness-of-fit, the internal measures of the training data alone are usually over optimistic. To honestly estimate the hit rate, cross-validation
is needed. Here, we adopt an approach to determining tree size by choosing the threshold value $\Delta_{\text{min}}$ so that the cross-validated estimate of hit rate is maximized. Let us start by looking at the empirical behavior of hit rate as $\Delta_{\text{min}}$ changes.

First, let us assume the training data $D$ has $n$ observations. The resubstitution estimate of hit rate is calculated as follows:

1. Use all the $n$ observations to construct a tree $T$;
2. Use tree $T$ to get predicted values of all these $n$ data points;
3. Compute the hit rate for $D$ based on the predicted values and the observed values.

Let us call it the resubstitution estimate of hit rate and denote by $H_{RS}$.

The cross-validation estimate of hit rate is calculated in a different fashion:

1. Randomly divide data $D$ into $V$ disjoint subsets of equal sizes: $D_1, D_2, \ldots, D_V$, the sample size of $D_v$ is (approximately) $n/V$;
2. For $v = 1, \ldots, V$, leave $D_v$ out and use the remaining subsets to construct a tree $T_v$, then use tree $T_v$ to predict values of data points in data $D_v$;
3. Compute the hit rate for $D_v$ based on the predicted values and the observed values;
4. If we denote $H(T_v, D_v)$ the hit rate computed this way, then the cross-validation estimate of hit rate is defined as the average of $H(T_v, D_v)$ ($v = 1, \ldots, V$):

$$H_{CV} = \frac{1}{V} \sum_{v=1}^{V} H(T_v, D_v).$$

To illustrate the behavior of these two estimates of hit rate as the tree size changes, we build trees at different values of $\Delta_{\text{min}}$ and a small fixed value of $\nu_{\text{min}} = 5$. Note that
tree size is related to $\Delta_{\text{min}}$. For instance, suppose that $\Delta_{\text{min}}$ is set sufficiently large so that no splits of the root node will result in $\Delta R(t) \geq \Delta_{\text{min}}$ (Equation (2.5)), then the resulting tree only has one terminal node, i.e., the root node. On the other hand, if $\Delta_{\text{min}}$ is set as zero, then a very large tree will be grown. In general, tree size decreases as $\Delta_{\text{min}}$ increases. Figure 2.5 shows a typical pattern of the behavior of $H^{RS}$ and $H^{CV}$ for different values of $\Delta_{\text{min}}$. Two important features are illustrated in Figure 2.5:

1. $H^{RS}$ varies more or less monotonically with $\Delta_{\text{min}}$ (tree size). As $\Delta_{\text{min}}$ increases (tree size decreases), $H^{RS}$ decreases.

2. $H^{CV}$ is small when $\Delta_{\text{min}}$ is either small or large (tree size is large or small) and attains a maximum value for an intermediate value of $\Delta_{\text{min}}$ (tree size).

Based on the empirical behavior of hit rate we observed in Figure 2.5, both too small and too large values of $\Delta_{\text{min}}$ yield low true hit rates. One way of getting the optimal model is to choose $\Delta_{\text{min}}$ to maximize $H^{CV}$. A description of the algorithm for choosing the optimal value of $\Delta_{\text{min}}$ appears in Appendix A.5. To summarize, we outline the steps of constructing an optimal tree from a training sample $D$.

1. For $\Delta_{\text{min}} = 1, \ldots, M$, where $M$ is an arbitrarily large integer, compute the cross-validation estimate of hit rate $H^{CV}$;

2. Choose the value of $\Delta_{\text{min}}$ at which $H^{CV}$ is maximized and denote it by $\Delta_{\text{opt}}$;

3. Construct a tree using the whole training sample $D$ at $\Delta_{\text{min}} = \Delta_{\text{opt}}$.

Note that generally the minimal node size $\nu_{\text{min}}$ also affects tree size. For the algorithm that uses cross-validation to determine tree size, we fix $\nu_{\text{min}} = 1$. 
Figure 2.5: Resubstitution hit rate ($H_{RS}$) and cross-validation hit rate ($H_{CV}$) as functions of $\Delta_{\text{min}}$. The triangles are values of $H_{RS}$ and the circles are values of $H_{CV}$. 
Chapter 3

Simulation Study Design

With complete data, MultiSCAM, which is designed to handle complete continuous responses, should work reasonably well. And in that case, it is expected that MultiSCAM would outperform our method which suffers from information loss due to discretization. However, when data censoring becomes a serious problem as discussed previously, it is not clear which method is better. In order to make a reasonable comparison, we need to know the performance of each method on both complete and censored data. We will compare the two methods in a simulation study.

In this chapter, we discuss algorithms for simulating complete and censored data. Section 3.1 reviews the way MultiSCAM works and the assumptions under which it works best. Section 3.2 describes an algorithm for simulating complete data under assumptions presented in Section 3.1. An algorithm for simulating censored data under a specific assumption on censoring mechanism is given in Section 3.3. New evaluation rules designed particularly for the simulation studies are described in Section 3.4.
3.1 The underlying assumption of MultiSCAM

Let \( Y = (Y_{ik})_{n \times q} \) be a matrix of continuous response and \( X = (X_{ij})_{n \times p} \) a matrix of binary covariates. For each \( j = 1, \ldots, p \), MultiSCAM computes the two-sample Hotelling’s \( T^2 \) statistic where the grouping is based on the value of \( x_{ij} \). The covariate that gives the minimum \( p \)-value of the \( T^2 \) statistic is chosen as the splitting variable. The \( p \)-value is adjusted for multiplicity because many tests are conducted for each split. The procedure is repeated for each resulting subnode until no significant adjusted \( p \)-value is found. For illustration purpose, let us assume the following tree has been built.

![Figure 3.1: A simple MultiSCAM tree](image)

Now, let us consider the question “What kind of data will produce this tree?”. Or in other words, “What is the underlying assumption of MultiSCAM?”. Ideally, if observations in node \( t_k \) are identically independently distributed \( N_q(\hat{\mu}_k, \hat{\Omega}_k) \), \( k = 1, 2, 3 \), where \( \hat{\mu}_k \) and \( \hat{\Omega}_k \) are sample mean vector and covariance matrix of node \( t_k \), and observations in
nodes $t_1, t_2$ and $t_3$ are independent, then MultiSCAM would identify such a tree structure. This is to say

$$Y|X_{37} = 0, X_{16} = 0 \sim N_q(\hat{\mu}_1, \hat{\Omega}_1)$$
$$Y|X_{37} = 0, X_{16} = 1 \sim N_q(\hat{\mu}_2, \hat{\Omega}_2)$$
$$Y|X_{37} = 1 \sim N_q(\hat{\mu}_3, \hat{\Omega}_3)$$

which is essentially the underlying data structure of tree in Figure 3.1. Consequently, we can simulate the response data based on the tree structure given in Figure 3.1 by generating $n_1 = 112$ identically independently distributed random vectors from $N_q(\hat{\mu}_1, \hat{\Omega}_1)$ for $X_{16} = 0$ and $X_{37} = 0$, $n_2 = 216$ from $N_q(\hat{\mu}_2, \hat{\Omega}_2)$ for $X_{16} = 1$ and $X_{37} = 0$, and $n_3 = 248$ from $N_q(\hat{\mu}_3, \hat{\Omega}_3)$ for $X_{37} = 1$. Repeating this process can simulate independent replicates of the response data.

### 3.2 Simulating complete data

The discussion in Section 3.1 tells us how to simulate data under a specific tree structure. All we need are a tree classifier and the information about mean and covariance for each node. An ideal simulation should result in a data set that bears a strong resemblance to the real data. For instance, in real data, many observations are left-censored, if we had observed the true response, i.e., the complete data without censoring, then the true values of censored observations would be less than those in real data. Therefore, the mean response values in complete data should be below those in the real data. Let us start with a real data set for which many observations are left-censored.
The response matrix $Z$ contains measured activities of 576 compounds in 10 essays (proteins). Of these $576 \times 10 = 5760$ activity measures, 2749 (47.7%) are censored to the left. Most (but not all) of the values at which the activity measures are censored are 4.0. The censored values have the following effects on the data:

- For each assay, the overall observed mean activity is greater than the actual mean activity.

- The censored values can inflate sample correlations among assays as shown in the following Example.

**Example.** Let us construct two small data sets of sample size 4: a data without censoring and a censored data as shown in Table 3.1.

<table>
<thead>
<tr>
<th>Obs</th>
<th>$Z_1$</th>
<th>$Z_2$</th>
<th>Obs</th>
<th>$Z_1$</th>
<th>$Z_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.5</td>
<td>5.5</td>
<td>1</td>
<td>5.5</td>
<td>5.5</td>
</tr>
<tr>
<td>2</td>
<td>6.0</td>
<td>6.0</td>
<td>2</td>
<td>6.0</td>
<td>6.0</td>
</tr>
<tr>
<td>3</td>
<td>3.5</td>
<td>2.8</td>
<td>3</td>
<td>*4.0</td>
<td>*4.0</td>
</tr>
<tr>
<td>4</td>
<td>3.0</td>
<td>3.5</td>
<td>4</td>
<td>*4.0</td>
<td>*4.0</td>
</tr>
<tr>
<td>$\rho$</td>
<td>0.95</td>
<td>$\rho$</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The sample correlation coefficient between $Z_1$ and $Z_2$ for the censored data is one, while the correlation coefficient of the “true” data is 0.95. Although this example is artificial, it illustrates the false correlation problem described above. Therefore, the effect of the imputed values on both mean response and covariance matrix has to be taken into account for simulating the complete data.
First, we grow a tree for the data using MultiSCAM. The tree has 19 terminal nodes. Recall that in the real data, all the censored values are replaced by the detection limits (e.g. 4.0). If we use the mean values of these 19 terminal nodes to simulate the complete data, the overall mean values of the resulting data set will be higher than the actual ones, because the plug-in values (e.g. 4.0) are greater than the actual one. To take this issue into account, we impute randomly generated activity levels for censored data in each of the five terminal nodes with the greatest percentage of censored values. The five nodes chosen each contains more than 60% censored observations. Imputed values are obtained by multiplying each censored value by a random number that has $U(0,1)$ distribution. This step alleviates both the false correlation problem and the problem with overall mean activity level. Thus, we obtain a new response data set by merging all the observations (both treated and untreated) in 19 nodes.

Next, we grow a new tree for the data with randomly imputed activity levels using MultiSCAM and denote the terminal nodes by $t_1, \ldots, t_K$. Denote the sample size, mean vector and covariance matrix for node $t_k$ by $n_k, \hat{\mu}_k$ and $\hat{\Omega}_k$, respectively.

Finally, for node $t_k$, we generate identically independently distributed samples of size $n_k$ from $N_q(\hat{\mu}_k, \hat{\Omega}_k)$, $k = 1, \ldots, K$. Therefore, we get response data from a mixture of multivariate normal distribution under the tree-structured classification $\{t_1, \ldots, t_K\}$. The algorithm is summarized in Appendix A.3. In the future, this algorithm is called the “complete-data generating algorithm” and is referred to as Algorithm A.3.
3.3 Simulating censored data

Having found a way for generating complete response data, we are now ready to consider simulating censored data. Two problems have to be addressed: the censoring mechanism, and the value at which an observation is censored. Before we get into details, let us introduce some notation used throughout this section. Let $Y^{\text{com}} = (Y_{ij}^{\text{com}})_{n \times q}$ be the simulated complete response matrix. The originally observed response matrix (real data) is denoted by $Z = (Z_{ij})_{n \times q}$. Associated with $Z$ is a censoring indicator matrix $D = (D_{ij})_{n \times q}$, where

$$D_{ij} = \begin{cases} 1, & \text{if } Z_{ij} \text{ is censored,} \\ 0, & \text{otherwise.} \end{cases}$$

Let $D_{j} = \sum_{i=1}^{n} D_{ij}$ denote the number of censored observations and $\overline{D}_{j} = D_{j}/n$ the observed censoring proportion in the $j$th column of $Z$. A matrix of independent and identically distributed $U(0, 1)$ random numbers $U = (U_{ij})_{n \times q}$ will be used in simulating the censoring mechanism.

Now, let us discuss the censoring mechanism. It is believed that the probability an observation is censored is related to its actual value. Specifically, we impose an assumption that the left-censoring probability is a decreasing function of the actual value,

$$P(Y_{ij}^{\text{com}} \text{ is censored} | Y_{ij}^{\text{com}} = y_{ij}) = \frac{1}{1 + \beta_{j} e^{y_{ij}}}.$$  

The parameters $\beta_{j}$ can be estimated by solving

$$\frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 + \beta_{j} e^{y_{ij}^{\text{com}}}} = \overline{D}_{j}, \quad j = 1, \ldots, q.$$  

resulting in $\hat{\beta}_{j}$. Based on the values of $\hat{\beta}_{j}$ and $Y_{ij}^{\text{com}}$, we stochastically determine whether
the $ij$th observation is censored or not.

\[
\begin{align*}
    Y_{ij}^{(\text{com})} &\quad \text{is censored,} \\
    Y_{ij}^{(\text{com})} &\quad \text{is not censored,}
\end{align*}
\]

\[
\text{if } \left( 1 + \hat{\beta}_j e^{Y_{ij}^{(\text{com})}} \right)^{-1} > U_{ij},
\]

\[
\text{otherwise.}
\]

In other words, we simulate a censoring indicator matrix $C = (C_{ij})_{n \times q}$ for matrix $Y$, where

\[
C_{ij} = \begin{cases} 
1, & \text{if } \left( 1 + \hat{\beta}_j e^{Y_{ij}^{(\text{com})}} \right)^{-1} > U_{ij}, \\
0, & \text{otherwise.}
\end{cases}
\]

Thus, we give a solution to the first problem, i.e., the censoring mechanism.

In order to simulate the censoring points, let us first consider the censoring points in the real data. Recall that the $j$th column of matrix $D$ indicates whether an observation is censored or not. Therefore, $D_j = \{Z_{ij} : D_{ij} = 1, 1 \leq i \leq n\}$ denotes the set of all censoring points for the $j$th assay in the original data. Similarly, $C_j = \{Y_{ij}^{(\text{com})} : C_{ij} = 1, 1 \leq i \leq n\}$ denotes the set of all observations which should be censored for the $j$th assay in the simulated complete data. Note that our algorithm for simulating the censoring mechanism discussed in this section does not guarantee that the size of $C_j$ (denoted by $C_j$) is equal to the size of $D_j$ (denoted by $D_j$), but it is expected that they are close. Without loss of generality, assume $C_j$ and $D_j$ are sorted in ascending order as follows:

\[
Y_{i_1j}^{(\text{com})} \leq Y_{i_2j}^{(\text{com})} \leq \cdots \leq Y_{i_{C_j}j}^{(\text{com})}
\]

and

\[
Z_{i_lj} \leq Z_{i_l'j} \leq \cdots \leq Z_{i_{D_j}'}.
\]

Our algorithm for simulating the censoring points simply replaces $Y_{i_{k,j}}^{(\text{com})}$ with some $Z_{i_{k'}j}$, where $Z_{i_{k'}j}$ is chosen so that its relative rank in $D_j (k'/D_j)$ matches the relative rank
of \(Y_{ij}^{(\text{com})}\) in \(C_j (k/C_j)\). For example, if \(C_j = D_j\), i.e., the number of censored observations in \(Y^{(\text{com})}\) is equal to the number of censored observations in \(Z\), we simply replace \(Y_{ij}^{(\text{com})}\) with \(Z_{i'j}\) for \(k = 1, 2, \ldots, D_j\). To put this in mathematical forms, let us define the following mappings.

- \(k_j(i)\) is the rank of \(Y_{ij}\) in \(C_j\); the domain of \(k_j(\cdot)\) is \(\{i : c_{ij} = 1, 1 \leq i \leq n\}\); and the range of \(k_j(\cdot)\) is \(\{1, 2, \ldots, C_j\}\).

- \(k'_j(i')\) is the rank of \(Z_{i'j}\) in \(D_j\); the domain of \(k'_j(\cdot)\) is \(\{i' : D_{i'j} = 1, 1 \leq i' \leq n\}\); and the range of \(k'_j(\cdot)\) is \(\{1, 2, \ldots, D_j\}\).

- \(i_j(k)\) is the inverse mapping of \(k_j(i)\).

- \(i'_j(k')\) is the inverse mapping of \(k'_j(i')\)

In order to generate the censoring points, we simply replace \(Y_{ij}^{(\text{com})} \in C_j\) by \(Z_{i'j} \in D_j\), where \(i' = i'_j(k'), k' = \max \{\lfloor k_j(i)D_j/C_j \rfloor, 1\}\), and \([x]\) is the largest integer not greater than \(x\).

The following example illustrates how the replacement is done.

**Example.** For simplicity, let us consider a single response variable. Assume we have observed the \(Z\) matrix and \(D\) matrix with sample size \(n = 10\) as shown in Table 3.2.

Note that \(D_{i1} = 1\) for \(i = 2, 7, 8\), we have \(D_1 = \{Z_{21}, Z_{71}, Z_{81}\} = \{4.00, 3.80, 4.30\}\) and \(D_{1} = 3\) by definition. Accordingly, assume the \(Y^{(\text{com})}\) matrix and \(C\) matrix are simulated as shown in Table 3.3.
Table 3.2: An example of the real response matrix $Z$ and its censoring indicator matrix $D$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Z_{i1}$</th>
<th>$D_{i1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.04</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>4.00</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5.83</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>6.09</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>7.45</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>6.76</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>3.80</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>4.30</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>5.96</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>5.94</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.3: An example of the simulated complete response matrix $Y^{(com)}$ and its censoring indicator matrix $C$

<table>
<thead>
<tr>
<th>$i$</th>
<th>$Y_{i1}^{(com)}$</th>
<th>$C_{i1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.66</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>3.29</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1.99</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>5.73</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>7.32</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>6.84</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>4.07</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>3.69</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>5.78</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>5.11</td>
<td>0</td>
</tr>
</tbody>
</table>

Since $C_{i1} = 1$ for $i = 2, 3, 8, 9$,

$$C_1 = \{Y_{21}^{(com)}, Y_{31}^{(com)}, Y_{81}^{(com)}, Y_{91}^{(com)}\}$$

$$= \{3.29, 1.99, 3.69, 5.78\},$$

and $C_1 = 4$. Now, we rewrite elements in $C_1$ and $D_1$ in ascending order:

$$C_1 = \{1.99, 3.29, 3.69, 5.78\} = \{Y_{31}^{(com)}, Y_{21}^{(com)}, Y_{81}^{(com)}, Y_{91}^{(com)}\}$$
and

\[ D_1 = \{3.80, 4.00, 4.30\} = \{Z_{71}, Z_{21}, Z_{81}\}. \]

To generate the values at which \( Y_{i1}^{(\text{com})} \) (\( i = 2, 3, 8, 9 \)) are censored, we replace the \( k \)th element in \( C_1 \) by the \( k' \)th element in \( D_1 \), where

\[ k' = \max \left\{ \left\lfloor \frac{kD_1}{C_1} \right\rfloor, 1 \right\} = \max \left\{ \left\lfloor \frac{3k}{4} \right\rfloor, 1 \right\}. \]

Or equivalently, the 1st, 2nd, 3rd and 4th elements in \( C_1 \) are replaced by the 1st, 1st, 2nd and 3rd elements in \( D_1 \), respectively. That is, \( Y_{31}^{(\text{com})} \), \( Y_{21}^{(\text{com})} \), \( Y_{81}^{(\text{com})} \) and \( Y_{91}^{(\text{com})} \) are replaced by \( Z_{71} \), \( Z_{71} \), \( Z_{21} \) and \( Z_{81} \), respectively. Finally, the simulated censored response matrix is denoted by \( Y^{(\text{cen})} \), where

\[ Y_{ij}^{(\text{cen})} = Y_{ij}^{(\text{com})} (1 - C_{ij}) + Z_{i'j} C_{ij}. \]

The simulated censored response matrix \( Y^{(\text{cen})} \) is shown in Table 3.4. As a comparison, the values of \( Y_{i1}^{(\text{com})} \) are also listed in Table 3.4.

Table 3.4: An example of the simulated response matrix \( Y^{(\text{cen})} \). A number in bold indicates that the corresponding cell is censored.
The algorithm for simulating the censored data is summarized in Appendix A.4. This algorithm is called the “censored-data generating algorithm” and is referred to as Algorithm A.4.

![Two-way scatter plots of a simulated censored data](image)

Figure 3.2: Two-way scatter plots of a simulated censored data

Figure 3.2 is the two-way scatter plots of the simulated censored data. It bears a
strong resemblance to the real data shown in Figure 2.3.

3.4 Evaluation rules

Using the algorithms discussed in previous sections, we can simulate both complete and censored response matrices. In this section, we discuss rules for evaluating the performance of the majority-vote method on the simulated data.

MultiSCAM, which is designed to handle complete continuous responses, should work reasonably well on the complete data. Thus, it serves as a benchmark in our evaluation. Given a tree built with MultiSCAM, each compound to be predicted belongs to a terminal node, according to its chemical structure (i.e. the values of the covariates). The activities of that compound against all the proteins of interest are then predicted by the mean vector of the node to which the compound belongs. Thus, MultiSCAM yields continuous predictions. In contrast, the majority-vote method discretizes the response and yields binary predictions. In practice, what we are really interested in is whether the activity value is large (active) or not (inactive). Hence, in order to compare the two methods in a common framework, it is reasonable to convert the predictions in MultiSCAM into binary.

Let us start our discussion by introducing some notations and definitions. Let \( Y^{(com)} \) denote a simulated complete data and \( Y^{(cen)} \) be the censored data generated from \( Y^{(com)} \). We use \( Y \) to represent a generic response matrix that can be either complete or censored. Throughout this section, we use superscripts \((com)\) and \((cen)\) for quantities related to complete and censored data, respectively. To proceed with the majority-vote method, we need to discretize \( Y^{(com)} \) and \( Y^{(cen)} \), and the resulting binary response matrices
are denoted by $Y_{ij}^{(bin,com)}$ and $Y_{ij}^{(bin,cen)}$, respectively. The cutoff for discretizing $Y_{ij}^{(com)}$ and $Y_{ij}^{(cen)}$ is denoted by $\tau_j$. i.e. $Y_{ij}^{(bin,com)} = I(Y_{ij}^{(com)} > \tau_j)$ and $Y_{ij}^{(bin,cen)} = I(Y_{ij}^{(cen)} > \tau_j)$. For both the complete data $Y^{(com)}$ and the corresponding censored data $Y^{(cen)}$, the true binary activity is defined as

$$Y_{ij}^{(true)} = Y_{ij}^{(bin,com)} = I\left(Y_{ij}^{(com)} > \tau_j\right),$$

i.e., the binary response converted from the complete data $Y^{(com)}$. In general, in a Multi-SCAM analysis, the prediction of response $Y_{ij}$ is defined as

$$\frac{1}{n(t)} \sum_{k \in t} Y_{kj},$$

where $t$ is the terminal node containing the $i$th compound and $n(t)$ is the size of node $t$. Because the fact that the $i$th compound belongs to node $t$ is totally determined by the corresponding covariate vector $x_i$ (see Figure 3.1), the above prediction can be denoted by

$$\hat{\mu}_j(x_i) = \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(bin)}$$

without any confusion. Similarly, a tree-structured analysis with the majority-vote method yields the following prediction of the proportion of active compounds in node $t$:

$$\hat{\pi}_j(x_i) = \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(bin)}.$$

To convert everything into binary, we define the following binary predictions for $Y_{ij}^{(bin)}$:

$$\hat{Y}_{ij}^{(MSCAM)} = I(\hat{\mu}_j(x_i) > \tau_j), \quad \text{for MultiSCAM}$$

$$\hat{Y}_{ij}^{(MV)} = I\left(\hat{\pi}_j(x_i) > \frac{1}{2}\right), \quad \text{for majority-vote}$$
which are compared with the true binary activity defined in Equation (3.1). Note that we have both complete and censored data ($Y^{(com)}$ and $Y^{(cen)}$), and for each data, two methods (MultiSCAM and the majority-vote method) produce two different predictions, we should have four different types of binary activity predictions. Namely,

\[
\hat{Y}_{ij}^{(com, MSCAM)}, \hat{Y}_{ij}^{(com, MV)}, \hat{Y}_{ij}^{(cen, MSCAM)}, \hat{Y}_{ij}^{(cen, MV)},
\]

where

\[
\hat{Y}_{ij}^{(com, MSCAM)} = I \left( \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(com)} > \tau_j \right), \tag{3.7}
\]
\[
\hat{Y}_{ij}^{(com, MV)} = I \left( \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(bin, com)} > \frac{1}{2} \right) = I \left( \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(true)} > \frac{1}{2} \right), \tag{3.8}
\]
\[
\hat{Y}_{ij}^{(cen, MSCAM)} = I \left( \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(cen)} > \tau_j \right), \tag{3.9}
\]
\[
\hat{Y}_{ij}^{(cen, MV)} = I \left( \frac{1}{n(t)} \sum_{k \in t} Y_{kj}^{(bin, cen)} > \frac{1}{2} \right). \tag{3.10}
\]

The diagram shown in Figure 3.3 illustrates how the four types of predictions are computed. Combining the true binary activity $\hat{Y}_{ij}^{(true)}$ in Equation (3.1) with any of the predicted activities in Equations (3.7)- (3.10), Table 3.5 can be constructed.

<table>
<thead>
<tr>
<th>Predicted</th>
<th>$Y_{ij} = 0$</th>
<th>$Y_{ij} = 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True activity</td>
<td>$Y_{ij}^{(true)} = 0$</td>
<td>$N_{00}$</td>
</tr>
<tr>
<td></td>
<td>$Y_{ij}^{(true)} = 1$</td>
<td>$N_{10}$</td>
</tr>
</tbody>
</table>

Hence the hit rate $H = N_{11}/(N_{01} + N_{11})$ can be computed. The four types of predictions yield four types of hit rate:

$H^{(com, MSCAM)}, H^{(com, MV)}, H^{(cen, MSCAM)}, H^{(cen, MV)}$.
Figure 3.3: True and predicted binary activity

The relevant comparisons will be

\[ H^{(\text{com}, \text{MSCAM})} \text{ vs } H^{(\text{com}, \text{MV})}, \]

and

\[ H^{(\text{cen}, \text{MSCAM})} \text{ vs } H^{(\text{cen}, \text{MV})}. \]
The evaluation rules discussed in this section only apply to simulation studies where the “true” activity is assumed known.
Chapter 4

Results of Simulation Study

This chapter describes results of a simulation study designed to compare Multi-SCAM and the majority-vote method. Complete and censored data are generated using the algorithms in Chapter 3 and performance is measured in terms of the hit rates described in Section 3.4. Section 4.1 compares the resubstitution estimates of hit rates for both MultiSCAM and the majority-vote method, where the $\Delta_{\text{min}}$ is fixed at 7 and $\nu_{\text{min}}$ is fixed at 5. Section 4.2 compares the test-sample estimates of hit rates for both MultiSCAM and the majority-vote method, where $\Delta_{\text{min}}$ is optimized by maximizing the cross-validation estimates of hit rate. Section 4.3 summarizes the results of the simulation study.

4.1 Hit rates comparison: resubstitution estimates

In this section, we first outline the strategy for the simulation experiment, then show the resulting hit rates, and finally, give an explanation of the results.
4.1.1 The design of experiment

Starting with the 576 × 10 pIC$_{50}$ data ($D_0$) introduced in Section 3.1, the complete-data generating algorithm (Appendix A.3, p. 72) outputs a tree ($T_1$) with 17 terminal nodes \{t_1, \ldots, t_{17}\}, after step 3. Let $n_k$, $\hat{\mu}_k$ and $\hat{\Omega}_k$ denote the size, mean vector and covariance matrix of node $t_k$, respectively ($k = 1, \ldots, 17$). In the rest of this section, we will use $n_k$, $\hat{\mu}_k$ and $\hat{\Omega}_k$ ($k = 1, \ldots, 17$) in the simulation without any changes. Repeating step 4 of the complete-data generating algorithm $B = 20$ times, we generate 20 independent simulated complete data sets ($Y^{(com)}_1, \ldots, Y^{(com)}_{20}$). From each of these 20 data sets, we generate a censored data set using the censored-data generating algorithm (Appendix A.4, p. 73). Thus, we get another 20 independent simulated censored data sets ($Y^{(cen)}_1, \ldots, Y^{(cen)}_{20}$). To clarify, the $i$th censored data set $Y^{(cen)}_i$ is generated from the $i$th complete data set $Y^{(com)}_i$ using the censored data generating algorithm; $Y^{(cen)}_i$ is the “observed data” and $Y^{(com)}_i$ is the “true data”.

Analyzing the simulated data sets using MultiSCAM and the majority-vote method results in various hit rates (Equations (3.11)-(3.14)) defined in Section 3.4. For example, analyzing $Y^{(com)}_i$ yields two hit rates: $H^{(com, MSCAM)}_i, H^{(com, MV)}_i$; and analyzing $Y^{(cen)}_i$ yields another two hit rates: $H^{(cen, MSCAM)}_i, H^{(cen, MV)}_i$ ($i = 1, \ldots, 20$). Notice that MultiSCAM always uses forward selection to determine the tree size. That is, given a level of significance $\alpha$ and a minimal node size $\nu_{\min}$, the root node is recursively partitioned until either no tests are significant at level $\alpha$ or the size of the resulting subnode is less than $\nu_{\min}$. In our study, $\alpha = 0.05$ and $\nu_{\min} = 5$. This section focuses on the case where the majority-vote method also uses forward selection to determine the tree size. Specifically,
the tree-growing algorithm (Appendix A.2) is applied with the minimal decrease in total
node impurity $\Delta_{\text{min}} = 7$ and the minimal node size $\nu_{\text{min}} = 5$, where the threshold values are
chosen arbitrarily. It is important to point out that the tree size determination algorithm
(Appendix A.5) is not applied to the analyses discussed in this section. Section 4.2 will deal
with the issue on tree size determination.

4.1.2 The results

Table 4.1: Resubstitution hit rates of MultiSCAM and the majority-vote method for 20 com-
plete and censored data sets. The 95% CI of $H^{(\text{com},\text{MV})} - H^{(\text{com},\text{MSCAM})}$ is (0.007, 0.019); the 95% CI of $H^{(\text{cen},\text{MV})} - H^{(\text{cen},\text{MSCAM})}$ is (0.021, 0.033).

<table>
<thead>
<tr>
<th>Data</th>
<th>Complete</th>
<th>Majority Vote</th>
<th>Censored</th>
<th>Majority Vote</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$H^{(\text{com},\text{MSCAM})}$</td>
<td>$H^{(\text{com},\text{MV})}$</td>
<td>$H^{(\text{cen},\text{MSCAM})}$</td>
<td>$H^{(\text{cen},\text{MV})}$</td>
</tr>
<tr>
<td>1</td>
<td>0.76</td>
<td>0.77</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>2</td>
<td>0.77</td>
<td>0.78</td>
<td>0.76</td>
<td>0.79</td>
</tr>
<tr>
<td>3</td>
<td>0.75</td>
<td>0.76</td>
<td>0.79</td>
<td>0.79</td>
</tr>
<tr>
<td>4</td>
<td>0.77</td>
<td>0.77</td>
<td>0.79</td>
<td>0.81</td>
</tr>
<tr>
<td>5</td>
<td>0.77</td>
<td>0.76</td>
<td>0.77</td>
<td>0.80</td>
</tr>
<tr>
<td>6</td>
<td>0.76</td>
<td>0.76</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
<td>0.79</td>
<td>0.75</td>
<td>0.80</td>
</tr>
<tr>
<td>8</td>
<td>0.77</td>
<td>0.79</td>
<td>0.79</td>
<td>0.80</td>
</tr>
<tr>
<td>9</td>
<td>0.77</td>
<td>0.77</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>10</td>
<td>0.76</td>
<td>0.78</td>
<td>0.78</td>
<td>0.80</td>
</tr>
<tr>
<td>11</td>
<td>0.75</td>
<td>0.75</td>
<td>0.78</td>
<td>0.80</td>
</tr>
<tr>
<td>12</td>
<td>0.75</td>
<td>0.77</td>
<td>0.74</td>
<td>0.79</td>
</tr>
<tr>
<td>13</td>
<td>0.78</td>
<td>0.78</td>
<td>0.75</td>
<td>0.80</td>
</tr>
<tr>
<td>14</td>
<td>0.76</td>
<td>0.78</td>
<td>0.76</td>
<td>0.81</td>
</tr>
<tr>
<td>15</td>
<td>0.75</td>
<td>0.78</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>16</td>
<td>0.78</td>
<td>0.78</td>
<td>0.77</td>
<td>0.79</td>
</tr>
<tr>
<td>17</td>
<td>0.74</td>
<td>0.74</td>
<td>0.76</td>
<td>0.77</td>
</tr>
<tr>
<td>18</td>
<td>0.74</td>
<td>0.77</td>
<td>0.75</td>
<td>0.78</td>
</tr>
<tr>
<td>19</td>
<td>0.76</td>
<td>0.80</td>
<td>0.79</td>
<td>0.82</td>
</tr>
<tr>
<td>20</td>
<td>0.74</td>
<td>0.77</td>
<td>0.75</td>
<td>0.79</td>
</tr>
<tr>
<td>Mean</td>
<td>0.76</td>
<td>0.77</td>
<td>0.77</td>
<td>0.80</td>
</tr>
</tbody>
</table>
Table 4.1 lists the hit rates of MultiSCAM and the majority-vote method for
the 20 complete \(Y_{1\text{,(com)}}, \ldots, Y_{20\text{,(com)}}\) and 20 censored \(Y_{1\text{,(cen)}}, \ldots, Y_{20\text{,(cen)}}\) data sets. As
discussed in Section 3.4, we compare \(H_{\text{com,MSCAM}}\) vs \(H_{\text{com,MV}}\) and \(H_{\text{cen,MSCAM}}\) vs \(H_{\text{cen,MV}}\). Note that within each row, the hit rates are correlated because they are com-
puted from the same data set; but across rows, the hit rates are independent because they
are computed from independent data sets. Accordingly, the paired-data, two-sample \(t\)-test
is used to compare hit rates. The two-tailed \(p\)-value for such a test on \(H_{\text{com,MSCAM}}\) vs
\(H_{\text{com,MV}}\) is \(p = 0.0002\). Note that the sample mean of \(H_{\text{com,MSCAM}}\) (0.76) is less than
that of \(H_{\text{com,MV}}\) (0.77), indicating that the majority-vote method is slightly better than
MultiSCAM in terms of hit rate with the complete data. The 95\% confidence interval for
\(H_{\text{com,MV}} - H_{\text{com,MSCAM}}\) is (0.007, 0.019). Similarly the paired-data two-sample \(t\)-test
on \(H_{\text{cen,MSCAM}}\) vs \(H_{\text{cen,MV}}\) has \(p\)-value < 0.0001, indicating that the majority-vote
method is slightly better than MultiSCAM in terms of hit rate with censored data. The
95\% confidence interval of \(H_{\text{cen,MV}} - H_{\text{cen,MSCAM}}\) is (0.021, 0.033).

The results for both complete and censored data presented in this section indicate
that the majority-vote method is better than MultiSCAM in terms of hit rate. However,
as noted in Section 2.6, there is often a discrepancy between the resubstitution hit rate
and the cross-validated hit rate that better reflects the true hit rate. Because the analyses
conducted in the section eventually have led to the discovery of the tree-size determination
algorithm discussed in Section 2.6, we present the results of the analyses after all.
4.2 Hit rates comparison: test-sample estimates

This section compares test-sample hit rates of MultiSCAM and the majority-vote method. We first outline the procedure of how trees are constructed and then study the performance of the majority-vote method on both complete and censored data.

4.2.1 The design of experiment

In general, the resubstitution estimates are overoptimistic. Although the results in Section 4.1 showed that the majority-vote method outperforms MultiSCAM in terms of hit rate, it is questionable that the resubstitution estimates of either or both methods are honest. We proceed as discussed in Section 2.6 and the following steps are applied to each of the 20 simulated complete data and the 20 simulated censored data. Let \( D \) denote the set of all row vectors of \( Y \), where \( Y \) denotes a simulated response matrix.

1. Each simulated data \( D \) (sample size \( n = 576 \)) is randomly split into two disjoint subsets: a training data \( D^{TR} \) with sample size \( n^{TR} = 400 \) and a test data \( D^{TS} \) with sample size \( n^{TS} = 176 \).

2. For MultiSCAM: Construct a tree using the training data \( D^{TR} \) and then compute the hit rate using the test data \( D^{TS} \). Let us denote this hit rate by \( H^{(MSCAM)}_{TS} \), i.e. the test-sample estimate of hit rate for MultiSCAM. Note that the level of significance \( \alpha = 0.05 \) and the minimal node size \( \nu_{\text{min}} = 5 \) are used and MultiSCAM uses forward selection to determine the tree size.

3. For the majority-vote method: Construct a tree using the training data \( D^{TR} \), where the threshold value \( \Delta_{\text{min}} \) is chosen by the tree-size determination algorithm with 10-
fold cross-validation. The test data $D^{TS}$ is then used to compute the hit rate. Let us denote this hit rate by $H^{(MV)}_{TS}$, i.e. the test-sample estimate of hit rate for the majority-vote method.

### 4.2.2 The results on complete data

Table 4.2: Test-sample hit rates comparison for the 20 complete data sets. The 95% CI of $H^{(com,MSCAM)}_{TS} - H^{(com,MV)}_{TS}$ is $(-0.002, 0.012)$.

<table>
<thead>
<tr>
<th>Data</th>
<th>$H^{(com,MSCAM)}_{TS}$</th>
<th>$H^{(com,MV)}_{TS}$</th>
<th>$\Delta_{opt}$</th>
<th>Tree Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.71</td>
<td>0.71</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>0.69</td>
<td>0.69</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
<td>0.66</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>0.73</td>
<td>0.72</td>
<td>8</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>0.73</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>0.75</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>0.77</td>
<td>0.78</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>8</td>
<td>0.69</td>
<td>0.71</td>
<td>8</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>0.72</td>
<td>0.70</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>0.69</td>
<td>0.69</td>
<td>22</td>
<td>7</td>
</tr>
<tr>
<td>11</td>
<td>0.70</td>
<td>0.75</td>
<td>13</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>0.74</td>
<td>0.74</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>0.71</td>
<td>0.71</td>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>14</td>
<td>0.76</td>
<td>0.76</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>15</td>
<td>0.73</td>
<td>0.76</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>0.76</td>
<td>0.76</td>
<td>9</td>
<td>13</td>
</tr>
<tr>
<td>17</td>
<td>0.70</td>
<td>0.70</td>
<td>23</td>
<td>6</td>
</tr>
<tr>
<td>18</td>
<td>0.76</td>
<td>0.76</td>
<td>11</td>
<td>13</td>
</tr>
<tr>
<td>19</td>
<td>0.73</td>
<td>0.73</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>0.72</td>
<td>0.72</td>
<td>8</td>
<td>19</td>
</tr>
<tr>
<td>Mean</td>
<td>0.722</td>
<td>0.727</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applying steps 1-3 in Section 4.2.1 to the 20 simulated complete data, we get the test-sample hit rates ($H^{(com,MSCAM)}_{TS}$ and $H^{(com,MV)}_{TS}$) shown in Table 4.2. The optimal value of $\Delta_{min}$ ($\Delta_{opt}$), and the tree size of the majority-vote method for each of the 20 complete data sets are also shown in Table 4.2. The two-sample paired $t$-test of
Table 4.3: The effect of $\Delta_{\text{min}}$ on hit rate for the 20 complete data sets. The trees are built at 5 different values of $\Delta_{\text{min}}$: $\Delta_{\text{opt}}$, $\Delta_{\text{opt}} \pm 1$, 3 and 48.

<table>
<thead>
<tr>
<th>Data</th>
<th>$H_{TS}^{(\text{com,SCAM})}$</th>
<th>$H_{TS}^{(\text{com,MV})}$</th>
<th>$\Delta_{\text{min}}$</th>
<th>3</th>
<th>$\Delta_{\text{opt}} - 1$</th>
<th>$\Delta_{\text{opt}}$</th>
<th>$\Delta_{\text{opt}} + 1$</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.70</td>
<td>0.68</td>
<td>0.71</td>
<td>0.71</td>
<td>0.71</td>
<td>0.72</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0.70</td>
<td>0.65</td>
<td>0.70</td>
<td>0.69</td>
<td>0.70</td>
<td>0.67</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0.65</td>
<td>0.66</td>
<td>0.66</td>
<td>0.66</td>
<td>0.67</td>
<td>0.63</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>0.72</td>
<td>0.66</td>
<td>0.72</td>
<td>0.72</td>
<td>0.73</td>
<td>0.70</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>0.71</td>
<td>0.65</td>
<td>0.73</td>
<td>0.73</td>
<td>0.71</td>
<td>0.68</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>0.75</td>
<td>0.74</td>
<td>0.75</td>
<td>0.75</td>
<td>0.76</td>
<td>0.73</td>
<td></td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
<td>0.74</td>
<td>0.77</td>
<td>0.78</td>
<td>0.79</td>
<td>0.73</td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>0.71</td>
<td>0.66</td>
<td>0.70</td>
<td>0.71</td>
<td>0.71</td>
<td>0.66</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>0.71</td>
<td>0.65</td>
<td>0.70</td>
<td>0.70</td>
<td>0.71</td>
<td>0.70</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>0.71</td>
<td>0.63</td>
<td>0.69</td>
<td>0.69</td>
<td>0.69</td>
<td>0.67</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>0.72</td>
<td>0.73</td>
<td>0.75</td>
<td>0.75</td>
<td>0.74</td>
<td>0.70</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>12</td>
<td>0.72</td>
<td>0.72</td>
<td>0.73</td>
<td>0.74</td>
<td>0.74</td>
<td>0.69</td>
<td></td>
<td>13</td>
</tr>
<tr>
<td>13</td>
<td>0.74</td>
<td>0.63</td>
<td>0.66</td>
<td>0.71</td>
<td>0.71</td>
<td>0.72</td>
<td></td>
<td>14</td>
</tr>
<tr>
<td>14</td>
<td>0.78</td>
<td>0.70</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
<td>0.70</td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>15</td>
<td>0.75</td>
<td>0.73</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
<td>0.75</td>
<td></td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>0.76</td>
<td>0.67</td>
<td>0.74</td>
<td>0.76</td>
<td>0.76</td>
<td>0.72</td>
<td></td>
<td>17</td>
</tr>
<tr>
<td>17</td>
<td>0.72</td>
<td>0.63</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.69</td>
<td></td>
<td>18</td>
</tr>
<tr>
<td>18</td>
<td>0.74</td>
<td>0.68</td>
<td>0.76</td>
<td>0.76</td>
<td>0.76</td>
<td>0.71</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>0.77</td>
<td>0.68</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td>0.73</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>0.73</td>
<td>0.68</td>
<td>0.72</td>
<td>0.72</td>
<td>0.72</td>
<td>0.69</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that in Table 4.2, $\Delta_{\text{opt}}$ ranges from 8 to 23 and the tree size ranges from 6 to 19. The choice of $\Delta_{\text{opt}}$ is totally data-dependent and so is the tree size. We examined the
performance of the tree-size determination algorithm (Appendix A.5) and the sensitivity of the determination of $\Delta_{\text{opt}}$ by computing hit rates for trees grown with various choices of $\Delta_{\text{min}}$. The values of $\Delta_{\text{min}}$ were: the optimal value $\Delta_{\text{opt}}$; $\Delta_{\text{opt}} \pm 1$; a small value $\Delta_{\text{min}} = 3$; and a large value $\Delta_{\text{min}} = 48$. The corresponding test-sample hit rates are listed in Table 4.3.

![Figure 4.1: $H_{CV}$ and $H_{TS}$ as functions of $\Delta_{\text{min}}$ for the 20 complete data sets. Each data set corresponds to a panel. The solid lines are $H_{TS}$ and the dotted lines are $H_{CV}$.](image)

As we can see, the results for $\Delta_{\text{min}} = \Delta_{\text{opt}} \pm 1$ are similar to the case where $\Delta_{\text{min}} = \Delta_{\text{opt}}$, i.e., there are no significant differences in hit rates between MultiSCAM and the majority-vote method. But the results for both $\Delta_{\text{min}} = 3$ and $\Delta_{\text{min}} = 48$ indicate that the majority-vote
method is worse than MultiSCAM. This is a result of either overfitting (for small $\Delta_{\text{min}}$) or underfitting (for large $\Delta_{\text{min}}$). This is illustrated by Figure 4.1. In Figure 4.1, both the cross-validation hit rate $H_{CV}^{(\text{com,MV})}$ and the test-sample hit rate $H_{TS}^{(\text{com,MV})}$ as functions of $\Delta_{\text{min}}$ for each of the 20 simulated complete data sets are plotted in each of the 20 panels. Each panel shows two curves: $H_{CV}^{(\text{com,MV})}$ (the dotted line) and $H_{TS}^{(\text{com,MV})}$ (the solid line).

The following facts can be observed from Figure 4.1:

1. As $H_{CV}^{(\text{com,MV})}$ increases, $H_{TS}^{(\text{com,MV})}$ has an increasing trend, and as $H_{CV}^{(\text{com,MV})}$ decreases, $H_{TS}^{(\text{com,MV})}$ has an decreasing trend;

2. In a small neighborhood of $\Delta_{\text{opt}}$ (at which $H_{CV}^{(\text{com,MV})}$ is maximized), $H_{TS}^{(\text{com,MV})}$ is either maximized or close to its maximum;

3. Both $H_{CV}^{(\text{com,MV})}$ and $H_{TS}^{(\text{com,MV})}$ are maximized at some intermediate values of $\Delta_{\text{min}}$.

Hence, for complete data analyses, we claim that the procedure of selecting $\Delta_{\text{opt}}$ is robust and it does a good job in terms of hit rate relative to MultiSCAM.

### 4.2.3 The results on censored data

Similarly, applying steps 1-3 in Section 4.2.1 to the 20 simulated censored data, we construct Table 4.4 showing the test-sample hit rates ($H_{TS}^{(\text{cen,MSCAM})}$ and $H_{TS}^{(\text{cen,MV})}$), the optimal value of $\Delta_{\text{min}}$ ($\Delta_{\text{opt}}$), and the tree size of the majority-vote method for each of the 20 censored data sets. The two-sample paired $t$-test of $H_{TS}^{(\text{cen,MSCAM})}$ vs $H_{TS}^{(\text{cen,MV})}$ has a $p$-value < 0.0001. Note that $H_{TS}^{(\text{cen,MV})} - H_{TS}^{(\text{cen,MSCAM})}$ has a sample mean=0.028 and its 95% confidence interval is (0.018, 0.039). So, we conclude that the majority-vote
Table 4.4: Test-sample hit rates comparison for the 20 censored data sets. The 95% confidence interval of $H_{TS}^{cen,MV} - H_{TS}^{cen,MS CAM}$ is (0.018, 0.039).

<table>
<thead>
<tr>
<th>Data</th>
<th>$H_{TS}^{cen,MS CAM}$</th>
<th>$H_{TS}^{cen,MV}$</th>
<th>$\Delta_{opt}$</th>
<th>Tree Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.70</td>
<td>0.77</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0.68</td>
<td>0.72</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
<td>0.69</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>0.76</td>
<td>0.73</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>0.66</td>
<td>0.71</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>0.73</td>
<td>0.76</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
<td>0.77</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>0.73</td>
<td>0.74</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>0.68</td>
<td>0.69</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>0.69</td>
<td>0.75</td>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>0.72</td>
<td>0.77</td>
<td>22</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>0.71</td>
<td>0.76</td>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>0.69</td>
<td>0.74</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>14</td>
<td>0.72</td>
<td>0.74</td>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>15</td>
<td>0.76</td>
<td>0.76</td>
<td>21</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.77</td>
<td>18</td>
<td>7</td>
</tr>
<tr>
<td>17</td>
<td>0.73</td>
<td>0.76</td>
<td>29</td>
<td>4</td>
</tr>
<tr>
<td>18</td>
<td>0.72</td>
<td>0.77</td>
<td>24</td>
<td>6</td>
</tr>
<tr>
<td>19</td>
<td>0.75</td>
<td>0.76</td>
<td>24</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>0.72</td>
<td>0.71</td>
<td>12</td>
<td>11</td>
</tr>
</tbody>
</table>

Mean | 0.716 | 0.745 |

The majority-vote method performs better than MultiSCAM in that it yields higher test-sample estimates of hit rates.

Again, notice that the range of $\Delta_{opt}$ is [8, 30] and the range of tree size is [4, 15].

To see how $\Delta_{min}$ affects the hit rate, we grow trees at five different values: $\Delta_{min} = 3$; $\Delta_{min} = \Delta_{opt}$; $\Delta_{min} = \Delta_{opt} \pm 1$; and $\Delta_{min} = 48$. Table 4.5 lists the corresponding hit rates. The results for $\Delta_{min} = \Delta_{opt} \pm 1$ are similar to $\Delta_{min} = \Delta_{opt}$, indicating that the majority vote method is better than MultiSCAM. For $\Delta_{min} = 3$, the result shows that the majority-vote method yields lower hit rates than MultiSCAM; for $\Delta_{min} = 48$, there is no evidence that the majority-vote method is better than MultiSCAM. Parallel to Figure 4.1,
Table 4.5: The effect of $\Delta_{\min}$ on hit rate for the 20 censored data sets. The trees are built at five different values of $\Delta_{\min}$: $\Delta_{\text{opt}}$, $\Delta_{\text{opt}} \pm 1$, 3 and 48.

<table>
<thead>
<tr>
<th>Data</th>
<th>$H_{\text{TS}}^{(\text{cen,MSCAM})}$</th>
<th>$H_{\text{TS}}^{(\text{cen,MV})}$</th>
<th>$\Delta_{\min} = 3$</th>
<th>$\Delta_{\text{opt}} - 1$</th>
<th>$\Delta_{\text{opt}}$</th>
<th>$\Delta_{\text{opt}} + 1$</th>
<th>48</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.70</td>
<td>0.67 0.76 0.72 0.72 0.77 0.77 0.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.68</td>
<td>0.65 0.72 0.72 0.72 0.72 0.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.67</td>
<td>0.62 0.69 0.69 0.69 0.69 0.67</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.76</td>
<td>0.66 0.73 0.73 0.73 0.73 0.71</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.66</td>
<td>0.61 0.71 0.71 0.71 0.71 0.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.73</td>
<td>0.74 0.76 0.76 0.76 0.76 0.72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
<td>0.76 0.77 0.77 0.77 0.77 0.74</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.73</td>
<td>0.68 0.75 0.74 0.74 0.74 0.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.68</td>
<td>0.65 0.69 0.69 0.69 0.69 0.68</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.69</td>
<td>0.66 0.75 0.75 0.75 0.75 0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.72</td>
<td>0.72 0.76 0.77 0.77 0.77 0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.71</td>
<td>0.68 0.75 0.76 0.76 0.76 0.69</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.69</td>
<td>0.63 0.73 0.74 0.74 0.74 0.72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.72</td>
<td>0.70 0.74 0.74 0.74 0.74 0.74</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.76</td>
<td>0.73 0.76 0.76 0.76 0.76 0.75</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>0.75</td>
<td>0.68 0.77 0.77 0.77 0.77 0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>0.73</td>
<td>0.62 0.76 0.76 0.76 0.76 0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>0.72</td>
<td>0.67 0.77 0.77 0.77 0.77 0.70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>0.75</td>
<td>0.67 0.75 0.76 0.76 0.76 0.73</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.72</td>
<td>0.67 0.71 0.71 0.72 0.72 0.74</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$H_{\text{TS}}^{(\text{cen,MV})} - H_{\text{TS}}^{(\text{cen,MSCAM})}$ = -0.042 0.026 0.028 0.028 0.0004

$p$-value of paired 2-sample $t$-test = 0.000 0.000 0.000 0.000 0.952

Figure 4.2 shows the curves of $H_{\text{TS}}^{(\text{cen,MV})}$ and $H_{\text{TS}}^{(\text{cen,MV})}$ as functions of $\Delta_{\min}$ for the 20 censored data sets, where the dotted lines are for $H_{\text{TS}}^{(\text{cen,MV})}$, and the solid lines are for $H_{\text{TS}}^{(\text{cen,MV})}$. As with the complete data analyses, a similar behavior of $H_{\text{TS}}$ and $H_{\text{CV}}$ can be observed. Accordingly, we conclude that the choice of $\Delta_{\text{opt}}$ is robust and the optimization of $\Delta_{\min}$ results in a significantly larger hit rate than MultiSCAM.
4.3 Summary of the simulation study

The results of simulation study in previous sections suggest the following conclusions:

1. For complete data, the majority-vote method shows no better performance than MultiSCAM in terms of hit rate.

2. For censored data, the majority-vote method yields significantly higher hit rate than MultiSCAM, though the difference is small (≈ 3%).
3. The procedure of optimizing $\Delta_{\text{min}}$ is robust, i.e., if we grow trees at $\Delta_{\text{opt}} \pm 1$ instead, there is no big changes in hit rate.

4. The tree-size determination algorithm (Appendix A.5) plays an important role in the analyses. It improves hit rate by optimizing the threshold value $\Delta_{\text{min}}$.

Figure 4.3 shows the test-sample hit rates and the cross-validation hit rate for the 20 complete data sets (as shown in Figure 4.1) and the 20 censored data sets (as shown in Figure 4.2) in a different fashion. The first row shows the 20 curves of the test-sample hit rates for complete data $H_{TS}^{\text{com,MV}}$ (left) and censored data $H_{TS}^{\text{cen,MV}}$ (right); the second row shows the 20 curves of the cross-validated hit rates for the complete data $H_{CV}^{\text{com,MV}}$ (left) and the censored data $H_{CV}^{\text{cen,MV}}$ (right); the third row shows the average curves of the test-sample hit rates (solid line) and the cross-validated hit rates (dotted line) for the complete data (left) and the censored data (right). Though some variability among replicates of simulation can be seen from the first and second rows, the average curves in the third row support claims we made in Section 4.2. That is, on average, the test-sample hit rate $H_{TS}$ and the cross-validation hit rate $H_{CV}$ satisfy the following:

1. When $\Delta_{\text{min}}$ is either small or large, both $H_{CV}$ and $H_{TS}$ are small;

2. As $\Delta_{\text{min}}$ initially increases, both $H_{CV}$ and $H_{TS}$ increase, then they reach their maxima at some intermediate values of $\Delta_{\text{min}}$ and begin to drop as $\Delta_{\text{min}}$ goes large, respectively;

3. At the value that maximizes $H_{CV}$, $H_{TS}$ is close to its maximum.

The fact that the average curve of $H_{TS}$ is always on the top of $H_{CV}$ might be due to the fact that the trees for computing the test-sample hit rate use all the training data with
Figure 4.3: $H_{CV}$ and $H_{TS}$ as functions of $\Delta_{\text{min}}$. Left panels = complete data, right panels = censored data. $H_{TS}$ = solid lines (first row), $H_{CV}$ = dotted lines (second row). The average curves are overlayed in the 3rd row.

400 observations while the trees for computing the cross-validation hit rate use part of the training data with only 360 observations.
Chapter 5

Discussions, Conclusions and

Future Work

5.1 Tree-structured methods: univariate to multivariate

In this section, we discuss tree-structured methods from a general point of view by explaining how the techniques discussed in Section 1.2.1 (tree-structured methods for univariate response) can be generalized to the case, where multivariate response is of concern, and comparing and contrasting the majority-vote method with classification and regression trees (CART) and Formal Inference-based Recursive Modeling (FIRM).

There are many ways of generalizing tree-structured methods from univariate to multivariate. Following the FIRM paradigm, one can apply multivariate test statistics for between-node difference to the splitting criteria and then use forward selection and multiplicity adjustment to determine the tree size. If the responses are continuous measures,
statistics such as two-sample Hotelling’s $T^2$ can be used for testing between-node difference. If the responses are categorical, a $\chi^2$-test based on multinomial distributions is a possible choice. In either case, a likelihood ratio test can serve as the splitting criterion, if an explicit probability model is assumed. In fact, MAID-M of Gillo and Shelly (1974) and MultiSCAM fall into this category in that both methods use statistical tests as their splitting criteria. The only difference between MAID-M and MultiSCAM is that MultiSCAM takes multiplicity into account, while MAID-M does not.

Following the CART paradigm, one can generalize the various node impurity measurements to the multivariate case and use them as the splitting criteria. The pruning algorithm can then be performed using the minimal cost-complexity criterion. In this sense, Segal (1992) and Zhang (1998) are generalizations of CART (Breiman et al. 1984). Specifically, Segal (1992) generalized the regression tree methodology to the multivariate case, while Zhang (1998) generalized the classification tree methodology to the multivariate case, where the responses are binary.

In review of the majority-vote method we proposed, it turns out that this method falls into neither the FIRM framework, nor the CART framework, though it bears more similarity to CART. In terms of splitting criteria, our method generalizes the misclassification rate in CART to the multivariate case and uses it as the basis of splitting. In terms of determining the tree size, however, our method differs from CART. In our approach, multiple trees are built at different values of the threshold $\Delta_{\text{min}}$, and an optimal value $\Delta_{\text{opt}}$ is chosen so that the cross-validation estimate of hit rate is maximized. Then the optimal threshold, $\Delta_{\text{opt}}$, is used in building a final tree in a forward direction. In contrast, CART
initially grows a very large tree, iteratively prunes this tree all the way back, thereby creates a nested sequence of trees, then selects the best tree from this sequence using test-sample or cross-validation estimates of error.

5.2 Conclusions

The results of the simulation study discussed in Chapter 4 show that the majority-vote method outperforms MultiSCAM for censored data in that it yields higher hit rates. This meets the expectation that our approach for handling incomplete data would be better than the current one, for which the censored observations are replaced with the detection limits.

For complete data, however, both methods perform equally in terms of hit rate. This result is a bit surprising. MultiSCAM, which uses all the information contained in the response data, should presumably perform better than the majority-vote method, which discretizes the response first, and then analyzes the resulting binary data. The reason behind this remains a puzzle. As seen in the first example discussed in Section 2.4.2, discretizing the response definitely suffers from information loss.

The following is an intuitive, if not fully satisfactory, explanation of why the majority-vote method yields higher hit rates than MultiSCAM does. Consider the splitting criteria in both methods. The splits of MultiSCAM rely on the between-node Mahalanobis distance, while the majority-vote method judges its splits on node impurity. As a result, a MultiSCAM tree would have terminal nodes that are as separated as possible, but a tree built with the majority-vote method would have terminal nodes that are as pure (i.e. the
node impurity is small) as possible. Notice that the hit rate is computed using the predicted values defined in Equation (3.5) and Equation (3.6) on page 47. In Equation (3.6), $\hat{\pi}_j(x_i)$ is the proportion of 1’s in node $t$, where $x \in t$. The node impurity defined in Equation (2.3) as a function of $\hat{\pi}_j(x_i)$ is maximized at $\hat{\pi}_j(x_i) = 1/2$ (see Equation (2.4)). Thus, a node with small impurity implies that $\hat{\pi}_j(x_i)$ is close to either 0 or 1. Consequently, the binary prediction for the majority-vote method (given by Equation (3.6)) is presumably more accurate than the binary prediction for MultiSCAM (given by Equation (3.5)).

5.3 Future work

As mentioned in Chapter 1, this work was motivated in hope of lessening the data incompleteness problem in a QSAR study. Although simulation study shows that the proposed method performs slightly better than the current approach, the data incompleteness problem is not completely solved. It is worth investigating other approaches to handling this problem. Multiple imputation (Little and Rubin 1987) combined with bagging (Breiman 1996) would be a possible choice. Parametric modeling of the response and the censoring mechanism also deserves a close look.

From a purely methodological point of view, it is interesting to compare the majority-vote method to Zhang’s generalized entropy approach, because both methods are concerned with multivariate binary responses. Statistical test-based methods have received less attention in the literature on tree-structured methods for multivariate binary responses. It is appealing to extend our study to that area. Possible candidates for the test statistics include a $\chi^2$-test derived from the multinomial distribution, and the likelihood ratio test.
Based on an explicit probability model.

Another research direction of our interest is to probe the possibility of conducting statistical tests on the node impurity measures. Although an exact probability distribution is hard to derive, resampling-based tests are possible.
Appendix A

Summary of Algorithms

A.1 Pseudo code of MultiSCAM

Initialize the threshold values for stopping rules:

the significant level $\alpha$ and the minimum node size $\nu_{\text{min}}$;

\[
\text{IF } n < \nu_{\text{min}} \text{ THEN} \newline
\text{STOP and declare a terminal node } t; \newline\]

\[
\text{ELSE} \newline\]

\[
\text{FOR } j = 1, \ldots, p; \newline\]

1. partition node $t$ into two groups according to the $j$th covariate:

\[
t_L(j) = \{Y_i \in t : x_{ij} = 0, i = 1, \ldots, n\} \text{ and } t_R(j) = \{Y_i \in t : x_{ij} = 1, i = 1, \ldots, n\}; \newline\]

2. compute the summary statistics of $t_L(j)$ and $t_R(j)$:

\[
\mathbf{Y}_L(j) = \frac{1}{n_L(j)} \sum_{Y_i \in t_L(j)} Y_i, \quad A_L(j) = \sum_{Y_i \in t_L(j)} (Y_i - \mathbf{Y}_L(j)) (Y_i - \mathbf{Y}_L(j))^T \newline
\mathbf{Y}_R(j) = \frac{1}{n_R(j)} \sum_{Y_i \in t_R(j)} Y_i, \quad A_R(j) = \sum_{Y_i \in t_R(j)} (Y_i - \mathbf{Y}_R(j)) (Y_i - \mathbf{Y}_R(j))^T \newline\]

where $n_L(j)$ and $n_R(j)$ are the sizes of nodes $t_L(j)$ and $t_R(j)$, respectively;

3. compute the pooled within-group covariance matrix: $V_{\text{pool}}(j) = \frac{1}{n - 2} (A_L(j) + A_R(j))$;
4. compute the Mahalanobis distance between $\mathbf{Y}_L(j)$ and $\mathbf{Y}_R(j)$:

$$T^2(j) = \frac{n_L(j)n_R(j)}{n} (\mathbf{Y}_L(j) - \mathbf{Y}_R(j))^\top V_{pool}(j)^{-1} (\mathbf{Y}_L(j) - \mathbf{Y}_R(j));$$

END FOR LOOP

Let $j_0 = \arg\max_{1 \leq j \leq p} T^2(j)$, and compute the raw $p$-value of $T^2(j_0)$, $rP(j_0)$, for node $t$;

Compute the Bonferroni adjustment of $rP(j_0)$, $aP(j_0) = rP(j_0)p$;

IF $aP(j_0) > \alpha$, THEN

STOP and declare a terminal node $t$;

ELSE

split $t$ on the $j_0$th covariate: $t_L = t_L(j_0)$ and $t_R = t_R(j_0)$;

REPEAT from the beginning for $t_L$ and $t_R$, respectively;

END IF

END IF

A.2 The tree-growing algorithm of the majority-vote method

Initialize the threshold values for stopping rules:

the minimum impurity decrease $\Delta_{min}$ and the minimum node size $\nu_{min}$;

IF $n < \nu_{min}$ THEN

STOP and declare a terminal node $t$;

ELSE

compute $R(t)$;

FOR $j = 1, \ldots, p$;

1. partition node $t$ into two groups according to the $j$th covariate:

$t_L(j) = \{Y_i \in t : x_{ij} = 0, i = 1, \ldots, n\}$ and $t_R(j) = \{Y_i \in t : x_{ij} = 1, i = 1, \ldots, n\}$;

2. compute $R(t_L(j))$ and $R(t_R(j))$;

END FOR LOOP

IF $\Delta_{min} < \Delta$ THEN

STOP and declare a terminal node $t$;

ELSE

split $t$ on the $j$th covariate: $t_L = t_L(j)$ and $t_R = t_R(j)$;

REPEAT from the beginning for $t_L$ and $t_R$, respectively;

END IF

END IF
3. compute the decrease in node impurity:

\[ \Delta(j) = n(t)R(t) - [n(t_L(j))R(t_L(j)) + n(t_R(j))R(t_R(j))]; \]

END FOR LOOP

Let \( j_0 = \arg \max_{1 \leq j \leq p} \Delta(j) \) and \( \Delta R(t) = \Delta(j_0) \),

IF \( \Delta R(t) < \Delta_{\text{min}} \), THEN

STOP and declare a terminal node \( t \);

ELSE

split \( t \) on the \( j_0 \)th covariate: \( t_L = t_L(j_0) \) and \( t_R = t_R(j_0) \);

REPEAT from the beginning for \( t_L \) and \( t_R \), respectively;

END IF

END IF

A.3 The complete-data generating algorithm

Starting from Data \( D_0 = \text{original data} (576 \times 10) \).

1. Data \( D_0 \xrightarrow{\text{MultiSCAM}} \text{Tree} T_0 = \{t'_1, \ldots, t'_{K'}\}; \)

2. For \( k = 1, \ldots, K' \),

if \( (> 60\% \text{ in } t_k \text{ are censored}) \) then censored value \( \times = \text{U}(0, 1) \rightarrow \text{Data } D_1; \)

3. Data \( D_1 \xrightarrow{\text{MultiSCAM}} \text{Tree} T_1 = \{t_1, \ldots, t_K\}; \)

4. For \( k = 1, \ldots, K' \),

generate \( n_k \) identically independently distributed \( \mathbf{Z}_i \sim N_q(\hat{\mu}_k, \hat{\Omega}_k) \rightarrow \text{Data } D_2 \), where \( \hat{\mu}_k \) and \( \hat{\Omega}_k \) are sample mean and covariance for node \( t_k \), respectively. Data \( D_2 \) is used as complete data in future analysis.
A.4 The censored-data generating algorithm

1. For \( j = 1, \ldots, q \), find \( \hat{\beta}_j \) by solving

\[
\frac{1}{n} \sum_{i=1}^{n} \frac{1}{1 + \hat{\beta}_j e^{Y_{ij}}} = \bar{D}_j, \quad j = 1, \ldots, q.
\]

2. Simulate the censoring indicator matrix \( C \):

\[
C_{ij} = \begin{cases} 
1, & \text{if } \frac{1}{1 + \hat{\beta}_j e^{Y_{ij}}} > U_{ij}, \\
0, & \text{otherwise.}
\end{cases}
\]

3. Simulate the censoring points by replacing \( Y_{ij} \in C_j \) with \( Z_{i'j} \in D_j \), where \( i' \) is related to \( i \) as described in Section 4.3.

A.5 The tree-size determination algorithm

This algorithm proceeds at a fixed small value of the minimal node size, \( \nu_{\min} \), e.g. \( \nu_{\min} = 1 \). Let \( D \) denote the training sample.

Randomly divide the training sample \( D \) into \( V \) disjoint subsets: \( D_1, \ldots, D_V \).

\textbf{FOR} \( \Delta_{\min} = 1, \ldots, M \), where \( M \) is a large integer

\textbf{FOR} \( v = 1, \ldots, V \),

- Construct tree \( T_v \) from \( D - D_v \);
- Predict data points in \( D_v \) using \( T_v \);
- Compute the hit rate for \( D_v \): \( H(T_v, D_v) \);

\textbf{END FOR LOOP}

Compute \( H_{CV}(\Delta_{\min}) = \frac{1}{V} \sum_{v=1}^{V} H(T_v, D_v) \);
Choose $\Delta_{\text{opt}}$ so that $H_{CV}(\Delta_{\text{min}})$ is maximized:

$$\Delta_{\text{opt}} = \arg \max_{1 \leq \Delta_{\text{min}} \leq M} H_{CV}(\Delta_{\text{min}}).$$
Appendix B

Proofs of Propositions

B.1 Proof of Proposition 2.1

By Equation 2.3:

\[ R(t) = \frac{1}{q} \sum_{k=1}^{q} R_k(t), \]

to show that \( n(t)R(t) \geq n(t_L(j))R(t_L(j)) + n(t_R(j))R(t_R(j)) \) for every \( j \in \{1, \ldots, p\} \), we only need to show that for each \( k = 1, \ldots, q \),

\[ n(t)R_k(t) \geq n(t_L(j))R_k(t_L(j)) + n(t_R(j))R_k(t_R(j)). \]

Without loss of generality, we will only show the result for \( k = 1 \). By Definition 2.5,

\[ n(t)R_1(t) = \sum_{Y_i \in \mathcal{I}} I \left( Y_{i1} \neq \hat{Y}_1(t) \right) \]

Notice that

\[ \hat{Y}_1(t) = I(p_1(t) \geq 1/2) = I \left( \sum_{Y_i \in \mathcal{I}} Y_{i1} \geq n(t)/2 \right) , \]
where \( p_1(t) = \{n(t)\}^{-1} \sum_{i \in t} Y_{i1} \) by Definition 2.4. Now, set \( n_1(t) = \sum_{i \in t} Y_{i1} \) and \( n_0(t) = n(t) - n_1(t) \). Note that \( n_1(t) \) and \( n_0(t) \) are number of 1’s and number of 0’s in node \( t \), respectively. Similarly, we can define \( n_1(t_L(j)), n_0(t_L(j)), n_1(t_R(j)) \) and \( n_0(t_R(j)) \). We have:

\[
\begin{align*}
n(t) &= n_0(t) + n_1(t) \quad \text{(B.1)} \\
n(t_L(j)) &= n_0(t_L(j)) + n_1(t_L(j)) \quad \text{(B.2)} \\
n(t_R(j)) &= n_0(t_R(j)) + n_1(t_R(j)) \quad \text{(B.3)}
\end{align*}
\]

and

\[
\begin{align*}
n_0(t) &= n_0(t_L(j)) + n_0(t_R(j)) \quad \text{(B.4)} \\
n_1(t) &= n_1(t_L(j)) + n_1(t_R(j)) \quad \text{(B.5)}
\end{align*}
\]

since the total number of 0’s and 1’s will remain the same after a split, respectively. Now, if \( n_1(t) \geq n(t)/2 \), i.e. \( n_1(t) \geq n_0(t) \), we have \( \hat{Y}_1(t) = I(n_1(t) \geq n(t)/2) = 1 \) by definition. Thus, \( n(t)R_1(t) \) is equal to the number of 0’s in node \( t, n_0(t) \). Similarly, if \( n_1(t) < n(t)/2 \), i.e. \( n_1(t) < n_0(t) \), then \( \hat{Y}_1(t) = I(n_1(t) \geq n(t)/2) = 0 \) and \( n(t)R_1(t) \) is equal to the number of 1’s in node \( t, n_1(t) \). Therefore, we have

\[
n(t)R_1(t) = \min(n_0(t), n_1(t)). \quad \text{(B.6)}
\]

The same arguments also apply to \( t_L(j) \) and \( t_R(j) \):

\[
\begin{align*}
n(t_L(j))R_1(t_L(j)) &= \min(n_0(t_L(j)), n_1(t_L(j))), \\
n(t_R(j))R_1(t_R(j)) &= \min(n_0(t_R(j)), n_1(t_R(j))).
\end{align*}
\]
By Equations (B.4) and (B.5),

$$\min (n_0(t), n_1(t)) \geq \min (n_0(t_L(j)), n_1(t_L(j))) + \min (n_0(t_R(j)), n_1(t_R(j))),$$

i.e.

$$n(t)R_1(t) \geq n(t_L(j))R_1(t_L(j)) + n(t_R(j))R_1(t_R(j)).$$

This proves Proposition (2.1).

Remarks. Equation (B.6) implies that $R(t) \leq \frac{1}{2}$. That is, the generalized misclassification rate is always in the range of $[0, 1/2]$.

### B.2 Proof of Proposition 2.2.

Assume that $Y$ has a probability density function $f(y)$ and let $F(y)$ denote its cumulative distribution function. Let $\mu = E(Y)$, $\sigma^2 = \text{Var}(Y)$ and $m$ be the median of $Y$, i.e. $F(m) = 1/2$. Let $Z_c = I(Y \geq c)$. It is easy to show that

$$E((Y - \mu)Z_c) = \int_{c}^{+\infty} (y - \mu)f(y)dy,$$

and

$$\text{Var}(Z_c) = F(c)(1 - F(c)).$$

Therefore, the correlation between $Y$ and $Z_c$ is given by

$$\rho(c) = \frac{\int_{c}^{+\infty} (y - \mu)f(y)dy}{\sigma \sqrt{F(c)(1 - F(c))}}.$$

Taking derivative with respect to $c$, we get

$$\rho'(c) = \frac{-(c - \mu)f(c)F(c)(1 - F(c)) - f(c)(1/2 - F(c)) \int_{c}^{+\infty} (y - \mu)f(y)dy}{\sigma (F(c)(1 - F(c)))^{3/2}}. \quad (B.7)$$
Note that
\[(c - \mu)f(c)F(c)(1 - F(c))\]
has the same sign \((\pm)\) as \((c - \mu)\). Also when \(c = m\),
\[f(c)(1/2 - F(c))\int_c^{+\infty} (y - \mu)f(y)dy = 0,\]
and
\[-f(\mu)(1/2 - F(\mu))\int_{\mu}^{+\infty} (y - \mu)f(y)dy\]
has the same sign as \((\mu - m)\). By Equation (B.7), we have
\[\rho'(\mu) = \frac{f(\mu)(F(\mu) - 1/2)\int_{\mu}^{+\infty} (y - \mu)f(y)dy}{\sigma (F(\mu)(1 - F(\mu)))^{3/2}}\]
and
\[\rho'(m) = \frac{-(m - \mu)f(m)F(m)(1 - F(m))}{\sigma (F(m)(1 - F(m)))^{3/2}} = \frac{-(m - \mu)f(m)(1/2)(1 - 1/2)}{\sigma ((1/2)(1 - 1/2)))^{3/2}} = \frac{2(\mu - m)f(m)}{\sigma}\]

Therefore, both \(\rho'(\mu)\) and \(\rho'(m)\) have the same sign as \((\mu - m)\). If \(\rho(c)\) has a unique local maximum at \(c_{\text{max}}\), i.e., \(\rho'(c) = 0\) has a unique solution at \(c_{\text{max}}\), then we have:

1. if \(\mu > m\), then \(\rho'(\mu) > 0\) and \(\rho'(m) > 0\), indicating that \(\rho(c)\) is increasing at \(\mu\) and \(c_{\text{max}}\) is to the right of \(\mu\);

2. if \(\mu < m\), then \(\rho'(\mu) < 0\) and \(\rho'(m) < 0\), indicating that \(\rho(c)\) is decreasing at \(\mu\) and \(c_{\text{max}}\) is to the left of \(\mu\);
3. if $\mu = m$, then $\rho'(\mu) = \rho'(m) = 0$, indicating that $\rho(c)$ is maximized at $c = \mu = m = c_{\max}$.

This proves Proposition 2.2 under the condition that $\rho(c)$ has a unique local maximum, i.e., $\rho'(c) = 0$ has a unique solution at $c_{\max}$. \qed

Remarks. Although the condition that $\rho(c)$ has a unique local maximum is not verified mathematically, Figure 2.2 is an empirical evidence that the condition holds for the data we are interested in.
Bibliography


Hand, D. J. (1997). *Construction and Assessment of Classification Rules*. Chichester: John Wiley & Sons Ltd.

296–302.


