DISCRIMINANT ANALYSIS: A FUNCTIONAL PERSPECTIVE

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER
FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
IN THE FACULTY OF SCIENCE AND ENGINEERING

2019

Diego Andrés Pérez Ruiz
School of Mathematics

Contents

Abstract **15**

Declaration **16**

Copyright Statement **17**

Dedication **18**

Acknowledgements **19**

1 Introduction **20**

1.1 Background and terminology for functional data 22

1.2 Notation 24

1.3 Major Tools for Functional Data 25

1.4 Misclassification 27

1.4.1 Decision rule based on Bayes' rule 27

1.4.2 Using loss functions 29

1.5 Thesis Structure 32

2 Some suggested methods for simulating functional data **34**

2.1 Simulation using a Gaussian Process (GP) 35

2.1.1 Algorithm to simulate using Gaussian Process (GP) 40

2.2 Simulation by using Fourier basis functions 41

2.3 Simulation of samples containing atypical observations 45

2.4 Real Data sets utilised in this thesis 48

2.4.1 Creation of real grouped dataset with imbalanced group size 50

2.5 Conclusions 53

3 k-Ranked Nearest Neighbours **54**

3.1 Introduction . . . 54

3.2 The k-RNN as an alternative of k-NN . . . 55

3.2.1 The k-RNN using depths . . . 57

3.3 Running example . . . 61

3.3.1 Choosing the value of k . . . 63

3.3.2 Functional depth . . . 66

3.3.3 Estimated probabilities . . . 70

3.4 The k-RNN as a regression approach - A model for the smoothing . . . 74

3.4.1 The k-RNN as a Moving Average. . . . 75

3.4.2 Smoother Matrix and Shrinking Smoothers . . . 76

3.4.3 The bias-variance trade-off . . . 80

3.4.4 A parametric bootstrap approach . . . 83

3.4.5 Bootstrap-t Confidence Intervals (CI) . . . 84

3.5 The signed distance integral as a classifier . . . 86

3.6 Dealing with Equal Samples Sizes . . . 89

3.7 Logistic regression . . . 90

3.7.1 A second covariate: Distance to the mode estimation . . . 92

3.8 Generalized additive models (GAMs) . . . 98

3.8.1 Preventing over-fitting in the data . . . 106

3.9 The GAM as a classifier . . . 112

3.9.1 Extending to more than two groups . . . 114

3.10 Simulations . . . 115

3.10.1 Comparison Methods . . . 116

3.10.2 Within maximum depth method . . . 120

3.11 Simulation Setup and Simulations Results . . . 121

3.11.1 Simulations Results . . . 122

3.12 Application to some Real data sets . . . 134

3.12.1 Point Misclassification . . . 137

3.13 Conclusions . . . 139

4 Dealing with Imbalanced Observations for functional data 140

4.1 Introduction . . . 140

4.2 Methods to dealing with unequal sample sizes . . . 141

4.3 Methods to evaluate imbalanced classifiers . . . 144

4.4 Methods to deal with imbalanced data for functional data . . . 149

4.4.1 A Bayesian Approach . . . 149

4.4.2 Oversampling techniques for functional data . . . 152

4.4.3 Performance Comparison . . . 158

4.5 Boundary observations . . . 160

4.5.1 Running Example . . . 162

4.5.2 Border Set . . . 163

4.5.3 Generating observations in the border set . . . 165

4.5.4 Improving the observations from the majority set as well as the minority set . . . 172

4.6 Simulation Setup . . . 174

4.7 Simulations Results . . . 175

4.8 Application to some Real imbalanced data sets . . . 179

4.9 Conclusions . . . 181

5 Principal Component Analysis for Functional Data 182

5.1 Introduction . . . 182

5.2 Problem formulation & Background . . . 184

5.3 Determining a finite dimensional space . . . 187

5.4 The Role of semimetrics in Functional Data . . . 189

5.4.1 Multivariate Partial Least Square Regression (MPLSR) . . . 191

5.5 Variable Kernel Density Estimation . . . 193

5.5.1 Nearest Neighbour Methods . . . 193

5.5.2 The Adaptive Kernel Method . . . 194

5.5.3 Choosing the Smoothing parameter h and k . . . 195

5.6 Statistical discrimination for functional data . . . 197

5.6.1 Statistical discrimination for more than two classes . . . 198

5.7 Simulations 199
5.7.1 Competing methods 201
5.7.2 Simulation Results 206
5.8 Real Data Study 222
5.9 Conclusions 226

6 Summary and Further Research **227**
6.1 Limitations and Further work 228
6.1.1 k- Ranked Nearest Neighbours for functional data 229
6.1.2 Dealing with Imbalanced Observations for functional data 229
6.1.3 Principal Component Analysis for Functional Data 230

Word count 53290

List of Tables

1.1 Difference in notation between the principal components in the multivariate and functional case for a fixed dimension p. 23

1.2 Loss functions for different actions and true classes. 30

2.1 Different sample sizes of each group for the real imbalanced data sets. . . . 51

3.1 Correlation matrix between the FM depth, different functional depths and the modal distance. 94

3.2 Summary of fitted logistic regression using the population label as a dependent variable. 94

3.3 Hosmer and Lemeshow goodness of fit (GOF) test applied to the simulated dataset. 97

3.4 The approximate significance of smooth terms. 101

3.5 Table showing the approximate significance of smooth terms. 104

3.6 The approximate significance of smooth terms when we consider an iteration between the signed depth and the distance to the mode. 106

3.7 Several performance measures for different GAMs. 113

3.8 Confusion Matrix or Error Matrix for three classes 0, 1 and 2. 115

3.9 Time and relative ratio of preprocessing, and classifying for the k-NN, k-RNN, GAM, Maximum Depth and α-Trimmed Mean method over one iteration. 123

3.10 Mean sensitivity rates, and their standard deviations in parentheses. 132

3.11 Mean specificity rates, and their standard deviations in parentheses. 133

3.12 Mean misclassification error rates and standard deviation for the methods when applied to the real dataset. 138

4.1 Confusion Matrix or Error Matrix for two classes 0 and 1. 145
4.2 Different scenarios for the Bayesian approach. 150
4.3 Several performance measures for different scenarios considered. 151
4.4 Observations in the border set, the area under the curve, precision/recall, F-measure and G-mean for different values of the threshold parameter. . . . 167
4.5 The value of the metrics before and after we oversample the observations in the border set. 170
4.6 The value of different metrics before and after oversampling the data. . . . 171
4.7 Confusion matrix for the simulated dataset before and after oversampling. . 171
4.8 The value of different metrics before and after oversampling the data, using nine nearest neighbours and the difference curve and sampling from both the majority and minority group. 173
4.9 Confusion matrix for the simulated dataset before and after oversampling, using using nine nearest neighbours and the difference curve and sampling from both the majority and minority group with a value of $\delta = 1$. 173
4.10 Confusion matrix for the simulated dataset before and after oversampling using using nine nearest neighbours and the difference curve and sampling from both the majority and minority group with a value of $\delta = 0.5$. 173
4.11 Different metric values for the proposed methods, across the different scenarios. The metrics are evaluated before and after oversampling the observations using a value of $\delta = 0.5$ and $\delta = 1$. 177
4.12 Different metrics values for the proposed methods, across the two different real datasets and considering an oversampling of both classes near the border. 180

5.1 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PCA semimetric. 216
5.2 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PCA semimetric. 217
5.3 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 200$ and using the PCA semimetric. 218

5.4 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 200$ and using the PCA semimetric. 219
5.5 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PLS semimetric. 220
5.6 Mean error rate, sensitivity rate, mean specificity and their variance in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PLS semimetric. 221
5.7 Missclassification error rate, sensitivity and specificity rates for the three real datasets and different metrics. 225

6.1 Distribution of the number of principal components in the simulations including a test set and minimising the mean integrate square error. 231
6.2 Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario testing over 100 simulated test dataset $n_1 = n_2 = 50$ and using the PCA semimetric. 232

List of Figures

1.1 Decision Boundaries. 29

2.1 Simulated functional data using a Gaussian Process $\mathcal{GP}(m(t), Cov(X(s), X(t)))$ with same mean function $m(t) = 80 \times (1-t) \times t^2$ and Stationary Covariance Function, using (top:) Model 1 with hyper-parameters $c = 100$, $\kappa = 0.1$ and $\mu = 2$, (middle:) Model 2 with hyper-parameters $c = 0.125$, $\kappa = 1.55$ and $\mu = 1.24$ and (bottom:) Model 3 with hyper-parameters $c = 3.5$, $\kappa = 1.85$ and $\mu = 1.35$ for 10 observations, in each group. 39

2.2 Decaying variances of the coefficients for the Fourier basis functions for (A) Scenario 1 and (B) Scenario 2. 43

2.4 Contaminated models generated over a finite time interval with a gap between the two time points set to be 0.150. For (top-left:) The Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model. 47

2.5 The near-infrared spectra of sugar in the orange juice samples. Group 1 consists of those with sucrose > 40 and Group 2 consists of those with sucrose < 40. 49

2.6 The NIR spectra of gasoline dataset. Group 1 consists of those with octane < 88 and Group 2 are those with octane > 88. 49

2.7 The phoneme dataset. The log-periodograms of three different phoneme classes. Group 1 correspond to the phoneme *sh*, Group 2 to the phoneme *iy* and Group 3 to the phoneme *dcl*. 51

2.8 The imbalance near-infrared spectra of sugar in the orange juice samples. (A) Boxplot of the sucrose content divided into two different groups according to the sucrose content. (B) Group 1 consists of those with sucrose > 30 and (B) Group 2 consists of those with sucrose < 30. 52

2.9 The imbalance NIR spectra of gasoline dataset. (A) Boxplot of the NIR spectra divided intro two different groups according to the octane content. (B) Group 1 consists of those with octane < 86 and (B) Group 2 are those with octane > 86. 52

3.1 Graphical representation of the reference curve when curves do not intersect. 60

3.2 Graphical representation of the reference curve when there is only one intersection when a larger proportion is positive. 60

3.3 Samples drawn from Gaussian process $\mathcal{GP}(m(t), K(s,t))$ with mean $m(t) = 80 * (1-t) * t^2$ and $K(s,t) = 0.1 * \exp(-100 * (s-t)^2)$ of size 200 observed over $M = 1000$ time points and a gap between the two time points to set to be 0.150. 62

3.4 Reference curve in solid line for the simulated data. 63

3.5 Missclassification error rate plotted against different neighbours for the simulated data. 66

3.6 Graphical representation of (A) the Tukey-depth mapping values of $F_{n,t}$ to $[0,1]$, and (B) the simplicial depth mapping $F_{n,t}$ to $[0.5, 1.0]$. 70

3.7 Graphical representation of the step function of the conditional probability of the signed depth belonging to the reference group Π_0. 72

3.8 The predicted probabilities for different values of k, using (top:) $k = 15$, (middle:) $k = 20$, and (bottom:) $k = 35$. 73

3.9 The first left and right singular eigenvectors for (top:) $n = 20$ and $k = 2$; (middle:) $n = 200$ and $k = 4$ and (bottom:) $n = 200$ and $k = 20$. 79

3.10 The first 20 ordered eigenvalues for (top-left:) $k = 3$; (top-right) $k = 4$ (bottom-left) $k = 5$ and (bottom-right) $k = 6$. 81

3.11 A 95% Bootstrap Confidence Intervals for the simulated data. 86

3.12 Kernel density estimation of the signed distance integral in terms of the first principal component score. The dashed vertical line at the value of zero indicates the reference curve (or signed depth of zero). 88

3.13 The first two principal component scores for the simulated data. 89
3.14 Modal curves, maximum and minimum curves for each population with a truncation point $T = 10$ for the simulated data. 93
3.15 The signed depth logit regression curve fitted to the simulated data. 95
3.16 The signed distance to the mode logit regression curve fitted to the simulated data. 96
3.17 Estimated smooth GAM function for $\hat{f}(X_1)$. 102
3.18 First and second estimated smooth GAM functions for the simulated dataset. 104
3.19 Hubert loss function with a threshold parameter $\delta = 1.345$ and a set of simulated residuals. 109
3.20 Scale Pearson residuals, the Huber residuals, an estimated GAM function for the signed depth and the fitted observations for (a) first iteration and (b) second iteration using a set of Huber weights. 111
3.21 Estimated surface for the two functional predictors the distance to the mode in the reference group and the signed depth in the simulated dataset when the response is 0 or 1. 113
3.22 Estimated surface for the two functional predictors the distance to the mode in the reference group and the signed depth in the simulated dataset with three different groups. 116
3.23 Boxplots of missclassification error rates obtained for the Uncontaminated Model. First row: Boxplots of the misclassification error for the same sample size $n_1 = n_2 = 50$ across the different Gaussian Processes. Second row: Boxplots of the missclassification error for a sample size of $n_1 = n_2 = 200$. 126
3.24 Boxplots of misclassification error rates obtained for the Asymmetric Contaminated Model for (First row:) sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$. 127
3.25 Boxplots of missclassification error rates obtained for the Linear Peak Contaminated Model for (First row:) sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$. 129
3.26 Boxplots of misclassification error rates obtained for the Shape Contamination Model for (First row:) same sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$. 131

3.27 Boxplots of the misclassification error rate when applied to the orange juice dataset. 135
3.28 Boxplots of the misclassification error rate when applied to the NIR gasoline spectra. 135
3.29 (a:) Boxplots of the missclassification error rate and (b:) signed depth against the distance to the mode, for the phoneme dataset. 136
3.30 Point misclassification error rates for the k-NN, k-RNN, GAM MD and α-TM classifiers applied to real datasets. 137

4.1 (A) Kernel density estimation for the observations in the minority and majority class with the posterior probabilities for the the signed depths to belong to (B) the minority class (B) and (C) the majority class. 151
4.2 A scatterplot of signed depth against the distance to the mode for the simulated data. The number of observations in each group are $n_0 = 50$ and $n_1 = 200$. 159
4.3 A scatterplot of signed depth against the distance to the mode after the original data is oversampled by bootstrapping the functional principal component scores. After oversampling the data both groups contain the same number of observations $n_0 = n_1 = 200$. 160
4.4 A set of four subfigures. 161
4.5 The misclassification error rated plotted against the number of neighbours for imbalanced data. 163
4.6 Border bands for the imbalanced data. The border observations are the observations in the minority class that fall in the closed interval of $[0.35, 0.65]$ with respect to the probability that it belongs to the population Π_0. 165
4.7 Observations in the border region for the simulated dataset. Observations in group 1 correspond to the majority class. Observations in group 0 correspond to those in the minority class and observations in group 2 correspond to those in the border set. 166
4.8 (a) Density strip and (b) first two principal components for observations in the majority and minority classes. 167
4.9 For (A) Observations in the border region for the simulated dataset and (B) new set of generated curves in the border region. 169

4.10 Misclassification error rates for the proposed methods, applied to the first Gaussian Process, across the different scenarios and considering an oversampling of both classes near the border. 176
4.11 Misclassification error rates for the proposed methods, applied to the second Gaussian Process, across the different scenarios and considering an oversampling of both classes near the border. 178

5.1 Boxplots of misclassification error rates obtained for the first scenario and across the different contamination models with the PC semimetric and a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 207
5.2 Boxplots of misclassification error rates obtained for the second scenario with the PC semimetric and different contamination model with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 208
5.3 Boxplots of misclassification error rates obtained for the first scenario and different contamination model with a sample size $n_1 = n_2 = 200$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 210
5.4 Boxplots of misclassification error rates obtained for the first scenario and different contamination models with a sample size $n_1 = n_2 = 200$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 211
5.5 Boxplots of misclassification error rates across different contamination models obtained for the first scenario using the PLS semimetric with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 213

5.6 Boxplots of misclassification error rates obtained for the second scenario using the PLS semimetric and different contamination model with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model 214
5.7 Kernel density estimates for the first four principal component scores for the orange juice dataset. 223
5.8 Kernel density estimates for the first four principal component scores for the NIR gasoline spectra. 224

The University of Manchester

Diego Andrés Pérez Ruiz
Doctor of Philosophy
Discriminant Analysis:
A functional perspective
June 4, 2019

Functional Data Analysis (FDA) provides information about curves that vary over a continuum. In this thesis, we propose two novel methodologies to classify a functional dataset, using supervised learning. The first methodology is based on Nearest Neighbours methods for functional data and classifies based on ranks of the functional signed depth. The proposed classifier uses the simplicity of the k-Ranked Nearest Neighbours (k-RNN) and its practical efficiency, exploiting the fact that the k-RNN provides conditional probabilities where the depth of an observed curve belongs to a particular group. Using this, we develop a probabilistic classifier and construct point-wise confidence intervals using a bootstrap approach. Following a generalized additive model, we propose a classifier based on the signed depth and the distance to the mode for functional observations. By means of a simulation study, we compare the performance of the proposed classifier against other nearest neighbours and depth classifiers. We also investigate the performance of the proposed classifier under different types of outliers common to these kinds of problems; we see that our proposed method works well under these different scenarios.

The second methodology we developed is based on log ratios of density estimates using Bayes' theorem. We propose a nonparametric adaptive density Bayesian classifier based on log ratios density estimates of functional principal component scores combined with different semimetrics. We study some of the main properties of the density estimator in a finite dimensional space and conduct a simulation study to investigate the performance of the proposed classifier under two semimetrics: the semimetric based on principal components scores and the semimetric based on partial least squares. We also compare the performance of the proposed classifier against different methods for simulated and real datasets.

Imbalanced sample sizes appear frequently in the classification problem and present multiple issues. We propose a method to strengthen observations that are at the boundary. We study different sampling methods to strengthen observations that are more susceptible to misclassification and we generate new curves by considering a linear combination of the observations in the border and the observations closes in depth.

Keywords: Functional Data, Nonparametric statistics, Kernel density estimation, k-ranked nearest neighbours, Bayesian classifier, Principal component analysis, Functional depth, Simulations.

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright Statement

Dedication

To the memory of

Jesús Ruiz Gordillo, Jesús Antonio Ruiz Palma and *Hilaria Pérez Pérez* .

Acknowledgements

I would like to thank to the CONACyT (Consejo Nacional de Ciencia y Tecnología) for the financial support from them.

Special thanks to my supervisor, Dr Peter J. Foster for being a constant source of interesting ideas and introducing me to the world of functional data.

I would like to thank to my parents Dorian Ruiz Palma and Florentino Pérez Pérez for providing me with the best environment to grow up and all the support during the time I spent in Manchester. Many thanks to my sister Dorian Andrea Pérez Ruiz and my brother Daniel Alberto Pérez Ruiz. Without them, none of this would be possible.

Many thanks to Diana Talía Alvarez Ruiz, for all her support, her love and encouragement.

Thanks to the examinators of this thesis.

Special thanks to Camilla Sammut-Powell for her support during the most difficult times.

Many thanks to my friends in the School of Mathematics and the statistics group and particular to my friends in the Alan Turing Building in the Room 1.122, Dávid Zoltán Szabó, Xiao Jiang (Mike), Lee Roy, Rahsed Alghanim and Chen Wang.

Finally, I would like to thank everybody that has helped or encouraged me in any way throughout the past three years in Manchester.

Chapter 1

Introduction

In functional data, an interest in classification arises in many fields of applications. In supervised classification for functional data, the goal is to construct a classifier that predicts the group label for each of the observed random curves. Since the beginning of the 21^{st} century the problem of classification for functional data has been studied from a multivariate perspective (Baillo and Cuevas, 2008). Many classification techniques have been adapted from the multivariate case to functional data and only a few of them are developed exclusively for functional data.

In real life, large problems fall into the framework of supervised classification. In such problems, the aim is to construct a function or decision rule that assigns new objects to one and only one of a pre-specified set of classes or groups, based on some descriptive information. The rule is usually constructed from a *training set*: data for which the true class labels are known. From the training set, we usually extract the information which is relevant to distinguish between the classes regarding the given measurements. The name supervised classification comes from the fact that the classes are known for the members of this initial data set, as if a *supervisor* has provided these class labels in advance (Hand et al., 2006). Once the decision rule is constructed, we are interested in evaluating the classification performance. For such purposes, different methods are proposed. This thesis studies the challenges in supervised classification for functional data and proposes two main contributions.

The 1^{st} contribution, described in Chapters 3 and 4, is based on using nearest neighbours methodology which refers to a family of techniques based on distances from observed functions to their k^{th} nearest amongst the sample cases. Their use in statistics dates back to the original report by Fix and Hodges Jr (1951) where it was mainly used for the k-nearest

neighbour classifier and density estimator. We study the k-RNN as an alternative of the k-NN for functional data and propose a first classifier based on ranks of signed depth. The proposed classifier uses the simplicity of the k-RNN making it efficient for practical implementation. Along with the signed depth, we propose a classifier based on generalized additive model. By introducing a second covariate to the study of the classifier, we investigate the performance of a logistic generalized additive model using the signed depth and the distance to the mode. This classifier is extended to more than two classes, and we compare its performance with different classifiers using a simulation study.

Another characteristic we investigate in this thesis is the performance of the k-RNN in the imbalanced case. This is where the proportion of data belonging to each class is not evenly distributed. Such scenarios appear more frequently in the classification problem. In the imbalanced case, the data is split into a minority and majority class and the minority class usually represents the most important concept to be learned. Usually, multiple problems arise in the imbalanced case; one problem is the classification of boundary observations. Boundary observations are observations near the classification borders. They are susceptible to being misclassified and thus are more important for classification. Using oversampling techniques, we propose a new method to strengthen observations that are near the borders and generate new observations by using a linear combination of the observations at the border and the observations closest in depth.

The 2^{nd} contribution is described in Chapter 5 and deals with a Bayesian classifier which is based on log ratios of density estimates computed using functional principal component scores. We propose a nonparametric adaptive density Bayesian classifier using log ratios density estimates of functional principal component scores, based on different semimetrics and a fixed dimension. We study some of the main properties of such a density estimator in a finite dimensional space and by means of a simulation study; we investigate the performance of the proposed classifier under two semimetrics: the semimetric based on principal components scores and the semimetric based on partial least squares. Finally, we compare the performance of the proposed classifier against different methods using simulated and real datasets.

We begin by setting the scene for the thesis; Section 1.1 introduces the background and terminology for functional data; Section 1.2 introduces the notation used; Section 1.3 introduces the major tools for functional data; Section 1.4 introduces the classification

problem and discusses the optimality of a Bayesian classifier. Finally, Section 1.5 gives the structure of the remaining chapters of this thesis.

1.1 Background and terminology for functional data

Within the field of functional data analysis, there exist two schools of thought based on how to conceptualise functional data. On one hand, some authors believe that functional data analysis can be considered as a smoothed version of multivariate data analysis and that it expresses the analytical tools for multivariate analysis in the language of functional analysis. On the other hand, the second line of development has been the statistical application of nonparametric function estimation (Silverman, 1986; Wahba, 1990; Green and Silverman, 1993; Eubank, 1999). In both cases, the primary goal is to analyse the entire set of functional observations.

This thesis follows a nonparametric approach using as few assumptions as possible. We assume that our observations are continuous functions $x_1(t), \ldots, x_n(t)$, for $t \in \mathcal{T}$, $\mathcal{T} = [a, b]$, which possess two main features (Ramsay, 2006):

Replication: Taking measurements on the same subject repeatedly over different functional space (usually time).

Smoothness: The underlying curve has a certain degree of smoothness.

Functional data make use of different tools from multivariate statistics and dimension reduction. Some of the common reduction techniques present in functional data are principal component analysis (PCA) and partial least squares (PLS). PCA was one of the first methods to be adapted to the functional case (Dauxois et al., 1982). We summarise the difference in notation between principal component analysis and functional principal component analysis as described in Table 1.1.

Table 1.1: Difference in notation between the principal components in the multivariate and functional case for a fixed dimension p.

	PCA	Functional PCA
Sample Space	$\mathbb{R}^p$	A function space $\mathcal{F}$
Parameter Space	$\theta \in \mathbb{R}^p$	$\mathbb{R}^p$, or a function space
Variables	$\mathbf{X}_i \in \mathbb{R}^p$	$X_i(t) \in \mathcal{F}$
Covariance	$Cov(\mathbf{X}_i, \mathbf{X}_j)$	$K(s,t) = Cov\{X(s), X(t)\}$

Functional data deals with a parameter space which can be multivariate $\mathbb{R}^p$, or a function space. The research in this area is still very active and focuses mainly on two approaches: a truly functional data approach and a multivariate approach.

When it is of interest to make inference about the parameter space (in $\mathbb{R}^p$), the most common approach is to project the data into a finite dimensional representation space. While, in a functional space a common approach is to do inference in a Hilbert space.

A Hilbert space is a space which generalises the notion of Euclidean space by extending the methods of vector algebra to high and infinite dimensional objects. An in-depth theory of Hilbert space is developed in Akhiezer and Glazman (2013). In the functional case the functions are infinite-dimensional objects in a $\mathcal{L}^2[a,b]$ space, i.e., the set of all real functions $X(t)$ defined on $\mathcal{T}$ satisfying $\int_a^b X^2(t)dt < \infty$. The $\mathcal{L}^2$ space is a Hilbert space with the inner product $\langle X_1(t), X_2(t)\rangle = \int_a^b X_1(t) \cdot X_2(t)dt$ for $X_1(t)$ and $X_2(t)$ and the equality $X_1(t) = X_2(t)$, $\forall\, t$ means

$$\int_a^b (X_1(t) - X_2(t))^2 dt = 0.$$

The $\mathcal{L}^2$ space is sufficient to handle most procedures considered in this thesis.

A single curve $X(t)$ represents a random function supported on an interval $t \in \mathcal{T}$ and $x(t)$ represents its observed value at point t. The mean of $X(t)$ is $\mu(t) = \mathbb{E}[X(t)]$; the variance is $\mathbb{V}ar(X(t)) < \infty$ and the covariance operator between two functions $X(s)$ and $X(t)$, at points s and $t \in \mathcal{T}$ is $K(s,t) = Cov\{X(s), X(t)\}$.

A functional dataset in a single sample can be defined as a collection of independent and identically distributed (i.i.d.) random functions which we will represent by $\{X_i(t),\, t \in \mathcal{T},\, i = 1, \ldots, n\}$. In practice, functional data are often observed at a grid of

data points. In this thesis, we will assume that each sample curve is observed on a common grid of data points $(t_1, t_2, \ldots, t_M)$, which are equally-spaced in the time interval. If all the sample functions are observed on $t_1, \ldots, t_M$, then we have a discretised functional data set represented by

$$\left\{x_i(t_j),\ i = 1, \ldots, n,\ j = 1, \ldots, M\right\}.$$

1.2 Notation

We represent a univariate random variables in upper case, for example X and Y, and their observed realisations are represented in lowercase, for example x and y or $x_1, \ldots, x_n$ for a sample of n observations of X. A multivariate random p-vector is represented by $\mathbf{X} = (X_1, \ldots X_p) \in \mathbb{R}^p$, while an observed vector is represented by $\mathbf{x} = (x_1, \ldots, x_p)$. The trace or sum of the diagonal elements of a square matrix $\mathbf{A}$ is written as $tr(\mathbf{A})$. Given an observed vector $\mathbf{x} = (x_1, \ldots, x_p)$, $diag(\mathbf{x})$ represents the $p \times p$ diagonal matrix with i^{th} diagonal element x_i.

The most common probability distributions like the Gaussian (normal) distribution and the binomial distribution are denoted by $N(\mu, \sigma^2)$ and $B(n, p)$, respectively. Similarly, the multivariate Gaussian distribution is written $N_p(\boldsymbol{\mu}, \Sigma)$. Where the dimension p is clear its inclusion in the notation will be suppressed.

For multivariate supervised classification, we consider the observed pairs $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ and the categorical variable y_i taking values in $\{0, 1, 2, \ldots, G-1\}$ for G classes or groups associated with the observations $\mathbf{x}_i \in \mathbb{R}^p$ and for some $p \in \mathbb{N}$.

For functional data, we have the sample functions $x_1(\mathbf{t}), \ldots, x_n(\mathbf{t})$ where $\mathbf{t} = (t_1, \ldots, t_M)$ denotes the vector of M common time points in the interval $[a, b]$ at which the sample functions are observed. In supervised classification for functional data we consider the observed pairs $(x_1(\mathbf{t}), y_1), \ldots, (x_n(\mathbf{t}), y_n)$. For the observed sample data which will be used to estimate a classification rule we know the true value of each y_i, $i = 1, \ldots, n$. Later we will also denote the G classes or groups by $\Pi_0, \Pi_1, \ldots, \Pi_{G-1}$.

1.3 Major Tools for Functional Data

To obtain uncorrelated, significant variables, several methods have been proposed. The most popular is Principal Component Analysis (PCA), popularised by Hotelling (1933). In the Hotelling (1933) approach, the variance of a linear combination of standardised variables is maximised. However, a different approach is the one suggested by Pearson (1901), where low rank approximations to the data matrix are fitted. In this section, we discuss how to formulate the tools utilised in later chapters of the thesis. We mainly discuss the Mercer's theorem (Mercer, 1909) in the functional form which relates to the singular value decomposition of the data matrix and implies that the Karhunen-Lòeve expansion or generalised Fourier expansion of the random function $X(t)$ is simply its representation in terms of the functional principal component basis $\psi_j(t)$.

Mercer's theorem (Mercer, 1909) Given a continuous symmetric non-negative definite kernel, K, there exists an orthonormal basis $\psi_j(t)$ on a close interval $\mathcal{T} = [a,b]$ consisting of eigenfunctions such that the eigenvalue sequences, θ_j, is non-negative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[a,b]$ and K can be represented as

$$K(s,t) = \sum_{j=1}^{\infty} \theta_j \psi_j(s) \psi_j^{'}(t), \tag{1.1}$$

where the convergence is absolute and uniform.

Mercer's theorem, implies that the Karhunen-Lòeve expansion or generalised Fourier expansion of a random function $X(t)$ is simply its representation in terms of the functional principal component basis. More precisely, we can consider the spectral decomposition of the variance covariance matrix be given by:

$$\begin{aligned} K(s,t) &= Cov\{X(s), X(t)\} \\ &= \sum_{j=1}^{\infty} \theta_j \psi_j(s) \psi_j(t) \end{aligned}$$

where $\theta_1 \geq \ldots \geq 0$ are the eigenvalues, with their respective orthonormal eigenfunctions $\psi_j(t)$ of $K(s,t)$. We can represent each random function in the functional dataset $\{X_i(t),\ i = 1,\ldots,n,\}$ in terms of a linear combination of the orthonormal principal components. More precisely we can write

$$X_i(t) = \sum_{j=1}^{\infty} \Xi_{ij}\psi_j(t) \quad \text{for} \quad i = 1,\ldots,n, \tag{1.2}$$

where Ξ_{ij}, for $j \geq 1$, $i = 1,\ldots,n$ are the functional principal component scores defined by

$$\Xi_{ij} = \theta_j^{-1/2} \int_{\mathcal{T}} (X_i(t) - \bar{X}(t))\psi_j(t)\ dt \quad \text{for} \quad i = 1,\ldots,n,\ \ j = 1,2,\ldots. \tag{1.3}$$

In practice, we consider a sample of functions $x_1(\mathbf{t}),\ldots,x_n(\mathbf{t})$ observed on a grid of points $\mathbf{t} = (t_1,\ldots,t_M)$. Thus, an estimator of the eigenvalues θ and the orthonormal eigenfunction $\psi_j(t)$ are denoted by $\widehat{\theta}$ and $\hat{\psi}_j(\mathbf{t})$ respectively. To find such estimators, in the functional case, we follow one of the following approaches: the discretisation - interpolation approach, the basis function expansion approach and the numerical quadrature approach. Such approaches are explained in detail in Ramsay et al. (2009). Each method aims to approximate the solutions of the functional eigen-equation:

$$\int_{\mathcal{T}} \widehat{Cov}(\mathbf{s},\mathbf{t})\psi(\mathbf{t})dt = \theta_j\psi_j(\mathbf{s}), \tag{1.4}$$

for positive eigenvalue/eigenfunction pairs, $\left(\theta_j, \psi_j(\mathbf{t})\right)$ and an estimator $\widehat{Cov}(\mathbf{s},\mathbf{t})$ of the covariance function. Mercer's theorem for functional data, guarantee that there exists some estimator of eigenvalues and orthonormal eigenfunctions, $\widehat{\psi}_j(\mathbf{t})$, such that for all pair of elements in $\mathbf{s}$ and $\mathbf{t}$:

$$\begin{aligned}
\widehat{Cov}\left\{x(\mathbf{s}),x(\mathbf{t})\right\} &= \widehat{Cov}(\mathbf{s},\mathbf{t}) \\
&= \frac{1}{n}\sum_{i=1}^{n}\left\{x_i(\mathbf{s}) - \bar{x}(\mathbf{s})\right\}\left\{x_i(\mathbf{t}) - \bar{x}(\mathbf{t})\right\} \\
&= \sum_{j=1}^{M} \hat{\theta}_j\hat{\psi}_j(\mathbf{s})\hat{\psi}_j(\mathbf{t}).
\end{aligned} \tag{1.5}$$

Now, we can represent each case in the functional dataset $\{x_i(\mathbf{t}),\ i = 1,\ldots,n,\}$ in terms of a linear combination of the orthonormal principal component scores. More precisely, $\hat{\psi}_j(\mathbf{t})$ corresponds to the estimated j^{th} functional principal component (eigenfunction) calculated from the sample and so we can write

$$x_i(\mathbf{t}) = \sum_{j=1}^{M} \hat{\Xi}_{ij}\hat{\psi}_j(\mathbf{t}) \quad \text{for} \quad i = 1,\ldots,n, \tag{1.6}$$

where $\hat{\Xi}_{ij}$, $j = 1,\ldots,M$; $i = 1,\ldots,n$ are the functional principal component scores defined by

$$\hat{\Xi}_{ij} = \widehat{\theta}_j^{-1/2} \int_{\mathcal{T}} (x_i(t) - \bar{x}(t))\hat{\psi}_j(t)\ dt. \tag{1.7}$$

Where the mean function, $\bar{x}(t)$, is estimated from the observed functional dataset and the integral is usually evaluated using numerical integration methods.

1.4 Misclassification

The goal of this section is to discuss minimax solutions for the 0-1 loss function. We start by exploring the decision rule based on the Bayes' rule.

1.4.1 Decision rule based on Bayes' rule

Let $y_1,\ldots,y_n$ be the true population indicator, with values in $\{0,1\}$, for the observations $x_1,\ldots,x_n \in \mathbb{R}$. We will assume that we are working in the one dimension but this can be extended to $p > 1$. Denote by $\pi_0 = \mathbb{P}(x \in \Pi_0)$ the prior probability that a randomly selected observation is in population Π_0, and by $\pi_1 = \mathbb{P}(x \in \Pi_1)$ the prior probability that a randomly selected observation is in population Π_1.

The posterior probability of $Y = j$ given x, for $j \in \{0,1\}$ can be calculated using Bayes' theorem. More specifically,

$$\begin{aligned} \mathbb{P}(Y = j \mid x) &= \frac{\mathbb{P}(x \text{ in population } \Pi_j \text{ and } x)}{\mathbb{P}(x)} \\ &= \frac{\mathbb{P}(x \mid x \text{ in population } \Pi_j)\mathbb{P}(x \in \Pi_j)}{\mathbb{P}(x)}. \end{aligned} \tag{1.8}$$

Where the denominator in equation (1.8) can be written as

$$\mathbb{P}(x) = \mathbb{P}(x \mid x \text{ in population } \Pi_0)\cdot\pi_0 + \mathbb{P}(x \mid x \text{ in population } \Pi_1)\cdot\pi_1.$$

The Bayes' rule assigns x to population Π_0 if

$$\mathbb{P}(x \text{ in population } \Pi_0 \mid x) > \mathbb{P}(x \text{ in population } \Pi_1 \mid x) \tag{1.9}$$

and to population Π_1 otherwise. Errors can happen two in different ways; we can have an error if we decide to assign x to population Π_0 but $x \in \Pi_1$ or by assigning x to population Π_1 but $x \in \Pi_0$, i.e., the probabilities of an error when classifying an observation x are $\mathbb{P}(x \in \Pi_0 \mid x)$ if we decide Π_1 and $\mathbb{P}(x \in \Pi_1 \mid x)$ if we decide Π_0. Where the minimum $\mathbb{P}(\text{Error} \mid x)$ be defined as $\min\{\mathbb{P}(x \in \Pi_0 \mid x), \mathbb{P}(x \in \Pi_1 \mid x)\}$.

The average probability of an error is defined as

$$\int_{-\infty}^{\infty} \mathbb{P}(\text{Error},\ x)dx = \int_{-\infty}^{\infty} \mathbb{P}(\text{Error} \mid x) \cdot \mathbb{P}(x)dx. \tag{1.10}$$

Bayes' rule minimises the average probability error and is optimum in that sense.

In a more general setting, we need a rule, $\mathcal{C}(\cdot)$, which will assign x to either one of the populations Π_0 or Π_1. This rule will divide the real line into two intervals or regions$\mathcal{R}_0$ and $\mathcal{R}_1$ such that $\mathcal{R}_0 \cup \mathcal{R}_1 = \mathbb{R}$. Then the probability of an error can be written as

$$\begin{aligned} \mathbb{P}(\text{Error}) &= \mathbb{P}(x \in \mathcal{R}_0 \mid x \in \Pi_1) + \mathbb{P}(x \in \mathcal{R}_1 \mid x \in \Pi_0) \\ &= \int_{\mathcal{R}_0} \mathbb{P}(x, x \in \Pi_1)dx + \int_{\mathcal{R}_1} \mathbb{P}(x, x \in \Pi_0)dx. \end{aligned} \tag{1.11}$$

We are free to decide the decision boundary between $\mathcal{R}_0$ and $\mathcal{R}_1$. To minimise $\mathbb{P}(\text{Error})$ for a given x we will assign x according to which is the smallest integrand in equation (1.11), i.e., if $\mathbb{P}(x, x \in \Pi_0) > \mathbb{P}(x, x \in \Pi_1)$ we assign x to population Π_0 as the error will be smallest or similarly if

$$\mathbb{P}(x \mid x \in \Pi_0) \cdot \pi_0 > \mathbb{P}(x \mid x \in \Pi_1) \cdot \pi_1.$$

Suppose that we have estimates of the probability density for observations in two populations, such densities can be represented in Figure 1.1 where $\hat{x}$ is a decision boundary.

Our decision rule will assign a new observation x^* to the population Π_0 if $x^* \in \mathcal{R}_0$ and to population Π_1 if $x^* \in \mathcal{R}_1$. For $x < \hat{x}$ an error arises when we misclassify x as x in Π_0 but it belongs to population Π_1. In this case, the overall error can be expressed in terms of the area of the probability densities in Figure 1.1. More precisely,

$$\mathbb{P}(x \in \mathcal{R}_0 \mid x \in \Pi_1) = \text{Area A} + \text{Area B}. \tag{1.12}$$

For $x > \hat{x}$, an error arises when we misclassify x as x in Π_1 but it belongs to population Π_0 and in this case the overall error is

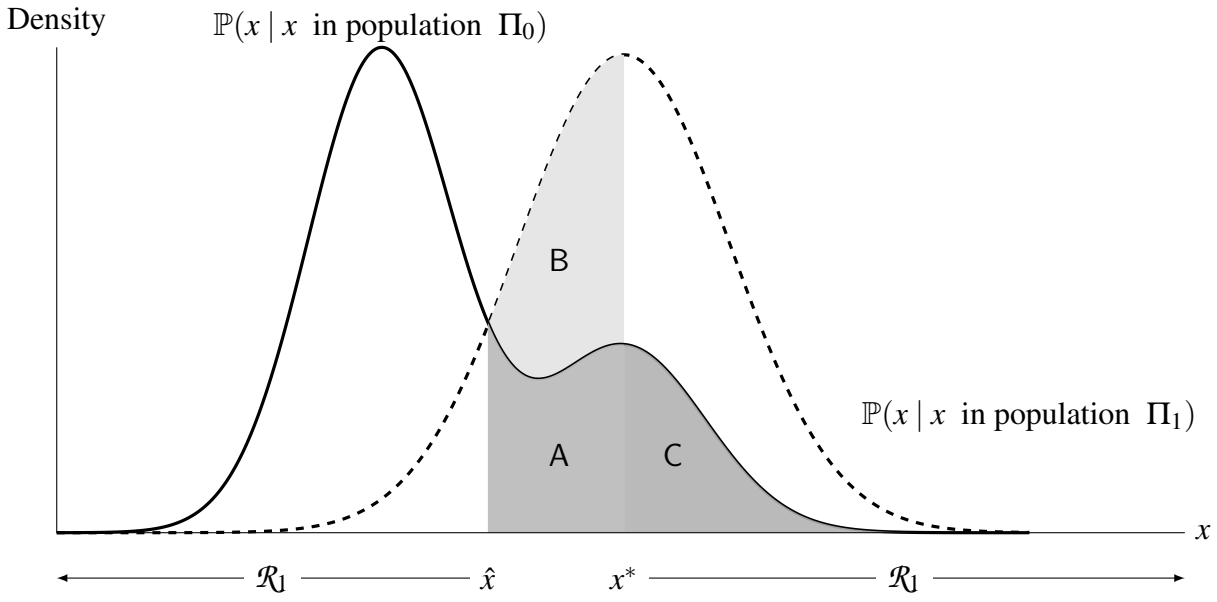


Figure 1.1: Decision Boundaries.

$$\mathbb{P}(x \in \mathcal{R}_1 \mid x \in \Pi_0) = \text{Area C}. \tag{1.13}$$

As we vary the location of $\hat{x}$, the combined areas of A and C stay the same but area B varies. Thus, the optimal choice for $\hat{x} = x_{opt}$ is where the two density curves cross, i.e., when

$$\mathbb{P}(x_0, x \in \Pi_0) = \mathbb{P}(x, x \in \Pi_1), \tag{1.14}$$

since Area B vanishes and the misclassification probabilities are as small as possible.

1.4.2 Using loss functions

Suppose now that we have an action α_0 which assigns x to population Π_0 and an action α_1, which assigns x to the population Π_1. Suppose also that a loss (or cost) function $\mathcal{L}(\cdot)$ is available. The loss function states exactly how costly each action is, and is used to convert a possible course of action determined by a probability into a decision. It also indicates the losses involved when taking particular actions, e.g.,

$$
\begin{aligned}
l_{00} &= \text{ loss in assigning } x \text{ to } \Pi_0 \text{ when } x \in \Pi_0, \\
l_{01} &= \text{ loss in assigning } x \text{ to } \Pi_0 \text{ when } x \in \Pi_1, \\
l_{11} &= \text{ loss in assigning } x \text{ to } \Pi_1 \text{ when } x \in \Pi_1, \\
l_{10} &= \text{ loss in assigning } x \text{ to } \Pi_1 \text{ when } x \in \Pi_0.
\end{aligned}
$$

We can summarise the losses as shown in Table 1.2.

Table 1.2: Loss functions for different actions and true classes.

	True Class	
	Class 0	Class 1
Action α_0	l_{00}	l_{01}
Action α_1	l_{10}	l_{11}

In this thesis we use the 0-1 loss function which assigns no loss to a correct decision and a unit loss to any error. Under this loss, all errors are equally costly. The risk corresponding to this loss function is the average probability of an error. For the two category problem, the losses takes the specific values $l_{00} = l_{11} = 0$ and $l_{10} = l_{01} = 1$, respectively.

The expected loss (or class conditional risk) associated with taking action α_0 is written as combination of weighted probabilities

$$\mathcal{R}(\alpha_0 \mid x) = l_{00} \cdot \mathbb{P}(x \in \Pi_0 \mid x) + l_{01} \cdot \mathbb{P}(x \in \Pi_1 \mid x), \tag{1.15}$$

and similarly, the expected loss associated with taking action α_1, is given by

$$\mathcal{R}(\alpha_1 \mid x) = l_{10} \cdot \mathbb{P}(x \in \Pi_0 \mid x) + l_{11} \cdot \mathbb{P}(x \in \Pi_1 \mid x). \tag{1.16}$$

Using the 0-1 loss function, we have that

$$
\begin{aligned}
\mathcal{R}(\alpha_0 \mid x) &= l_{01} \cdot \mathbb{P}(x \in \Pi_1 \mid x) = \mathbb{P}(x \in \Pi_1), \\
\mathcal{R}(\alpha_1 \mid x) &= l_{10} \cdot \mathbb{P}(x \in \Pi_0 \mid x) = \mathbb{P}(x \in \Pi_0).
\end{aligned}
$$

We choose the action which minimises the conditional risk, i.e., assigning x to population Π_0 if $\mathcal{R}(\alpha_0 \mid x) < \mathcal{R}(\alpha_1 \mid x)$ or to population Π_1 otherwise.

This can be written as

$$\begin{aligned}\frac{\mathbb{P}(x \mid x \in \Pi_0)}{\mathbb{P}(x \mid x \in \Pi_1)} &> \frac{(l_{01}-l_{11})\cdot\pi_1}{(l_{10}-l_{00})\cdot\pi_0}\\ &> \frac{\pi_1}{\pi_0},\end{aligned}$$

if $l_{01} = l_{10} = 1$. We now specifically make our action a function of x such that

$$\alpha(x) = \begin{cases}\text{Assign to } \Pi_0 \text{ if } \quad x \in \mathcal{R}_0.\\ \text{Assign to } \Pi_1 \text{ if } \quad x \in \mathcal{R}_1.\end{cases}$$

We have divided $\mathbb{R}$ into $\mathcal{R}_0 \cup \mathcal{R}_1$ to denote the intervals for x leading to assigning x to either population Π_0 or Π_1. The overall risk $(O\mathcal{R})$ is given by

$$\begin{aligned}O\mathcal{R} &= \int_{\mathbb{R}} R(\alpha(x) \mid x)\,\mathbb{P}(x)\,dx\\ &= \int_{\mathcal{R}_0} [l_{00}\mathbb{P}(x \mid x \in \Pi_0)\cdot\pi_1 + l_{01}\mathbb{P}(x \mid x \in \Pi_1)\cdot\pi_1]\,dx\\ &\quad + \int_{\mathcal{R}_1} [l_{10}\mathbb{P}(x \mid x \in \Pi_0)\cdot\pi_0 + l_{11}\mathbb{P}(x \mid x \in \Pi_1)\cdot\pi_0]\,dx. \qquad (1.17)\end{aligned}$$

Therefore, under the 0-1 loss function, we can rewrite equation (1.17) as

$$\begin{aligned}O\mathcal{R} &= \int_{\mathcal{R}_0} \mathbb{P}(x \mid x \in \Pi_1)\cdot\pi_1 dx + \int_{\mathcal{R}_1} \mathbb{P}(x \mid x \in \Pi_0)\cdot\pi_0 dx\\ &= \int_{\mathcal{R}_0} \mathbb{P}(x, x \in \Pi_1)dx + \int_{\mathcal{R}_1} \mathbb{P}(x, x \in \Pi_0)dx.\end{aligned}$$

We have shown that to minimise the average probability of an error, which is equivalent to the overall risk here, we choose the decision boundary to be the value of x that satisfies $\mathbb{P}(x, x \text{ in } \Pi_1) = \mathbb{P}(x, x \text{ in } \Pi_0)$. Good classifiers, minimise the maximum possible overall risk.

1.5 Thesis Structure

Chapter 2 describes in detail different simulation settings and techniques to simulate functional data. Functional data can be seen as sample paths of a stochastic processes that vary over a continuum. We explore the flexibility of the Gaussian process to generate different functions and combine this with the kernel or covariance function. To set up our simulations, we focus on the stationary covariance function. This type of covariance function is a flexible covariance function that generates rough or smooth curves according to the combination of parameters. A second method we explore to is the simulation method based on Fourier basis. An advantage of this method is that we generate functions allowing the basis coefficients to be chosen randomly from a Normal distribution with a decaying variance as the the number of coefficients increases. This chapter also explains how we contaminate the data, by introducing atypical observations. Finally, we introduce the real (balance and imbalanced) datasets on which we based our simulations.

Chapter 3 starts by introducing the k-RNN as an alternative of the k-NN. We then propose a first classifier based on signed depth. Using a running example, we explore different features of the classifier including choosing the value of k and using different functional depths. Motivated by the simplicity of the k-RNN classifier, we explore how the k-RNN classifier can be interpreted in terms of conditional probabilities and as a moving average. Utilising the fact that the k-RNN classifier obtains conditional probabilities that the signed depth of the a curve belongs to a particular group, we construct point-wise confidence intervals for the estimated probability that an observed signed depth corresponds to a particular group. Using logistic regression, we follow a generalized additive model to estimate the predicted probabilities in the k-RNN classifier based two covariates: the signed depth and the signed distance to the mode. At the end of this chapter some simulations with artificial and real datasets are shown to demonstrate the performance.

Chapter 4 deals with the problem of imbalanced observations. In the imbalanced case, the number of observations in the majority class exceeds the number of observations in the minority class, thus different problems arise. We explore boundary observations in terms of the nature of the classification rule and propose a method to strengthen the minority observations in the boundary. Our method is based on sampling techniques involving the functional principal component scores. We generate new curves by considering a linear

combination of the observations in the border and the observations closest in depth. We apply our proposed methodology to simulated and real datasets.

In Chapter 5 we propose a novel nonparametric adaptive density Bayesian classifier using log ratios of density estimates computed using functional principal component scores. It is based on different semimetrics and linked to a particular dimension. The adaptive kernel density estimation plays an important role in high dimensions. We study some of the main properties of such a density estimator with selected particular dimensions. By means of a simulation study, we investigate the performance of the proposed classifier under two semimetrics. The first semimetric we study is the semimetric based on principal components scores. The second semimetric is the semimetric based on partial least squares. For the adaptive approach, we also investigated the behaviour of the classification under different norms. This approach was applied to real and simulated datasets.

Chapter 2

Some suggested methods for simulating functional data

Simulations can have different aims and play an important role in statistics (Tocher, 1967). In functional data, simulations are usually used to make inference about the distribution of an estimator or to estimate the variability associated with a descriptive statistic to construct its confidence interval. In supervised classification, simulations of functional data are used to illustrate the performance of the classifier under different scenarios.

The aim of this chapter is to describe in detail different simulation settings and techniques to simulate functional data. Such techniques provide the basis to carry out simulations for the proposed methods in the later chapters. We describe two different approaches to simulate functional data. The first approach is motivated by Gaussian Processes, a continuous-time process that can generate different functions, forced to pass through specific points and in which the degree of smoothness can be controlled by the covariance function. The second approach is the motivated by Fourier basis. We choose the Fourier basis coefficients to be independent and normally distributed such that the contribution tend to decay as the number of basis increase.

The methods of generating samples of functional data following a given structure will be extended to including sample curves which are atypical from the main body of the data. Firstly, such curves can be specified to differ with respect to magnitude if such a curve is distant from the mean curve of the underlying process. Secondly, we can specify curves which follow a different pattern or shape from the majority. For example, such a curve could be tending in an opposite direction to the rest or be very irregular when the others are quite

smooth. We will look to contaminate samples of regular function with sub-samples of such atypical observations to be able to create a more demanding scenario with which to evaluate our classification methodology. We will use the abbreviated form *atypicals* to refer to such simulated curves later in the thesis.

We begin in Section 2.1 by providing a brief description of the methods considered for simulating functional data using a Gaussian Process and the influence of covariance functions to generate irregular curves. Section 2.2 discuss the role of Fourier basis functions in simulating functional data. Section 2.3 explains in detail different models to contaminated the data. Section 2.4 discusses the real datasets utilised in this thesis and the creation of imbalanced real datasets. Finally, in Section 2.5 we state some conclusions.

2.1 Simulation using a Gaussian Process (GP)

Functional data can be thought of as sample paths of a stochastic process with a mean and covariance function that varies over a continuum. This approach uses the covariance function of a stochastic process as the fundamental tool for assessing the variability. A very important class of continuous-time processes is that of Gaussian Processes which arise in many applications. The flexibility of the Gaussian Process allows us to generate different functions and forces us to pass through some specific points. For a more detailed introduction on Gaussian Processes see Rasmussen (2006), Stein (2012) and Cressie (1993).

A Gaussian Process, $\{X(t)\}_{t\in T}$ indexed by a set T (often time), is a collection of random variables such that the joint distribution of any finite number of them follows a Normal distribution.

More formally we may say that a stochastic process in continuous time is Gaussian if and only if, for every finite set of indices $\mathbf{t} = (t_1,\ldots,t_M) \in T$, the joint distribution of the random variables $X(t_1),\ldots,X(t_M)$ is multivariate normal with mean vector $(m(X(t_1)),\ldots,m(X(t_M)))$ and covariance function

$$Cov(X(\mathbf{s}),X(\mathbf{t})) = \mathbb{E}\left[(X(\mathbf{s})-m(\mathbf{s}))\,(X(\mathbf{t})-m(\mathbf{t}))\right] \quad \forall\, s,t,\in T. \tag{2.1}$$

For $\mathbf{t} = (t_1,\ldots,t_M)$ the random variable X is observed at a finite set of time points, the covariance matrix for $X(\mathbf{t})$ can be written as

$$Cov(X(\mathbf{t})) = \begin{bmatrix} Cov(X(t_1),X(t_1)) & Cov(X(t_1),X(t_2)) & \dots & Cov(X(t_1),X(t_M)) \\ Cov(X(t_2),X(t_1)) & Cov(X(t_2),X(t_2)) & \dots & Cov(X(t_2),X(t_M)) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X(t_M),X(t_1)) & Cov(X(t_M),X(t_2)) & \dots & Cov(X(t_M),X(t_M)) \end{bmatrix}.$$

Therefore, a Gaussian Process, is specified by its mean function $\mathbb{E}[X(t)] = m(t)$ and the covariance function $Cov(X(s),X(t))$. The first two moments of a Gaussian Process are sufficient for a complete characterisation of the process. It can be convenient to assume that the mean function is simply zero everywhere. The covariance function $Cov(X(s),X(t))$ can be more simply denoted by $K(s,t)$. A Gaussian Process is stationary in the mean if $\mathbb{E}[X(t)] = m(t) = \mu(t)$ and constant $\forall\, t$. It is said to be second order stationary if $Cov(X(s),X(t)) = K(s,t)$ only depends on the value of $|s-t|$.

In this thesis, we write a Gaussian Process as $\mathcal{GP}(m(t),Cov(X(s),X(t)))$. The covariance function is a crucial component in a Gaussian Process. It encodes the degree of similarity between $X(t)$ and $X(s)$ as a function of s and t and importantly, the covariance function must be positive semi-definite.

There are many possible choices of the covariance function, and we can specify a wide range of models just by specifying the covariance function of the Gaussian Process. To begin understanding the types of curves that we can express by using a Gaussian Process, we start by briefly describing the most commonly used covariance functions in the literature. For more types of covariance functions see Rasmussen (2006).

The Squared Exponential Covariance Function The Squared Exponential Covariance Function has the form

$$Cov(X(s),X(t)) = \sigma^2 \exp\left(-\frac{(X(s)-X(t))^2}{2l^2}\right). \tag{2.2}$$

This covariance function has two parameters which specify the shape of the covariance function. The first parameter is the length-scale l, which describes how smooth the function is. Small length-scale value generates roughness functions, while large values generates smooth functions that change slowly. The second parameter, σ^2, is a scaling factor and can be interpreted as signal variance, i.e., $\mathbb{V}ar(X(t)) = \sigma^2$, a constant $\forall\, t$. A small value of σ^2

characterises functions tend to stay close to their mean value, over the interval $\mathcal{T}$, while larger values allow for more variation.

The Periodic Covariance Function The Periodic covariance function is a function which repeats itself and has the form

$$Cov(X(s),X(t)) = \sigma^2 \exp\left(-\frac{2\sin^2(\pi \times |X(s)-X(t)|/P)}{l^2}\right). \tag{2.3}$$

The period P simply determines the distance between repetitions of the function. While the length scale l determines the smoothness of the function. As in the square exponential covariance function, small length scale value means that the function values can change quickly, while large values have the opposite effect.

The Locally Periodic Covariance Function The form of the Locally Periodic Covariance Function is given by

$$\begin{aligned} Cov(X(s),X(t)) = \sigma^2 \exp\left(-\frac{2\sin^2(\pi \times |X(s)-X(t)|/P)}{l^2}\right) \times \\ \exp\left(-\frac{(X(s)-X(t))^2}{2l^2}\right), \end{aligned} \tag{2.4}$$

where the interpretation of the parameters is the same as the Squared Exponential Covariance and the Periodic Covariance Function.

The Linear Covariance Function The Linear Covariance Function is one of the simpler linear covariance functions. Its form is given by

$$Cov(X(s),X(t)) = \sigma_a^2 + \sigma_b^2 (X(t)-c)(X(s)-c), \tag{2.5}$$

where σ_a^2 and σ_b^2 are both positive and c is a finite constant. A restriction on the parameters we need is that

$$\frac{\sigma_a^2}{\sigma_b^2} > (X(t)-c)^2,$$

in order that $\mathbb{V}ar(X(t)) > 0, \forall\, t$.

The Stationary Covariance Function The Stationary Covariance Function has the form

$$Cov(X(s),X(t)) = \kappa \cdot \exp\{-c\,|s-t|^{\mu}\}, \tag{2.6}$$

with the hyper-parameters c, κ and μ that control the roughness of the curves. Increasing μ and κ yields softer curves, while increasing c results in more irregular curves. This family of models was previously studied in Wood and Chan (1994) for a simulation of stationary Gaussian vector fields.

We can find different similarities between the previously described covariance function. For example, in the Squared Exponential Covariance Function and the Periodic Covariance Function, both can control how the function values change according to the length scale. On one hand, the Locally Periodic Covariance Function is a combination of the Squared Exponential Covariance and the Periodic Covariance Function, which results in functions which are periodic, but can slowly vary over time. On the other hand, the Stationary Covariance Function describes a different covariance function involving different parameters which play a different role to those in the other covariance functions described above.

To simulate from a Gaussian Process, we consider the Stationary Covariance Function. As we can see from equation (2.6), it depends on the hyper-parameters c, κ and μ , which relate to the variability of curves generated by this Gaussian Process. We can use the value of the parameter c to control how close simulated curves are to each other. A large value of approximately 10^2 can generate curves which are close together, while curves more disperse can be generated with a value of $c < 10$. A combination of the value of κ and μ controls the roughness of the curves, for a combination of a small $\kappa < 1$ and $\mu > 2$, we can generate smooth curves. To generate roughness curves, we start by increasing the value of $\kappa > 1$ and $\mu > 2$. Increasing $\kappa > 2$ and $\mu > 2$, generates very rough curves.

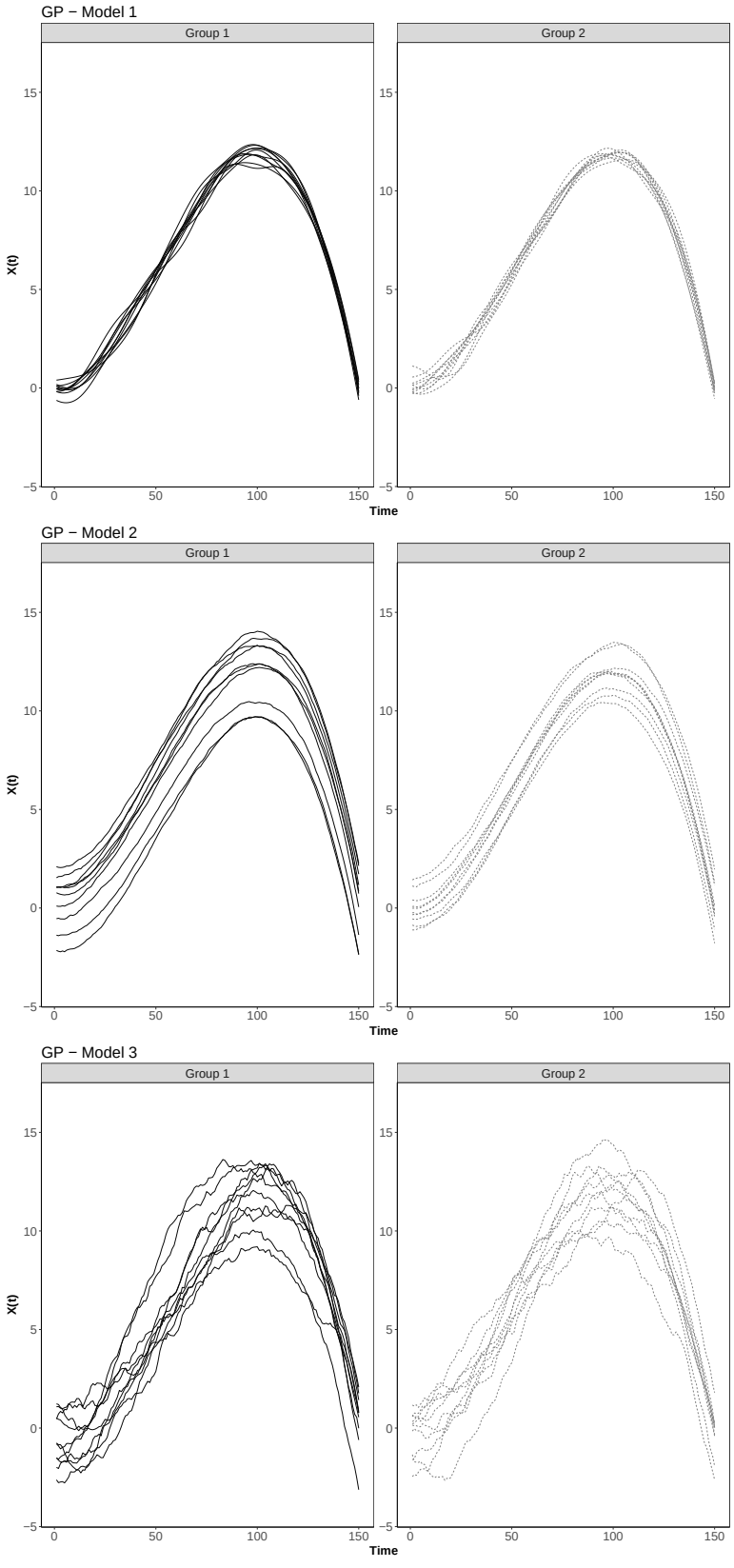


Figure 2.1: Simulated functional data using a Gaussian Process $\mathcal{GP}(m(t), Cov(X(s), X(t)))$ with same mean function $m(t) = 80 \times (1-t) \times t^2$ and Stationary Covariance Function, using (top:) Model 1 with hyper-parameters $c = 100$, $\kappa = 0.1$ and $\mu = 2$, (middle:) Model 2 with hyper-parameters $c = 0.125$, $\kappa = 1.55$ and $\mu = 1.24$ and (bottom:) Model 3 with hyper-parameters $c = 3.5$, $\kappa = 1.85$ and $\mu = 1.35$ for 10 observations, in each group.

2.1.1 Algorithm to simulate using Gaussian Process (GP)

We now detail an algorithm to simulate from a Gaussian Process. Given a random number generator that generates multivariate normal distributed random numbers, we can sample from a Gaussian stochastic process by sampling from a multivariate normal distribution. To simulate from a Gaussian Process with a pre-specified choice of a mean function and covariance function, $\mathcal{GP}(m(\mathbf{t}), Cov(X(\mathbf{s}), X(\mathbf{t})))$, we use the algorithm described below.

Step 1. Give a positive integer, M, representing the number of points in the process, to simulate for each curve in the time interval $\mathcal{T}$.

Step 2. Generate an equally spaced time sequence $\mathbf{t} = (t_1, \ldots, t_M) \in \mathcal{T}$, starting at $t_1 = 0$.

Step 3. Generate a M-dimensional random vector, $\mathbf{X}$, simulated from a multivariate normal distribution with mean vector $\mathbf{0}$ and positive definite covariance function, i.e., $\mathbf{X} \sim N(\mathbf{0}, Cov(X(\mathbf{s}), X(\mathbf{t})))$. Where the $M \times M$ elements of $Cov(X(\mathbf{s}), X(\mathbf{t})$ are specified by the choice of the covariance function.

Step 4. Generate a discretised curve according to $X(\mathbf{t}) = m(\mathbf{t}) + \mathbf{X}$ for a pre-specified choice of mean function $m(\mathbf{t})$.

Step 5. Repeat Step 2 - 4 to generate n vectors.

This represents our simulated discretised sample functions observed at M time points. We simulate from three different Gaussian Process using the previous algorithm. Model 1 considers a simulation of a Gaussian Process $\mathcal{GP}(m(t), Cov(X(s), X(t)))$ with mean function $m(t) = 80 \times (1-t) \times t^2$ and Stationary Covariance Function with hyper-parameters $c = 100$, $\kappa = 0.1$ and $\mu = 2$. Model 2 considers the same mean and Stationary Covariance Function with hyper-parameters $c = 0.125$, $\kappa = 1.55$ and $\mu = 1.24$. While Model 3 uses a Gaussian Process $\mathcal{GP}(m(t), Cov(X(s), X(t)))$ with mean and Stationary Covariance Function as in Model 1 but with hyper-parameters $c = 3.5$, $\kappa = 1.85$ and $\mu = 1.35$. For the three different model, we generate two different groups. The groups differs in the mean function, for group 1, the mean function is $m_0(t) = 80 \times (1-t) \times t^2$, while for the second group, we set $m_1(t) = \delta + m_0(t)$, with $\delta = 0.25$ and we generate 200 curves in each group. We set $M = 1000$, to have a finite grid, the sample functions are equally spaced with a gap between the two time points set to be 0.150. A graphical representation of a sample of ten curves

from such simulations is shown in Figure 2.1. In this figure the points have been connected by straight lines to create joined up curves for the plot.

2.2 Simulation by using Fourier basis functions

In this section we consider a method to simulate functional data using Fourier basis functions. To this end we start by defining a Fourier basis system on the interval $[a,b]$. This is

$$\begin{aligned}
\phi_0(t) &= 1 \\
\phi_1(t) &= \cos\left(\frac{\pi t}{L}\right) \\
\phi_2(t) &= \sin\left(\frac{\pi t}{L}\right) \\
\phi_3(t) &= \cos\left(\frac{2\pi t}{L}\right) \\
\phi_4(t) &= \sin\left(\frac{2\pi t}{L}\right) \\
&\vdots
\end{aligned} \tag{2.7}$$

where $L = (b-a)/2$. Observe that the first basis function is constant, while the successive basis functions are either cosine or sine functions with arguments being multiplied by integers $1,2,\ldots$. Such functions, $\phi_0(t),\phi_1(t),\phi_2(t),\ldots$ have the property that we can represent a function $X(t)$ by taking a linear combination of these basis functions, i.e.,

$$\begin{aligned}
X(t) &= \frac{\xi_0}{2} + \sum_{j=1}^{\infty}\left(\xi_j \cos\left(\frac{\pi jt}{L}\right) + \xi_{j+1}\sin\left(\frac{\pi jt}{L}\right)\right) \\
&= \sum_{j=0}^{\infty}\xi_j\phi_j(t)
\end{aligned} \tag{2.8}$$

for particular values of the ξ_j's. In practice, we can represent a sample function $x(\mathbf{t})$ as in equation (2.8) by replacing summing to infinity with to $2M$. More precisely,

$$x(\mathbf{t}) = \sum_{j=0}^{2M}\xi_j\phi_j(\mathbf{t}). \tag{2.9}$$

To simulate a single function $x(\mathbf{t})$, on $[a,b]$, we can obtain a random coefficient ξ_j for $j = 0,\ldots,2M$, by assuming it is a sample value from a univariate normal distribution $N(0,\sigma_j^2)$.

We will set $\sigma_0^2 > \sigma_1^2 > \ldots > \sigma_{2M}^2$, so the values of the weights tend to diminish as j increases. If we just require the values of $x(\mathbf{t})$ at $t_1, \ldots, t_M \in [a,b]$ we can evaluate them as follows

$$\begin{bmatrix} x(t_1) \\ x(t_2) \\ x(t_3) \\ \vdots \\ x(t_M) \end{bmatrix} = \begin{bmatrix} 1 & \phi_1(t_1) & \phi_2(t_1) & \ldots & \phi_{2M-1}(t_1) & \phi_{2M}(t_1) \\ 1 & \phi_1(t_2) & \phi_2(t_2) & \ldots & \phi_{2M-1}(t_2) & \phi_{2M}(t_2) \\ 1 & \phi_1(t_3) & \phi_2(t_3) & \ldots & \phi_{2M-1}(t_3) & \phi_{2M}(t_3) \\ \vdots & \vdots & \vdots & \ldots & \vdots & \vdots \\ 1 & \phi_1(t_M) & \phi_2(t_M) & \ldots & \phi_{2M-1}(t_M) & \phi_{2M}(t_M) \end{bmatrix} \begin{bmatrix} \xi_0 \\ \xi_1 \\ \xi_2 \\ \vdots \\ \xi_{2M} \end{bmatrix}.$$

To generate a sample of size n curves using Fourier basis functions, we repeat the same process obtaining a new random set of $(\xi_0, \ldots, \xi_{2M})$ for each new curve and the $\mathbb{V}ar(\xi_j) = \sigma_j^2$ is set according to one of the following scenarios:

Scenario 1. The basis coefficients ξ_{ij} are chosen to be independent and normally distributed with mean 0 and decaying variance $\sigma_j^2 = 2.5e^{-j/2}$, $j = 0, \ldots, 2M$; $i = 1, \ldots, n$.

Scenario 2. The basis coefficients ξ_{ij} are chosen to be independent and normally distributed with mean 0 and decaying variance $\sigma_j^2 = 2.5e^{-j/20}$, $j = 0, \ldots, 2M$; $i = 1, \ldots, n$.

For computational purpose, the simulations are implemented via the R package fda, we use the function **create.fourier.basis** to create an Fourier basis with specified period and the function **eval.basis** to evaluate the Fourier basis functions at different time points.

Figure 2.2(A) shows a slow decrease in terms of the variability of the basis coefficeints, while a rapid decrease in terms of the variance can be seen in Figure 2.2(B). In fact in Scenario 1 more than 90% of the variability is explained by the first 15 basis coefficients, while in Scenario 2 it is necessary to use only the first five principal basis coefficients explain the same variability. Figure 2.3 shows the shape of the simulation of Gaussian Process $\mathcal{GP}(m(t), K(s,t))$ for Scenario 1 and Scenario 2.

Another way to simulate functional data is by means of trigonometric functions. Simulated functional data using trigonometric functions was proposed by Ferraty et al. (2013). In such cases, the simulated functions are generated in the restricted interval $[0, \pi]$ and use a weighted cumulative distribution of univariate normal distributions. Let F_1 be the cumulative distribution function of a univariate normal distribution $N(\mu_1, \sigma_1^2)$ and F_2 the cumulative distribution function of a univariate normal distribution $N(\mu_2, \sigma_2^2)$ with some weights ω_1 and

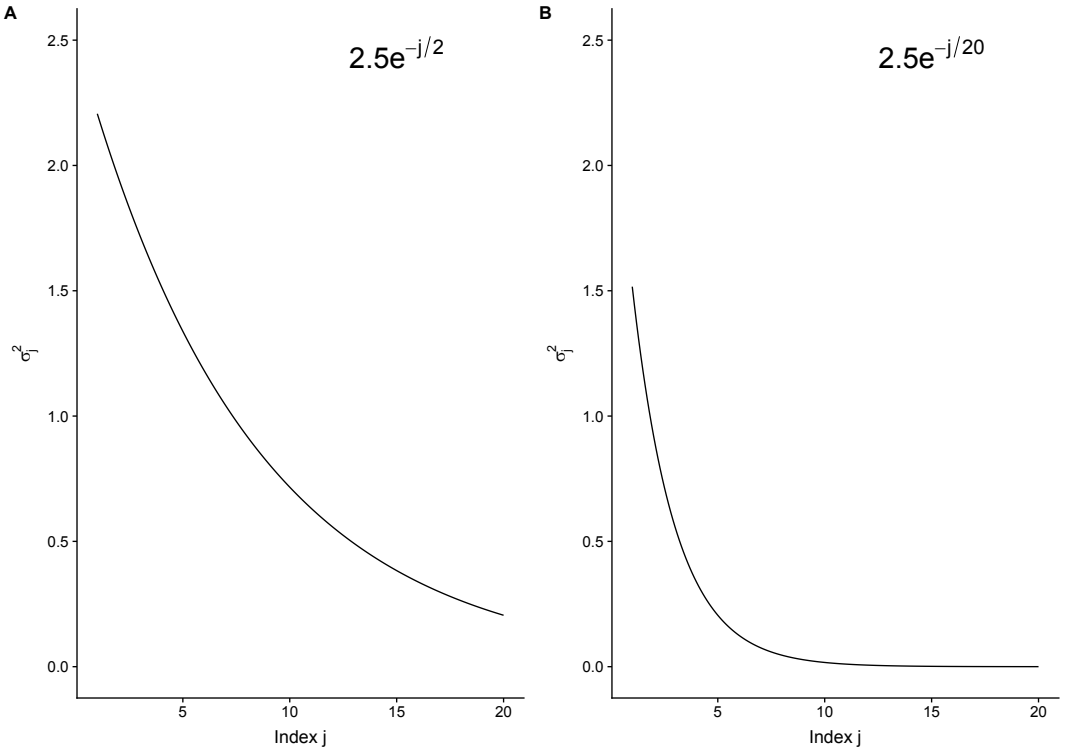


Figure 2.2: Decaying variances of the coefficients for the Fourier basis functions for (A) Scenario 1 and (B) Scenario 2.

ω_2 such that $\omega_1 + \omega_2 = 1$. The simulation by means of trigonometric function generates realisations from an independent, identically distributed random variable Z_i according to

$$Z_i \sim F_Z = \omega_1 F_1 + \omega_2 F_2, \tag{2.10}$$

and generates a sample function, at $\mathbf{t} = (t_1, \ldots, t_M)$ according to

$$x_i(\mathbf{t}) = Z_i \cos(2\mathbf{t}), \quad \text{for} \quad i = 1, \ldots, n. \tag{2.11}$$

A disadvantage of this method is that choosing a high value of the variance does not allow for heterogeneity in the data; the functions are similar smooth functions following a trigonometric defined function.

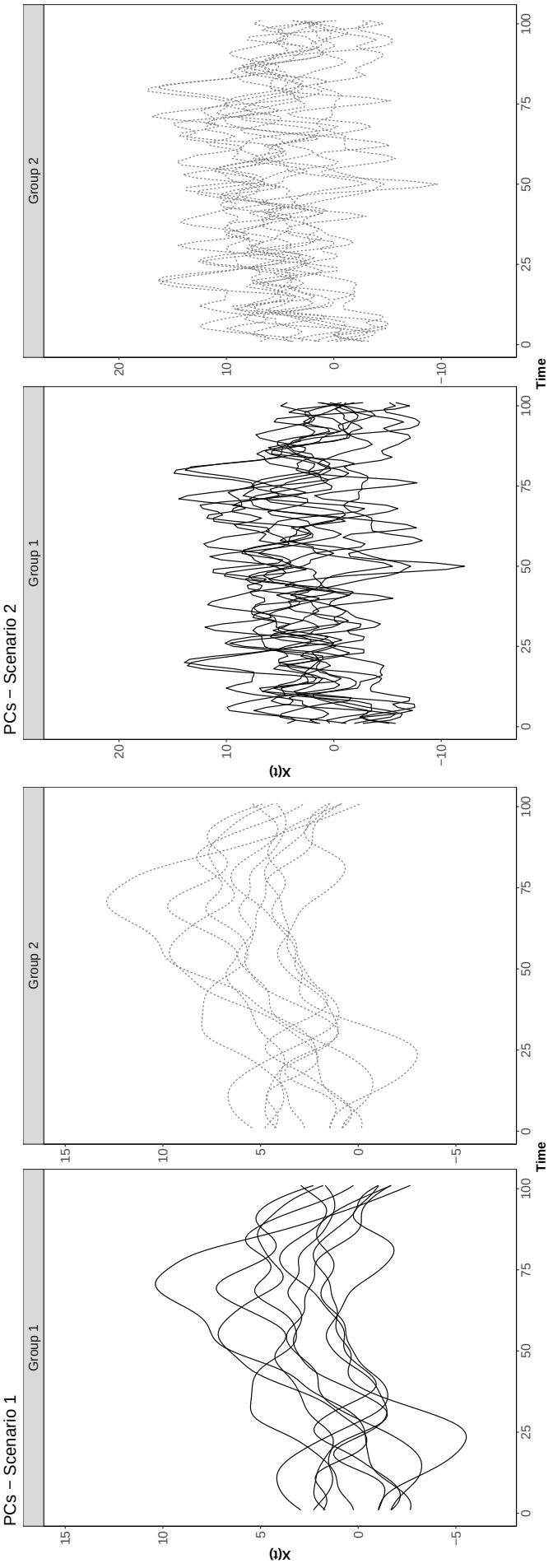


Figure 2.3: Simulated functional data with a mean function $m(t) = 80 \times (1-t) \times t^2$ for (left:) simulated according to Scenario 1 and (right:) simulated according to Scenario 2. The simulated functions are generated over $M = 1000$ time points equally spaced.

2.3 Simulation of samples containing atypical observations

A sample containing atypical curves is said to be *contaminated*. In this section, we describe models by which we can generate such contaminated samples. These methods will be applied to generating samples of data to be used in our simulations in Chapter 3, Chapter 4 and Chapter 5.

We consider the following models: a base model (or uncontaminated model), $\mathcal{M}_1$, the asymmetric contaminated model, $\mathcal{M}_2$. These models are called Linear Asymmetric Contamination Models (LACM) as in Kwon et al. (2016). We also consider a linear peak contamination model, $\mathcal{M}_3$ and a shape contamination model $\mathcal{M}_4$. Such models are specified below. The main difference between the models consists of introducing asymmetry into the curves and generating noisy data in the peaks (Kwon et al., 2016; Cuevas et al., 2007).

Model 1 - Uncontaminated Model This model is one of the simplest models and assumes that there are no atypical observations in the data. The uncontaminated model generates random functions according to

$$X_i(\mathbf{t}) = \mu(\mathbf{t}) + G_i(\mathbf{t}), \quad \text{for} \quad i = 1, \ldots, n. \tag{2.12}$$

Where $\mu(\mathbf{t})$ is the mean function and $G_i(\mathbf{t})$ is a realisation for the i^{th} sample curve from the Gaussian Process $\mathcal{GP}(\mathbf{0}, K(s,t))$ using a particular choice of the covariance function.

Model 2 - The Asymmetric Contaminated Model Let $\mathsf{M} \in \mathbb{R}$ be a constant determining the degree of movement of the atypical away from the uncontaminated observations. The symmetric contaminated model generates a functional dataset according to

$$X_i(\mathbf{t}) = \mu(t) + G_i(\mathbf{t}) + \mathsf{M} \cdot C_i \quad \text{for} \quad i = 1, \ldots, n, \tag{2.13}$$

where C_i takes the values of 1 with probability α and 0 with probability $1-\alpha$ for $0 < \alpha < 1$. The Asymmetric Contaminated Model generates curves which appear to be different in magnitude from the rest of the curves. By modifying the value of α we can control the expected number of asymmetric curves generated. In the simulations in Chapter 3 and Chapter 5 the value of α is set to be 0.25.

Model 3 - The Linear Peak Contaminated Model The Linear Peak Contaminated Model generates curves that are contaminated only in a short subinterval of length l. For a fixed value of l and a random number $T_i \sim U(0,1)$ the Linear Peak Contaminated Model generates a functional random variables according to

$$X_i(\mathbf{t}) = \begin{cases} \mu(\mathbf{t}) + G_i(\mathbf{t}) + \mathsf{M} \cdot C_i & \text{for} \quad T_i \leq t \leq T_i + l \quad \text{and} \quad i = 1, \ldots, n, \\ \mu(\mathbf{t}) + G_i(\mathbf{t}) & \text{for} \quad t \notin [T_i, T_i + l]. \end{cases}$$

Where $l \in \mathbb{R}$ is a constant that determines the time length of the of the peak and M is a constant determining the magnitude of the systematic movement away from $\mu(\mathbf{t})$. As before, C_i takes the values of 1 with probability α and 0 with probability $1-\alpha$ for $0 < \alpha < 1$. Again, $\alpha = 0.25$ is used for the simulations in Chapter 3 and Chapter 5.

Model 4 - Shape Contamination Model This model generates curves that have a different shape from the other curves, but are not necessarily far away from the main body of the distribution in terms of distance. To generate such curves, we consider a mixture of Gaussian Process. And we include functions with a different mean but the same covariance structure as the Gaussian Process used to generate the data. Let $G_1(t)$ be a $\mathcal{GP}(\mathbf{0}, K(s,t))$ and covariance function $K(s,t)$ and let $G_2(t)$ be a $\mathcal{GP}(m(t), K(s,t))$ with mean $m(t) = \mu_2(t) = 70 \times (1.1 - t) \times t^2$ but with the same covariance function as in $G_1(t)$. The Shape Contamination Model generates a functional random sample according to

$$X_i(\mathbf{t}) = \varepsilon \times G_{i1}(\mathbf{t}) + (1-\varepsilon) \times G_{i2}(\mathbf{t}), \quad \text{for} \quad i = 1, \ldots, n, \tag{2.14}$$

where ε is a Bernoulli random variable, $Be(p)$, where $p = 0.025$ is the probability of contamination. $G_{i1}(\mathbf{t})$ and $G_{i2}(\mathbf{t})$ are realistations of $G_1(\mathbf{t})$ and $G_2(\mathbf{t})$ for the i^{th} sample curve.

Figure 2.4 shows the simulation of the contaminated models for 100 functions in each group. The curves are equally spaced with a gap between the two time points set to be 0.150. The simulated functions are generated according to a Gaussian Process with the same mean function, $m(t) = 80 \times (1-t) \times t^2$, and Stationary Covariance Function with hyper-parameters $c = 100$, $\kappa = 0.1$ and $\mu = 2$. Similar to the simulations using Gaussian Process the groups differs in the mean function. We consider the mean function of the first group to be $m_0(t) = 80 \times (1-t) \times t^2$, while for the second group, we set $m_1(t) = \delta + m_0(t)$, with $\delta = 0.25$.

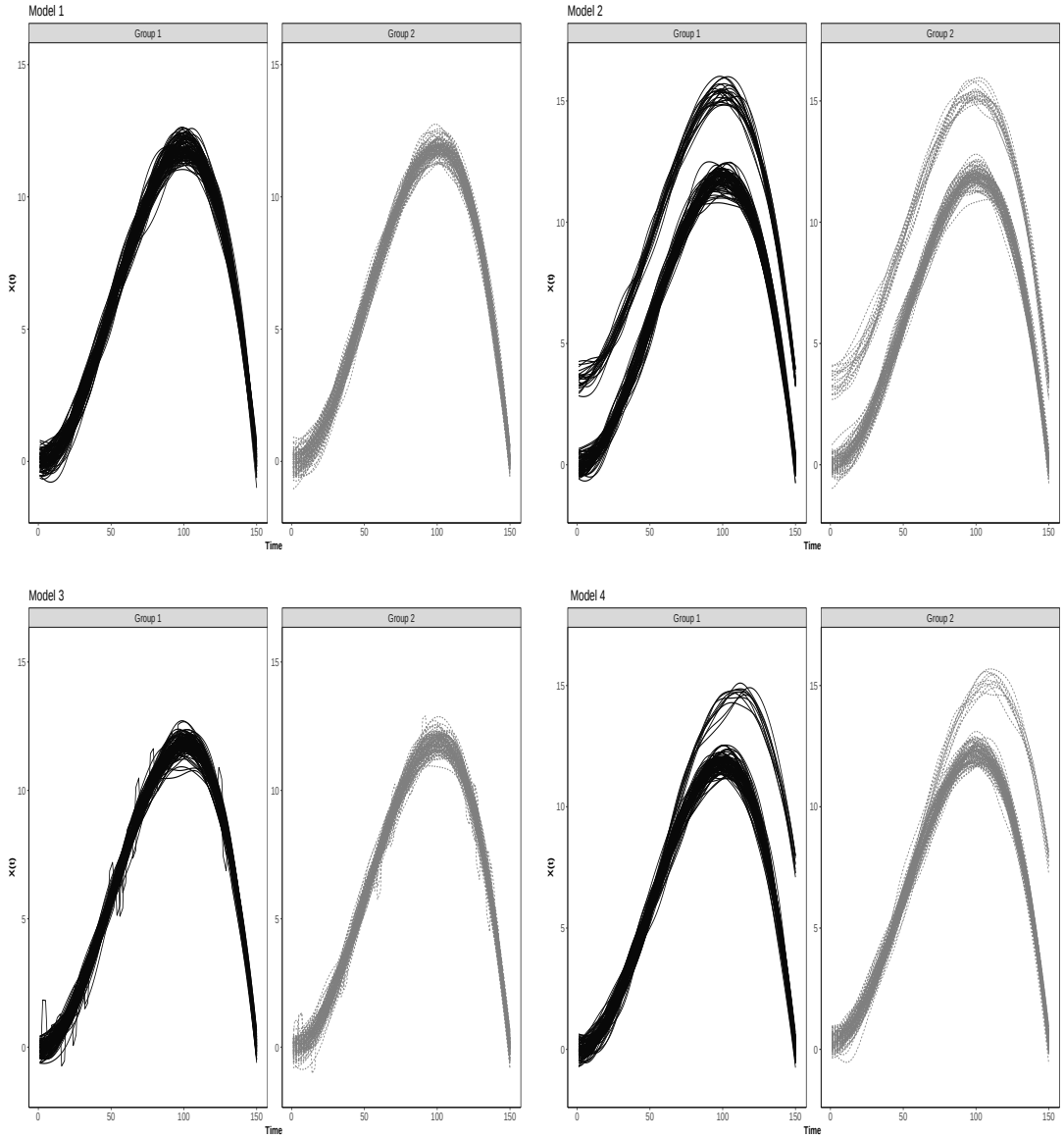


Figure 2.4: Contaminated models generated over a finite time interval with a gap between the two time points set to be 0.150. For (top-left:) The Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model.

2.4 Real Data sets utilised in this thesis

We consider three different real datasets. The first two real datasets are datasets for binary classification while the third dataset contains more than two groups.

The Orange Juice dataset: The orange juice dataset discussed in Li et al. (1996) is a well-known dataset corresponding to the near-infrared spectra of sugar (saccharosa) in the orange juice samples. The original dataset has a total of 218 orange juice samples collected from different countries in Europe. The curves are illuminated by a light beam at 700 equally spaced wavelengths, $t_1,\ldots,t_{700}$ in the near-infrared range of 1100 and 2500 nm. For each wavelength t, and each index i, the absorption $\log(1/R)$, where R is the light reflection on the sample surfaced of radiation, was measured. Along with the dataset, it contains a sucrose variable, a response, of an orange juice from its observed near-infrared spectrum. To form equally balanced groups, we split the original data $\left\{x_i^{OJ}(t_j),\ \text{for } j=1,\ldots,700 \text{ and } i=1,\ldots,218\right\}$ into two equally balanced groups using the sucrose content. The first group contains $n_0 = 109$ curves (sucrose > 40) and the second group is formed by $n_1 = 109$ curves with (sucrose < 40). We can observe from this dataset that there are some curves which appear to be different from the rest of the curves. Curves that are different with respect to shape and distance from the mayority of the curves can be seen in Group 1 in Figure 2.5, while what appear to be some atypical curves can also be seen in Group 2 in Figure 2.5.

The NIR spectra of gasoline dataset: The NIR spectra of gasoline dataset (Kalivas, 1997) is a data set containing sixty gasoline samples with a specific octane number. The samples were measured using the diffuse reflectance calculated using $\log(1/R)$, where R is the light reflectance on the surface, which varies from 900 to 1700nm in 2nm intervals giving a total of 401 wavelengths. The NIR spectra functional dataset can be represented as $\left\{x_i^{NIR}(t_j),\ \text{for } j=1,\ldots,401 \text{ and } i=1,\ldots,60\right\}$. To form equally balanced groups, we split the data into two equally balanced groups according to the octane content. The first population Π_0 is formed by $n_0 = 30$ curves with (octane < 88) and the second population Π_1 is formed by $n_1 = 30$ curves with (octane > 88). The gasoline dataset divided into two equally balanced groups is depicted in Figure 2.6.

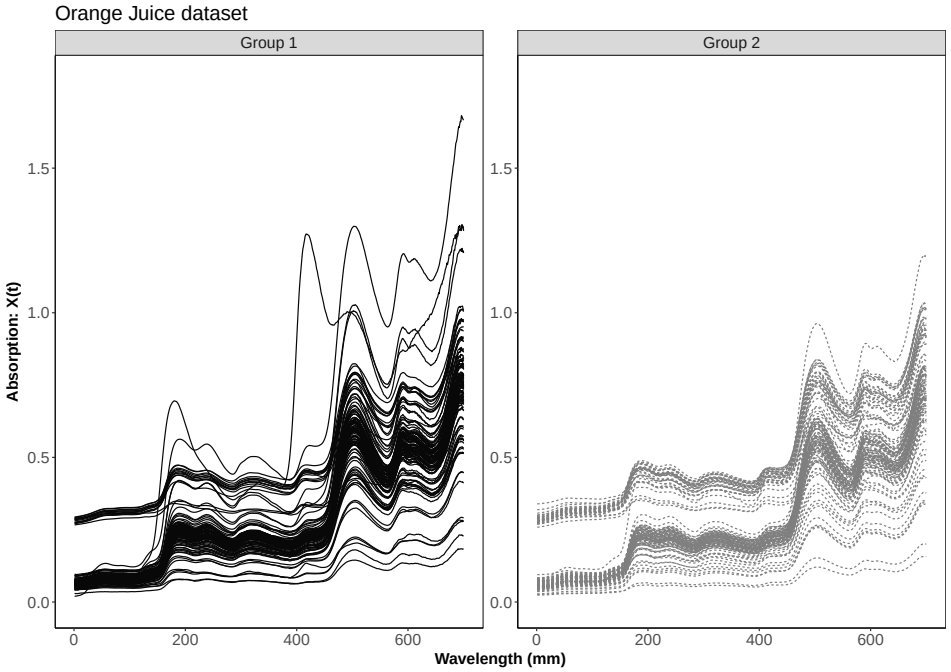


Figure 2.5: The near-infrared spectra of sugar in the orange juice samples. Group 1 consists of those with sucrose > 40 and Group 2 consists of those with sucrose < 40.

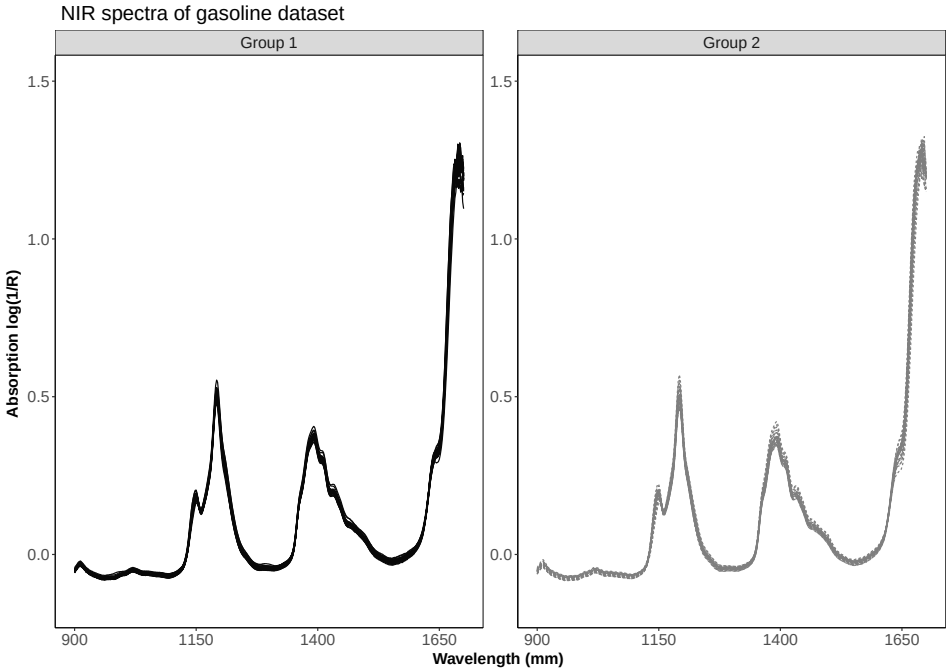


Figure 2.6: The NIR spectra of gasoline dataset. Group 1 consists of those with octane < 88 and Group 2 are those with octane > 88.

The phoneme dataset The third real dataset that we consider in this thesis is the digitised speech phoneme data, also known as the phoneme dataset. The phoneme dataset is used

in Ferraty and Vieu (2003) and Hastie et al. (1995). It is a widely used dataset in speech recognition and it is available from the website: `www-stat.stanford.edu/ElemStatLearn`. The data were extracted from the TIMIT database (TIMIT Acoustic-Phonetic Continuous Speech Corpus, NTIS, US Dept of Commerce). The original dataset was formed by selecting five phonemes for classification based on digitized speech from this database. The phonemes are considered the following: *sh* as it sounds in the word *she*, *dcl* as in the word *dark*, *iy* as the vowel in *she*, *aa* as the vowel in the word *dark*, and *ao* as the first vowel in the word *water*. However, we consider a reduced version of the dataset, with only the three different phoneme classes *sh*, *iy* and *dcl*. From each speech frame, the log-periodograms are constructed from a 32ms recording of a male pronouncing the three different phonemes. Thus, the data consists of 1200 log-periodograms each recorded at 256 equally-spaced points, with known classes (phoneme) memberships. For the three phonemes classes *sh*, *iy* and *dcl*, there are 400 samples of log-periodograms in each. The reason we consider these three groups of log-periodograms is that they are difficult to distinguish between them due to the overlap of the curves.

For the purpose of the plot we have smoothed the 256 observations in a single log-periodogram using a linear polynomial kernel smoother with a smoothing parameter chose for each curve using least squares cross-validation. For implementation the *locpoly* function in the R package **MASS** was used. We kept only the least noisy part by truncating all the curves to the interval spanning the first 150 observed values as shown in Figure 2.7.

2.4.1 Creation of real grouped dataset with imbalanced group size

In real classification problems, the scenario of imbalanced group sample sizes appears frequently. In binary classification, strongly imbalanced classes can lead to unsatisfactory results. The main problem with imbalanced groups arise when we have many more samples functions in one group than the other. When creating a classification rule with such data we have far more information about the distribution of function in one group than the other. Such a rule may then be quite inaccurate. We discuss more in detail the scenario of imbalance observations for functional data in Chapter 4. However, we start describing how imbalanced datasets can be created from the first two binary real dataset. For the first imbalanced dataset, we used the sucrose content in the orange juice data to separate the data into two populations, while for the gasoline dataset, we use the octane content.

To form unequally groups in the orange juice dataset, we split the original data into two

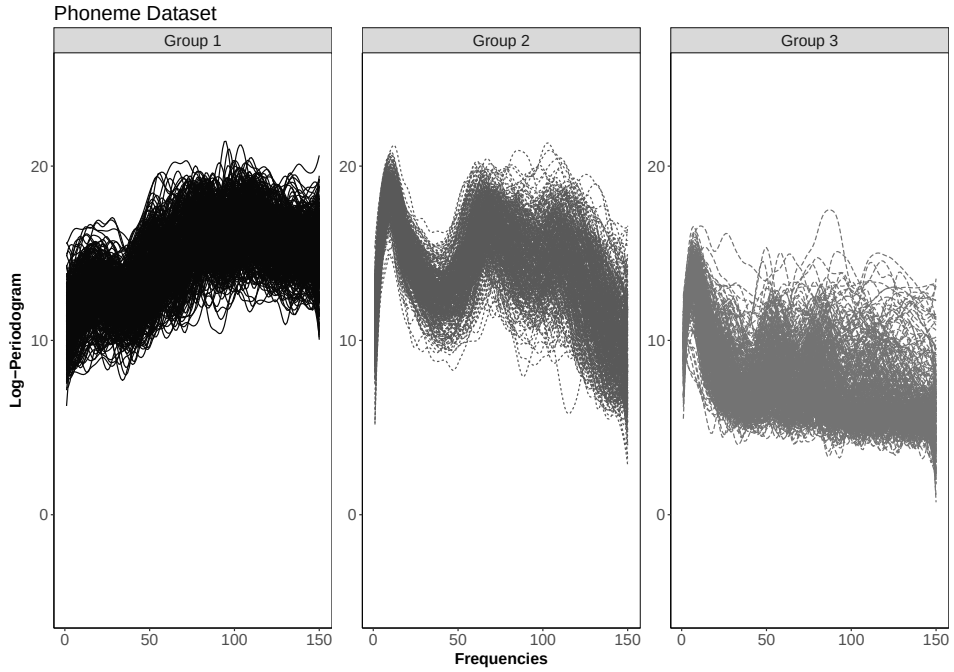


Figure 2.7: The phoneme dataset. The log-periodograms of three different phoneme classes. Group 1 correspond to the phoneme *sh*, Group 2 to the phoneme *iy* and Group 3 to the phoneme *dcl*.

groups according to a threshold of the univariate variable sucrose content. By considering this, the new dataset is an imbalanced dataset, with $n_0 = 48$ (sucrose < 30) curves in the first population and $n_1 = 170$ curves (sucrose > 30) in the second population Π_1. Figure 2.8 shows the near-infrared spectra of sugar in these two orange juice samples.

To create the second imbalanced dataset, we consider the NIR spectra of gasoline dataset. As we mentioned before, the original data set contains 60 gasoline samples each associated with a specific octane content. To form two imbalanced groups, we split the data into two groups according to the octane content. The first group is formed by $n_0 = 15$ curves with (octane < 86) and the second group is formed by $n_1 = 45$ curves with (octane > 86). Table 2.1 summarise the sample sizes in each group.

Table 2.1: Different sample sizes of each group for the real imbalanced data sets.

	Real Dataset 1	Real Dataset 2
n_0	48	15
n_1	170	45

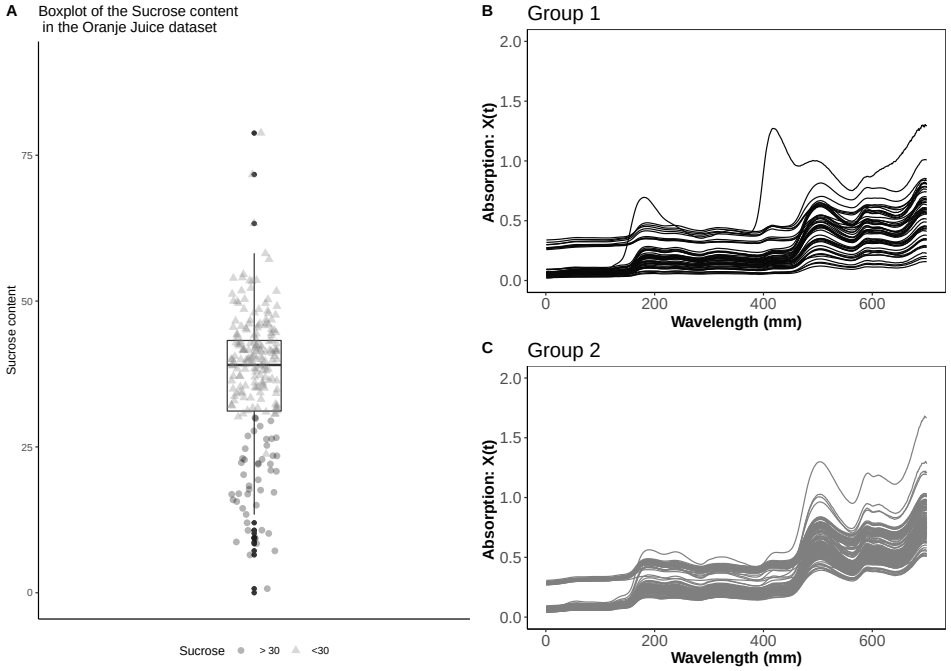


Figure 2.8: The imbalance near-infrared spectra of sugar in the orange juice samples. (A) Boxplot of the sucrose content divided into two different groups according to the sucrose content. (B) Group 1 consists of those with sucrose > 30 and (B) Group 2 consists of those with sucrose < 30.

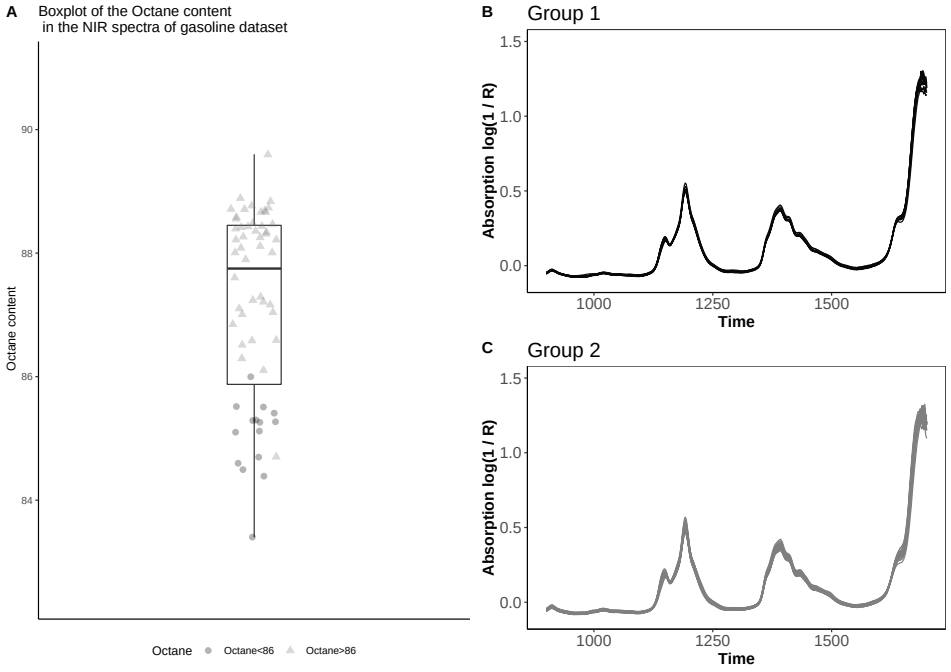


Figure 2.9: The imbalance NIR spectra of gasoline dataset. (A) Boxplot of the NIR spectra divided intro two different groups according to the octane content. (B) Group 1 consists of those with octane < 86 and (B) Group 2 are those with octane > 86.

2.5 Conclusions

This chapter introduced different settings and techniques for simulating functional data. We saw that the simulation methods for functional data use Gaussian Processes or Fourier basis functions. Such techniques provide the basis to carry out simulations for the proposed methods in the later chapters. By simulating data using Gaussian Processes we saw that the degree of smoothing in the data can be controlled by modifying the parameter of the covariance function.

The second approach is motivated by Fourier basis functions and the representation of a curve as a linear combination of these basis functions. The method considered here generates functions allowing the basis coefficients to be chosen randomly from a Normal distribution with a decaying variance as the the number of coefficients increases.

Finally, we described the real datasets that we use in the simulation studies and the creation of imbalanced datasets. We described three real datasets that are part of our simulation studies, including a real dataset with more than two groups.

Chapter 3

k-Ranked Nearest Neighbours

3.1 Introduction

The k-Nearest Neighbours (k-NN) rule is one of the popular rules among the machine learning community. The report written on February 1951 by Evelyn Fix and J.L. Hodges, Jr. contained not only work on probability density estimation and nonparametric discriminant analysis, but also contained one of the most simple rules of discrimination analysis, the k-NN rule. Functional data makes use of different nonparametric tools to address certain statistical problems, and the use of nearest neighbours techniques, due to its straight forward implementation, is not an exception.

The k-Ranked Nearest Neighbours (k-RNN) algorithm using functional depth is the motivation behind this chapter and it serves as the basis of our first contribution which is based on using nearest neighbours methods. We start this chapter by proposing a new algorithm to classify curves from different groups using a k-RNN approach. Our algorithm is based on using functional depth as a means of ranking the curves relative to the *center* of the data. We start by describing the proposed algorithm in detail along with a running example. Later, we exploit the fact that our k-RNN algorithm can be interpreted in terms of a conditioned probability function which can be used to classify new curves. In particular, we use a generalized additive logistic regression model to estimate the conditional probability function with covariates corresponding to the signed depth and the distance to the mode for functional observations. By means of an extensive simulation study we compare the performance of the proposed classifier against other nearest neighbours and depth classifiers. We also looked at the performance of the proposed classifier on real datasets.

The structure of this chapter is as follows: Section 3.2 describes the proposed algorithm for supervised classification based on functional depth. Section 3.3 illustrates various parts of the proposed methodology using a simulated Gaussian Process. In Section 3.4 we describe the k-RNN as a regression approach and we introduce the k-RNN as a moving average. Section 3.5 and Section 3.6 introduces the signed distance integral as a classifier and proposes a Bayesian approach to classify new observations using the signed depth. The use of logistic regression to model the signed depth and the distance to the mode, is presented in Section 3.7. A discussion about generalized additive models (GAM) is given in Section 3.8 along with an iterative procedure to prevent over-fitting of the training data. Section 3.9 discusses the extension to more than two groups. Section 3.10 describes the simulation study we carry out, along with the methods that will be used to compare with our proposed classifier based on signed depth and distance to the mode. Results of the comparisons are given in Section 3.11, while Section 3.12 details the real datasets applications. Finally, conclusions are stated in Section 3.13.

3.2 The k-RNN as an alternative of k-NN

The k-nearest-neighbour classifier k-NN, proposed by Fix and Hodges Jr (1951), is a non-parametric classification rule that searches in a metric space. Given a fixed value of k, it assigns an unlabelled observation, $\mathbf{x}_0 \in \mathbb{R}^p$, to the class with the maximum number of representatives in the set of k labeled observations closest (in metric) to $\mathbf{x}_0$. In the case of a tie, a random tie breaking procedure is used. Formally, consider a vector in which the observations are coupled with a group membership indicator variable, i.e., $\{(\mathbf{x}_i, y_i)\}$ for $i = 1, \ldots, n$, where y_i is called the group label (or class labels) and $y_i \in \{0, 1, 2, \ldots, G-1\}$ for G classes or groups associated with the the observations $\mathbf{x}_i \in \mathbb{R}^p$ and for some $p \in \mathbb{N}$. We assume that each of $\mathbf{x}_i$'s and y_i's represent observations of the random vectors $\mathbf{X}$ and $\mathbf{Y}$. In the classical k-NN algorithm, a new observation $\mathbf{x}_0$ is classified by finding the k nearest observations, using a distance metric, i.e., $d_{(i)} = \left\|\mathbf{x}_{(i)} - \mathbf{x}_0\right\|$, among these k neighbours and assigning it the label which is most frequently represented among the k nearest neighbours, where the distance metric is commonly taken to be the Euclidean norm.

The k-NN classifier can be modified by introducing a degree of robustness to the algorithm. One approach is introduced by using the concept of ranks, which provide a monotonic

ordering for the data. Suppose that F is a probability distribution in $\mathbb{R}^p$, then a depth function, $D(\cdot) : \mathbb{R}^p \to \mathbb{R}^+$, measures how deep a central observation $\mathbf{x}_i \in \mathbb{R}^p$ is with respect to F or with respect to a sample $\{\mathbf{x}_1, \ldots \mathbf{x}_n\}$. Such a function $D(\cdot)$ can provide an ordering of the points $\mathbf{x}_i \in \mathbb{R}^p$. Observations with high values of depth correspond to central observations, and the point $\mathbf{x}_i$ that maximises the depth function represents the *centre* of the distribution.

Several depth definitions for multivariate data have been proposed. For instance we can find the Halfspace (Tukey) depth (Tukey, 1975), Mahalanobis depth (Mahalanobis, 1936), the simplicial depth (Liu et al., 1999), the projection depth (Donoho, 1982), the simplicial volume depth (Zuo and Serfling, 2000), the Oja depth (Zuo and Serfling, 2000) and the Zonoid depth (Koshevoy and Mosler, 1997), among others. All of these have the same purpose, i.e., to measure how central a given point $\mathbf{x}_i \in \mathbb{R}^p$ is with respect to F or with respect to a sample.

Consider the Mahalanobis depth to measure how deep is an arbitrary point $\mathbf{x}_0 \in \mathbb{R}^p$ with respect to some observations $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. It assigns to $\mathbf{x}_0$ its degree of centrality according to $D(\mathbf{x}_0 \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = (1 + (\mathbf{x}_0 - \bar{\mathbf{x}})' \hat{\Sigma}^{-1} (\mathbf{x}_0 - \bar{\mathbf{x}}))^{-1}$, where $\bar{\mathbf{x}}$ is the mean vector and $\hat{\Sigma}$ is the covariance matrix of the data points. The vector $\mathbf{x}_0$ which will maximise $D(\mathbf{x}_0 \mid \mathbf{x}_1, \ldots, \mathbf{x}_n)$ is $\bar{\mathbf{x}}$ and, as such, can be interpreted as the center of the empirical distribution of the data. Values of $D(\mathbf{x}_0 \mid \mathbf{x}_1, \ldots, \mathbf{x}_n)$ which are close to 1 will indicate that $\mathbf{x}_0$ is close to $\bar{\mathbf{x}}$ and so to the center of the distribution of the data.

The k-RNN classifier can be applied to multivariate and univariate data. For a new observation $\mathbf{x}_0 \in \mathbb{R}^p$ and a known multivariate depth function, we start by computing the depth for the new observation with respect to the combined sample of the new observation and the available observations, then we rank all the observations according to the depth. For a fixed value of k, and to predict the label of the new observation, we select k observations above the ranking of $\mathbf{x}_0$, and k observations below the ranking of $\mathbf{x}_0$. The new observation is classified by a majority vote in this $2k$ neighbourhood.

The k-NN classifier can be extended to the functional case. However, there are two main difficulties. The first one is that in the infinite dimensional space, the k-NN classifier is not consistent, as discussed by Cérou and Guyader (2006) (in finite dimensional space the k-NN classifier is consistent, i.e., that the probability of error converges to the Bayesian error, irrespective of the distribution of $(\mathbf{x}, \mathbf{y})$ see Stone (1977a)). The second difficulty is that most of the multivariate depth measures are computationally intensive and are not appropriate for functional data as discussed by López-Pintado et al. (2010) and Liu et al. (1990).

Therefore, different concepts of depth for functional data have been proposed. Fraiman and Muniz (2001) introduced a concept of depth for functional observations based on the integrals of univariate depths, while López-Pintado and Romo (2009) introduced the notion of functional depth based on the graphic representation of the functions and the bands these graphs determine in the plane. In the functional case, functional depth again introduces the notion of a *centre* as the maximal depth point and can this provide an ordering of the observations inwards to the *centre*. Functional depths are useful for outlier detection and rank test as shown in Hubert et al. (2015) and López-Pintado and Romo (2009).

Our goal is to introduce the k-RNN for functional data as an alternative to classifiers based on functional depth. This work is motivated by Anderson (1966), who originally proposed the k-RNN classifier for multivariate data and was rediscovered in Bagui et al. (1995). The k-RNN classifier has been studied by Bagui et al. (2003) with some applications in the context of multivariate data and by Bagui and Pal (1995) who extended the idea to more than two populations. Discussion of using the k-RNN classifier for functional data has not up to now been considered in the literature.

3.2.1 The k-RNN using depths

We start by describing the k-RNN classifier using depths for the classification of functional data. Let us consider a binary classification task and suppose that we observe an equally balanced collection of curves coming from two different populations Π_0 and Π_1, taking values in a functional space, $\mathcal{F}$, in the same close interval $t \in [a,b]$. We denote by n_0 and n_1 the number of observations in each group. The total sample size, N, is obtained by adding the number of observations in each population.

In supervised classification, we observe a finite number of curves coupled with a group membership $\{(x_i(t), y_i)\}$. The group label y_i takes values according to

$$y_i = \begin{cases} 0 & \text{if } \; x_i(t) \in \Pi_0, \\ 1 & \text{if } \; x_i(t) \in \Pi_1, \end{cases}$$

for i's in the total sample size and the aim is to construct a classifier $C(\cdot) : \mathcal{F} \to \{0,1\}$, which maps a new observation $x_0(t)$ to a predicted label. Usually, classifiers are constructed based on a training sample $\mathcal{T}rain = \{(x_i(t), y_i)\}$ for $i = 1, \ldots, n_{\mathcal{T}rain}$, which contains a sample of the two populations Π_0 and Π_1, and where the model *learns* from the data and a test sample

$\mathcal{T}est = \{(x_i(t), y_i)\}$ for $i = 1, \ldots, n_{\mathcal{T}est}$, is used to evaluate a given model.

Suppose that a finite number of curves, $\{(x_i(t), y_i)\,,\ t \in \mathcal{T},\ i = 1, \ldots, N\}$, coupled with a group membership is available and contains an equally balanced sample of the two populations Π_0 and Π_1. To classify using the k-RNN classifier, we begin by ranking all the observations in the functional dataset by their depth values. The depth of $x_0(t)$ is calculated with respect to the combined samples and it determines where the observation falls in terms of its ranking when combined with both groups. Given $k \in \mathbb{Z}^+$, the unknown observation, $x_0(t)$, is assigned to the population Π_i if the majority of the k rank nearest neighbours of $x_0(t)$ come from population Π_i $(i = 0, 1)$, and in case of ties, break the ties randomly.

The k-RNN classifier using depths as described has the drawback that two or more curves which are in different groups and positions in the plane $G(x(t)) = \{(t, x(t))\,;\ \ t \in \mathcal{T}\}$, relative to the *center*, can have the same depth. Therefore, to overcome this difficulty, we propose to introduce a new method to assign a sign to each rank by considering the position of each curve relative to a reference curve.

To achieve an ordering of the curves, we need to select a reference curve from the sample. One such choice is the curve in the combined groups which have the largest sample depth. The choice of the particular reference curve is not unique, but serves to provide an order of the sample functions. For instance, alternatives, such as the maximum or the minimum curve will still provide an ordering to the observations.

Subsequently, we will choose the reference curve, $x_{ref}(t)$, to be the sample median curve calculated using the two groups of data based on using a depth measure which is a maximum at the sample median. The concept of the sample median for functional data will be discussed in detail in Section 3.7.1. More formally,

$$x_{ref}(t) = \max_{1 \le i \le N} D(x_i(t)).$$

If there is not a unique highest value, the reference curve is taken to be the average of the curves maximising the depth function; this will be discussed in more detail in Section 3.3.2. The next step is to assign a sign to each curve $x_i(t)$ in the sample; to achieve this we compute the **signed distance integral**

$$\int_{\mathcal{T}} \big(x_i(t) - x_{ref}(t)\big)\, dt, \tag{3.1}$$

which enables us to order or rank the curves in a monotonic fashion from *top* to *bottom* or vice versa. The **signed depth (sdp)** of an observed curve $x_i(\mathbf{t})$, is then calculated as

$$sdp(x_i(\mathbf{t})) = sgn\{D(x_i(\mathbf{t}))\} \times D(x_i(\mathbf{t})), \tag{3.2}$$

where

$$sgn\{D(x_i(\mathbf{t}))\} = \begin{cases} +1 & \text{if } \int_{\mathcal{T}} \left(x_i(t) - x_{ref}(t)\right) dt > 0, \\ -1 & \text{Otherwise,} \end{cases}$$

for $i \in \{1,\ldots,N\}$. Therefore, for curves deemed above the reference curve, the signed depth will be positive. While for curves below the reference curve, the signed depth will correspond to a negative depth. By doing this, we obtain a new set of paired variables $\{(sdp(x_i(\mathbf{t})), y_i)\}$ for $i = 1,\ldots,N$.

The reference curve plays an important role in the case of the k-RNN. It is used to determine the sign of each of the depth values, via the signed distance integral in equation (3.1), which assigns a sign to each depth by considering each curve position relative to the reference curve. However, it can be the case where curves intersect the reference curve.

Intersections of a curve or curves with the reference curve can happen in two different cases. In the first case, where curves do not intersect, curves with a positive signed distance integral judge to be above the reference curve, while curves with a negative signed distance integral are below the reference curve. In the second case, when a function crosses the reference curve, the signed distance integral is divided into different regions. In the regions where the curve $x_i(\mathbf{t})$ is below $x_{ref}(\mathbf{t})$, the signed distance integral calculates the area between $x_i(\mathbf{t})$ and $x_{ref}(\mathbf{t})$ as being negative. Similarly, for the regions where the curve $x_i(\mathbf{t})$ is above $x_{ref}(\mathbf{t})$, the signed integral calculates the area between $x_i(\mathbf{t})$ and $x_{ref}(\mathbf{t})$ as positive. The overall value is obtained by adding all the areas in such regions which will result in either a positive or negative value for the integral in (3.1).

Figure 3.1 shows a graphical description of this procedure in the case that curves do not intersect while Figure 3.2 show the case when there is only one cross of the curve with the reference curve.

Formally, the k-RNN classifier based on the signed depths (sdp), may be described by the following steps.

Suppose that the signed depths of a sample of functions are available, $\{(sdp(x_i(\mathbf{t})), y_i)\}$

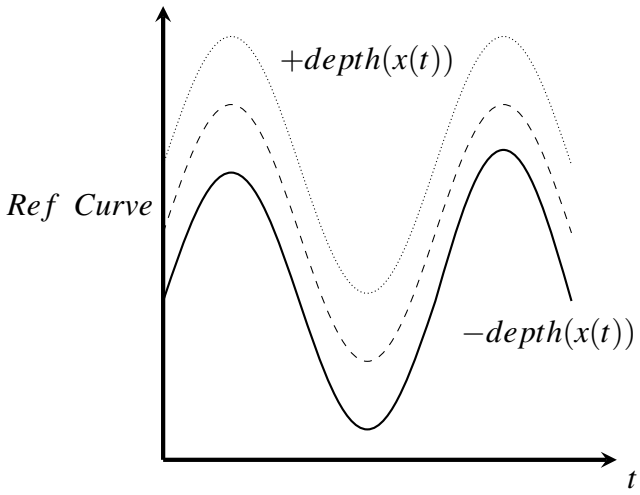


Figure 3.1: Graphical representation of the reference curve when curves do not intersect.

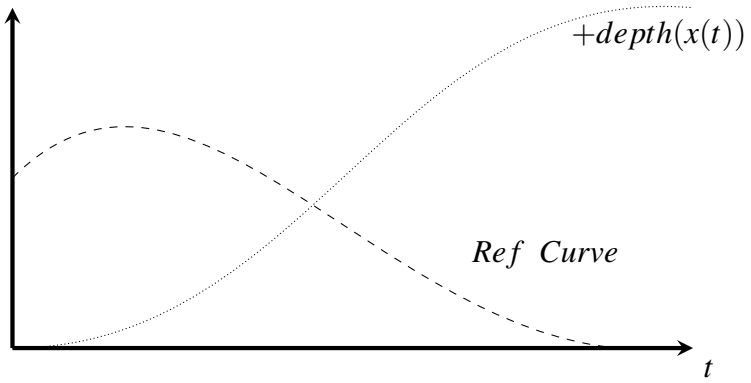


Figure 3.2: Graphical representation of the reference curve when there is only one intersection when a larger proportion is positive.

for $i = 1,\ldots,N$, contain an equally balanced sample of the two populations Π_0 and Π_1. To classify a new observation $x_0(\mathbf{t})$ using the k-RNN classifier, we start by ranking all the observations by their depth values. We then calculate the depth of $x_0(\mathbf{t})$ with respect to the combined samples and assign a sign to the depth of the new observation according to the signed distance integral, forming the signed depth of the new observation $sdp(x_0(\mathbf{t}))$. Then, we take k observations below the $sdp(x_0(\mathbf{t}))$ and k observations above the $sdp(x_0(\mathbf{t}))$ and form a $2k$ neighbourhood formally defined in Section 3.4.1. If the signed depth of the new observation we want to classify falls near one of the boundaries, the k-RNN use as many observations as available. We summarise the k-RNN signed depth classifier in Algorithm 1.

Algorithm 1: The k-RNN algorithm based on signed depth (sdp) for functional data.

Input : A depth function $D(\cdot)$, a fixed $k \in \mathbb{Z}^+$ and a functional dataset $\{(x_{ij}(\mathbf{t}), y_{ij})\}$, for $j = 0, 1$ in the two group case. The respective group sizes are n_0 and n_1 with $N = n_0 + n_1$

Output : Classifications for the functional dataset.

1 Calculate $D(\cdot)$ for each $\{(x_{ij}(\mathbf{t}), y_{ij})\}$ in the combined samples and then compute the $sdp(x_{ij}(\mathbf{t}))$ for $i = 1, \ldots, N$ and $j = 0, 1$ with reference to $x_{ref}(\mathbf{t})$, the median curve calculated using the two groups of the data.

2 Rank all $\{(x_{ij}(\mathbf{t}), y_{ij})\}$, $i = 1, \ldots, N$; $j = 0, 1$ according to $sdp(\cdot)$.

3 Let $x_0(\mathbf{t})$ be a new curve we want to classify to a predicted label.

4 Compute the signed depth of $x_0(\mathbf{t})$, according to equation (3.2).

5 Rank the signed depth of $sdp(x_0(\mathbf{t}))$ among the signed depths of the sample functions in both groups.

6 Consider the observations having ranks k above and k below $x_0(\mathbf{t})$. Assign $sdp(x_0(\mathbf{t}))$ to the population for which the majority of the k-RNN's belong.

7 **if** *exactly k observations belong to each group* **then**

8 break the tie randomly with probability $1/2$.

9 **else if** *there are not k observations smaller and k larger than* $sdp(x_0(\mathbf{t}))$ *(or the other way around as occurs in the boundary regions)* **then**

10 just use those available.

3.3 Running example

To illustrate various parts of the methodology of the k-RNN classifier, we consider a Gaussian process to simulate from functional data. Let $\mathcal{GP}(m(t), K(s,t))$ be a Gaussian process with mean $m(t) = 80 * (1-t) * t^2$ and $K(s,t) = 0.1 * \exp(-100 * (s-t)^2)$, $t \in [0, 150]$ as detailed in Chapter 1. We generate two different populations Π_0 and Π_1, that differ in terms of the mean function, with $n_0 = 200$ and $n_1 = 200$ curves in each population. The mean function for the population Π_0 is $m_0(t) = 80 * (1-t) * t^2$ while the mean for the population Π_1 is replaced by $m_1(t) = \delta + m_0(t)$ where $\delta = 0.35$, i.e., in Π_1 the mean function is shifted upwards by δ units over the whole range of t. The simulated curves are observed over a grid of $t_1, \ldots, t_{1000}$.

With a gap between the two time points set to be 0.150. Figure 3.3 shows a simulation of the Gaussian process with two different populations Π_0 and Π_1.

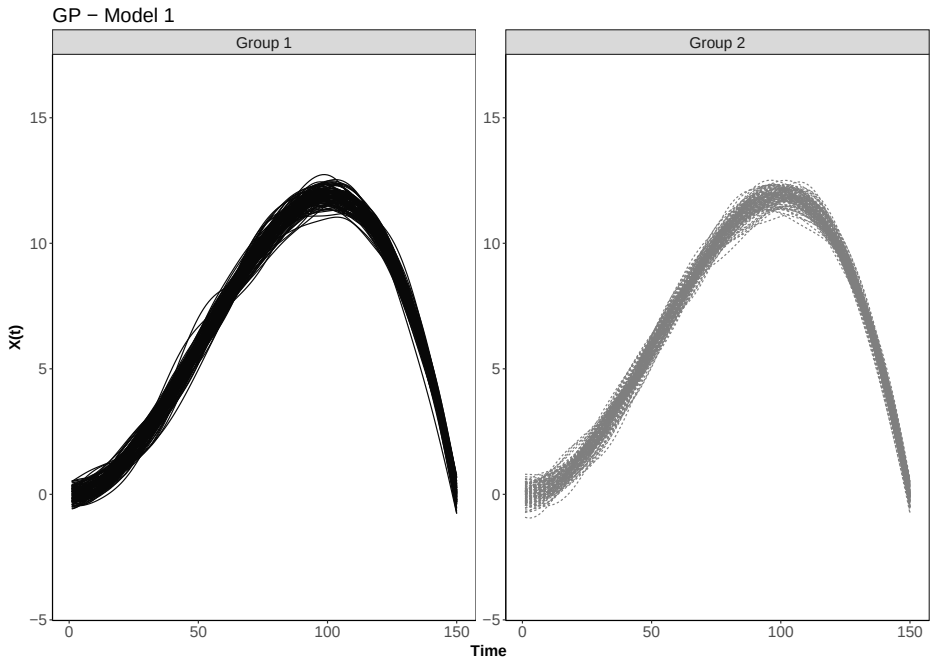


Figure 3.3: Samples drawn from Gaussian process $\mathcal{GP}(m(t),K(s,t))$ with mean $m(t) = 80*(1-t)*t^2$ and $K(s,t) = 0.1*\exp(-100*(s-t)^2)$ of size 200 observed over $M = 1000$ time points and a gap between the two time points to set to be 0.150.

Observe that δ, allows us to control the overlapping of both groups. A large value of δ generates populations far apart from each other and vice versa. To start showing how we achieve an ordering of the curves and how we select the reference curve, we started by considering the Fraiman-Muniz depth. The Fraiman-Muniz depth is strictly defined in Section 3.3.2.

Depth functions will be explained in more detail in Section 3.3.2. We start by ranking all the simulated data according to the Fraiman-Muniz depth. We choose the reference curve, $x_{ref}(\mathbf{t})$, to be the curve that maximises the Fraiman-Muniz depth with respect to the combined samples.

Figure 3.4 shows the reference curve, for the simulated data as a solid line. The dashed lines in Figure 3.4 shows the the lower and upper bands for each population in the running example. The upper bands are calculated by determining the proportions of time each curve stays above the other curves in each group and the lower bands are determined by estimating the proportion of time that the curve stays below the other curves. For example, if we consider curves $x_k(\mathbf{t})$ and $x_j(\mathbf{t})$ $(k \neq j)$ in the same group and using $\mathbf{t}$ as the discretised vector

$(t_1, \ldots, t_M)$ then we count the number of t_i's for which

$$x_k(t_i) > x_j(t_i).$$

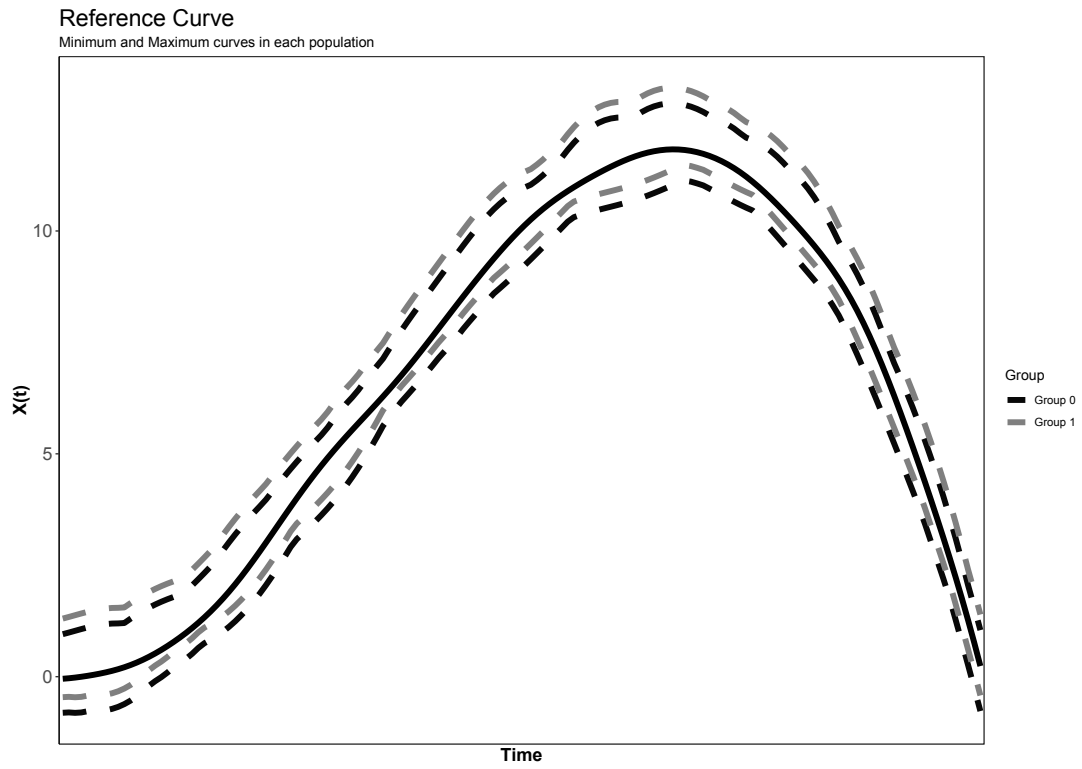


Figure 3.4: Reference curve in solid line for the simulated data.

3.3.1 Choosing the value of k

Cross-validation is widely used in supervised classification problems to chose the value of the k, known as the tuning parameter. For the k-RNN, by training and testing the classifier on separate subsets of the data, we get an estimate of the misclassification error rate as a function of the number of neighbours. This estimate admits many properties (see Stone (1977b) for a discussion of asymptotic consistency and efficiency).

As previously discussed, the k-RNN classifier depends on two main considerations, the first consideration is the depth function $D(\cdot)$ and the second considerations is the number of the nearest neighbours considered, k, which plays an important role. To determine the value of k, we consider R-fold cross-validation, which can be described with the following four steps.

Step 1. For the pairs of observations $\{(sdp(x_i(\mathbf{t})), y_i)\}$, where $i = 1, \ldots, N$, divide the data set of signed depth observations into R subsets (or folds) and let $F_1, \ldots, F_R$, be the indices of the signed depth in each of the folds. Note that the size of each subset is either $\lfloor N/R \rfloor$ or $\lfloor N/R \rfloor + 1$, where $\lfloor x \rfloor = \max\{m \in \mathbb{Z}^+ \mid m \leq x\}$.

Step 2. For $\kappa = 1, \ldots, R$ and a fixed $k_{max} \in \mathbb{Z}^+$, consider a training set formed by $(sdp(x_i(\mathbf{t})), y_i)$, $\forall i \notin F_\kappa$ and then validate on the the pairs $(sdp(x_i(\mathbf{t})), y_i), \forall i \in F_\kappa$. For each value of k in $k \in \{1, \ldots, k_{max}\}$, classify the observations according to the k-RNN classifier and estimate the misclassification error as

$$\mathcal{E}rr_\kappa(k) = \sum_{i \in F_\kappa} \frac{1}{n_\kappa} \mathbb{1}_{\{y_i \neq \hat{y}_i\}},$$

where the indicator variable $\mathbb{1}_{\{y_i \neq \hat{y}_i\}}$ takes the value of 1 when the classifier makes the right prediction and zero otherwise, $\hat{y}_i$ is the predicted label for y_i and n_κ is the number of observations in the κ^{th} fold.

Step 3. For each value of k in $k \in \{1, \ldots, k_{max}\}$, compute the average error over all folds, given by

$$CV(k) = \frac{1}{R} \sum_{\kappa=1}^{R} \mathcal{E}rr_\kappa(k).$$

Step 4. Choose the value of k that minimises the average error, i.e.,

$$k^* = \underset{k \in \{1, \ldots, k_{max}\}}{\arg\min} \; CV(k). \tag{3.3}$$

This R-fold cross-validation procedure estimates the misclassification error rate as a function of the number of neighbours. Usually, a large value of R is desirable because there is a better performance estimate and the training size is closer to the full data size but reasonable values can be achieved for $R = 10$ (Kohavi et al., 1995).

The one standard deviation rule for cross-validation, also known as one standard error rule is an alternative rule for choosing the value of the tuning parameter. It is an empirical rule applied in the literature without a theoretical justification, for example in Breiman et al. (1984), Friedman et al. (2001) and Tibshirani et al. (2015). The one standard error rule for

choosing number of k nearest neighbours, as opposed to the rule in equation (3.3), can be described as follows.

Suppose that the cross validation curve initially decreases rapidly then plateaus, the minimum occurs somewhere in the flat valley region, where the function $CV(k)$ is constant except for up-down changes within the one standard error range. In the one standard error rule, we select the value of k whose error is within one standard error of the minimal misclassification error. If k^* denotes the value that minimises the cross-validation curve $CV(k)$, we then move k in the direction until it is satisfied, subject to the restriction

$$CV(k) \leq CV(k^*) + SE(k^*),$$

where the standard error (SE) of the CV is given by:

$$SE(k^*) = \frac{\widehat{SD}(k^*)}{\sqrt{R}},$$

and the sample standard deviation of $CV_1(k^*), \ldots, CV_R(k^*)$, is obtained by

$$\widehat{SD}(k^*) = \sqrt{\mathbb{V}ar(CV_i(k^*))} \quad \text{for} \quad i = 1, \ldots, R.$$

To choose the value of k for the simulated data and use it in the k-RNN classifier, the first step is to implement a $R = 10$ fold Cross-Validation. For the running example, using equal sized groups, we varied the number of neighbours $k \in [1, 40]$ and averaged the misclassification rate over ten folds. The results of calculating the misclassification rate for a 10-Fold Cross Validation can be seen in Figure 3.5.

Note from Figure 3.5 that several local minima of the cross-validation function can be seen. For the simulated data a minimum misclassification rate can achieved using eleven nearest neighbours, therefore the value that minimises the CV error curve is $k^* = 11$ with a misclassification error rate of $CV(11) = 0.1525$. Applying the one standard error rule selects $k = 18$ neighbours; the misclassification error rate does not decrease, but instead increases the number of neighbours. Considering a model with the same misclassification error rate but less neighbours involves less computational time.

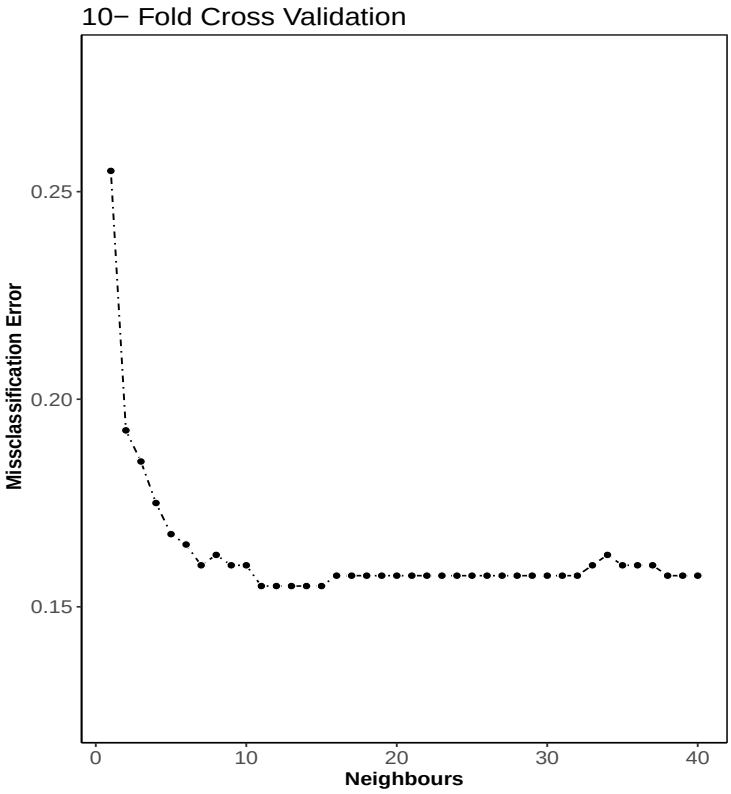


Figure 3.5: Missclassification error rate plotted against different neighbours for the simulated data.

3.3.2 Functional depth

The second consideration we discuss is the depth function $D(\cdot)$. In particular, let us consider the multivariate depth followed by the functional depth.

To start introducing the multivariate depth, we started by considering the one dimensional depth. Suppose we observe one dimensional observations $x_1, \ldots, x_n \in \mathbb{R}$, usually ranked in an ascending or descending order. For a univariate dataset, the median is defined as the order statistic of rank $(n+1)/2$ when n is odd, and as the average of the order statistics of ranks $n/2$ and $(n+2)/2$ when n is even. When $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^p$, the notion of median can be generalised; instead of being a single value in the univariate case, the depth is a point with maximal depth.

In the one dimensional case, the depth is given by

$$\min\left\{\#\left\{x_i \leq x\right\}, \#\left\{x_i \geq x\right\}\right\},$$

where # counts the number of observations according to a corresponding condition, i.e.,

those above and below the point x. Under this consideration, univariate points may be ranked from the outside inward by assigning depth 1 to the furthest data point, the second smallest and second largest data points depth 2, etc., and the median is the point (or points) with maximal depth.

In the multivariate case, we can find different depth functions, all have the same purpose, i.e., to measure how deep (or central) a given point $\mathbf{x} \in \mathbb{R}^p$ is with respect to its underlying distribution, F, or with respect to a data cloud, admitting certain properties (Zuo and Serfling, 2000). We focus on two multivariate depths, the half-space depth and the simplicial depth.

For visualisation purposes, Tukey (1975) introduced the notion of halfspace depth (HD) and depth contours. Let F be a probability distribution in $\mathbb{R}^p$ where $p \geq 1$, and assume that F is an absolutely continuous common distribution of the sample $\mathbf{x}_1, \ldots, \mathbf{x}_n$. For $\mathbf{x} \in \mathbb{R}^p$ the half-space depth (or half-space Tukey depth) by Tukey (1975) and developed by Donoho and Gasko (1992), with respect to the underlying distribution F, is defined as

$$TD(\mathbf{x}) = \inf_{H} \{F(H);\ H \text{ is a closed half-space in } \mathbb{R}^p \text{ and } \mathbf{x} \in H\}.$$

The minimal probability attached to any closed halfspace with $\mathbf{x}$ on the boundary. In particular, the sample halfspace depth of $\mathbf{x}$ is the minimum fraction of data points in any closed halfspace containing $\mathbf{x}$.

In contrast to associated depth contours to the depth function, there exists the simplicial depth by Liu et al. (1990). The simplicial depth at the point $\mathbf{x} \in \mathbb{R}^p$ is defined by

$$SD(\mathbf{x}) = \mathbb{P}_F\left(\mathbf{x} \in S[\mathbf{x}_1, \ldots, \mathbf{x}_{p+1}]\right),$$

where $S[\mathbf{x}_1, \ldots, \mathbf{x}_{p+1}]$ denotes the closed simplex formed by $(p+1)$ observations from F. In other words, the simplicial depth represents the number of subsets that contain $\mathbf{x}$ in their convex hull.

A case of interest in the Tukey and the simplicial depth appears when $p = 1$. In the univariate case, the Tukey depth takes the form

$$TD(x) = \min\{F(x), 1 - F(x)\}.$$

While, on the real line the simplicial depth at x counts how many times the point x is in the closed simplex with vertices in the sample; when $p = 1$,

$$SD(x) = 2F(x)\left(1 - F(x)\right),$$

which can be defined through the one dimensional depth

$$SD(x) = 1 - \left|\frac{1}{2} - F(x)\right|.$$

The corresponding sample versions of the Tukey depth and the simplicial depth are defined by replacing F by the empirical distribution F_n.

In the functional case, similar to the multivariate case, we look for an estimator of the functional median, which allows us to rank the curves based on their depth. This leads to the notion of functional depth, see López-Pintado and Romo (2009) and Cuevas et al. (2006) for more details. For functional data, we want to consider a function that provides an order of the observations outwards from the centre. Thus, we focus in the oldest depth of functional data: the Fraiman-Muniz (FM) depth proposed by Fraiman and Muniz (2001).

The Fraiman-Muniz depth is a function that not only provides an order of the observations outwards from the centre but also considers a mapping from the functional space to the real positive line $D(\cdot) : \mathcal{F} \to \mathbb{R}^+$. Let $\{x_i(\mathbf{t}),\ i = 1,\ldots,N\}$ be a collection of observed functional variables and let $F_{N,t}$ be the empirical distribution of the collection of functional random variables. Then, for every $t \in [a,b]$ the quantity

$$Z_i(t) = D\left(x_i(t)\right), \quad \text{for} \quad i = 1,\ldots,N, \tag{3.4}$$

provides a univariate depth of $x_i(t)$ at time $t \in \mathcal{T}$, with respect to the collection of random variables. In this way, at every single point t, we rank the curves according to their depths $Z_i(t)$. The Fraiman-Muniz depth of the function $x_i(\mathbf{t})$ is defined as the integral:

$$FM_i\left(x_i(\mathbf{t})\right) = \int_a^b Z_i(t)dt, \quad \text{for} \quad 1 \leq i \leq N. \tag{3.5}$$

The choice of the univariate depth function, $D(\cdot)$, modifies the behaviour of the FM depth and, for instance, the deepest curve may have a different meaning, depending on this selection. To exemplify this point, consider the two multivariate depths on the real line, the Tukey depth and the simplicial depth. Observe that we can write the version of equation (3.5) appropriate for a sample in the following form:

$$\begin{aligned}\text{Simplicial - } FM_i\left(x_i(\mathbf{t})\right) &= \int_a^b 1 - \left|\frac{1}{2} - F_{N,t}(x_i(t))\right| dt \\ &= \int_a^b \left[\frac{1}{2} + \min\left\{F_{N,t}(x_i(t)), 1 - F_{N,t}(x_i(t))\right\}\right] dt \end{aligned} \tag{3.6}$$

and

$$\text{Tukey - } FM_i\left(x_i(\mathbf{t})\right) = \int_a^b \min\left\{F_{N,t}(x_i(t)), 1 - F_{N,t}(x_i(t))\right\} dt, \tag{3.7}$$

where

$$F_{N,t}(x_i(\mathbf{t})) = \frac{1}{N}\sum_{j=1}^{N} \mathbb{1}_{\left\{x_j(\mathbf{t}) \leq x_i(\mathbf{t})\right\}},$$

represents the empirical distribution function that counts the number of times $x_j(\mathbf{t})$ is below $x_i(\mathbf{t})$. To visualise the mapping of both functions, we set $M = 100$, thus the functions are equally spaced with a gap between them to 0.01.

Figure 3.6 shows the implementation of the simplicial-FM and the Tukey-FM depths. We can observe that the simplicial-FM depth maps Simplicial - $FM_i\left(x_i(\mathbf{t})\right) \to [0.5, 1.0]$, while the Tukey-FM depth maps Tukey - $FM_i\left(x_i(\mathbf{t})\right) \to [0, 1]$. Additionally, the simplicial- FM depth takes values with a mean of 0.75, while the Tukey-FM depth takes values with a mean of 0.5. From the definition, it is easy to see that the variance for the Tukey depth is four times larger than that of the univariate-FM depth. This relationship is observed by calculating the ratio

$$\mathbb{V}ar(\text{Tukey - } FM_i\left(x_i(\mathbf{t})\right))/\mathbb{V}ar(\text{Simplicial - } FM_i\left(x_i(\mathbf{t})\right)).$$

Also, it is easy to transform from one depth measure to another using the following relationship:

$$\text{Tukey - } FM_i\left(x_i(\mathbf{t})\right) = 2\left(\text{Simplicial - } FM_i\left(x_i(\mathbf{t})\right) - \frac{1}{2}\right). \tag{3.8}$$

For a discretised functional dataset, $\left\{x_i(t_j),\ i = 1, \ldots, N, \quad j = 1, \ldots, M\right\}$, for each observation we can compute the Simplicial-FM in terms of the sum

$$FM_{n,i}\left(x_i(t_j)\right) = \sum_{j=2}^{M}(t_j - t_{j-1})\left[1 - \left|\frac{1}{2} - \frac{1}{N}\sum_{k=1}^{N}\mathbb{1}_{\left\{x_k(t_j) \leq x_i(t_j)\right\}}\right|\right], \tag{3.9}$$

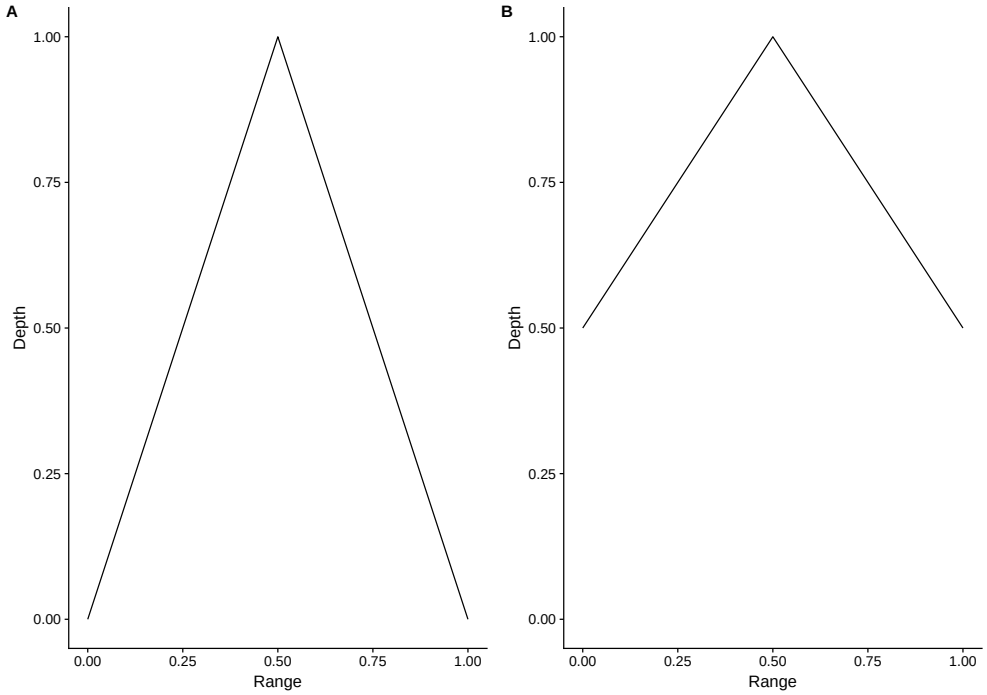


Figure 3.6: Graphical representation of (A) the Tukey-depth mapping values of $F_{n,t}$ to $[0,1]$, and (B) the simplicial depth mapping $F_{n,t}$ to $[0.5,1.0]$.

for M equally spaced points such that the value of $(t_j - t_{j-1})$ is $1/M$.

Different depth measures lead to different estimators of the median function. We can observe that different depths map to different real and positive intervals. Therefore, the estimation of the median curve differs with respect to the functional depth function used. For our purposes, we use the version given by Tukey-FM.

3.3.3 Estimated probabilities

For classification, a useful approach to predict the population from which a signed depth originates is to consider the probabilities that an observed signed depth $sdp(x(\mathbf{t}))$ belongs to each of the two populations Π_0 or Π_1. In the binary case, this provides a way to estimate the conditional probability for each group. Here we describe how the k-RNN procedure can be interpreted in terms of such conditional probabilities.

Let $(sdp(X(t)), Y)$ be a random pair, for $t \in \mathcal{T}$, with two possible outcomes for Y. To classify $sdp(X(t))$, we estimate the conditional probability,

$$\mathbb{P}\Big(Y = y \mid sdp(X(t)) = sdp(x(\mathbf{t}))\Big). \tag{3.10}$$

We can construct an estimate of equation (3.10) using the k-RNN algorithm introduced in Section 3.2. First, observe that $sdp(x_1(\mathbf{t})), \ldots, sdp(x_n(\mathbf{t}))$ represent the signed depths for a collection of curves and y_i denotes the vector of the membership indicator variable where $y_i = 1$ if $sdp(x_i(\mathbf{t})) \in \Pi_1$ and $y_i = 0$ if $sdp(x_i(\mathbf{t})) \in \Pi_0$. Given a value of $k \in \mathbb{Z}^+$, the k-RNN classifier starts defining half-closed intervals between the observations of the form $[sdp(x_{i-1+k}(\mathbf{t})), sdp(x_{i+1+k}(\mathbf{t})))$, given by

$$\Big\{ sdp(x_j(\mathbf{t})) \in \mathbb{R};\ \ sdp(x_{i-1+k}(\mathbf{t})) \leq sdp(x_j(\mathbf{t})) < sdp(x_{i+1+k}(\mathbf{t})) \Big\}. \tag{3.11}$$

Thus, for a known population, Π_0, an estimator of equation (3.10) is given by

$$\hat{\mathbb{P}}\Big(y_i = 0 \mid sdp(x(\mathbf{t}))\Big) = \frac{1}{2k} \sum_{i \in [sdp(x_i(\mathbf{t})), sdp(x_{i+1}(\mathbf{t})))} y_i, \tag{3.12}$$

for i's in the total sample size. An advantage of this approach is that we only need to consider the conditional probabilities with respect to one group, since the conditional probability for Π_1 adds to one. Another advantage of considering this approach is that the estimated probabilities can be visualised as a step function form.

Figure 3.7 represents the step function for the depths against the probability that the depth belongs to population Π_0. The predicted conditional probabilities reveal an increasing shape with respect to the reference group. This is because observations from population Π_1 have a negative signed depth while observations with positive signed depth are more likely to belong to population Π_0.

By considering a graphical representation of the step function of the conditional probability of the signed depth belonging to the reference group, we can classify a new observation $x_0(\mathbf{t})$ by calculating its signed depth $sdp(x_0(\mathbf{t}))$. And for a fixed value of k, the $P(y = 0 \mid sdp(x_0(\mathbf{t})))$ can be read off from the already generated plot and assigned to population Π_0 if $P(y = 0 \mid sdp(x_0(\mathbf{t}))) > 0.5$.

Observe that equation (3.10) summarises the dependence of the class label on the signed depth. A more general type of function that summarises such dependence and produces a smooth function is the running-mean smoother or moving average. The **moving average** approach is explained in detail in Section 3.4.1.

There are two main features we want to consider about this approach. The first feature is when an observation falls into one of the boundaries. When we are interested in classifying

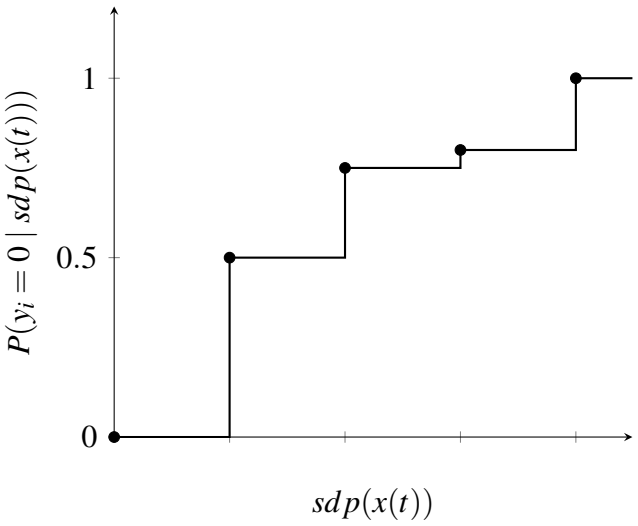


Figure 3.7: Graphical representation of the step function of the conditional probability of the signed depth belonging to the reference group Π_0.

an observation that falls into one of the boundaries, the k-RNN classifier uses as many observations available. However, this influences the estimation of the conditional probabilities. If the number of observations is not the same above and below, we say we have an unbalanced number of observations, and the estimated probabilities only consider the observations available. By considering this, we induce a bias for the conditional probabilities in the reference group.

The second feature we want to illustrate is the fact that the estimated probabilities vary with respect to the number of neighbours we consider. To illustrate how the estimated probabilities vary, we consider different values for the number of neighbours, and we estimate the conditional probabilities. Figure 3.8 shows the different behaviours of the conditional probabilities when $k = 15, 25$ and 35, respectively.

Observe that by varying the number of neighbours k, we can achieve a wide range of flexibility in the estimated conditional probability, with small k corresponding to a more flexible fit, and large k being less flexible. From Figure 3.8, we notice that a larger number of nearest neighbours produces flattened curves, and the semi-closed intervals become narrow around the signed depth. For small values of nearest neighbours, observations near the boundaries show conditional probabilities below 0.50, while increasing the numbers of neighbours tends to increase the conditional probability of belonging to Π_0 more than 0.50.

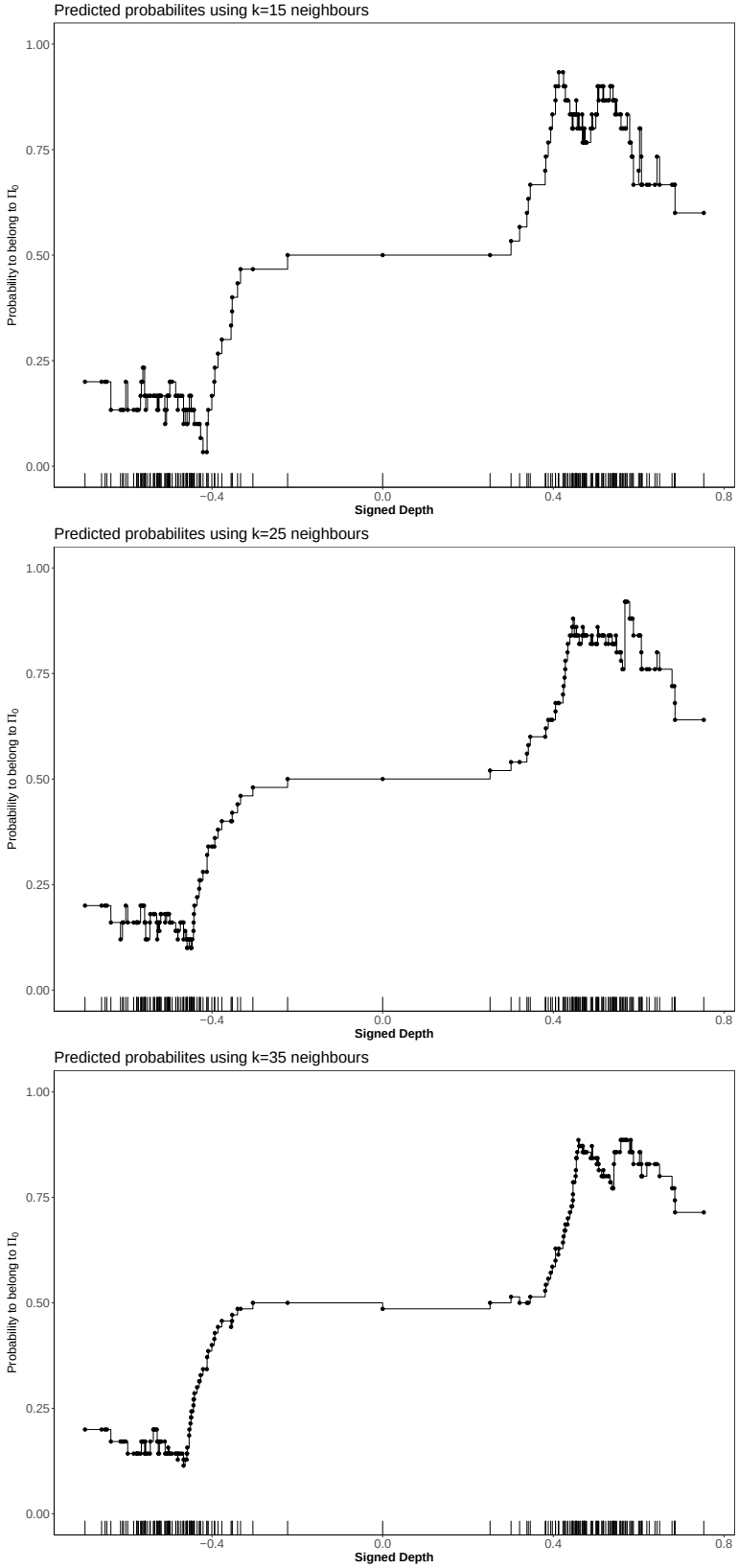


Figure 3.8: The predicted probabilities for different values of k, using (top:) $k = 15$, (middle:) $k = 20$, and (bottom:) $k = 35$.

3.4 The k-RNN as a regression approach - A model for the smoothing

To study the smoothing problem in detail and to look in more detail how the moving average works, we start by introducing the k-RNN algorithm as a regression approach. Assume that we can relate the response Y to a predictor X in the following way

$$Y = f(X) + \varepsilon, \tag{3.13}$$

where ε denotes the uncorrelated random error, with expectation $\mathbb{E}(\varepsilon) = 0$ and $\mathbb{V}ar(\varepsilon) = \sigma^2$. To study such dependence, we consider a linear regression model of the form

$$\mathbb{E}[Y \mid X] = f(X), \tag{3.14}$$

where the function $f(X)$ is an arbitrary unspecified function. A regression procedure considers describing the dependence of the mean of a response variable Y as a function f (generally unknown to some degree) of a variable (or covariate) X. Equation (3.13) suggests that, given some data, we can estimate the dependence of the mean of Y, conditional on the covariate as follows

$$\mathbb{E}[Y \mid X = x] = f(x), \tag{3.15}$$

and under this formal framework the **smoother** is defined as an estimate of the function $f(X)$ that is less variable than Y. Smoothers do not assume any rigid form of the dependence, thus sometimes they can be referred to as *nonparametric smoothers*, e.g., see Hastie and Tibshirani (1987).

Formally, a nonparametric smoother is a tool for summarising the relationship between the response Y and one or more covariates $X_1, \ldots X_p$. It produces an estimate of the response that is less variable than Y; see Hastie (2017). If the dependence is linear, we can estimate a regression line which provides a useful summary about the relationship. For more about the topic, see Conover and Conover (1980), Silverman (1986) and Hastie (2017).

Smoothers usually depend on two main parameters: a smoother matrix denoted by $\mathbf{S}$ and a span which we will describe later and can have two main uses. The first use is to estimate the dependence of the mean of Y on the covariates $X_1, \ldots X_p$ and the second use is to make a

prediction of Y for new values of X.

3.4.1 The k-RNN as a Moving Average.

When Y is a categorical variable, to smooth Y we can average the values of Y in each category. By doing this, we capture the relationship between Y on X which is smoother than the Y values themselves. This simple averaging process is the conceptual basis for smoothing in the more general setting. Most smoothers, attempt to mimic category averaging through local averaging, i.e., we average the Y values of observations having predictor values close to a target value. Usually, this averaging is done in neighbourhoods around the target value.

The k-RNN assumes that our target value $sdp(x_0(\mathbf{t}))$ is equal to $sdp(x_i(\mathbf{t}))$. If we have replicates of $sdp(x_i(\mathbf{t}))$ we can just simply average the y values as our estimate $f\big(sdp(x_i(\mathbf{t}))\big)$. However, we usually do not have replicates, so instead we can average y values corresponding to values close to $sdp(x_i(\mathbf{t}))$. We can define a neighbourhood $N_k(sdp(x_i(\mathbf{t})))$, which represents the indices of these observations $sdp(x_i(\mathbf{t}))$ inside this neighbourhood and close to $sdp(x_i(\mathbf{t}))$. A formal definition of a this neighbourhood $N_k(sdp(x_i(\mathbf{t})))$ for the k-RNN classifier is

$$N_k(sdp(x_i(\mathbf{t}))) = \{\max\{i-k,1\},\ i-1,\ i+1,\ldots,\min\{i+k,n\}\}. \tag{3.16}$$

For example, if an observation is in the fifth place $sdp(x_5(\mathbf{t}))$ out of a total of $n = 10$ observations using $k = 2$, the neighbouhood $N_2(sdp(x_5(\mathbf{t})))$ is going to be formed by the observations of the signed depth $sdp(x_i(\mathbf{t}))$ with indices

$$\begin{aligned} N_2(sdp(x_5(\mathbf{t}))) &= \{\max\{5-2,1\},\ldots,\min\{5+2,10\}\} \\ &= \{3,4,6,7\}. \end{aligned}$$

The *running mean* smoother or moving average for the k-RNN classifier, with a span of $2k$ has the following form

$$\hat{f}_k(sdp(x_i(\mathbf{t}))) = \frac{1}{2k} \sum_{j \in N_k(sdp(x_i(\mathbf{t})))} y_j. \tag{3.17}$$

As we saw before, a drawback of this method is its behaviour near the endpoints. Note that if it is not possible to take k points to the left or to the right of $sdp(x_i(\mathbf{t}))$, we take as many

as possible resulting in a non symmetric neighbourhood. Usually in the k-RNN the span is an even number which includes k y_i's below $sdp(x_i(\mathbf{t}))$, k y_i's above $sdp(x_i(\mathbf{t}))$, resulting in an even span of the form $2k$.

This simple smoother is popular in the time series literature and for evenly-spaced time-series data see Zivot and Wang (2003). However, a disadvantage of working with the moving average is that in practice does not work well and it tends to be wiggly and flattens out trends near the endpoints and hence can be severely biased (Buja et al., 1989; Trexler and Travis, 1993).

One of the most popular approaches when we consider the study of a classifier in functional data is to consider a regression model. Some of the approaches in the literature estimate posterior probabilities in a nonparametric manner. In fact, equation (3.17) provides an estimator of the conditional probabilities which can be formulated in terms of conditional expectations

$$\mathbb{P}\left(y_i = 0 \mid sdp(x(\mathbf{t}))\right) = \mathbb{E}\left(y_i = 0 \mid sdp(x(\mathbf{t}))\right). \tag{3.18}$$

Note that equation (3.18) corresponds to the same conditional probability as in equation (3.10) using the fact that conditional probabilities can be expressed in terms of conditional expectations.

3.4.2 Smoother Matrix and Shrinking Smoothers

Smoothers have the advantage that they can be expressed as a mapping acting on the response vector $\mathbf{y} = \{y_1, \ldots, y_n\}$. Note that the linearity is in $\mathbf{y}$ and we can write the smoother as

$$\mathbf{y}_k = \mathbf{S}_k \mathbf{y}, \tag{3.19}$$

where $\mathbf{S}_k$ is called the smoother matrix $\mathbf{S} : \mathbb{R}^n \to \mathbb{R}^n$. Therefore we can consider it as an estimator of a regression function and a conditional probability. For simplicity, we refer to the case where the conditional probability we are interested in estimating is the one provided in equation (3.10).

The smoother matrix depends on the number of neighbours k and the observed signed depths $\{sdp(x_1(\mathbf{t})), \ldots, sdp(x_n(\mathbf{t}))\}$. In this case, the matrix $\mathbf{S}_k$ which depends on k is a non-symmetric band matrix whose matrix elements are zero outside a diagonally bordered

band and the range is determined by the number of neighbours k we consider to estimate such a matrix. The smoother matrix for the k-RNN as a running mean smoother is a square matrix of dimension $n \times n$ given by

$$\mathbf{S}_k = \left\{ s_{\{i,j\}} \right\} = \begin{cases} 0 & \text{if} \quad j < i-k \quad \text{or} \quad j > i+k; \quad k \geq 0, \\ \frac{1}{2k} & \text{otherwise}, \end{cases}$$

for $i = 1, \ldots, n$, and $j = 1, \ldots, n$. The value of k forms the lower and the upper limit and the values inside the diagonal band are given by the probabilities $1/2k$. A running-mean smoother not only estimates the number of observations of one particular population but also estimates the probability that the signed depth of the new observation $x_0(\mathbf{t})$ belongs to the reference population Π_0. An example of the smoother matrix for the k-RNN using six ranked signed depth and $k = 2$ is given by

$$\mathbf{S}_2 = \begin{bmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\ 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \end{bmatrix}.$$

Notice that the truncated neighbourhood is near the boundaries. A smoother can also be described qualitatively by the eigenvalue and singular value decompositions of the corresponding smoother matrix. We consider the eigenvalue decomposition of the k-RNN classifier as a running mean smoother. Let $\{\mathbf{u}_1, \ldots, \mathbf{u}_n\}$ be an orthonormal basis of the eigenvectors of $\mathbf{S}$ with a sequence of non decreasing eigenvalues $\theta_1 \geq \theta_2 \ldots \geq \theta_n$. The smoother matrix $\mathbf{S}$ admits a decomposition of the following form

$$\begin{aligned} \mathbf{S} &= \mathbf{U}\mathbf{D}_\theta\mathbf{U}^{'} \\ &= \sum_{j=1}^{n} \theta_j \mathbf{u}_j \mathbf{u}_j^{'}, \end{aligned} \tag{3.20}$$

where $\mathbf{D}_\theta$ is a diagonal matrix with the diagonal formed by the eigenvalues of $\mathbf{S}$. The columns of $\mathbf{U}$ and $\mathbf{U}^{'}$ are called the left and right singular eigenvector of $\mathbf{S}$, respectively. The k-RNN as a running mean smoother is a smoother where the smooth matrix $\mathbf{S}$ is not symmetric.

However, the singular value decomposition is always a real-valued function. We investigate the behaviour of the running mean smoother by applying a singular value decomposition to the smoother matrix.

To show this behaviour, we consider different values of k and different samples sizes n and calculate the smooth matrix $\mathbf{S}$. Then a singular value decomposition for the smooth matrix is obtained. Results of this behaviour can be seen in Figure 3.9 and Figure 3.10. The results show that the first two eigenvalues capture most of the variability which decreases as the number of neighbours decreases. In addition, we plot the first ordered eigenvalues and we investigate the behaviour of the singular eigenvectors.

We considered exploring the effect of the sample size on the first two singular vectors. First, we show the results of applying the singular value decomposition and plotting the right and left eigenvector for $n = 20$ with $k = 2$, $n = 200$ with $k = 4$ and $n = 200$ with $k = 20$ which can be seen in Figure 3.9.

These demonstrate that the first two singular vectors tend to be a quadratic function. Next, we investigate if the smoother S belongs to the class of shrinking smoothers. For a vector of observations $\mathbf{y} = \{y_1, \ldots, y_n\}$, we say a smoother S is a shrinking smoother (Hastie, 1996; Hastie and Tibshirani, 1987) if

$$\|S\mathbf{y}\| \leq \|\mathbf{y}\|,$$

and strictly shrinking if

$$\|S\mathbf{y}\| < \|\mathbf{y}\|,$$

for all the values of $\mathbf{y}$, where $\|\cdot\|$ denotes the Euclidean norm. Note that this is related in the case of its singular values. If all of its singular values of the smoother matrix $\mathbf{S}$ are ≤ 1 the smoother is a shrinking smoother and strictly shrinking if < 1. For the k-RNN as a moving average, the largest singular value is slightly greater than one and therefore is not a member of the class of shrinking smoothers.

Observe that equation (3.20) can be expressed as $\mathbf{S}\mathbf{u}_j = \theta_j \mathbf{u}_j$ for $j = 1, \ldots, n$, and if S is a shrinking smoother, it shrinks the component of $\mathbf{y}$ along $\mathbf{u}_j$ by an amount θ_j as in equation (3.20).

An advantage of considering a shrinking smoother is that prediction accuracy can sometimes be improved by shrinking, e.g., Copas (1983). Under some special considerations, a smoother S can be a shrinking smoother. In fact Buja et al. (1989) provides some conditions

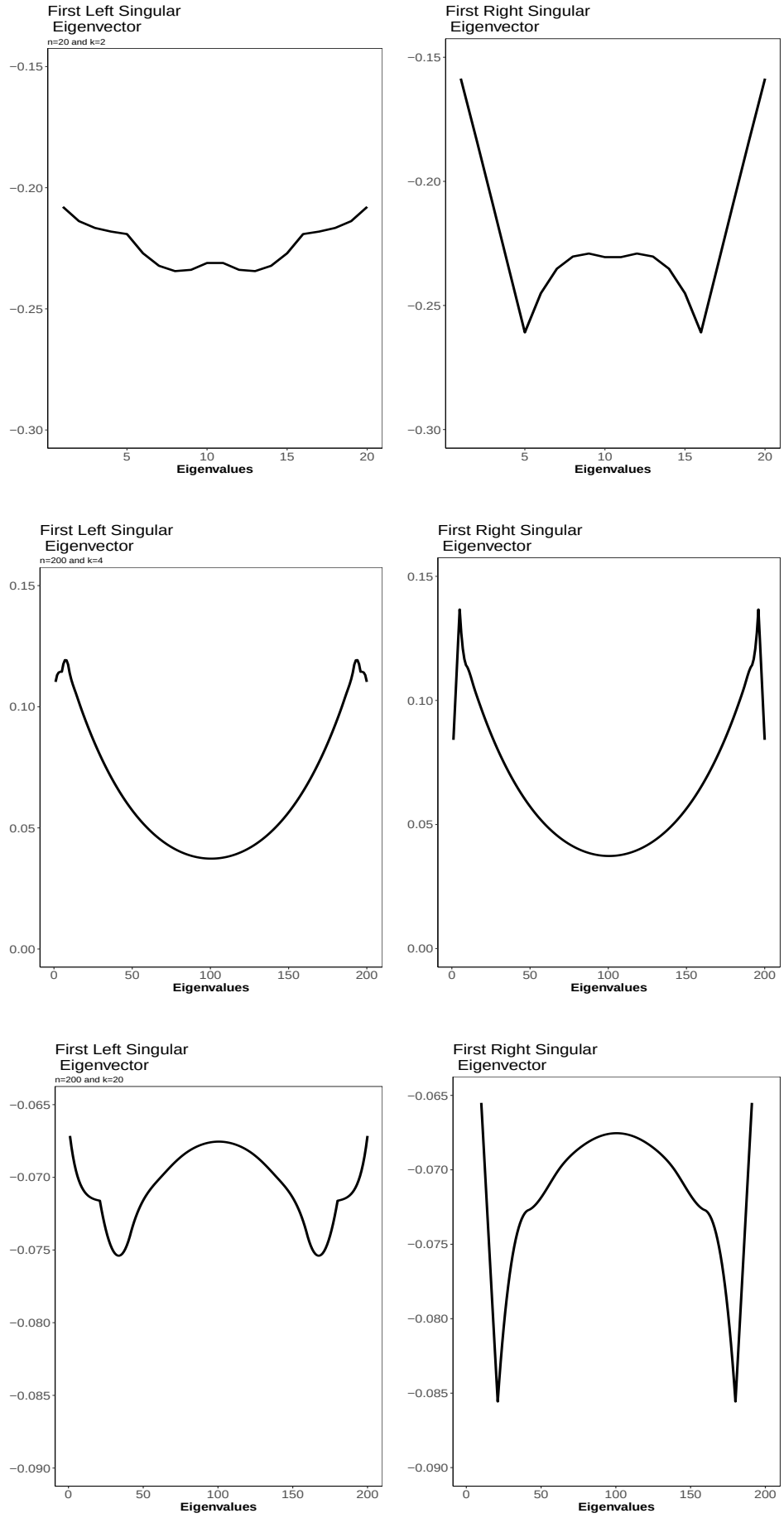


Figure 3.9: The first left and right singular eigenvectors for (top:) $n = 20$ and $k = 2$; (middle:) $n = 200$ and $k = 4$ and (bottom:) $n = 200$ and $k = 20$.

to guarantee that S is shrinking. If the smoother matrix $\mathbf{S}$ is double stochastic, the smoother S is a shrinking and symmetric smoother with non-negative elements that are shrinking if each row adds to 1. However, we can find counter examples, such as the cubic spline smoother, with a smoother matrix with negative elements, and real positive eigenvalues less than one. Demmler and Reinsch (1975) provide the mathematical proof.

A major use of a smoother is to prediction. Thus, given data on a response variable and associated predictor variables X_i, our aim is to find a function of the X_i values which is a good predictor of Y. Suppose that

$$y_i = \hat{f}_k(sdp(x_i(\mathbf{t}))) + \varepsilon_i$$

is the regression model where $f_k(\cdot)$ is the true function and the errors are uncorrelated with $\mathbb{E}(\varepsilon_i) = 0$ and common variance σ^2. When $y_i \sim Bin(1, p)$ the variance - covariance matrix of $S\mathbf{y}$ is given by

$$\mathbb{V}ar(S\mathbf{y}) = \mathbf{S}' \operatorname{diag}\{p_i(1-p_i)\}\mathbf{S}. \tag{3.21}$$

Point-wise confidence bands can be useful to get an idea of how variable the function $\hat{f}_k(sdp(x_i(\mathbf{t})))$ is. However, they are not be very helpful when we want to see how variable the function $\hat{f}_k(sdp(x_i(\mathbf{t})))$ is as a whole. Equation (3.21) provides an estimator of the variance - covariance matrix that can be used to form point-wise standard-error bands. Under normality assumptions, this can be used to form point-wise error bands for the probabilities that the signed depth belongs to a particular group Π_0 or Π_1, therefore we can get an idea of how variable our estimator is. In fact, Tibshirani and Hastie (1987) discuss this approach to form point-wise confidence bands for the estimated smoother. From a Bayesian perspective, Wahba (1983) provides evidence that posterior bands derived from a Bayesian model for smoothing splines can have good sampling properties. An alternative to forming the point-wise standard errors for the estimated smoother is to use a parametric bootstrap approach discussed in Section 3.4.4.

3.4.3 The bias-variance trade-off

In smoothing, there is a trade-off between the bias and variance of the estimate, and this trade-off is governed by the smoothing parameter or span. In the case of the k-RNN as

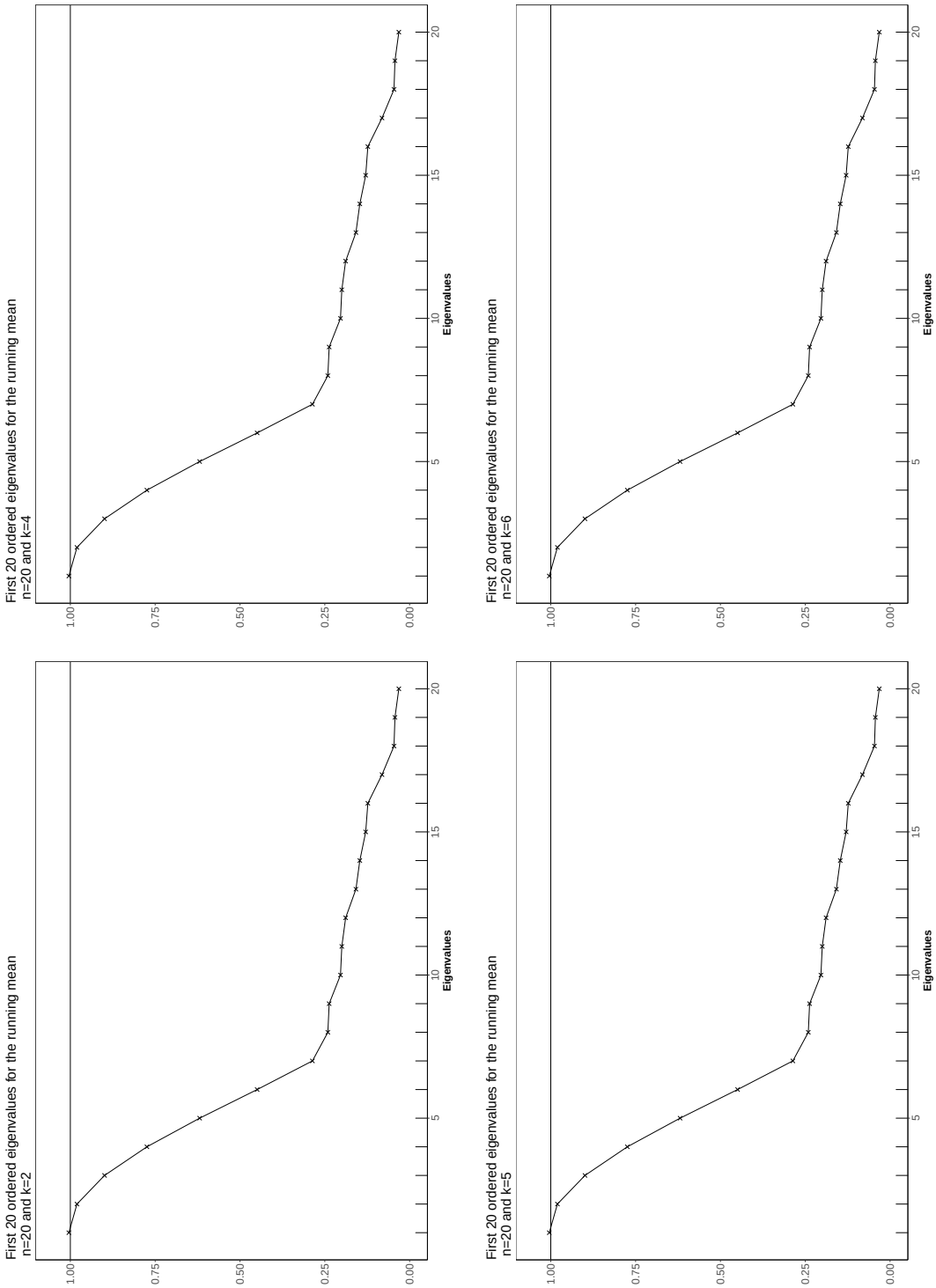


Figure 3.10: The first 20 ordered eigenvalues for (top-left:) $k = 3$; (top-right) $k = 4$ (bottom-left) $k = 5$ and (bottom-right) $k = 6$.

a regression approach, we detailed the trade-off between the bias and the variance. First, assume that the observation $sdp(x_i(\mathbf{t}))$ is near the middle of the data, therefore, we are containing the total $2k$ observations. The fitted running mean smoother for the k-RNN can be written as

$$\begin{aligned}\hat{f}_k(sdp(x_i(\mathbf{t}))) &= \frac{1}{2k}\sum_{j\in[sdp(x_i(\mathbf{t})),sdp(x_{i+1}(\mathbf{t})))} y_j \\ &= \frac{1}{2k}\sum_{j\in N_k(sdp(x_i(\mathbf{t})))} y_j \end{aligned} \tag{3.22}$$

The expected value of $\mathbb{E}\left[\hat{f}_k(sdp(x_i(\mathbf{t})))\right]$ can be written as

$$\mathbb{E}\left[\hat{f}_k(sdp(x_i(\mathbf{t})))\right] = \frac{1}{2k}\sum_{j\in N_k(sdp(x_i(\mathbf{t})))} f(sdp(x_j(\mathbf{t}))), \tag{3.23}$$

while the variance can be expressed as

$$\mathbb{V}ar\left[\hat{f}_k(sdp(x_i(\mathbf{t})))\right] = \frac{\sigma^2}{2k}. \tag{3.24}$$

We can get an approximate expression for the bias in terms of $f(\cdot)$ using a Taylor series expansion for the function $f(x_j) = f(sdp(x_j(\mathbf{t})))$ and assuming that the quantity $\Delta = sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t}))$ is small. The Taylor series expansion, around $sdp(x_j(\mathbf{t}))$ is given by

$$\begin{aligned} f(sdp(x_j(\mathbf{t}))) = f(sdp(x_i(\mathbf{t}))) &+ (sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t})))f'(sdp(x_i(\mathbf{t}))) \\ &+ \frac{(sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t})))}{2} f''(sdp(x_i(\mathbf{t}))) \\ &+ O(sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t}))) \end{aligned} \tag{3.25}$$

where $O(sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t})))$ is the remainder. To explicitly derive the quadratic term, let $y_j = j^2/2$, so that $f''(y_j) = 1$ and all high order differentiations are zero. In this case, the first two terms of the Taylor series in equation (3.25) are the same. At the value of $j = 0$, the running mean smoother for $f(sdp(x_j(\mathbf{t}))) = j^2/2$ is given by

$$\frac{1}{2k}\sum_{j=1}^{k} j^2 = \frac{(k+1)(2k+1)}{12}.$$

Thus, the first two terms of the Taylor Series expansion can be written as

$$f(sdp(x_j(\mathbf{t})) = f(sdp(x_i(\mathbf{t})) + \frac{(k+1)(2k+1)}{12} f^{''}(sdp(x_j(\mathbf{t})). \tag{3.26}$$

After some algebra, we can see that the bias in $f_k(sdp(x_i(\mathbf{t})))$ is given by

$$\begin{aligned} \text{Bias}\left[f_k(sdp(x_i(\mathbf{t})))\right] &= \mathbb{E}\left[\hat{f}_k(sdp(x_i(\mathbf{t}))) - f_k(sdp(x_i(\mathbf{t})))\right] \\ &\approx \frac{(k+1)(2k+1)}{12} f^{''}(sdp(x_i(\mathbf{t})))\Delta^2, \end{aligned} \tag{3.27}$$

where $\Delta^2 = \left(sdp(x_j(\mathbf{t})) - sdp(x_i(\mathbf{t}))\right)^2$. An optimal choice of the span is the one that trades the bias against the variance. One reasonable criterion is the Mean Square Error (MSE). We select the value of the span that minimises the MSE, which relates the number of neighbours in the following way:

$$\begin{aligned} MSE(f_k(sdp(x_i(\mathbf{t})))) &= Bias^2\left[f_k(sdp(x_i(\mathbf{t}))\right] + \mathbb{V}ar\left[f_k(sdp(x_i(\mathbf{t}))\right] \\ &= \frac{((k+1)(2k+1))^2}{144}\left\{f^{''}(sdp(x_i(\mathbf{t})))\right\}^2\Delta^4 + \frac{\sigma^2}{2k}. \end{aligned} \tag{3.28}$$

From equation (3.28) we can see that increasing the value of the span k, the variance tends to decrease but on the other hand increases the bias. However, decreasing the value of k increases the variance but tends to decrease the bias. Therefore there is a trade-off between the bias and the variance.

Using the derived expressions in equation (3.23) and equation (3.24) the k that minimises the mean square error is approximately,

$$k_{opt} \approx \left\{\frac{9\sigma^2}{2\Delta^4\left(f^{''}(sdp(x_i(\mathbf{t})))\right)^2}\right\}^{1/5}.$$

However, this is not useful in practice since we do not know the function $f^{''}(sdp(x_i(\mathbf{t})))$.

3.4.4 A parametric bootstrap approach

Since the work of Efron and Tibshirani (1993), bootstrap techniques have received increasing popularity in Statistics. Despite its general applicability, bootstrap techniques have been mainly applied in univariate and multivariate data. There exists comparably less work on bootstrapping functional data, with some exceptions of Hall and Vial (2006). Bootstrap

techniques have been widely used for determining the critical value for testing hypotheses and constructing confidence intervals (Gervini, 2008). Different approaches were developed to estimate confidence bands following smoothing splines. From a Bayesian approach, Wahba (1990) discusses confidence bands for smoothing splines and Nychka (1988) provides evidence that the posterior confidence band derived from the Bayesian model has good properties. The goal of this subsection is to provide confidence intervals for the conditional probability of observing population Π_0 given the signed depths. Observe that equation (3.21) provides an expression for the variance-covariance matrix of the response. When y_i is assumed to follow a 0/1 random variable we can assume that $y_i \sim Bin(1,p)$ where p is the probability that the depth belongs to the reference group and $Bin(1,p)$ denotes the Bernoulli distribution with probability p of success.

An advantage of considering a parametric bootstrap approach in this setting is that we can resample from a known parametric distribution. Assuming that $y_i \sim Bin(1,p)$ and we have some fixed value of k, for all the $sdp(x_1(\mathbf{t})),\ldots,sdp(x_n(\mathbf{t}))$, the conditional probability of observing the reference group is denoted by $p_1(y_1 = 0 \mid sdp(x_1(\mathbf{t}))),\ldots,p_n(y_n = 0 \mid sdp(x_n(\mathbf{t})))$. For the given value of k, we select all the ranked signed depths $sdp_{(1)}(x_1(\mathbf{t})),\ldots,sdp_{(n)}(x_1(\mathbf{t}))$ in $N_k(sdp(x_i(\mathbf{t})))$ and simulate a Bernoulli distribution y_i with probability $p_i(y_i = 0 \mid sdp(x_i(\mathbf{t})))$ for each $sdp_{(n)}(x_i(\mathbf{t})) \in N_k(sdp(x_i(\mathbf{t})))$. Using this approach, we can simulate as desired to obtain a bootstrap sample of the conditional probability to belong to the reference population. We denote our bootstrap samples for the conditional probability as

$$p_1^{*1}(y_1 = 0 \mid sdp(x_1(\mathbf{t}))), p_1^{*2}(y_1 = 0 \mid sdp(x_1(\mathbf{t}))),\ldots,p_1^{*B}(y_1 = 0 \mid sdp(x_1(\mathbf{t}))),$$

and we repeat this for each $sdp_{(i)}(x_i(\mathbf{t})) \in N_k(sdp(x_i(\mathbf{t})))$. Note that this approach allows us to have a bootstrap sample for each observed depth in the training set and we can compute point-wise bootstrap confidence intervals.

3.4.5 Bootstrap-*t* Confidence Intervals (CI)

We can obtain a bootstrap sample of the conditional probability that $sdp(x_i(\mathbf{t}))$ belongs to the reference group, from which we can estimate some characteristics of the corresponding sampling distribution. There are several approaches to estimating the end-points of a confidence interval (CI) using the bootstrap method. We focus on the *bootstrap-t* interval

which generalises the well-known Student-t method of constructing confidence intervals. The algorithm for constructing the confidence intervals for the conditional probability proceeds as follows:

Step 1. For a fixed value of k, generate the neighbourhood $N_k(sdp(x_i(\mathbf{t})))$ and B bootstrap samples

$$p_i^{*1}, p_i^{*2} \ldots, p_i^{*B},$$

for each $sdp_{(i)}(x_i(\mathbf{t})) \in N_k(sdp(x_i(\mathbf{t})))$ and $i = 1, \ldots, n$.

Step 2. For $1 \leq j \leq B$ and $i = 1, \ldots n$, compute the Z-Score $Z^*(j)$ given by,

$$Z^*(j) = \frac{\hat{p}_i^{*j} - \hat{p}_i}{\hat{\sigma}_j^*},$$

where $\hat{p}_i^{*j}$ is the value of $\hat{p}_i$ for the j^{th} bootstrap sample and $\hat{\sigma}_j^*$ is the estimated standard error of $\hat{p}_i^{*j}$ from the j^{th} bootstrap sample.

Step 3. The α^{th} percentile of the Z-Score $Z^*(j)$ is estimated by the value $\hat{t}^{(\alpha)}$ such that

$$\frac{\mathbb{1}_{\left\{Z^*(j) \leq \hat{t}^{(\alpha)}\right\}}}{B} = \alpha$$

Step 4. Then an α- *bootstrap-t* confidence interval for the conditional probability is given by

$$\left(\hat{p}_i - \hat{t}^{(1-\alpha)}\hat{\sigma}, \hat{p}_i - \hat{t}^{(\alpha)}\hat{\sigma}_i\right),$$

for each $i = 1, \ldots, n$ and $\hat{\sigma}_i$ the estimated standard error of $\hat{p}_i$.

Using the procedure described, we can estimate a point-wise confidence interval for the probability that the depth of the new observation belongs to the reference group. For the simulated data and for all the signed depths in our running example

$$\{sdp(x_1(\mathbf{t})) \ldots, sdp(x_n(\mathbf{t}))\},$$

a 95% confidence interval using a total of $B = 100$ bootstrap samples for each signed depth of the conditional probability that the observed depths are in the reference group was constructed. Results of our implementation can be seen in Figure 3.11.

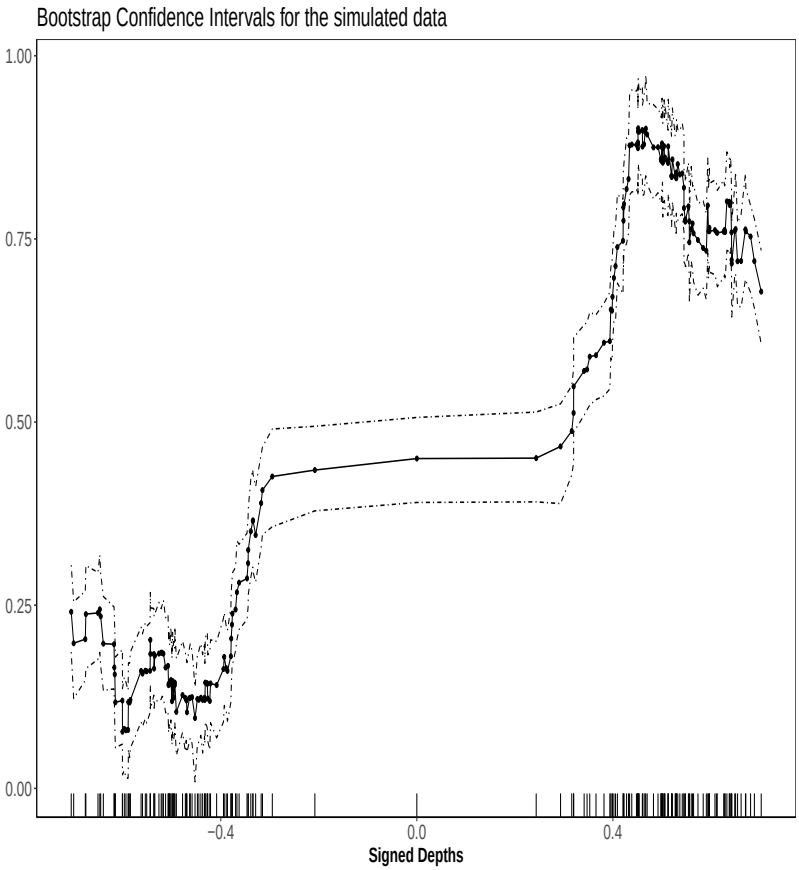


Figure 3.11: A 95% Bootstrap Confidence Intervals for the simulated data.

3.5 The signed distance integral as a classifier

Note, that all the previous work was motivated in terms of the signed depth which requires using the signed distances. However, we can consider the signed distance integral as a measure of dissimilarity between the curves and the reference curve. The goal of this section is to introduce the signed distance integral as an alternative to classification given in terms of the principal component scores. To achieve this goal we start by considering the expansion of the curve for the signed distance integral in equation (3.1) in terms of the principal component scores.

Consider $X(t)$ to be a function on a closed interval $\mathcal{T} = [a,b]$. Using the Karhunen-Lòeve expansion for random functions, we can express a curve in terms of a linear combination of the principal component scores and a complete orthonormal basis functions $\{\psi_1(t), \psi_2(t)\ldots\}$. Let $\{X_i(t),\ t \in \mathcal{T}\}$ for $i = 1,\ldots,n$, be a collection of random curves and consider that each of the curves is centered, i.e., $X_i(t) - \bar{X}(t)$. The expansion for each curve in terms of the principal component scores is given by

$$X_i(t) = \sum_{j=1}^{\infty} \Xi_{ij}\psi_j(t) \quad \text{for} \quad i = 1,\ldots,n, \tag{3.29}$$

where the matrix Ξ_{ij} for $j \geq 1$ consists of the functional principal component scores. The reference curve, $X_{ref}(t)$, admits an expansion of the following form:

$$X_{ref}(t) = \sum_{j=1}^{\infty} \xi_j^{ref}\psi_j(t). \tag{3.30}$$

Substituting equation (3.30) into equation (3.1), we can observe that

$$\begin{aligned}
\int_{\mathcal{T}} \big(X_i(t) - X_{ref}(t)\big)\, dt &= \int_{\mathcal{T}} \Big(\sum_{j=1}^{\infty} \Xi_{ij}\psi_j(t) - \xi_j^{ref}\psi_j(t) \Big) dt \\
&= \int_{\mathcal{T}} \Big(\sum_{j=1}^{\infty} \left(\Xi_{ij} - \xi_j^{ref}\right) \psi_j(t) \Big) dt \\
&= \sum_{j=1}^{\infty} \left(\Xi_{ij} - \xi_j^{ref}\right) \int_{\mathcal{T}} \psi_j(t) dt \\
&= \sum_{j=1}^{\infty} \left(\Xi_{ij} - \xi_j^{ref}\right),
\end{aligned} \tag{3.31}$$

for $i = 1,\ldots,n$. By Fubini's theorem due to Fubini (1907) we can interchange summation with integration here because the second integrand in equation (3.31) is infinite. Note that the integral $\int_{\mathcal{T}} \psi_j(t)dt$ integrates to one using the fact that we are working on a complete orthonormal basis. In practice, we can not compute infinite sums, but we can approximate the sum using a small finite number of principal components. Thus, an approximation to equation (3.31) is given by

$$\int_{\mathcal{T}} \big(X_i(t) - X_{ref}(t)\big)\, dt \approx \sum_{j=1}^{p} \left(\Xi_{ij} - \xi_j^{ref}\right). \tag{3.32}$$

To construct a classifier using equation (3.32), we select a fixed dimension p, and for a new observation $x_0(\mathbf{t})$, we estimate the principal component score $\hat{\xi}_{0p}$ and the corresponding eigenfunction $\hat{\psi}_0(\mathbf{t})$. Then, we assign the new observation to the population Π_0 if

$$\sum_{j=1}^{p} \left(\hat{\xi}_{0j} - \hat{\xi}_j^{ref}\right) > 0.5.$$

An advantage of considering the signed distance integral as a classifier is that we can not only classify a new observation, but we can also visualise the signed distance integral.

We start by exploring the density of the signed depth values using one functional principal component score. Results of the estimated density using one principal component score can be seen in Figure 3.12, where a total of 35.51% of the variance can be explained for the simulated dataset.

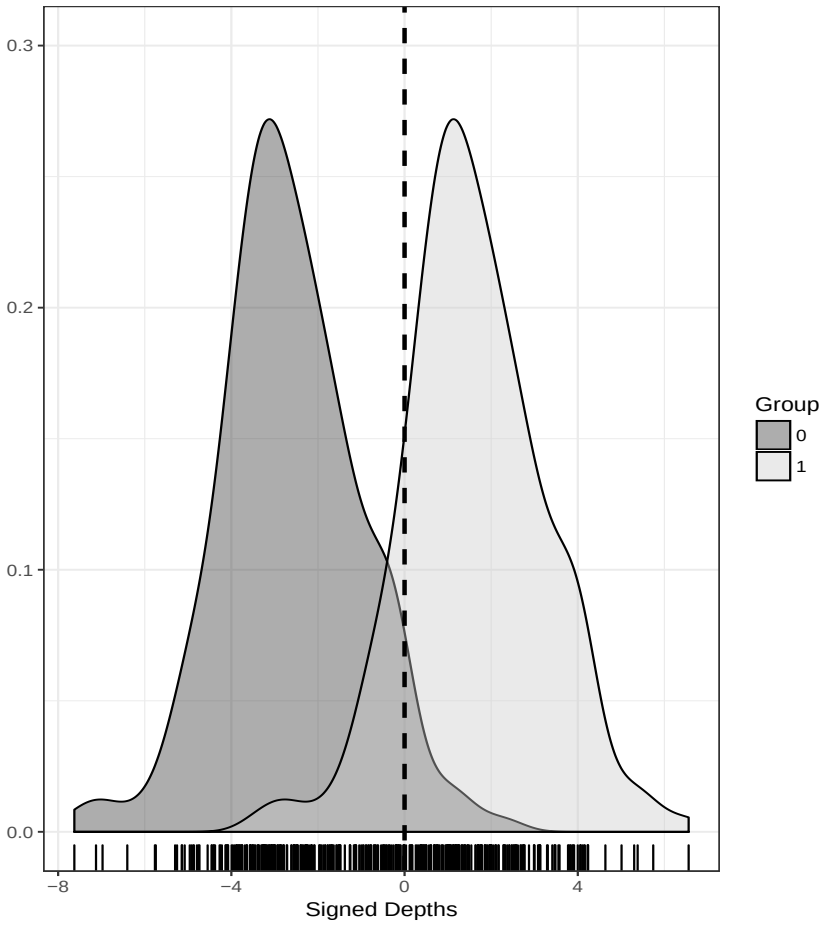


Figure 3.12: Kernel density estimation of the signed distance integral in terms of the first principal component score. The dashed vertical line at the value of zero indicates the reference curve (or signed depth of zero).

Next, we consider more that one principal component score. For the simulated dataset, the first two principal components account for 49.40% of the variability of the data, with the first eigenfunction capturing most of the variability. The plot of the first two principal component scores, in Figure 3.13, reveals a clear structure and distinguishes between two groups.

An interesting feature in Figure 3.13 is the area when there is an overlap in the two groups. We called this region of overlap to be the *border line* of a classifier. To achieve better prediction, most of the classifiers attempt to learn the borderline of each class from the

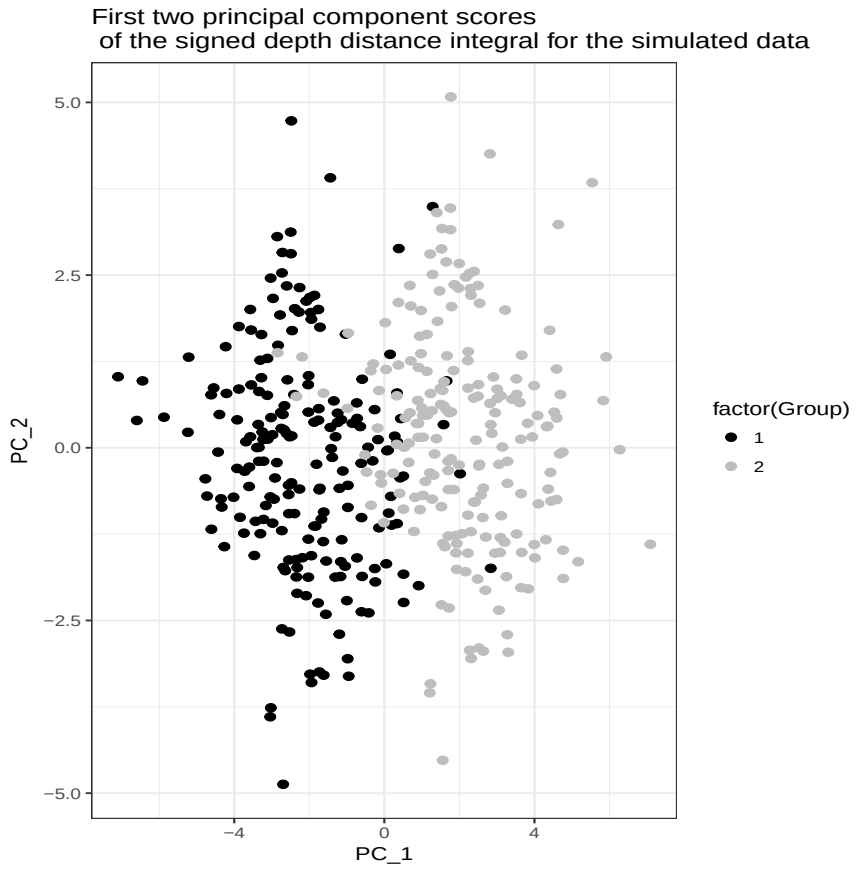


Figure 3.13: The first two principal component scores for the simulated data.

training data. We discuss this issue with more detail in Chapter 4.

3.6 Dealing with Equal Samples Sizes

In this section we explore the behaviour of a Bayesian classifier and we explore how we can construct a Bayesian classifier when dealing with equal and unequal sample sizes. We first, start exploring the Bayesian approach when the number of observations is the same in each group.

To start with, assume that we are considering two classes. Let $f_0(sdp(x_i(\mathbf{t})))$ for $i = 1,\ldots,n$, be the density of the signed depth in population Π_0. Suppose

$$\pi_G \geq 0 \quad \text{for } G = 0,1. \tag{3.33}$$

are the prior probabilities of the observed signed depths belonging to each of the groups. The resulting posterior probability that the observed signed depth belongs to group 0 (reference

group) is given by

$$\mathbb{P}(sdp(x_i(\mathbf{t})) \in \Pi_0 \mid sdp(X_i(t)) = sdp(x_i(\mathbf{t}))) = \frac{f_0(sdp(x_i(\mathbf{t})))\pi_0}{f_0(sdp(x_i(\mathbf{t})))\pi_0 + f_1(sdp(x_i(\mathbf{t})))\pi_1}, \tag{3.34}$$

where $f_1(sdp(x_i(\mathbf{t})))$ represents the density of the signed depth observations in the population Π_1 and π_1 is the prior probability that the observed signed depth is in population Π_1.

The Bayes' rule classifier assigns a new observed depth $sdp(x_0(\mathbf{t}))$ to the class with the highest posterior probability. The density $f_0(sdp(x_i(\mathbf{t})))$ is estimated using kernel density estimation while the prior probabilities are estimated by considering the number of curves in each group, i.e., $\hat{\pi}_1 = \frac{n_1}{n_0+n_1}$ and $\hat{\pi}_0 = \frac{n_0}{n_1+n_0}$, respectively.

When dealing with equal sample sizes, we can simplify equation (3.34) and assign a new signed depth $sdp(x_0(\mathbf{t}))$ to the group with the highest density. In other words,

$$\mathbb{P}(sdp(x_0(\mathbf{t})) \in y_0 \mid sdp(X_i(\mathbf{t})) = sdp(x_i(\mathbf{t}))) = \max\left\{f_0(sdp(x_0(\mathbf{t}))), f_1(sdp(x_0(\mathbf{t})))\right\}. \tag{3.35}$$

A random assignment can be used (in case of a tie) to break the tie between the appropriate classes. The Bayes' rule classifier can be rewritten in an equivalent form by pairwise comparisons of posterior probabilities. We define the *log-odds* as follows:

$$\begin{aligned} L(sdp(x_i(\mathbf{t}))) &= \log\left\{\frac{f_0(sdp(x_i(\mathbf{t})))\pi_0}{f_1(sdp(x_i(\mathbf{t})))\pi_1}\right\} \\ &= \log\left\{\frac{f_0(sdp(x_i(\mathbf{t})))}{f_1(sdp(x_i(\mathbf{t})))}\right\} + \log\left\{\frac{\pi_0}{\pi_1}\right\}. \end{aligned} \tag{3.36}$$

We discuss the Bayes' rule classifier for imbalance data in Chapter 4.

3.7 Logistic regression

When the response variable is dichotomous and the data analysis is aimed at relating this outcome to the predictors, one simple approach for modelling binary data is: **the logistic regression**. Logistic regression is a special case of a broad class of models known as generalized linear models (GLMs). Generalized linear models consists of a link function linking a random component and a systematic component.

Assume that the response Y is a random variable with distribution in the exponential family, i.e., the random variable Y is assumed to have probability density of the form

$$f_Y(y,\theta,\phi) = \exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y,\phi)\right\},$$

where θ is called the natural parameter and ϕ is called the dispersion or scale parameter, known as the random component of the model. Generalized linear models also assume that the expectation of Y denoted by μ is related to the covariates $X_1,\ldots,X_p$, through a systematic component $\eta = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p$ via $\mathbb{E}(Y) = g^{-1}(\eta)$, where the function $g(\cdot)$ is the link function. When the response Y takes values in $\{0,1\}$, a natural choice for $g^{-1}(\cdot)$ is the inverse logistic link function

$$g^{-1}(x) = \frac{\exp(x)}{1+\exp(x)}.$$

Then, the resulting model is known as logistic regression. In logistic regression, one compares the probabilities of each class by modelling the log ratio as a linear relationship. Let $p_0 = \mathbb{P}(Y = 0 \mid X_1,\ldots,X_p)$ and $p_1 = 1 - p_0 = \mathbb{P}(Y = 1 \mid X_1,\ldots,X_p)$. The log ratio (or logit) of the response probability is given by the log odds

$$\log\left(\frac{p_1}{p_0}\right) = \log\left(\frac{p_1}{1-p_1}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p, \tag{3.37}$$

where $\{\beta_0,\beta_1,\beta_2,\ldots,\beta_p\} \in \mathbb{R}^{p+1}$. The unknown regression coefficients are often estimated using maximum likelihood (OLS, WLS, and GLS) and we derive the expressions

$$p_1 = \frac{\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p)}{1+\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p)},$$

and

$$p_0 = 1 - p_1 = \frac{1}{1+\exp(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_p X_p)}.$$

Logistic regression can be applied to classification. In the multivariate regression model, the probabilities p_0 and p_1 naturally lead to a logistic regression discriminant rule. Suppose we have a set of n independent realisations of these random variables denoted by $\{(y_1,x_{i1},\ldots,x_{ip}),\ldots,(y_n,x_{n1},\ldots,x_{np})\}$. To classify a new observation $x_1^{new},\ldots,x_p^{new}$ with a logistic regression discriminant rule, the first step is to derive estimates $\hat{\beta}_0,\hat{\beta}_1,\hat{\beta}_2\ldots,\hat{\beta}_n$ of the coefficients $\beta_0,\beta_1,\beta_2,\ldots,\beta_p$. Then, we estimate

$$\hat{p}_{new} = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 x_1^{new} + \hat{\beta}_2 x_2^{new} + \ldots + \beta_p x_p^{new})}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 x_1^{new} + \hat{\beta}_2 x_2^{new} + \ldots + \hat{\beta}_p x_p^{new})},$$

and assign to $x_1^{new}, \ldots, x_p^{new}$ to the class 0 if $\hat{p}_{new} > 0.5$. A drawback of the logistic regression discriminant rule is when we have more than two groups. Even though the logistic regression discriminant rule can be extended to more than two groups, such generalisation require multiple pairwise comparisons, which in the case of many groups can be complex and tedious.

Observe that a sensible model for the probabilities in Figure 3.15 is to model such probabilities using logistic regression and predict the class membership by only considering a single covariate (the signed depth). However, this is not the only option and in fact we can consider a second covariate.

3.7.1 A second covariate: Distance to the mode estimation

In classification problems for functional data, where the population consists of two or more well-separated sub-populations, the mean and the median functions are not representative of any sub-population (Delaigle and Hall, 2010). To avoid this, we can consider the *modal* function which represents the most likely function in one of the sub-populations. The mode estimation has been studied by Hall and Heckman (2002) and Gasser et al. (1998).

Our goal is to introduce the mode as an alternative to the median for estimating the location for functional data. We start by describing the estimation of the mode in terms of the functional principal component scores and we consider using the signed distance to the mode as a second covariate.

Motivated by the Karhunen-Lòeve expansion, we can represent a function $X(t)$ in terms of the principal component scores. Let h_j be the density of the j^{th} principal component score. The modal function is given by

$$X(t)_{mode} = \sum_{j=1}^{\infty} \theta_j^{1/2} m_j \psi j(t), \tag{3.38}$$

which, for each j, has the j^{th} principal component score ξ_j equal to the mode m_j of h_j. In a finite sample, the $x(\mathbf{t})_{mode}$ can be estimated by

$$\hat{x}(\mathbf{t})_{mode} = \sum_{i=1}^{T} \hat{\theta}_j^{1/2} \hat{m}_j \hat{\psi}_j(\mathbf{t}), \tag{3.39}$$

where $\hat{\psi}$ and $\hat{\theta}$ are estimators of $\psi_j(\mathbf{t})$ and θ_j, the orthonormal eigenfunctions and eigenvalues of the spectral decomposition of the variance-covariance matrix, respectively, $\hat{m}_j$ is the mode of h_j, obtained as a value which produces a local maximum of $\hat{h}_j$, where $\hat{h}_j$ an estimator of h_j, usually estimated using kernel density estimation methods and T is a truncation point. For implementation purposes we consider the value of $T = 10$, since the effect of changing T from 10 to higher values is almost indistinguishable. While the mode estimation was carried out using the Parzen's kernel mode estimator, i.e, taking the value maximising the kernel density estimate, implemented via the R package modeest. An advantage of considering the mode is that it is a more robust function and not as susceptible to the problem of atypicals as the mean and in some cases the median. In our running example, an implementation of the modal curve for the two populations can be see in Figure 3.14.

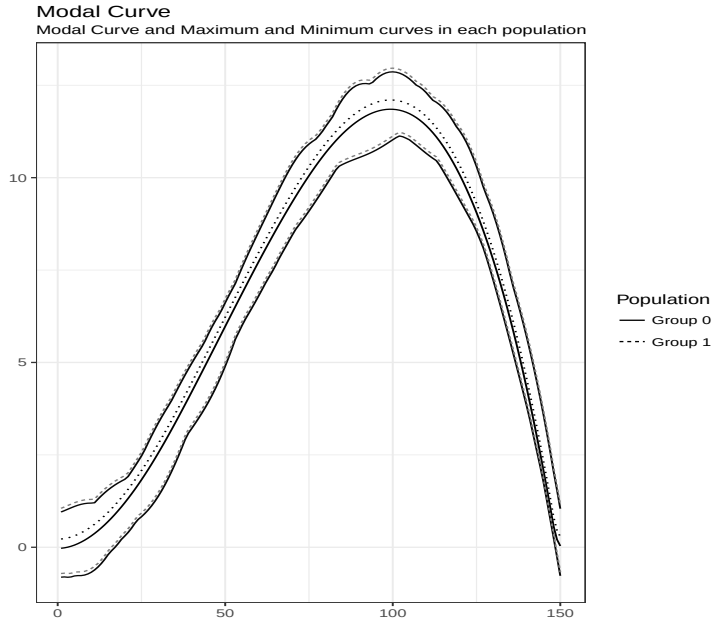


Figure 3.14: Modal curves, maximum and minimum curves for each population with a truncation point $T = 10$ for the simulated data.

To investigate if the distance to the mode can be used as a second covariate in our analysis and to avoid multicollinearity, we start investigating the correlation between the distance to the modal function and other different depths. We consider the FM depth, the h-modal depth, the random projection (RP) depth and the double random projection (DRP) depth. The correlation matrix for the simulated data set can be found in Table 3.1.

Observe that the correlation between the FM depth and the modal distance $\rho = 0.307$ suggests that we can include the modal distance as a second covariate in our proposed model.

Table 3.1: Correlation matrix between the FM depth, different functional depths and the modal distance.

	FM Depth	h-Modal	RP	DRP	Modal distance
FM Depth	1	-	-	-	-
h-Modal	0.684	1	-	-	-
RP	-0.964	-0.484	1	-	-
DRP	-0.964	-0.484	0.999	1	-
Modal distance	0.307	0.295	-0.278	-0.279	1

Moreover, the modal function also makes use of the signed distance to the mode as a second covariate in our analysis.

Now, we can consider a model with two independent variables, the signed depth $X_1 = sdp(X_i(\mathbf{t}))$ for $i = 1,\ldots,n$, and the signed distance to the mode $X_2 = sdm(X_i(\mathbf{t}))$ for $i = 1,\ldots,n$, and a response variable Y, constrained to be either zero and one indicating the true class membership. By considering this, we can fit a logistic regression using R to the following set of models: Model 1 using $X_1 = sdp(X_i(\mathbf{t}))$ as a single covariate, Model 2 using $X_1 = sdp(X_i(\mathbf{t}))$ and $X_2 = sdm(X_i(\mathbf{t}))$ and Model 3 including an interaction term, $X_1 : X_2$. The results of fitting a logistic regression are shown in Table 3.2, giving a summary of the fitted model which includes the estimated coefficients, standard errors of the coefficients, the number of observations, the log likelihood and the AIC.

Table 3.2: Summary of fitted logistic regression using the population label as a dependent variable.

Variable	Model 1	Model 2	Model 3
Signed Depth	4.70***	9.42***	17.96***
	(0.52)	(2.41)	(4.79)
Signed Distance		0.86**	0.595***
		(0.05)	(0.13)
Signed Depth : Signed Distance			-2.06**
			(0.75)
Observations	200	200	200
Log Likelihood	-61.28	-29.16	-24.4
Akaike Inf. Crit.	124.57	62.33	54.84

$^{***}p < 0.01$, $^{**}p < 0.05$, $^{*}p < 0.1$

Figure 3.15 shows the fitted line to examine the relationship between the groups and the signed depth logit regression curve fitted to the simulated data. Some atypicals observations

can be seen to have very large positive depth values from the reference group influencing the logit curve. Figure 3.16 shows the fitted line to examine the relationship between the groups and the signed distance to the mode logit regression curve fitted to the simulated data.

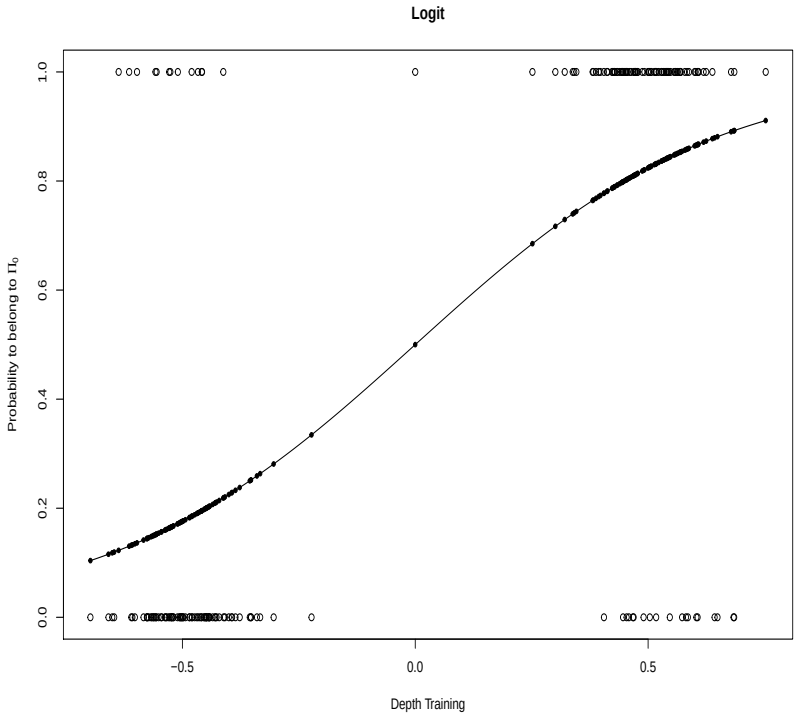


Figure 3.15: The signed depth logit regression curve fitted to the simulated data.

Measuring the performance of the logistic regression discriminant rule using the misclassification error provides a way to understand the data. Yet, when we consider modelling the estimated probabilities as a binary regression, we want to assess the adequacy of a fitted logistic regression model to the simulated data. For such a purpose Lemeshow and Hosmer Jr (1982), Hosmer and Lemesbow (1980) and Hosmer Jr et al. (2013) proposed a goodness of fit test called the Lemeshow goodness of fit test for the logistic regression model. The test is based on a contingency table type approach to develop the Lemeshow goodness of fit statistics, $\hat{C}$, obtained by calculating the Pearson Chi-Square statistic from the $g \times 2$ table of observed and estimated expected probabilities. The Lemeshow goodness of fit statistic, $\hat{C}$ is defined as

$$\hat{C} = \sum_{k=1}^{g} \frac{\left(O_k - n_k^{'}\bar{\pi}_k\right)^2}{n_k^{'}\bar{\pi}_k(1-\bar{\pi}_k)}, \tag{3.40}$$

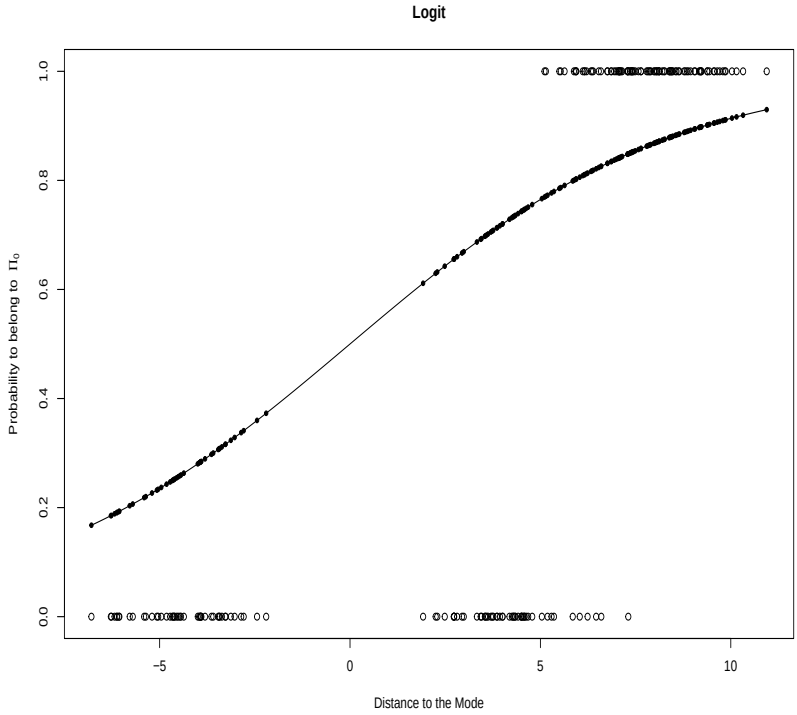


Figure 3.16: The signed distance to the mode logit regression curve fitted to the simulated data.

where $n_k^{'}$ represents the total number of observations in the k^{th} group, c_k denotes the number of covariate patterns, which is defined as every possible combination of a model's independent variables, in the k^{th} decile. The number of responses among the c_k covariate patterns are given by

$$O_k = \sum_{j=1}^{c_i} y_j,$$

and the average estimated probabilities are given by

$$\bar{\pi}_k = \sum_{j=1}^{c_i} \frac{m_j \hat{p}_j}{n_k^{'}}.$$

In Hosmer and Lemesbow (1980), it is demonstrated that the distribution of the statistic, $\hat{C}$ is well approximated by $\chi^2_{(g-2)}$. The results of applying the Hosmer-Lemeshow goodness of fit test, to the simulated data in Figure 3.3 and for the three different models are shown in Table 3.3. Observe that the corresponding p-values for model 1, model 2 and model 3 are less than 0.05. This indicates that a logistic regression model using a single covariate, two covariates and two covariates and an interaction term is not an appropriate fit for this

particular simulated dataset. The test was carried out using the R package generalhoslem and using 10 groups.

Table 3.3: Hosmer and Lemeshow goodness of fit (GOF) test applied to the simulated dataset.

Hosmer and Lemeshow goodness of fit (GOF) test			
Model 1	χ^2 = 67.398	df = 8	p-value = 1.616e-11
Model 2	χ^2 = 32.449	df = 8	p-value = 7.737e-05
Model 3	χ^2 = 59.196	df = 8	p-value = 6.7e-10

3.8 Generalized additive models (GAMs)

Generalized additive models (GAMs) proposed by Hastie and Tibshirani (1987), Hastie and Tibshirani (1990) and Wood (2006), are a more flexible extension of Additive Models. Here, we discuss the additive logistic model for binomial response data. Generalized additive models provide an extension of the standard linear regression model by allowing smooth functions of the covariates. More precisely, they model a transformed mean response as a sum of smooth functions of individual covariates. For a chosen link function $g(\cdot)$, we have

$$g\left(\mathbb{E}\left[Y \mid X_1,\ldots,X_p\right]\right) = f_0 + \sum_{j=1}^{p} f_j(X_j), \tag{3.41}$$

where $f_0 \in \mathbb{R}$, and f_j are smooth functions. Implicit in equation (3.41) is the assumption that $\mathbb{E}\left[f_j(X_j)\right] = 0$, otherwise there would be free constants in each of the functions. We are interested in estimating $\mathbb{E}[Y|X_1,\ldots,X_p]$. An estimator for $\mathbb{E}[Y|X_1,\ldots,X_p]$ is naturally given by

$$g^{-1}\left(\hat{f}_0 + \sum_{j=1}^{p} \hat{f}_j(X_j)\right),$$

where the estimated smooth functions are fitted usually by a Generalized Local Scoring Algorithm (GLSA), Backfitting or Penalised least-squares; see Hastie and Tibshirani (1987) for more.

A generalized additive model with the inverse logistic link function can serve to generalize the linear predictor of the logit function with an additive one, i.e.,

$$\log\left\{\frac{\mathbb{P}\left(Y=1 \mid X_1,\ldots,X_p\right)}{1-\mathbb{P}\left(Y=1 \mid X_1,\ldots,X_p\right)}\right\} = f_0 + \sum_{j=1}^{p} f_j(X_j). \tag{3.42}$$

As an alternative to Hastie and Tibshirani (1987), Generalized Additive Models by Wood (2006) can be represented using penalised regression splines, usually estimated by penalised regression methods where the smooth components are modelled using splines or another appropriate function basis. The appropriate degree of smoothness for the function can be estimated from the data using general cross validation. GAMs allow the data to determine the shape of the response curves, rather than being limited by the shapes available in a parametric class. For this reason, GAM modelling provides a more flexible tool for data exploration than standard GLM modelling as discussed by He et al. (2006).

To start explaining approach to the Generalized Additive Models by Wood (2006), assume the simple model with one smooth function of a single covariate, $f_1(X_1)$, and let it have a basis representation given by

$$f_1(X_1) = \sum_{j=1}^{q_1} \beta_{1j} b_{1j}(X_1), \tag{3.43}$$

where β_{1j} are the unknown parameters for f_1, q_1 is the number of unknown parameters of f_1, and $b_{1j}(t)$ is the j^{th} basis function. A single univariate function can be represented using cubic splines with knots occurring at specific locations or wherever is a datum. For the regression splines, the locations of the knots must be chosen. Let us denote the locations knots for f_1 by $\left\{x_{1i}^* : i = 1,\ldots,q_1-2\right\}$. Let the basis be given by $b_{11}(x) = 1$, $b_{12}(x) = x$ and $b_{1\ i+2} = R(x, x_{1i}^*)$ for $i = 1 \ldots q_1 - 2$, where

$$\begin{aligned} R(x,z) = &\left[(z-1/2)^2 - 1/12\right]\left[(z-1/2)^2 - 1/12\right]/4 \\ &- \left[(|x-z|-1/2)^4 - 1/2(|x-z|-1/2)^2 + 7/240\right]/24. \end{aligned} \tag{3.44}$$

An additive model of this form can be written in the linear form $\mathbf{y} = \mathbf{X}\beta + \varepsilon$, where the i^{th} row of the model matrix is given by:

$$\mathbf{X}_i = \left[1, x_{1i}, R(x_1, x_{11}^*), R(x_1, x_{12}^*), \ldots, R(x_1, x_{1q_1-2}^*)\right]. \tag{3.45}$$

An alternative to controlling smoothness is to add a *wiggliness* penalty to the least squares fitting objective. Then, the penalised regression spline fitting problem is to minimise

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \int_0^1 [f_1''(X_1)]^2 dX_1. \tag{3.46}$$

where the penalty term for $f_1(X_1)$ can be written as

$$\lambda_1 \int_0^1 [f_1''(X_1)]^2 dX_1 = \boldsymbol{\beta}' \mathbf{S}_1 \boldsymbol{\beta},$$

and the matrix $\mathbf{S}_1$ is a matrix of known values and contains zero everywhere except for the entries $\mathbf{S}_{i+2,j+2} = R_1(x_1, x_{1j}^*)$ for $i, j = 1, \ldots, q_1 - 2$. The λ_1 parameter controls the weight associated to the objective of making the function f_1 smooth. Some special cases are when $\lambda_1 = 0$, i.e., no constraint on in the function f_1, and when $\lambda_1 = \infty$ the function f_1 has to be linear. In summary, the penalised regression spline fitting problem minimises

$$\|\mathbf{y}-\mathbf{X}\boldsymbol{\beta}\|^2+\boldsymbol{\beta}^{'}\mathbf{S}_1\boldsymbol{\beta}, \tag{3.47}$$

with respect to $\boldsymbol{\beta}$. Using some algebra, it can be see that the penalised least square estimator of $\boldsymbol{\beta}$ is given by

$$\boldsymbol{\beta}=\left(\mathbf{X}^{'}\mathbf{X}+\lambda_1\mathbf{S}_1\right)^{-1}\mathbf{X}^{'}\mathbf{y}.$$

Similarly, the influence or hat matrix $\mathbf{H}$, can be written as

$$\mathbf{H}=\mathbf{X}\left(\mathbf{X}^{'}\mathbf{X}+\lambda_1\mathbf{S}_1\right)^{-1}\mathbf{X}^{'}.$$

To choose λ, we use general cross validation (GCV). Choosing the adequate smoothing parameter is an important issue; if the smoothing parameter is too high, then the data is over smoothed, and if the parameter is too low, then the data will be under smoothed. Choosing the smoothing parameter using GCV has computational advantages over ordinary cross validation. For the model in equation (3.49) the general cross validation score can be written as

$$GCV(\lambda_1)=\frac{n\sum\limits_{i=1}^{n}\left(\mathbf{y}_i-\hat{f}_1(\mathbf{x}_{1i})\right)^2}{tr\left(\mathbf{I}-\mathbf{H}\right)^2}, \tag{3.48}$$

where $\mathbf{I}$ represents the identity matrix, $\hat{f}_1(\mathbf{x}_{1i})$ is the fitted function using the observations from the first covariate X_1, and $\mathbf{H}$ is the hat matrix. Usually the GCV score is obtained by a search for each value of $\lambda_1=(\lambda_{1i})$ for $i=1,\ldots,m$.

We follow a generalized additive model to estimate the predicted probabilities in the k-RNN classifier based on signed depth and the signed distance to the mode. To remain consistent with our previous investigation using the logistic regression approach in Section 3.7, we considered the same set of models: Model 1 using $X_1=sdp(X_i(\mathbf{t}))$ as a single covariate, Model 2 using $X_1=sdp(X_i(\mathbf{t}))$ and $X_2=sdm(X_i(\mathbf{t}))$ and Model 3 including an interaction term, $X_1:X_2$. The approach we consider was implemented via the R package mgcv.

A first proposed model - One covariate

Consider a model based on one covariate $X_1=sdp(X_i(\mathbf{t}))$. The proposed model to fit for predicting the class membership is in the form of an additive smooth function with a single covariate X_1 : the signed depth. The probabilities for each group are given by

$$\mathbb{P}[Y=1 \mid X_1] = \frac{\exp\{f_0 + f_1(X_1)\}}{1+\exp\{f_0 + f_1(X_1)\}},$$

and

$$\mathbb{P}[Y=0 \mid X_1] = \frac{1}{1+\exp\{f_0 + f_1(X_1)\}},$$

respectively, with $f_0 \in \mathbb{R}$, and f_1 is a smooth function. Equivalently, the logit transformation can be expressed as

$$\log\left\{\frac{\mathbb{P}[Y=1 \mid X_1]}{\mathbb{P}[Y=0 \mid X_1]}\right\} = f_0 + f_1(X_1), \tag{3.49}$$

and instead of directly estimating the probabilities, we estimate the smooth function f_1. For the classification problem with groups of data from two population we have known population labels $Y=1$ and $Y=0$ for all the sample data, i.e., we observe (X_{1i}, Y_i) for $i=1,\ldots,N$ and we use them to estimate the constant f_0 and the function $f_1(X_1)$. For a new observation with observed covariate X_{10} we will assign this to population Π_1 if the GAM estimated $\mathbb{P}[Y=1 \mid X_{10}] > \mathbb{P}[Y=0 \mid X_{10}]$ and to population Π_0 otherwise.

One way of measuring the flexibility of the fitted model is to define the *effective degrees of freedom*. This can be defined as the trace, i.e., $tr(\mathbf{A}) = tr(\mathbf{XH})$. The maximum of $tr(\mathbf{A})$ is the number of parameters less the number of constraints. While the smoothing parameters vary, from zero to infinity, the effective degrees of freedom moves smoothly between these limits.

In this GAM implementation, the smooth function f_1 is represented using penalised regression splines and the Generalized Cross Validation (GCV) technique is used to estimate smoothing parameters. Results of the fitted model to the simulated data can be seen in Table 3.4, where the smoothing parameter is given by $\lambda_1 = 0.074$.

Table 3.4: The approximate significance of smooth terms.

Smooth terms	edf	Ref.df	Chi.sq	p-value
S(Signed Depth)	5.631	6.977	112.1	< 0.0001

In the last column of Table 3.4, we can observe a p-value. This is testing the significance of smooth terms. The details behind this calculation are found in Section 4.8.5 of Wood (2006). Clearly, there is a strong evidence that the Signed Depth is important.

The estimated smooth GAM function for the simulated data is displayed in Figure 3.17 along with 95% confidence limits shown using dashed lines. We observe that atypicals influence the smooth curve on the left-hand side by leading to higher than expected probability estimates when the depth has a large negative value. For instance, some observations are pulling up the estimated smooth GAM function for $\hat{f}(X_1)$.

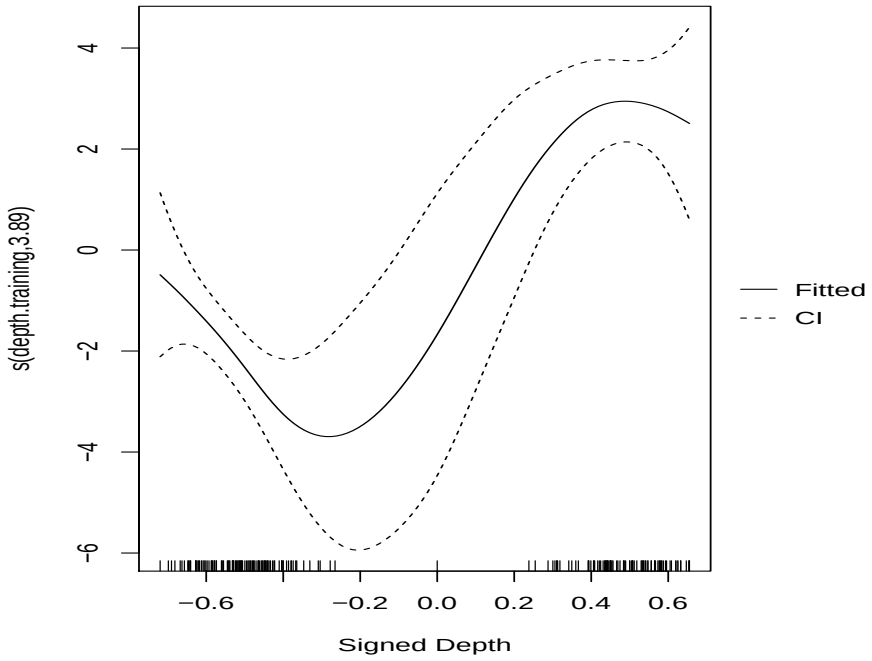


Figure 3.17: Estimated smooth GAM function for $\hat{f}(X_1)$.

The confidence intervals for the component functions of a GAM are constructed using a Bayesian approach, which are obtained by simulating from the posterior distribution of interest. More details on the form of the posterior distribution can be found in Wood (2006). The Bayesian confidence intervals are obtained from the quantiles of the posterior distribution. Some previous approaches to finding confidence intervals involved the use of bootstrap techniques, but Bayesian confidence intervals are cheaper than performing one bootstrap replicate.

A second proposed model - Two covariates

In the second model, suppose that we have two covariates variables $X_1 = sdp(X_i(\mathbf{t}))$ and $X_2 = sdm(X_i(\mathbf{t}))$ and a response variable. We are interested in predicting the class membership

in the form of additive smooth functions of two covariates X_1 : the signed depth and X_2 : the signed distance to the mode. As before, the probabilities for each group are given by

$$\mathbb{P}[Y=1 \mid X_1, X_2] = \frac{\exp\{f_0 + f_1(X_1) + f_2(X_2)\}}{1+\exp\{f_0 + f_1(X_1) + f_2(X_2)\}},$$

and

$$\mathbb{P}[Y=0 \mid X_1, X_2] = \frac{1}{1+\exp\{f_0 + f_1(X_1) + f_2(X_2)\}},$$

with $f_0 \in \mathbb{R}$, and f_1 and f_2 are smooth functions. Equivalently, the logit transformation can be expressed as

$$\log\left\{\frac{\mathbb{P}[Y=1 \mid X_1, X_2]}{\mathbb{P}[Y=0 \mid X_1, X_2]}\right\} = f_0 + f_1(X_1) + f_2(X_2), \tag{3.50}$$

In this GAM implementation, each smooth function is fitted using penalised regression splines and the Generalized Cross Validation (GCV) technique is used to estimate the smoothing parameters. Note that in terms of the estimation, we need to add a penalty term for $f_2(X_2)$, which is given by

$$\lambda_2 \int_0^1 [f_2''(X_2)]^2 dX_2 = \boldsymbol{\beta}' \mathbf{S}_2 \boldsymbol{\beta}.$$

Now, the penalised regression spline fitting problem is to minimise

$$\|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}' \mathbf{S}_1 \boldsymbol{\beta} + \boldsymbol{\beta}' \mathbf{S}_2 \boldsymbol{\beta}, \tag{3.51}$$

with respect to $\boldsymbol{\beta}$. To choose λ_1, λ_2, as with the model with a single covariate, we use general cross-validation. For the model in equation (3.50), the general cross validation score can be written as

$$GCV(\lambda_1, \lambda_2) = \frac{n \sum_{i=1}^{n} \left(\mathbf{y}_i - (\hat{f}_1(\mathbf{x}_{1i}) + \hat{f}_2(\mathbf{x}_{2i})\right)^2}{tr(\mathbf{I} - \mathbf{H})^2}, \tag{3.52}$$

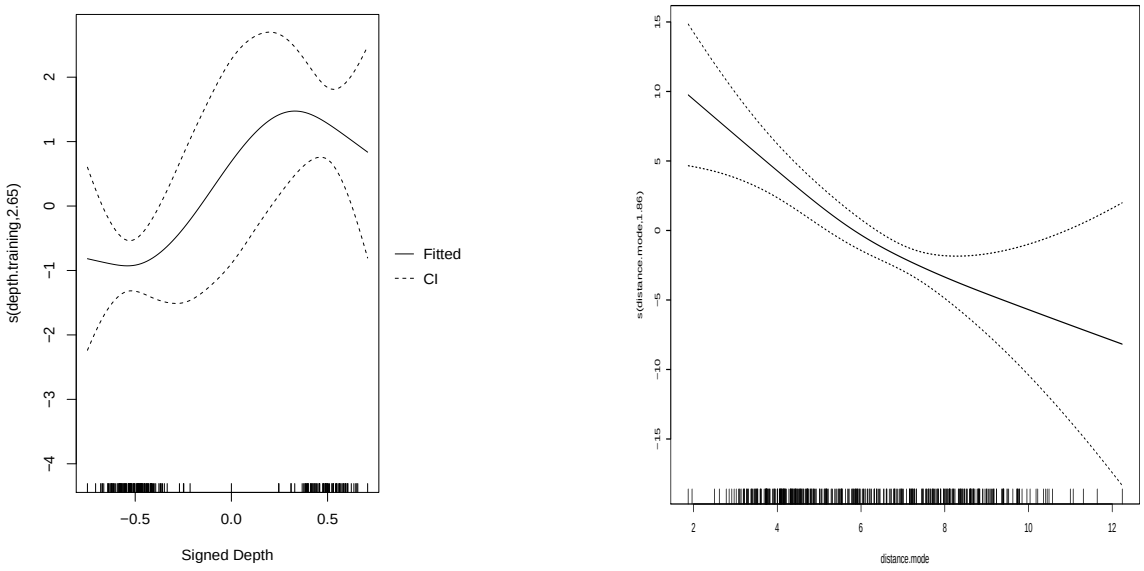
where $\mathbf{I}$ represents the identity matrix, $\hat{f}_1(\mathbf{x}_{1i})$ is the fitted function using the observations from the first covariate X_1 and $\hat{f}_2(\mathbf{x}_{2i})$ is the fitted function using the observations from the second covariate X_2. The hat matrix $\mathbf{H}$ is written as

$$\mathbf{H} = \mathbf{X}\left(\mathbf{X}'\mathbf{X} + \lambda_1 \mathbf{S}_1 + \lambda_2 \mathbf{S}_2\right)^{-1} \mathbf{X}'.$$

In this case, the GCV score is obtained by a grid search for each combination of $(\lambda_{1i}, \lambda_{2j})$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$. We applied the proposed model to the simulated dataset. Results of the fitted smoothed terms using two covariates are shown in Table 3.5, where the values of the smoothing parameters are $\lambda_1 = 0.12765$ and $\lambda_2 = 0.0283$. Figure 3.18 shows the fitted functions for the signed depth and the signed distance.

Table 3.5: Table showing the approximate significance of smooth terms.

Smooth terms	edf	Ref.df	Chi.sq	p-value
s(Signed Depth)	4.467	5.457	38.09	< 0.0001
s(Signed Distance)	1.858	2.253	29.69	< 0.001



(a) (b)

Figure 3.18: First and second estimated smooth GAM functions for the simulated dataset.

We can observe from Figure 3.18a, that the estimated smooth function for the signed depth is influenced by atypicals, while Figure 3.18b shows that the estimated smooth function for the distance to the mode seems to be linear for this particular dataset. In terms of the approximate significance of smooth terms, both terms, the signed depth and the signed distance to the mode, seem to be important in our model.

A third proposed model - An interaction term

One might add an interaction term like $f_{12}(X_1, X_2)$ to equation (3.50) and estimate with a surface smoother. We can consider a pairwise interaction between two the two covariates X_1 and X_2, where the probabilities for each group are given by

$$\mathbb{P}[Y = 1 \mid X_1, X_2] = \frac{\exp\{f_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1, X_2)\}}{1 + \exp\{f_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1, X_2)\}},$$

and

$$\mathbb{P}[Y = 0 \mid X_1, X_2] = \frac{1}{1 + \exp\{f_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1, X_2)\}}.$$

The logit transformation can be expressed as

$$\log\left\{\frac{\mathbb{P}[Y = 1 \mid X_1, X_2]}{\mathbb{P}[Y = 0 \mid X_1, X_2]}\right\} = f_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1, X_2), \tag{3.53}$$

where $f_0 \in \mathbb{R}$, and f_1, f_2 and f_{12} are smooth functions. As the components of this model are not identifiable, we need to add some restrictions in $f_{12}(X_1, X_2)$. A convenient choice is

$$\mathbb{E}\left[f_{12}(X_1, X_2) \mid X_j\right] = 0 \quad \text{for} \quad j = 1, 2. \tag{3.54}$$

Observe that such restrictions are analogous to those put on each $f_j(X_i)$, $\mathbb{E}\left[f_j(X_i)\right]$. The model in equation (3.53) is a hierarchical model, which means that the subterms $f_1(X_1)$ and $f_2(X_2)$ are in the model as well. Using penalised regression splines, the function f_{12} in equation (3.53) has a penalty of the form

$$\int \left(\frac{\partial^2 f_{12}^2}{\partial X_1^2}\right)^2 + 2\left(\frac{\partial^2 f_{12}}{\partial X_1 \partial X_2}\right)^2 + \left(\frac{\partial^2 f_{12}^2}{\partial X_2^2}\right)^2 dX_1 dX_2. \tag{3.55}$$

We can observe that the expression involves second order terms. To see how we can write the penalty form, let $\mathbf{d} = d_j(X_1) = b_j''(X_1)$ be the basis expansion for the second derivative of the function $f''(X_1)$, then, the first term in equation (3.55) can be written as

$$\begin{aligned}\int_0^1 \left(f''(X_{12})\right)^2 dX_1 &= \int_0^1 \boldsymbol{\beta}' \mathbf{d}\mathbf{d}' \boldsymbol{\beta} dX_1 \\ &= \boldsymbol{\beta}' \int_0^1 \mathbf{d}\mathbf{d}' dX_1 \boldsymbol{\beta} \\ &= \boldsymbol{\beta}' \mathbf{S} \boldsymbol{\beta},\end{aligned} \tag{3.56}$$

with $\mathbf{S} = \int_0^1 \mathbf{d}\mathbf{d}' dX_1$. Treating each integral on the right hand side of equation (3.55) in a similar way, we can observe that the penalty term can be written as $\beta'\mathbf{S}\beta$ where the coefficient matrix $\mathbf{S}$ is given by

$$\mathbf{S} = \int_0^1 \mathbf{d}_{\mathbf{X_1},\mathbf{X_1}}(X_1,X_1)\mathbf{d}_{\mathbf{X_1},\mathbf{X_1}}(X_1,X_1)' + 2\mathbf{d}_{\mathbf{X_1},\mathbf{X_2}}(X_1,X_2)\mathbf{d}_{\mathbf{X_1},\mathbf{X_2}}(X_1,X_2)' + \mathbf{d}_{\mathbf{X_2},\mathbf{X_2}}(X_2,X_2)\mathbf{d}_{\mathbf{X_2},\mathbf{X_2}}(X_2,X_2)', \quad (3.57)$$

and $\boldsymbol{\beta}$ is a parameter vector with β_j in its j^{th} element. The coefficient matrix can be expressed in terms of the known basis functions b_j providing that these possess at least two (integrable) derivatives with respect to X_1 and X_2. For the simulated dataset, results of the fitted smoothed terms using the interactions terms are shown in Table 3.6.

Table 3.6: The approximate significance of smooth terms when we consider an iteration between the signed depth and the distance to the mode.

Smooth terms	edf	Ref.df	Chi.sq	p-value
S(Signed Depth)	4.3974	5.392	38.398	< 0.0001
S(Signed Distance)	1.0009	1.001	14.264	< 0.001
S(Signed Depth, Signed Distance)	0.8706	2.000	2.712	< 0.05

In this case, the GCV score is obtained by a grid search for each combination of $(\lambda_{1h}, \lambda_{2i}, \lambda_{12j})$ for $h = 1,\ldots,2$ and $i = 1,\ldots,25$ and $j = 1,\ldots,1.5$. We applied the proposed model to the simulated dataset. Results of the fitted smoothed terms using two covariates are shown in Table 3.5, where the values of the smoothing parameters are given by $\lambda_1 = 0.132125$, $\lambda_2 = 22.7966$ and $\lambda_{12} = 0.003786$. Here, the interaction between the signed depth and the signed distance to the mode seems to be significant.

3.8.1 Preventing over-fitting in the data

Unfortunately, generalized additive models can be very sensitive to the presence of a small proportion of observations that deviate from the assumed model. Therefore, a few atypical observations could seriously affect the non-parametric estimates of the smooth regression function. We observed such behaviour in Figure 3.17 and Figure 3.18. Where for instance, some observations are pulling up the estimated smooth GAM functions.

To overcome this situation, we proposed an iterative procedure for down-weighting such observations, which is part of the methodology but very useful in our case. There are multiple

ways to approach the estimation of the generalized additive models. We followed the more stable approach of a GAM fitting process in Wood (2006).

The estimation for GAM models proceeds as follows:

Step 1. Construct basis functions and a set of one or more quadratic penalty coefficient matrices for each smooth term.

Step 2. Obtain a model matrix for the parametric component of the GAM. These matrices are combined to produce a complete model matrix and a set of penalty matrices for the smooth terms. Iteratively re-weighted Least Squares (IRLS) is then used to estimate the model; at each iteration of the IRLS, a penalised weighted least squares model is run, and the smoothing parameters of that model are estimated by GCV.

Step 3. Repeat the process until convergence is achieved.

Before, explaining in detail the iterative procedure, first consider the estimation of the GAM when the distribution of $Y_i \mid X_i$, $i = 1,\ldots,n$, belongs to an exponential family. Let $Y_1,\ldots,Y_n$ be independent random variables following a GAM model with associated covariates $\mathbf{X} \in \mathbb{R}^p$ and let $\mu_i = g^{-1}(\eta_i)$, $\eta_i = \boldsymbol{\beta}'\mathbf{X}_i$, $\upsilon_i = v(\mu_i)$ and v be a known function. Suppose we have a set of n independent realisations of these random variables denoted by $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, then the maximum likelihood estimate (MLE) satisfies the following equations

$$\sum_{i=1}^{n} \left(\frac{y_i - \mu_i}{\upsilon_i} \right) \frac{\partial \mu_i}{\partial \boldsymbol{\beta}} \Big|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}}. \tag{3.58}$$

Using some algebra, we can see that these equations can be written as

$$\sum_{i=1}^{n} (y_i - \mu_i) \frac{\partial \eta_i}{\partial \mu_i} W_i \mathbf{x}_i = \mathbf{0}, \tag{3.59}$$

where the weights are given by

$$W_i = \upsilon_i^{-1} \left(\frac{\partial \mu_i}{\partial \eta_i} \right). \tag{3.60}$$

To solve equation (3.59), a penalised iterative re-weighted least squares (P-IRLS) algorithm is used. The algorithm consists of two steps which are repeated until convergence:

1. Given the current parameter estimates $\beta^{(j)}$, and a corresponding estimated mean response vector $\mu^{(k)}$, calculate:

$$W_i \propto \frac{1}{\upsilon\left(\mu_i^{(k)}\right) g'\left(\left(\mu_i^{(k)}\right)\right)} \quad \text{and} \quad z_i\left(\boldsymbol{\beta}^{(j)}\right) = \eta_i(\boldsymbol{\beta}^{(j)}) + \left(y_i - \mu_i(\boldsymbol{\beta}^{(j)})\right) \frac{\partial \eta_i}{\partial \mu_i}\left(\boldsymbol{\beta}^{(j)}\right).$$

2. Minimize

$$\left\| \sqrt{\mathbf{W}}\,(\mathbf{z} - \mathbf{X}\boldsymbol{\beta}) \right\|^2 + \lambda \boldsymbol{\beta}' \mathbf{S} \boldsymbol{\beta},$$

with respect to $\boldsymbol{\beta}$ to obtain $\boldsymbol{\beta}^{(k+1)}$, where $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_p]$ is the matrix of known values and $\mathbf{W}$ is a diagonal matrix such that $W_{ii} = W_i$.

A drawback of the penalised iterative re-weighted least squares schema is that it can only be used to estimate the model coefficients, $\boldsymbol{\beta}$, given the smoothing parameter λ. To perform down-weighting on atypicals, we first start by considering a loss function. The role of this loss function is to give less weight to the vector of scaled Pearson residuals, $\mathbf{r}$, of a first fitted GAM model which does not uses weights.

In our implementation, we consider the Huber loss function introduced in Huber (1964) and Huber (1973). The Huber loss function, for a vector of residuals $\mathbf{r} = \{r_1, \ldots, r_n\}$ is defined piecewise by

$$L_\delta(\mathbf{r}) = \begin{cases} \frac{1}{2}\mathbf{r}^2, & \text{for } |\mathbf{r}| \leq \delta, \\ \delta(|\mathbf{r}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

where δ is the down-weighting threshold. Observe that the Huber loss function is a quadratic function for small values of r. Thus, this function gives less weight to the observations that are far from the centre as demonstrated in Figure 3.19.

When the residuals follows a standard Gaussian distribution, usually the value of the threshold parameter is $\delta = 1.345$. When we are dealing with generalized additive model, a better way to check the model fit is to consider scaled Pearson residuals. Scaled Pearson residuals are raw residuals divided by the standard deviation of the data. More precisely,

$$r_{P_i} = \frac{r_i - \hat{\mu}_i}{\sqrt{v(\hat{\mu}_i)}}. \tag{3.61}$$

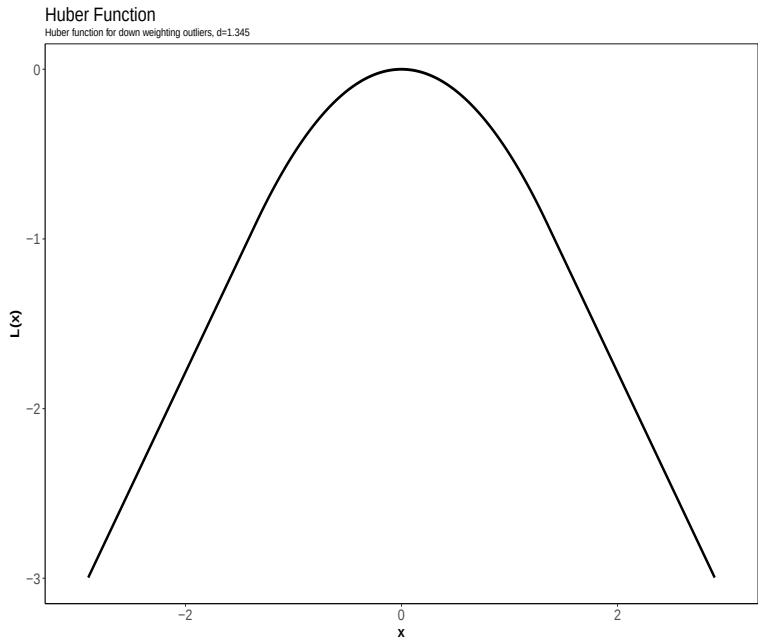


Figure 3.19: Hubert loss function with a threshold parameter $\delta = 1.345$ and a set of simulated residuals.

The name of Pearson residuals is taken from the fact that for the Poisson distribution, the Pearson residuals are defined as the signed square root of the component of the Pearson χ^2 goodness of fit statistics. Note that observations with an absolute Pearson residual value larger than δ will be down-weighted; the larger the residual, the lower the weight. For down-weighting atypicals, we consider a set of Huber weights using the residuals previously fitted. In our case we set $\delta = 1.65$, which produced good results in this situation. Note that as the value of δ gets larger, the robust fit resembles the classic one.

The procedure for down-weighting atypicals can be summarised in the following steps.

Step 1. Fit a GAM model which does not use weights. We called this model M.

Step 2. Find the scaled Pearson residuals from the model M.

Step 3. Find a set of Huber weights using the residuals from Step 2.

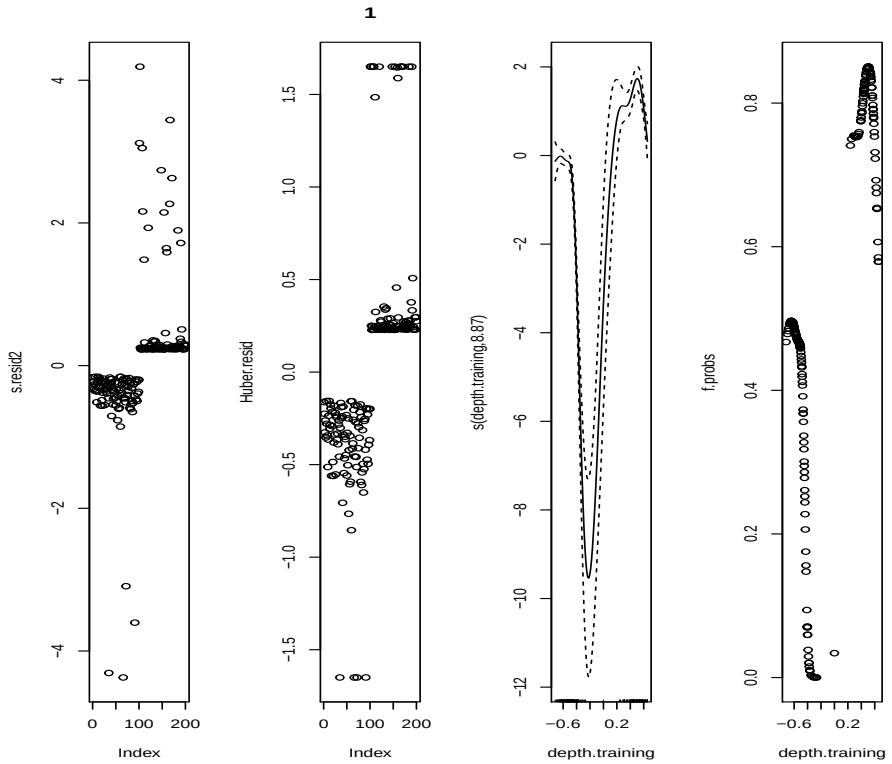
Step 4. Fit a new model now using the weights determined in Step 3 and now let this be M.

Step 5. Iterate steps 2 to 4 until the difference between the log likelihood of the previous model is small.

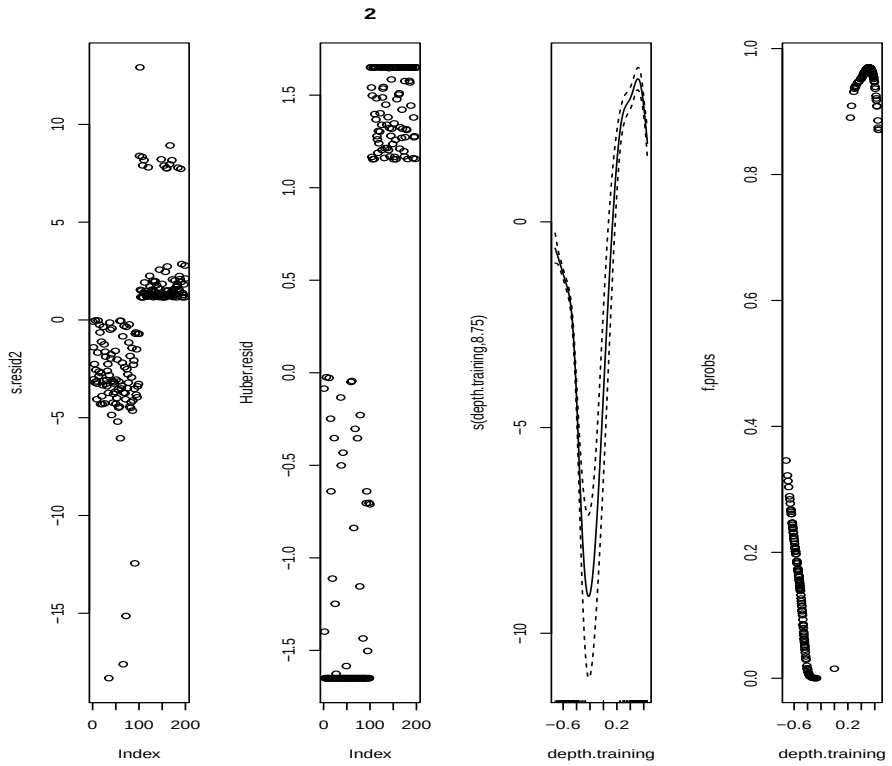
However, a drawback about down-weighting the effects of atypical in the data is that

this process tries to guard against the model overfitting on the training set which could then possibly lead to it not working as well on independent data.

In terms of the implementation, we use the weights option in the **gam()** function in the mgcv package to down-weight atypicals. Figure 3.20 shows the first two iterations of this procedure. Figure 3.20(a) shows the value of the scaled Pearson residuals, the Huber residuals, an estimated GAM function and the fitted probabilities for the signed depth. We observe some atypicals in the data and we see observations with a Huber residual greater than 1.5. After the second iteration, the fitted probabilities in Figure 3.20(b) show the outlier observations down-weighted. Comparing Figure 3.20(a) and Figure 3.20(b) we can observe that the atypicals are down-weighted after the first iteration.



(a)



(b)

Figure 3.20: Scale Pearson residuals, the Huber residuals, an estimated GAM function for the signed depth and the fitted observations for (a) first iteration and (b) second iteration using a set of Huber weights.

3.9 The GAM as a classifier

In this section we explain how we can use the distance to the mode and an interaction term between those two variables to estimate the predicted probabilities and predict the group membership of a new curve, $x_0(t)$. We start by describing the binary problem and then we extend this approach to more than two groups.

Recall that the logit transformation can be expressed as

$$\log\left\{\frac{\mathbb{P}[Y=1 \mid X_1,X_2]}{\mathbb{P}[Y=0 \mid X_1,X_2]}\right\} = f_0 + f_1(X_1) + f_2(X_2) + f_{12}(X_1,X_2), \tag{3.62}$$

where $f_0 \in \mathbb{R}$, and f_1, f_2 and f_{12} are smooth functions. The probabilities for each group are given by $\mathbb{P}[Y=1 \mid X_1,X_2]$ and $\mathbb{P}[Y=0 \mid X_1,X_2]$ as defined in Section 3.8. As we saw before, GAMs provide a flexible tool to estimate such probabilities. For a two class problem and equally balanced number of observations in each group, we can assign a curve $x(t)$, with covariates X_1 : the signed depth and X_2 : the signed distance to the mode, according to the following rule

$$C(X_1,X_2) = \begin{cases} 1 & \text{if } \hat{\mathbb{P}}[Y=1 \mid X_1,X_2] > 0.5 \\ 0 & \text{if } \hat{\mathbb{P}}[Y=0 \mid X_1,X_2] < 0.5 \end{cases}$$

where $\hat{\mathbb{P}}[Y=1 \mid X_1,X_2]$ and $\hat{\mathbb{P}}[Y=0 \mid X_1,X_2]$ are estimated as we described in Section 3.8. To classify a new observation, $x_0(t)$, we first derive estimates of $\hat{f}_1(X_1), \hat{f}_2(X_2)$ and $\hat{f}_{12}(X_1,X_2)$, then we calculate

$$p_{new} = \frac{\exp(\hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_{12}(X_1,X_2))}{1 + \hat{f}_1(X_1) + \hat{f}_2(X_2) + \exp(\hat{f}_{12}(X_1,X_2))}, \tag{3.63}$$

and assign the new observation to population Π_1 if $p_{new} > 0.5$. Similarly, we can consider using the subterms $f_1(X_1)$ and $f_2(X_2)$ to classify new observations. Table 3.7 shows the classification results for three different GAMs. Model 1 refers to the model using the signed depth as a single covariate. Model 2 is the model formed by two covariates, the signed depth and the signed distance to the mode. Model 3 is the model formed by using an interaction between the two covariates.

We decided to use model 3 with the interaction terms because it achieves a higher accuracy compared with the models formed by considering the subterms $f_1(X_1)$ and $f_2(X_2)$. An advantage of working with with an interaction between two covariates is that we can

Table 3.7: Several performance measures for different GAMs.

Model	Measure of Performance		
	Error Rate	Accuracy	AUC
Model 1	0.1525	0.8475	0.8475
Model 2	0.0625	0.9375	0.9375
Model 3	0.0325	0.9675	0.9675

visualise the interaction between the distance to the mode and the signed depth. For the running example, we considering the population Π_0 as the reference group. The estimated classification surface can be seen in Figure 3.21. It shows the signed depth against the signed distance to the mode along with the contours of the surface fitted using the interaction terms. The contour edges are set to be 0.5, because observations above this level are assigned to population Π_0 and observations below are assigned to population Π_1.

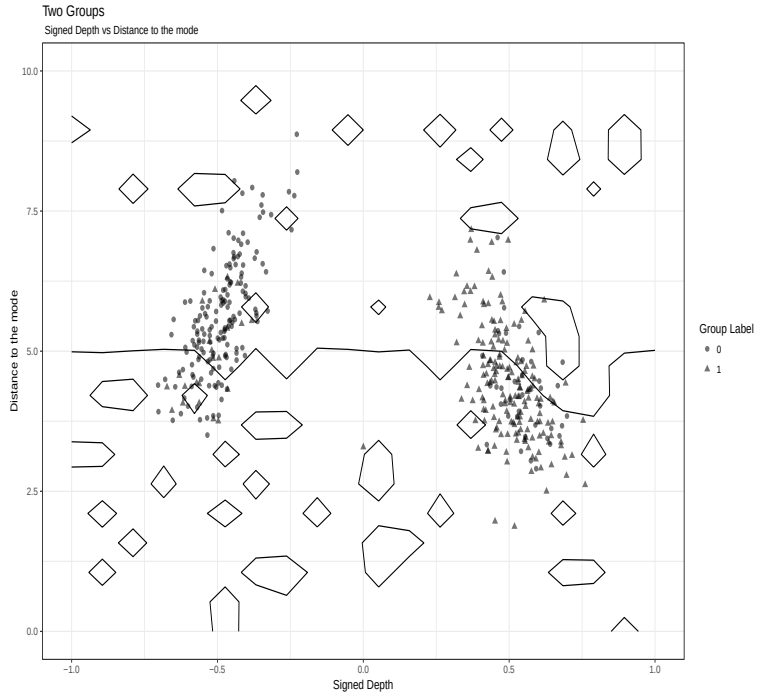


Figure 3.21: Estimated surface for the two functional predictors the distance to the mode in the reference group and the signed depth in the simulated dataset when the response is 0 or 1.

3.9.1 Extending to more than two groups

In many applications, we may, for instance, be considering different curves generated from different groups. In this section, we consider extending the proposed approach to more than two classes. We start by exploring the extension of the logistic additive model regression to classification with multiple classes.

Let $\{0,1,2,\ldots,G-1\}$ be the G classes or groups associated with the the observations. When we consider more than two classes, the model is called multinomial logistic regression which generalises the logistic regression to multiple classes. This is similar to fitting G separate logit models. The fitting procedure uses the log-odds ratio and assumes that the log-odds ratio for class i relative to population Π_0 has the following form

$$\log\left\{\frac{\mathbb{P}[Y=i \mid X_1,X_2]}{\mathbb{P}[Y=0 \mid X_1,X_2]}\right\} = f_0^i + f_1^i(X_1) + f_2^i(X_2) + f_{12}^i(X_1,X_2), \tag{3.64}$$

for $i \in \{1,2,\ldots,G-1\}$. Here $f_1^i(X_1)$, $f_2^i(X_2)$ and $f_{12}^i(X_1,X_2)$ are smooth functions considering only observations from the i^{th} group. As before, the conditional probability for population Π_0 is given by

$$\mathbb{P}[Y=0 \mid X_1,X_2] = \frac{1}{1+\exp\{f_1(X_1)+f_2(X_2)+f_{12}(X_1,X_2)\}}.$$

while the conditional probability of observing population Π_i is given by

$$\mathbb{P}[Y=i \mid X_1,X_2] = \frac{\exp\left\{f_1^i(X_1)+f_2^i(X_2)+f_{12}^i(X_1,X_2)\right\}}{1+\exp\left\{f_1^i(X_1)+f_2^i(X_2)+f_{12}^i(X_1,X_2)\right\}}.$$

In the same sense, this provides a comparison between observations in population Π_i and Π_0.

To demonstrate the multinomial logistic regression for more than two groups, we start simulating an additional group. Let $\mathcal{GP}(m(t),K(s,t))$ be a Gaussian process with mean $m(t) = 80*(1-t)*t^2$ and $K(s,t) = 0.1*\exp(-100*(s-t)^2)$ as specified in Chapter 1. Then, we generate three different populations Π_0, Π_1 and Π_2 with equal sample sizes: $n_0 = n_1 = n_2 = 200$ curves in each population. The mean for Π_0 is $m_0(t) = 80*(1-t)*t^2$ where the mean for Π_1 is replaced by $m_1(t) = \delta + m_0(t)$ and the mean for population Π_2 is $m_2(t) = 2\delta + m_0(t)$ where $\delta = 0.25$. In terms of computing, the package mgcv includes multinomial logistic regression which we implemented.

We assume that the categories of the response variable are coded 0,1, or 2. In the three outcome category model we require to compute two logit functions. We decided to use

population Π_0 as the reference population and form logits comparing population Π_1 and Π_2 to it. More specifically, we fit

$$\log\left\{\frac{\mathbb{P}[Y=1 \mid X_1, X_2]}{\mathbb{P}[Y=0 \mid X_1, X_2]}\right\} = f_0^1 + f_1^1(X_1) + f_2^1(X_2) + f_{12}^1(X_1, X_2), \tag{3.65}$$

and

$$\log\left\{\frac{\mathbb{P}[Y=2 \mid X_1, X_2]}{\mathbb{P}[Y=0 \mid X_1, X_2]}\right\} = f_0^2 + f_1^2(X_1) + f_2^2(X_2) + f_{12}^2(X_1, X_2). \tag{3.66}$$

Observe that we only form logits comparing population Π_0 versus Π_1 and Π_0 versus Π_2, because the logit function comparing Π_1 versus Π_2 is the difference between these two logits. Next, we examine the predicted probability in each group and classify the observation according to the highest probability. Results in terms of the confusion matrix for three different classes are shown in Table 3.8

Table 3.8: Confusion Matrix or Error Matrix for three classes 0, 1 and 2.

	Predicted Class 0	Predicted Class 1	Predicted Class 2
True Class 0	181	17	2
True Class 1	63	81	56
True Class 2	4	7	189

Observe that observations in class 1 are often misclassified because they are very similar to observations in class 0 and class 2. For visualisation, we estimate the surface for the two functional predictors in the three groups. Results of the estimated surface can be seen in Figure 3.22.

3.10 Simulations

In this section, we conduct a simulation study to compare the performance of the k-RNN classifier based on depth, and the GAM additive model with the two proposed covariates and outlier down-weighting. To compare the proposed methods, we consider a variety of scenarios; we mainly explore the effectiveness of the methods for a set of different simulation methods for functional data and for different contaminated models. To show the robustness of our method we introduce atypicals observations in our data according to the models introduced in Chapter 2.

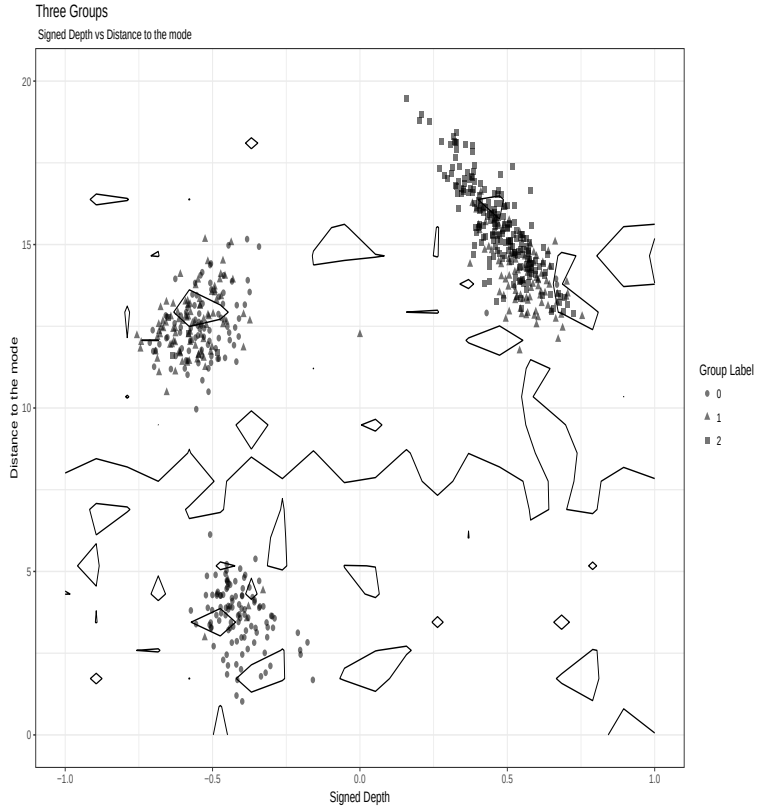


Figure 3.22: Estimated surface for the two functional predictors the distance to the mode in the reference group and the signed depth in the simulated dataset with three different groups.

3.10.1 Comparison Methods

We start by describing the methods that will be used to compare our proposed classifier based on signed depth and distance to the mode. The competing methods we use here are methods based on depth functions. The first method we start describing is the k nearest neighbour method based on depth, the k ranked nearest neighbour, the GAM classifier based on two covariates, the distance to the α - trimmed mean and finally we describe the within maximum depth method. The last two methods we compare are methods derived from the classification rules, based on depth introduced in López-Pintado and Romo (2006).

The k-NN based on depths

The k-NN rule, introduced by Fix and Hodges Jr (1951), classifies each element in the test set by finding the k nearest observations in the training set and assigning the observation to the most frequent class among these k neighbours. In this setting, we consider a modification of the k-NN rule. Instead of considering the Euclidean distance to measure the observations, we

based our computations on the univariate Euclidean distance between depths. The basic idea behind the k-NN is simple and intuitive: close patterns, in depth, are more likely to belong to the same class. The k-NN based on depths predicts the label of a new observation, $x_0(\mathbf{t})$, based on the classes associated to the k closest to the new curve, using a majority decision rule. The k-NN based on depth can be summarised in Algorithm 2.

Algorithm 2: The k-NN Algorithm based on depths.

Input : The set of training curves of size n_{train} and a test set of curves of size n_{test}. A depth function $D(\cdot)$ and a fixed value of k.

Output : Classifications for the set of test curves.

1 For a depth function, compute the functional depth value of the curves in the training set.

2 **foreach** *Curve in the test set* **do**

3 Compute the depth $D(\cdot)$ of the curve with respect to the combined sample of the new observation and the curve in the training set.

4 Compute the univariate Euclidean distance among the different depths and select the set of k closest (in depth) curves to the new observation.

5 Assign a label based on the predominance of a particular class in this k neighbourhood.

6 If there are exactly k observations closest in depth, classify according to either of the two populations with probabilities 1/2 to break the tie.

The value of k is selected by a grid search that minimises the misclassification error of the curves in the training set and the functional depth we consider is the Tukey-FM depth.

The k-RNN based on depths

The k-RNN based on depth was introduce in Section 3.2. To classify a new observation $x_0(\mathbf{t})$ using the k-RNN we begin by ranking the data by their depths. After we obtain the ranks, we calculate to depth of $x_0(\mathbf{t})$ with respect to the combined samples and determine where it falls in terms of its ranking when combined with both groups. The detailed algorithm is summarised in Algorithm 1.

The GAM Classifier

The proposed GAM classifier use two covariates X_1, the signed depth and X_2, the distance to the mode of the reference group, which are available for a response Y. Using a generalized additive model, we get estimates of the probabilities $\mathbb{P}[Y = 0 \mid X_1, X_2]$ and $\mathbb{P}[Y = 1 \mid X_1, X_2]$. For a new observation, $x_0(t)$, the GAM classifier calculates

$$p_{new} = \frac{\exp(\hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_{12}(X_1, X_2))}{1 + \exp(\hat{f}_1(X_1) + \hat{f}_2(X_2) + \hat{f}_{12}(X_1, X_2))}, \tag{3.67}$$

where $\hat{f}_{12}(X_1, X_2)$ is a smooth function that describes the interaction between the two covariates X_1 and X_2. The generalized additive model classifier assigns the new observation to population Π_0 if $p_{new} > 0.5$. In case of atypicals, the methodology developed in Section 3.8.1 is applied.

The α - trimmed mean

L-estimates or L-statistics form an important class of estimators in nonparametric statistics. Working with L- statistics provides the advantage that L - estimates are often a robust statistic (Tyler, 2008). Examples of such estimators are the trimmed mean and the Winsorized mean. L-estimates are defined in terms of a linear combination of order statistics. In particular, the trimmed means are defined in terms of the average of the most central $(1-\alpha)n$ observations, for $\alpha \in [0, 1]$, and constitute estimates that range from the sample mean to the sample median; see Maronna et al. (2006).

In the functional case, the version of the α - trimmed mean, due to Fraiman and Muniz (2001) and López-Pintado and Romo (2009), is defined as the average of the $n - [n\alpha]$ deepest curves from the sample $x_1(\mathbf{t}), \ldots, x_n(\mathbf{t})$, where $[n\alpha]$ denotes the integer part of $n\alpha$. Let $x_{(1)}(\mathbf{t}), \ldots, x_{(n)}(\mathbf{t})$ be the ordered sample of curves, where $x_{(1)}(\mathbf{t})$ represents the deepest observation and $x_{(n)}(\mathbf{t})$ is the least deep one. The α - trimmed mean estimator is defined as

$$\hat{m}_n^{\alpha}(\mathbf{t}) = \frac{\sum_{i=1}^{n-[n\alpha]} x_{(i)}(\mathbf{t})}{n - [n\alpha]}. \tag{3.68}$$

A functional binary classifier, based on the α - trimmed mean first calculates the distance from the new observation to the trimmed mean of each group. The steps can be summarised as follows:

Step 1. For a fixed value of α, compute the trimmed mean for each group $\hat{m}^{\alpha}_{ni}(\mathbf{t})$ and $i \in \{0,1\}$.

Step 2. Calculate the distance, from a new observation $x_0(\mathbf{t})$ to the trimmed mean for each group $\hat{m}^{\alpha}_{n0}(\mathbf{t})$ and $\hat{m}^{\alpha}_{n1}(\mathbf{t})$. Usually the distance between the new observation and the the trimmed mean of each group $d\left(x_0(\mathbf{t}), \hat{m}^{\alpha}_{ni}(\mathbf{t})\right)$ is calculated as

$$\begin{aligned} d\left(x_0(\mathbf{t}), \hat{m}^{\alpha}_{ni}(\mathbf{t})\right) &= \left\| x_0(\mathbf{t}) - \hat{m}^{\alpha}_{ni}(\mathbf{t}) \right\|_1 \\ &= \int_a^b \left| x_0(t) - \hat{m}^{\alpha}_{ni}(t) \right| dt \end{aligned} \tag{3.69}$$

for $i \in \{0,1\}$ for curves observed in a close interval $\mathcal{T} = [a,b]$.

Step 3. Classify $x_0(\mathbf{t})$ to the population such that the distance to the trimmed mean is the minimum, more specifically,

$$d\left(x_0(\mathbf{t}), \hat{m}^{\alpha}_{ni}(\mathbf{t})\right) = \min_{i \in \{0,1\}} d\left(x_0(\mathbf{t}), \hat{m}^{\alpha}_{ni}(\mathbf{t})\right).$$

A more robust version of the α - trimmed mean algorithm is the Trimmed Weighted Averaged Distance. This method, proposed in López-Pintado and Romo (2006), obtains the distance from an observation to a group as a weighted average of distances to each element in the group. The weights are determined by the depths of the points within the group. Thus, the influence of any observation on the distance depends on its depth. However, as pointed out by López-Pintado and Romo (2006), the result depends strongly on the number of observations in each group. An alternative approach, termed the Trimmed Weighted Average Distance (TWAD), considers only the $m \leq n_0, n_1$ deepest observations from each group to compute the distance. The Trimmed Weighted Average Distance is given by

$$TWAD\left(x(\mathbf{t}), \Pi_g\right) = \frac{\sum_{i=1}^{m} d\left(x(\mathbf{t}), x_{(i)}(\mathbf{t})\right) D\left(x_{(i)}(\mathbf{t})\right)}{\sum_{i=1}^{n_g} D\left(x_{(i)}(\mathbf{t})\right)}. \tag{3.70}$$

where $D(\cdot)$ is a depth function and $d\left(x(\mathbf{t}), x_{(i)}(\mathbf{t})\right)$ is the distance calculated as

$$\begin{aligned} d(x(\mathbf{t}), x_{(i)}(\mathbf{t})) &= \left\| x(\mathbf{t}) - x_{(i)}(\mathbf{t}) \right\|_1 \\ &= \int_a^b \left| x(t) - x_{(i)}(t) \right| dt, \end{aligned} \tag{3.71}$$

for the observed functions $x_i(\mathbf{t})$ with $t \in [a,b]$. A classifier based on the Trimmed Weighted Average Distance calculates the distance from an observation to a group as a weighted average of distances to each element in the group and classifies a new observation $x_0(\mathbf{t})$ to the group which the Trimmed Weighted Average Distance is the minimum.

3.10.2 Within maximum depth method

The within maximum depth (WMD) classification method, computes the depth values of the curves relative to the data in both groups. Originally proposed in Ghosh and Chaudhuri (2005), the WMD classifier is one of the simplest classifiers based on depth. An advantage of the within maximum depth (WMD) classification method is that it does not assume any specific parametric form or any probability distribution for the populations. Instead, they classify a new observation into the class with respect to which has the maximum depth.

However, within maximum depth classifiers have the drawback that they lead to a linear or a quadratic classifier. So, if we are interested in a more flexible procedure than just a classifier with a linear or quadratic class boundaries, then within maximum depth classifiers will not be an appropriate choice. A maximum depth classifier assigns a new observation $x_0(\mathbf{t})$ to the population with the maximum depth. More specifically,

$$d_D(x_0(\mathbf{t})) = \arg\max_j D(x_{n_j}(\mathbf{t})) \quad \text{for} \quad j \in \{0,1\}. \tag{3.72}$$

where n_j is the number of observations from the j^{th} group and $D(x_{n_j}(\mathbf{t}))$ is the depth of the curve $x(\mathbf{t})$ in the j^{th} group. Observe that we can modify equation (3.72) by multiplying by the prior probabilities of the classes. More generally,

$$d_D(x_0(\mathbf{t})) = \arg\max_j \pi_j D(x_{n_j}(\mathbf{t})) \quad \text{for} \quad j \in \{0,1\}. \tag{3.73}$$

where π_j is the prior probability that the observed depth belongs to the j^{th} population. In the multivariate case, as pointed out in Ghosh and Chaudhuri (2005), this type of classifiers leads to a linear or quadratic classifier depending the type of depth and if a common matrix is used for the observations in each population.

3.11 Simulation Setup and Simulations Results

In this section we compare the performance of the proposed k-RNN based on the signed depth, the generalized additive model, the k-NN, the classifier based on maximum depth and the classifier based on the trimmed mean depth. We start by setting the simulation parameters. We compare the performance with different ways to simulate a Gaussian Process using different kernel functions, different models to introduce atypicals in the data, and different sample sizes. To investigate their performance, when we introduce atypicals we follow the methods discussed in Section 2.3.

To start with, we generated curves from three different Gaussian Processes.

1. Model 1: Simulation using a $\mathcal{GP}(m(t),K(s,t))$ with mean $m(t) = 80(1-t)t^2$ and $K(s,t) = 0.1\exp\left\{-100(s-t)^2\right\}$.

2. Model 2: Simulation using a $\mathcal{GP}(m(t),K(s,t))$ with mean $m(t) = 80(1-t)t^2$ and $K(s,t) = 1.55\exp\left\{-0.125\,|s-t|^{1.25}\right\}$.

3. Model 3: Simulation using a $\mathcal{GP}(m(t),K(s,t))$ with mean $m(t) = 80(1-t)t^2$ and $K(s,t) = 1.85\exp\left\{3.5\,|s-t|^{1.35}\right\}$.

A graphical representation of the above models can be seen in Section 2.1. For the clarity and conciseness, we restrict our simulation study to these methods. For each of the three main models, we consider two different sample sizes, $n = 50$ and 200 and consider the contamination models: No contamination Model, Asymmetric Contamination, Linear Peak Contamination and Shape Contamination. A low sample size generates a low contamination rate in the data while a large sample size generates more atypicals in our data. The first and third models generate smooth curves, while fairly rough sample curves are generated using the second model model.

Additionally, a single non contamination model and a sample size of $N = 400$ was run as a pilot study discussed in the simulation results to measure the burden of the computational time. We ran 100 simulations for each scenario. In each case we sampled the curves on 150 equidistant points over the close interval $[0,150]$. For all the functional depths, we use a discretised version of their definitions as we explained in Section 3.3.2 . We perform the comparison among methods in terms of misclassification, specificity and sensitivity.

For the classification method using maximum depth, we consider the use of the random projection (RP) depth, which is based on medians and is robust to atypicals in the data. To reduce variability, the RP depth is implemented by projecting the discretised trajectories over 50 independent normalised standard Gaussian random directions and taking the average. While for the alpha trimmed mean classifier, we consider the Trimmed Weighted Average distance with $m = n - [n\alpha]$ and $\alpha = 0.2$ to increase robustness.

The simulations have been conducted in R using the functions in the packages depth, depthTools, and for the implementation of the Tukey-FM depth we use the version in the fda.usc, although a similar function can be found in the depth packages. We use the library MASS for the simulation of Multivariate Normal Distribution used in the simulation of Gaussian Process, modeest for the estimation of the mode of the kernel density estimation, pROC for the specificity and sensitivity and mgcv for the generalized additive model. We implemented our own version of the contaminated models.

3.11.1 Simulations Results

We start by discussing our simulation results over the different simulation models and samples containing atypical observations. We compare the newly proposed methods against the competing methods and we compare the performance in terms of misclassification, specificity and sensitivity and we report their means and standard deviations in the next tables. However, most methods of functional data, especially methods involving depth computation, can be time consuming. Hence, we ran a pilot study to measure the computational time required to classify a total of $N = 400$ observations split into two equally size groups $n_1 = n_2 = 200$ which we repeated over a total of 100 simulations.

A Pilot Study

Most methods in functional data and for functional classification tend to be time consuming and if these methods involve parameter tuning, smoothing technique and preprocessing data, the computational time significantly increases. The methods proposed here are not an exception. Most of our proposed methods require simulating random variables, parameter tuning and multiple integrations. However, when running multiple scenarios the computational time is an important practical issue.

The goal of this pilot study is to measure the computational time required for five classification methods and a total of $N = 400$ curves equally split. To have an idea of the relative computational burden of the different methods, we ran the simulations using R version 3.4.3 on a Windows 7 computer with an Intel Core i5-3470 CPU at 3.20GHz and 8GB of RAM. To measure the time, we use the function system.time() which returns the user, system and elapsed CPU time. In this case, we report the elapsed time, which is the is the wall clock time taken to execute the function. User CPU time is the time spent by the current process while the system CPU time returns the time spent by the operating system. Note that to avoid extra computations, we only consider the functions that produce the necessary output and for functions that produce a plot, the plotting option is disabled. Notice that we load the required libraries only once so there is no extra computational time for loading the libraries. To start measuring the time, we begin preprocessing the data; this is a common procedure in all the different methods and involves loading the required libraries, loading our functions, creating the functional objects and simulating the data. After this procedure is done, we then apply the corresponding classification method. Thus, the total measured computational time involves preprocessing the data, and classifying according to the classification methods. Going through all these procedures constitutes an iteration and this is repeated a total of 100 times.

Table 3.9: Time and relative ratio of preprocessing, and classifying for the k-NN, k-RNN, GAM, Maximum Depth and α-Trimmed Mean method over one iteration.

Proceduce	Time (in seconds)	Relative Computational Ratio
Preprocessing	23.005	2875.271
k-NN	26.977	3371.629
k-RNN	27.214	3401.252
GAM	0.008	1.000
Maximum Depth	213.123	26636.790
α-Trimmed Mean	1.422	177.744

Table 3.9 shows the time in seconds to run each classification procedure and the relative computational ratio of the different methods. We can observe that if we take the smallest CPU time (0.008 sec) for the generalised additive model as a unit, the relative computational ratio which measures the performance is defined as the corresponding quotients CPU(k-NN)/CPU(GAM), CPU(k-RNN)/CPU(GAM), CPU(Maximum Depth)/CPU(GAM) and CPU(α-Trimmed Mean)/CPU(GAM). Observe that the GAM method has the smallest CPU time(0.008 sec) while the largest CPU time is for the Maximum Depth classifier. Our

implementation, in terms of the code, is efficient due to three main reasons.

1. Firstly, we vectorise the functions when it was necessary.

2. Secondly, in order to avoid loops in R, we consider using the apply function.

3. Thirdly, we use the replicate function to simulate from the Gaussian Process several times instead of considering a loop.

Note that the computational time can vary according to the platform used. Using Windows 7 and going over a single iteration takes 4.77 minutes, including the functions to calculate the misclassification error, specificity and sensitivity and the necessary functions to save the results and preprocess as functional data objects. However, when we increase to 100 iterations, the total computational time is 8.28 hours per contamination model and a fixed sample size. A similar computational time is obtained when we consider a different method to introduce atypicals in the data and a different same sample size.

Multiple scenarios lead us to big computational times. In fact if we run all the different scenarios in a single computer (without parallelisation) we can expect more than 192 hours for a total of 2 different sample sizes, 4 contamination models and 3 different Gaussian Process. To speed up the computations we use the High Throughput Computing (HTC) system at the University of Manchester by using the EPS Condor cluster. The idea is to consider many independent computations running over many machines, which is a particularly suitable when we consider multiple simulation studies. A more detailed introduction to the Condor cluster can be found here: `http://condor.eps.manchester.ac.uk/`. An advantage of using the EPS Condor cluster is that we can request the number of CPUs in the machine.

Using the EPS Condor cluster there is a reduction in the computational time to 5.48 hours per contamination model and a large sample size. On average the same computational time is obtained when we consider different contamination models and different sample sizes. For the small sample size scenario, the computational time reduces dramatically to 1.54 hours. The results are obtained requesting 4 CPUs and using a Linux architecture. This encourages users to use the EPS Condor cluster to save computational time. An example of the submitted file use to run a single model can be found in Appendix 6.1.3.

Numerical Simulation Results

We compare the performance of the k-NN, the k-RNN, the generalized additive model classifier, the classifier based on maximum depth and the classifier based on the trimmed mean depth and for each contaminated model and each sample size we summarise the misclassification error rate in terms of boxplots and the mean specificity and mean sensitivity along with the standard deviation in a tabular form. For methods that require a value of k to be chosen, the number of neighbours is optimised in each simulation experiment.

To start with, we consider the first model for which data is simulated and not contaminated by atypicals. We also consider three different scenario: a scenario where the data is generated following a Gaussian Process described in Model 1, a scenario where the data is generated simulated a Gaussian Process described in Model 2 and a scenario where the data follows a Gaussian Process with parameters described in Model 3. For all the different scenarios, we use a modest sample size scenario of $n_1 = n_2 = 50$, based on previous simulation studies for functional data, for example Delaigle and Hall (2012), and a bigger sample size of $n_1 = n_2 = 100$. We repeat the simulation experiment 100 times. We present the comparison of the k-NN (KNN), the k-RNN (KRNN), the generalized additive model using the signed depth and the distance to the mode (GAM), the maximum depth classifier (MD) and the α -trimmed to the mean classifier (TM).

Figure 3.23 shows that the proposed method based on the signed depth and the distance to the mode outperforms all its competitors, when the data are not contaminated by atypicals. In second place, the k-RNN method performs better that the k-NN based on depth. This is because the k-RNN based on depth uses more robust rankings of nearest neighbours. In terms of the variability of the misclassification error, we observe that all the proposed methods exhibit roughly the same variability. Among the other depth classifiers, the distance trimmed to the mean method performs better than the maximum depth classifier. In terms of the sensitivity and specificity; we summarise the results of all the models in Table 3.10 and Table 3.11.

Figure 3.24 shows the boxplots of the misclassification error rate considering an Asymmetric Contamination Model described in Section 2.3. We can observe that generalized additive model based on the signed depth and the distance to the mode, outperforms all its competitors. Under this scenario, the k-NN and the k-RNN methods also perform favourably

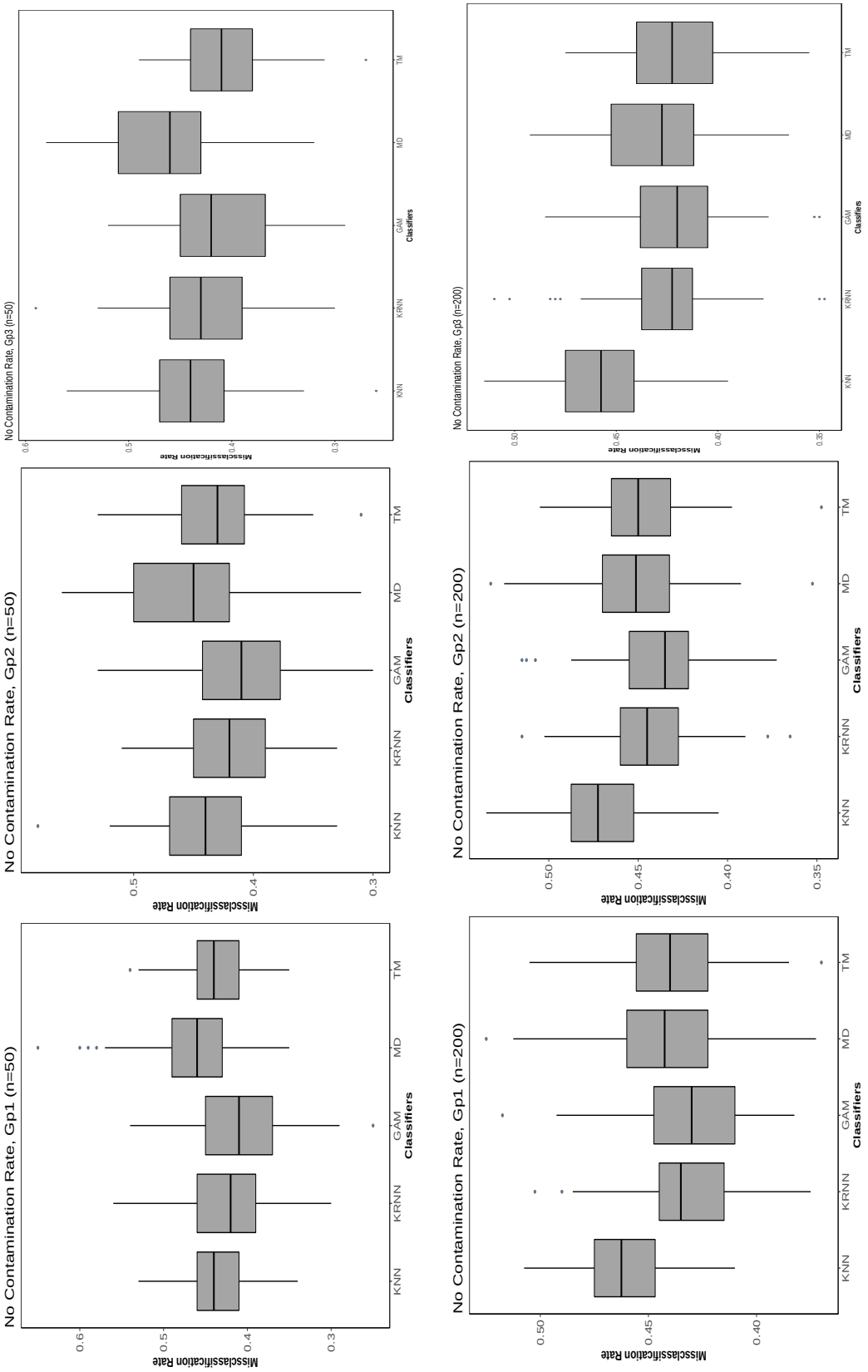


Figure 3.23: Boxplots of missclassification error rates obtained for the Uncontaminated Model. First row: Boxplots of the misclassification error for the same sample size $n_1 = n_2 = 50$ across the different Gaussian Processes. Second row: Boxplots of the missclassification error for a sample size of $n_1 = n_2 = 200$.

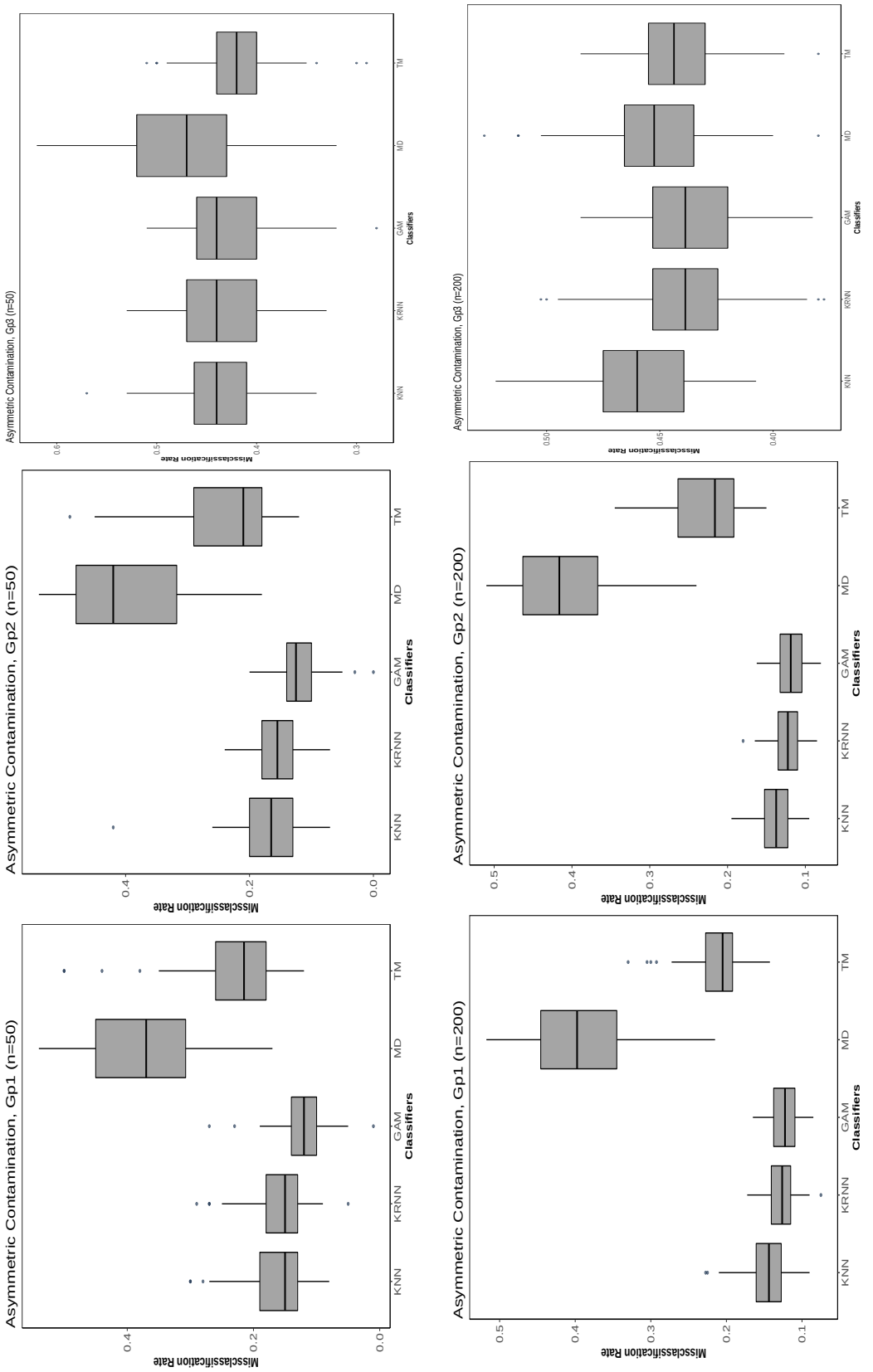


Figure 3.24: Boxplots of misclassification error rates obtained for the Asymmetric Contaminated Model for (First row:) sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$.

with the generalized additive model. However, the maximum depth classifier does not perform as well. This is because when we introduce asymmetric contamination, the depth values change significantly and there is more variation in the data. We can also observe that the misclassification error variability is higher for the maximum depth classifier, while there is small variability for the generalized additive model based on the signed depth and the distance to the mode. We can observe that in second place, the k-RNN method performs as good as the k-NN based on depth. Now, we consider introducing atypicals to the data using the asymmetric contamination model. The sensitivity and specificity for this model is summarised in Table 3.10 and Table 3.11.

Figure 3.25 shows the boxplot of the misclassification error rate considering the peak contamination model. We can observe that all the methods except the k-NN perform favourable. In this scenario, we can observe that the trimmed mean method slightly outperforms the other methods with a small misclassification error rate. This is caused by the type of contamination considered under this scenario. When the contamination occurs in a short time interval, the depth of the contaminated curves is low, and if there is an overlap in the curves and depth, classification procedures based on depth are robust. In terms of the variability of the misclassification error, we can observe that for the first two simulated Gaussian Processes, the variability is small across different classifiers while for the third simulated Gaussian Process, the misclassification error exhibits higher variability.

Finally, we consider the scenario where we introduce the shape contamination. Figure 3.26 shows the results of considering shape contamination model and different samples size. We can observe that all models considered perform better than the k-NN method, which is the worst model due not to being robust to the the contamination model. Moreover, the misclassification error variability seems to decrease using a bigger sample size and remains almost the same for the other methods.

We summarise the results of the sensitivity and specificity in Table 3.10 and Table 3.11, respectively. Table 3.10 summarises the specificity over all the different models divided by Gaussian Process, simulated model and sample size. While Table 3.11 shows the results of considering the specificity for all the different models using the same division.

For the first Gaussian Process, the mean sensitivity rate, averaged across all the simulations, shown in Table 3.10 is higher for generalized additive classifier based on the signed

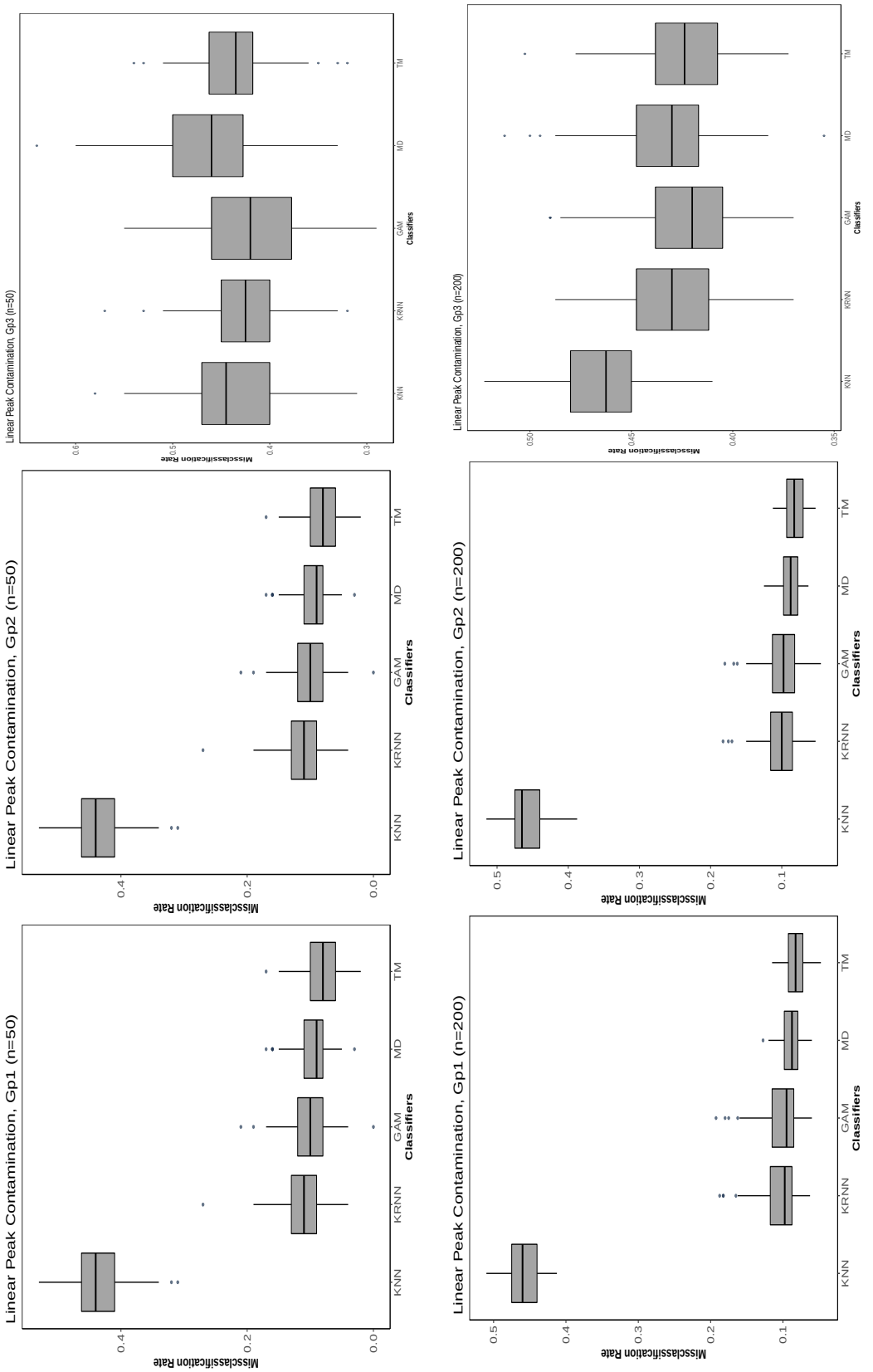


Figure 3.25: Boxplots of missclassification error rates obtained for the Linear Peak Contaminated Model for (First row:) sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$.

depth and the distance to the mode and the same behaviour is exhibited for a larger sample size. In the second Gaussian Process, the mean sensitivity seems to be higher for the generalized additive model, except in some situations when the maximum depth classifier outperforms the other methods.

In terms of the mean specificity rate and Gaussian Process 1, we can observe in Table 3.11 that the highest rates are achieved for the k-RNN method for Models 1 and 2 and the α -trimmed to the mean classifier for Model 3 and Model 4. For the second Gaussian Process, we can observe that there is a higher specificity for the classifier based on two covariates except for some contamination models. For last Gaussian Process, our proposed classifier has a higher specificity across different scenarios.

Overall, we prefer a classifier with high values for all the three measures, sensitivity, specificity and accuracy. When there is no contamination in the data our proposed method based on the signed depth and the distance to the mode outperforms all its competitors and exhibits a high values of specificity and sensitivity. In the Asymmetric Contaminated Model, we also have the same behaviour with the k-NN and the k-RNN methods also performing favourably. In the peak contamination model, methods other than the generalized additive model based on signed depth and signed distance to the mode also have small misclassification error. However, it depends on the type of contamination considered. Finally, the shape contamination model is a difficult scenario where other classifiers like the trimmed mean, perform better than our proposed classifier based on two covariates.

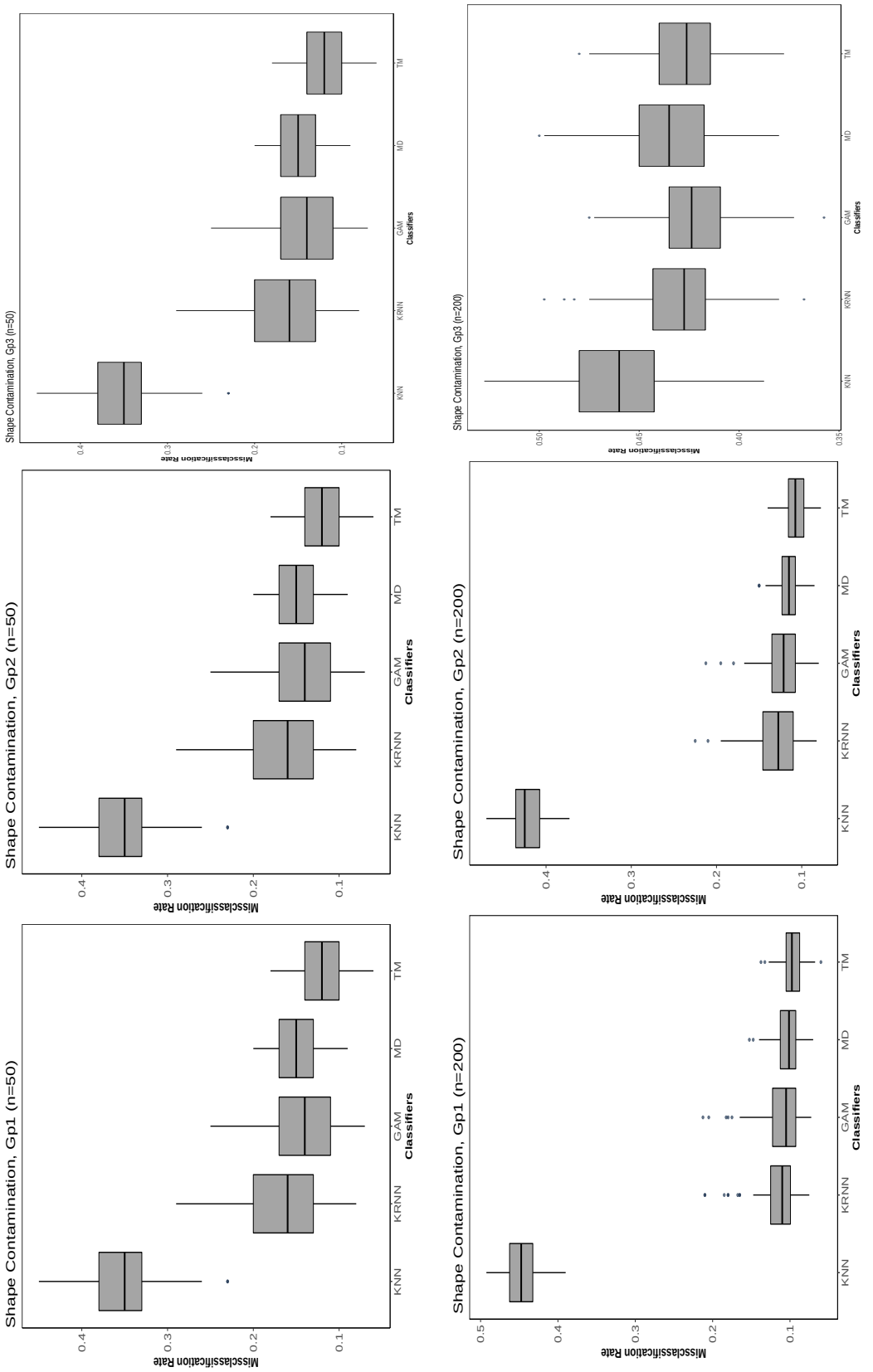


Figure 3.26: Boxplots of misclassification error rates obtained for the Shape Contamination Model for (First row:) same sample size $n_1 = n_2 = 50$ across the different Gaussian Processes and (Second row:) for a sample size of $n_1 = n_2 = 200$.

Table 3.10: Mean sensitivity rates, and their standard deviations in parentheses.

	$n_1 = n_2 = 50$				$n_1 = n_2 = 200$			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
GP1								
k-NN	0.5537 (0.0350)	0.8316 (0.0562)	0.5523 (0.0378)	0.6367(0.0458)	0.5312(0.0186)	0.8685(0.0291)	0.5345(0.0204)	0.5460(0.0217)
k-RNN	0.5697 (0.0467)	0.8103 (0.0490)	0.8868 (0.0516)	0.8193(0.0531)	0.5599(0.0235)	0.8601 (0.0266)	0.9052 (0.0484)	0.8860(0.0398)
GAM	0.5927 (0.0647)	0.8826 (0.0482)	0.9135 (0.0575)	0.8849(0.0611)	0.5681(0.0273)	0.8746 (0.0267)	0.9145 (0.0518)	0.8884(0.0411)
Max Depth	0.5300 (0.0802)	0.6026 (0.0804)	0.9095 (0.0334)	0.8621(0.0240)	0.5569(0.0271)	0.5875 (0.0629)	0.9116 (0.0183)	0.9008(0.0196)
α-TMD	0.5651(0.0407)	0.8073 (0.0918)	0.9205(0.0317)	0.8525 (0.0246)	0.5597(0.0244)	0.8350 (0.0499)	0.9182 (0.0192)	0.8983(0.0181)
GP2								
k-NN	0.5540 (0.0497)	0.8446(0.0613)	0.5550 (0.0505)	0.5541(0.0296)	0.5255 (0.0218)	0.8793 (0.0277)	0.5347 (0.0236)	0.5679(0.0209)
k-RNN	0.5692(0.0408)	0.9058(0.0452)	0.5666 (0.0421)	0.5842(0.0678)	0.5506 (0.0256)	0.8692 (0.0255)	0.9079(0.0481)	0.8777(0.0342)
GAM	0.5891(0.0555)	0.8886(0.0508)	0.5824 (0.0599)	0.6040(0.0663)	0.5639 (0.0290)	0.8842 (0.0228)	0.9138(0.0516)	0.8917(0.0438)
Max Depth	0.5422(0.0657)	0.8044(0.0783)	0.5350 (0.0705)	0.4951(0.1398)	0.5451(0.0355)	0.5738 (0.0585)	0.9100(0.0176)	0.8906(0.0134)
α-TMD	0.5654(0.0417)	0.7178(0.0845)	0.5628 (0.0421)	0.5603(0.0610v	0.5514(0.0259)	0.8034 (0.0744)	0.9146(0.0179)	0.8817(0.0140)
GP3								
k-NN	0.5537(0.0479)	0.5517(0.0418)	0.5504(0.0363)	0.5610(0.0525)	0.5367(0.0235)	0.5363 (0.0218)	0.5301(0.0195)	0.5329(0.0223)
k-RNN	0.5624(0.0468)	0.5584(0.0435)	0.5718(0.0500)	0.5833(0.0515)	0.5696(0.0278)	0.5565 (0.0252)	0.5649(0.0261)	0.5637(0.0221)
GAM	0.5933(0.0551)	0.5728(0.0510)	0.5951(0.0519)	0.6043(0.0578)	0.5822(0.0302)	0.5654 (0.0265)	0.5803(0.0285)	0.5786(0.0240)
Max Depth	0.5392(0.0997)	0.5262(0.0657)	0.5467(0.0829)	0.5551(0.0731)	0.5698(0.0284)	0.5460 (0.0252)	0.5670(0.0277)	0.5650(0.0247)
α-TMD	0.5911(0.0452)	0.5808(0.0405)	0.5988(0.0433)	0.5961(0.0395)	0.5792(0.0246)	0.5582 (0.0192)	0.5761(0.0241)	0.5732(0.0203)

Table 3.11: Mean specificity rates, and their standard deviations in parentheses.

	$n_1 = n_2 = 50$				$n_1 = n_2 = 200$			
	M_1	M_2	M_3	M_4	M_1	M_2	M_3	M_4
GP1								
k-NN	0.5871(0.0634)	0.8448 (0.0522)	0.5891(0.0628)	0.6750(0.0580)	0.5508(0.0288)	0.8421(0.0346)	0.5515(0.0304)	0.5657(0.0298)
k-RNN	0.6053(0.0709)	0.8862 (0.0455)	0.8999(0.0648)	0.8504(0.0723)	0.5792(0.0294)	0.8881(0.0256)	0.8982(0.0571)	0.8925(0.0504)
GAM	0.5897(0.0587)	0.8799 (0.0464)	0.8984(0.0656)	0.8480(0.0658)	0.5738(0.0312)	0.8809(0.0234)	0.8964(0.0609)	0.8992(0.0529)
Max Depth	0.5402(0.0826)	0.6804 (0.1151)	0.9009(0.0365)	0.8426(0.0380)	0.5559(0.0302)	0.6476(0.0802)	0.9122(0.0162)	0.8937(0.0205)
α-TMD	0.5644(0.0392)	0.7481 (0.0573)	0.9188(0.0346)	0.9105(0.0421)	0.5598(0.0248)	0.7562(0.0243)	0.9172(0.0166)	0.9074(0.0187)
GP2								
k-NN	0.5814(0.0673)	0.8204 (0.0747)	0.5809(0.0655)	0.5793(0.0703)	0.5386(0.0349)	0.8477(0.0278)	0.5514(0.0335)	0.5951(0.0311)
k-RNN	0.6057(0.0560)	0.7838 (0.0663)	0.6002(0.0603)	0.6027(0.0581)	0.5661(0.0312)	0.8848(0.0275)	0.8976(0.0553)	0.8694(0.0506)
GAM	0.5915(0.0599)	0.8728 (0.0604)	0.5843(0.0636)	0.5931(0.0550)	0.5618(0.0265)	0.8788(0.0254)	0.8982(0.0577)	0.8741(0.0570)
Max Depth	0.5407(0.0563)	0.3946 (0.2070)	0.5294(0.0749)	0.5105(0.1068)	0.5478(0.0311)	0.6419(0.0828)	0.9148(0.0179)	0.8793(0.0193)
α-TMD	0.5660(0.0432)	0.7996 (0.1227)	0.5631(0.0421)	0.5599(0.0605)	0.5518(0.0263)	0.7500(0.0332)	0.9207(0.0179)	0.9055(0.0199)
GP3								
k-NN	0.5869(0.0687)	0.5851 (0.0803)	0.5859(0.0653)	0.5983(0.0715)	0.5532(0.0317)	0.5521(0.0328)	0.5494(0.0366)	0.5539(0.0402)
k-RNN	0.6000(0.0737)	0.5826 (0.0610)	0.6126(0.0670)	0.6121(0.0676)	0.5841(0.0305)	0.5681(0.0275)	0.5816(0.0304)	0.5824(0.0304)
GAM	0.5917(0.0534)	0.5748 (0.0503)	0.5927(0.0481)	0.5967(0.0533)	0.5782(0.0286)	0.5622(0.0227)	0.5766(0.0294)	0.5773(0.0256)
Max Depth	0.5436(0.0789)	0.5350 (0.1027)	0.5475(0.0736)	0.5537(0.0660)	0.5709(0.0285)	0.5466(0.0615)	0.5705(0.0286)	0.5665(0.0246)
α-TMD	0.5905(0.0430)	0.5809(0.0401)	0.5982(0.0433v	0.5974(0.0405)	0.5796(0.0252)	0.5581(0.0191)	0.5759(0.0236)	0.5734(0.0205)

3.12 Application to some Real data sets

In this section, we apply our methods to three different real datasets previously introduce in Section 2.4 to illustrate the performance of our proposed classification method, based on signed depth and distance to the mode, and its competitors. For binary classification, the datasets we analysed are the orange juice dataset and the NIR gasoline spectra. While for more than two groups, we consider the phoneme dataset with three groups.

As we mentioned before, classifiers are constructed based on a training and a test sample. To demonstrate how our proposed classifier works when a training $\mathcal{T}rain$ and a $\mathcal{T}est$ sample is available, we start analysing the orange juice dataset. For the orange juice dataset, we start by randomly splitting the data into a training and test set. The size of the learning set is set to be equal to $n_{\mathcal{T}rain} = 150$ and the size for the test set is $n_{\mathcal{T}est} = 68$. The number of observations in each group is considered to be equally balanced. We have constructed the 100 samples to be random observations but subject to each group having an equal number of observations. All the competitor methods previously described were applied. For the k-NN and the k-RNN methods the value of k is selected such that it minimises the misclassification error in the training set and then we apply them to the test set. Figure 3.27 shows the boxplot of the misclassification error rate of these 100 samples of the real dataset. For this example, we can observe that the classifier based on the generalised additive model with two covariates outperforms all the different methods. In the second place, the classifier based on maximum depth perform best while the k-NN perform worst.

The second real dataset we consider is the NIR gasoline spectra. As before, for this real dataset, we start splitting the data into a random training and test sample. The size of the training set is set to be equal to $n_{\mathcal{T}rain} = 80$, while the size of the test set is $n_{\mathcal{T}est} = 58$ and the data in both groups is equally balanced. We generated 100 random samples and we apply the five different classifiers. Boxplots of the misclassification error rate over these samples can be seen in Figure 3.28. As before, for the k-NN and the k-RNN methods, the value of k is selected such that it minimises the misclassification error in the training set and then apply to the test set.

From Figure 3.28 we can observe that the proposed classifier based on the generalised additive model considering both covariates outperforms the other methods. Also, we can observe that the methods based on k-NN and k-RNN perform very similarly.

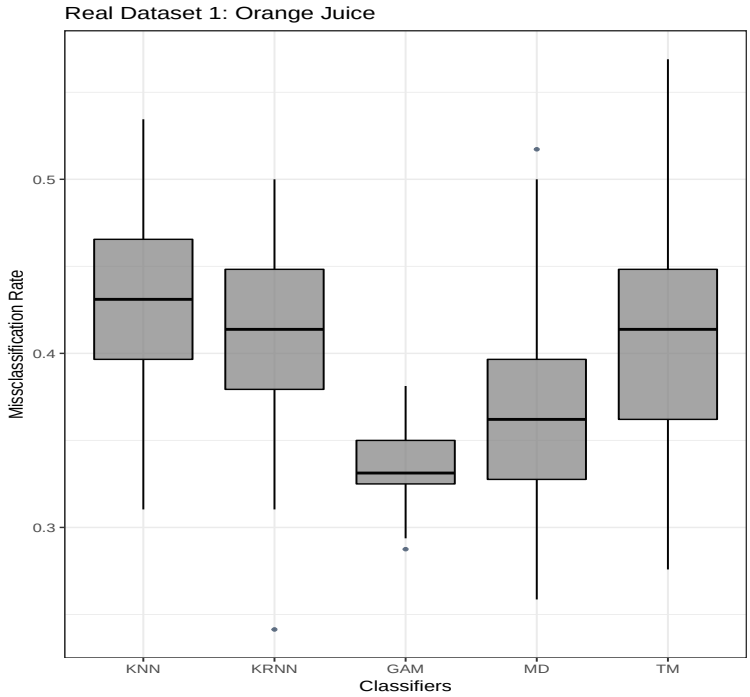


Figure 3.27: Boxplots of the misclassification error rate when applied to the orange juice dataset.

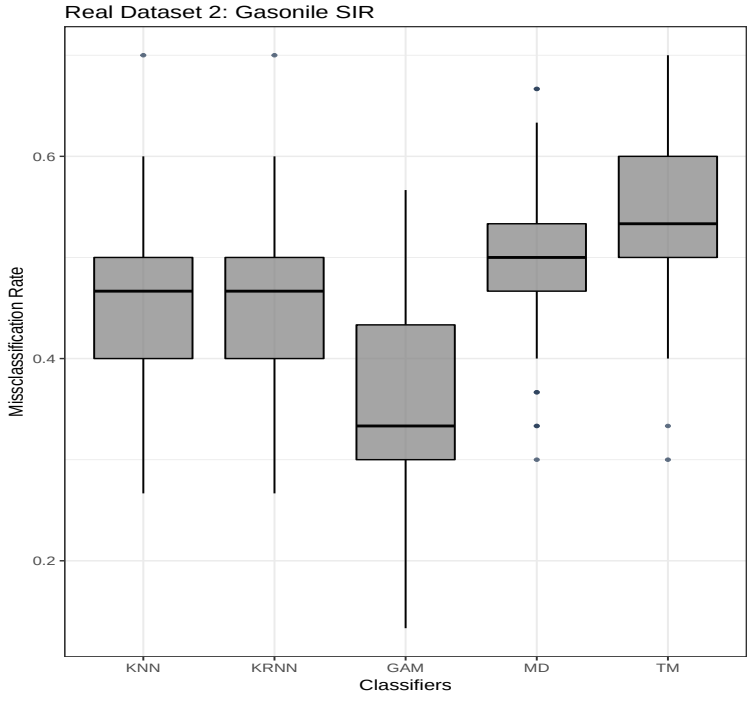
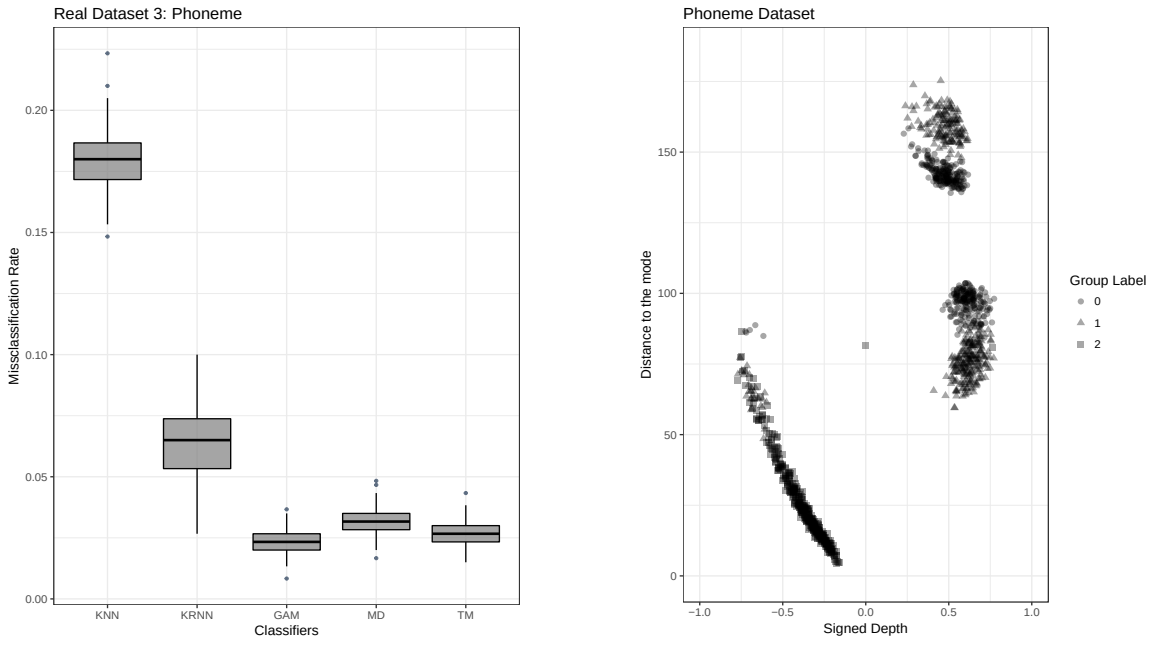


Figure 3.28: Boxplots of the misclassification error rate when applied to the NIR gasoline spectra.

The last real dataset we consider is the phoneme dataset used in Ferraty and Vieu (2003) widely used in Hastie et al. (1995). In this dataset we consider $n_1 = n_2 = n_3 = 400$ curves in

each group. From each speech frame, we compute the log-periodogram, which is the one suitable for speech recognition. Thus the data consists of 1200 log-periodograms of length 256, with know class (phoneme) memberships. For this real dataset, we created a training set of 600 observations based on $n_0 = n_1 = n_2 = 200$ curves in each class, while the remaining curves were considered to be in the test set. Then, we repeated this procedure 100 times. Figure 3.29 shows the boxplots of the misclassification error rate over 100 iterations for the five different classification methods we considered.

An interesting feature we can observe in Figure 3.29b, which is a plot of the two covariates used for classification, is that we can clearly distinguish three main groups formed. Each of these groups belongs to each phoneme class. This explains the good performance of the classifier based on the signed depth and the distance to the mode.



(a) (b)

Figure 3.29: (a:) Boxplots of the missclassification error rate and (b:) signed depth against the distance to the mode, for the phoneme dataset.

We summarise the results of the applying to real datasets in Table 3.12. We can observe from Table 3.12, that in general, the proposed classifier demonstrates a better performance than their maximum depth classifier counterparts as shown in the three real data examples. Also for more than two groups, the proposed classifier not only performed better but also correctly identified clusters in the data.

3.12.1 Point Misclassification

We also computed point misclassification error rates which are shown in Figure 3.30. The results of the point misclassification error rate for the five methods show that the best method with respect to the misclassification error rate is the generalised additive model using the distance to the mode and the signed depth as covariates.

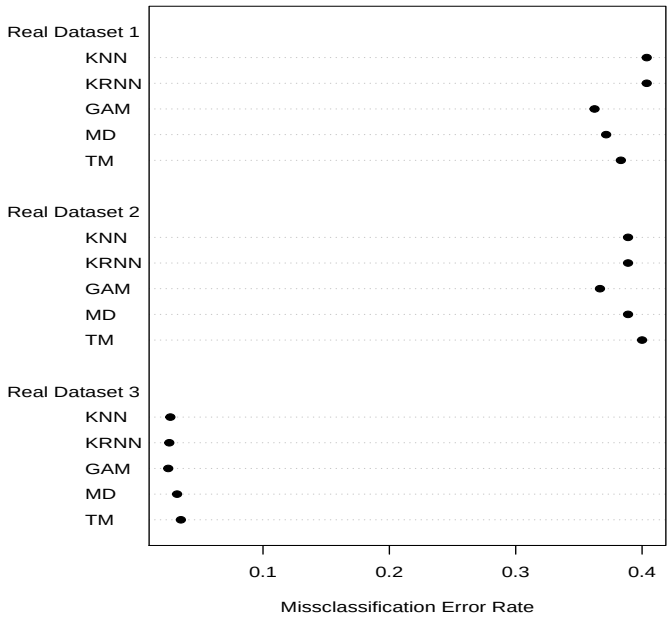


Figure 3.30: Point misclassification error rates for the k-NN, k-RNN, GAM MD and α-TM classifiers applied to real datasets.

To conclude this simulation study, we observe through an extensive simulation method that the proposed method based on the signed depth and the distance to the mode is a robust classifier under different outlier scenarios. Also, by considering the computational time, we observe that it is faster than the other compared methods and does not require any optimisation in term of the parameters, which is required for the k-NN and the k-RNN.

Table 3.12: Mean misclassification error rates and standard deviation for the methods when applied to the real dataset.

	k-NN	k-RNN	GAM	MD	α-TM
Real Dataset_1	0.4264 (0.0471)	0.4110 (0.0505)	0.3368 (0.0187)	0.3634 (0.0562)	0.4141 (0.0590)
Real Dataset_2	0.4540 (0.0768)	0.4540 (0.0768)	0.3463 (0.1040)	0.4960 (0.0702)	0.5367 (0.0678)
Real Dataset_3	0.4306 (0.0123)	0.3135 (0.0161)	0.0236 (0.0055)	0.0315 (0.0059)	0.0267 (0.0052)

3.13 Conclusions

Through this chapter, we introduced a new classifier for functional data based on a generalized additive model using two covariates, the signed to the depth and the distance to the mode. We started by considering the study of the k-RNN classifier using functional depths and introducing the signed distance integral, with respect to the reference curve, to assign a sign to the depth of a function $x_i(\mathbf{t})$. By considering this, we proposed a nearest neighbour type of algorithm based on the signed depth for functional data.

By means of a running simulated example, we model the k-RNN as a regression type approach and we developed bootstrap confidence intervals for the probability that one curve belong to an specific population. By introducing generalized additive models, we showed the performance of the classifier using the distance to the mode and the signed depth. We compare the performance of the proposed classifier against different robust classifiers, the k-NN based on depth, the k-RNN, which uses the ranking of the depth, and classifiers based on a linear combination of a functional depth. Our proposed classifiers, outperform the other classifiers based on robust measures. In all the examples, the proposed classifier performs better than the competitor methods, suggesting that in these examples, there is atypicals contained in the data. In the presence of more atypicals in the data, the proposed method performs better to the other competitors.

Even though we only consider a single functional depth there are other functional depths available and could be useful to consider evaluating the proposed classifier in comparison with these functional depths.

Chapter 4

Dealing with Imbalanced Observations for functional data

4.1 Introduction

Imbalanced sample sizes is a common scenario in the classification problems. When the imbalance in the data is not too extreme, a Bayesian classifier can overcome such a difficulty by incorporating the sample size via a modification of the prior probabilities. However, strongly imbalanced classes often lead to unsatisfactory results with respect to the prediction of new observations, especially for the small class.

In this chapter we use *imbalanced* to refer to the case where the number of observations in the majority class exceeds the number of observations in the minority class. This definition has also been used for this special setting by He and Garcia (2009), Chawla et al. (2002), Chawla et al. (2003) and Chawla et al. (2004). The word unbalanced or imbalanced can be used interchangeably - among which the word imbalanced seems to be more popular.

In the literature, there is little research which has focused on the supervised classification problem with imbalanced samples sizes in either the multivariate case or the functional case. In the multivariate case, we can find the works undertaken by Chawla et al. (2004), García et al. (2012), López et al. (2013), Japkowicz et al. (2000) and Han et al. (2005). The functional case, has even less attention, with only Lin and Chen (2012) who have studied imbalanced classifiers for high dimensional data by incorporating a new strategy in the training phase to account for different sample sizes.

When dealing with imbalanced observations two main problems arise. First, traditional

classifiers might behave undesirably, because it does not take into consideration the distribution of the minority class. The second problem is concerned with boundary observations. Boundary observations are observations near to the borderline and are therefore more susceptible to being misclassified than the ones far from the borderline. Thus, accurate classification of these would reduce the overall error rate.

This chapter proposes different methods to strengthen the borderline minority observations in terms of the functional principal component scores. The proposed methods are based on bootstrapping techniques in terms of the principal component scores, applied to resampling functions in the minority class. However, when this approach is insufficient, we also generate boundary observations from the majority class. We explore different methods to generate new curves by considering a linear combination of the observations in the border and curves closest in depth. We apply our proposed methodology to simulated and real data sets.

The structure of this chapter is as follows. Section 4.2 describes different methods to deal with unequal sample sizes. In Section 4.3 we discuss the methods to evaluate imbalanced classifiers. Section 4.4 deals with methods for imbalanced data for the functional case. Section 4.5 explains what is understood for boundary observations and different methods to generate observations in the border set. Section 4.6 describes the simulation setup and results are shown in Section 4.7. Section 4.8 describe the application to real imbalanced data sets. Finally, conclusions are stated in Section 4.9.

4.2 Methods to dealing with unequal sample sizes

When we are working with imbalanced observations, usually, traditional machine learning algorithms such as k-NN, decision trees and neural networks among others behave undesirably; see Akbani et al. (2004). In this scenario, classifiers can have good accuracy for the majority class but poor accuracy for the minority class. This occurs because the distribution of the imbalanced dataset is not taken into consideration. Along with this, most classification algorithms aim to minimise the error rate and they ignore the unequal distribution of the data and implicitly assume that all misclassification errors cost equally.

To tackle this problem, a number of different solutions were previously proposed for

example, Estabrooks et al. (2004). These solutions include many different forms of resampling techniques and adjusting the misclassification costs of the classes, so misclassification of minority instances is weighted as more important than misclassification of majority instances. However, resampling techniques have been proved to be more efficient and provide a balanced distribution. Balanced distributions can be obtained in different ways; the most common approaches to obtain balanced distributions are:

- Over-sampling of minority class.
- Under-sampling of majority class.

In this section, we briefly describe the most popular methods of under-sampling the majority class and over-sampling the minority class. For an in depth overview of the different techniques see Chawla (2009).

Under-sampling of majority class. This consists of down-sizing the majority class by removing observations, usually at random, until the dataset is balanced. There are several under-sampling methods proposed in recent years and some of the methods to under-sampling the minority class are based on a k-NN approach, the most common being the condensed nearest neighbours and the one side selection approach, Kubat et al. (1997). On one hand, we can assume that many observations of the majority class are redundant and that by removing some of them at random, the data distribution will not change significantly. On the other hand, by removing some observations it is a possibility that we are removing relevant observations from the dataset, losing information. In practice, this technique is often adapted since it is simple and speeds up the computation.

Over-sampling of the minority class. One approach consists of up-sizing the minority class at random. Up-sizing consists of replicating the minority class until the two classes have an equal number of observations. However, by doing this, we increase the risk of over-fitting by biasing the model towards the minority class, since it makes exact copies of the minority class observations, Guo et al. (2008). Some drawbacks of this approach are that the increase in the size of the data increases the computational time and if the original dataset is large, we cannot add new valuable minority observations.

The synthetic minority oversampling technique (SMOTE) is one of the most popular techniques for minority oversampling. It was proposed for multivariate data and introduced

by Lusa et al. (2013). The main idea of the SMOTE is to create a new minority class of observations by interpolating several minority class instances.

The SMOTE uses the k nearest neighbours approach; for a fixed value of k, neighbours from the neighbourhood are randomly chosen and added to the minority class until both classes have the same length. By using this technique, the minority class is over-sampled, and new observations are introduced by interpolating rather than over-sampling with replacement. Let $\mathbf{x}_i \in \mathbb{R}^p$ be a vector of features in the minority class. The SMOTE algorithm searches its k nearest neighbours and one neighbour is randomly selected, say $\mathbf{x}^{'}$. Then for a random number $\delta \in [0,1]$, the SMOTE algorithm generates an artificial sample as

$$\mathbf{x}_{new} = \mathbf{x}_i + (\mathbf{x}^{'} - \mathbf{x}_i) \times \delta. \tag{4.1}$$

Details of the SMOTE algorithm can be found in Chawla et al. (2002). Some other derivations of the SMOTE are SMOTEboost, Chawla et al. (2003); RAMOBoost, Chen et al. (2010); Kernel Based SMOTE, Mathew et al. (2015); and borderline SMOTE, Han et al. (2005). In this chapter, we focus on observations near the boundary, borrowing some concepts from the borderline SMOTE.

The borderline SMOTE is a method in which only the boundary observations of the minority class are over-sampled. It considers a fixed set of k nearest neighbours and over-samples or strengthens the observations that are near the boundary. Depending upon the amount of oversampling required, neighbours from the k nearest neighbours are randomly chosen. The borderline SMOTE calculates the $k = 5$ nearest neighbours of the same class for every minority example and then select some examples at random according to an oversampling rate. After that, new synthetic observations are generated along the line between the minority example and its selected nearest neighbours according to equation (4.1). The borderline SMOTE only over-samples or strengthens the boundary minority examples, Han et al. (2005). A drawback of the borderline SMOTE is that when the number of minority class samples is particularly smaller than that of the majority class, then most of the minority class samples are regarded as noise. Thus, few synthetic samples are generated which improve the accuracy of the method, leading to little overall improvement.

4.3 Methods to evaluate imbalanced classifiers

Evaluating the classifiers of an imbalanced dataset plays an important role. In fact, the accuracy (or misclassification error rate) is not a reliable metric. For instance, consider an extremely imbalanced dataset with $n_0 = 10$ and $n_1 = 500$ observations in each class, respectively. Even when the classifier classifies all of the observations to group one, completely misspecifying group zero, the accuracy is 0.98 which is high because there are many more majority examples than minority examples. Under these circumstances, accuracy does not reflect reliable prediction for the minority class. Thus, more reasonable evaluation metrics are needed. In this section, we start by briefly reviewing the methods to evaluate imbalanced classifiers.

In the imbalanced case, we can group the evaluation metrics, according to Ferri et al. (2009) and Caruana and Niculescu-Mizil (2004), into two main groups. These are:

1. Threshold metrics.

2. Ranking methods and metrics.

The threshold metrics consider the overall performance of the algorithm on all the classes in the dataset. These can have multiple or a single class focus. When the threshold metrics have only a single class focus, the most common performance measures are the accuracy and the misclassification rate. Because the varying degree of importance of the different classes is not considered, the threshold metrics for a single class focus do not perform well in the imbalanced situation, unless the class ratio is taken into consideration (Caruana and Niculescu-Mizil, 2004). To handle this, different pairs of distinct measures are considered, often referred to as a multiple class focus. Examples of measures with a multiple class focus are the sensitivity/specificity, precision/recall, Geometric mean (G-mean), F-measure, Generalized Index of Balanced Accuracy (IBA), Macro-Averaged Accuracy, Mean-Class-Weighted Accuracy, Optimised Precision and Adjusted Geometric Mean, among others.

Ranking methods and metrics are considered based on the fact that cost or other information related to the classification is not known. For this purpose, it more useful to use evaluation methods that enable visualisation or summarise the performance over the range of the classifier. The receiver operating characteristics (ROC) and the area under the curve (AUC) are examples of ranking metrics for such purposes. Most of the criteria for the performance of the classifiers are based on the confusion matrix.

The confusion matrix, also known as the error matrix, is a two by two table that visualises the performance of the classifier in supervised classification. Each column represents the instances in a predicted class and each row represents the instances in an actual class (or vice versa). The name, confusion matrix, arises from the fact that it makes it easy to see if the system is confusing two classes. The shape of the confusion matrix is a square matrix containing all the possible classes in both the horizontal and vertical directions and lists the classes along the top of a table as the predicted outputs, and then down the left-hand side as the targets.

Suppose we have a binary classification problem with two populations: Π_0 and Π_1. If we count the number of times that the predicted output was class 0 when truly belongs to class 0, and similarly for class 1. We can construct a square matrix. Table 4.1 shows an example of the associated two by two confusion matrix. The row position represents the actual class label, while the column position represents the predicted class label. True Positive (TP) and True Negative (TN) denote the number of positive and negative examples that are classified correctly, while False Negative (FN) and False Positive (FP) denote the number of misclassified positive and negative examples, respectively.

Table 4.1: Confusion Matrix or Error Matrix for two classes 0 and 1.

	Predicted Class 0	Predicted Class 1
True Class 0	TP	FN
True Class 1	FP	TN

To evaluate the performance of our classifier we will consider threshold and ranking metrics. More specifically, we will consider the F- measure, G-mean, the Generalized Index of Balanced Accuracy (IBA) and the area under the curve (AUC) as metrics to evaluate the performance of the imbalanced classifiers. These various metrics attempt to get a better summary of how good the classifier is when the data are imbalanced. We now briefly describe the evaluation metrics.

The F - measure

The F-measure is a popular evaluation metric for imbalanced problems (Japkowicz, 2013). Also known as the weighted harmonic mean of precision and recall, it is a combination of recall and precision rates. The precision is the percentage of positive predictions made by the classifier that are correct:

$$\text{Precision} = \frac{TP}{TP+FP},$$

and the recall is the percentage of true positives that are correctly detected by the classifier.

$$\text{Recall} = \frac{TP}{TP+FN}.$$

The recall rate explains the prediction accuracy among minority class instances, while precision is the rate of correct predictions among all instances predicted to belong to the minority class. It indicates how many of the positive predictions are correct (He and Garcia, 2009). The F-measure is defined as the harmonic mean of recall and precision, which tends to be closer to the smaller of the two. In other words,

$$\text{F-measure} = \frac{(1+\beta^2) \times \text{Precision} \times \text{Recall}}{\beta^2 \times \text{Precision} + \text{Recall}}.$$

Note that the F-measure depends on $\beta \in [0, \infty)$ which is used to control the influence of recall and precision separately. When $\beta = 0$, then F-measure reduces to precision and conversely when $\beta \to \infty$ then the F-measure approaches the recall rate. When $\beta = 1$, the F-measure is suggested to integrate these two measures as an average. The F-measure is high when both recall and precision are high and can be adjusted by changing the value of β. In practice, the value of β is usually set to 1, i.e.,

$$\text{F-measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}.$$

For interpretation, a high F-measure value ensures that both recall and precision are reasonably high.

G-Mean

When the performance of both classes is concerned, both the True Positive Rate (TP) and the True Negative Rate (TN) are expected to be high simultaneously. A metric that captures both rates is the G-mean proposed in Kubat et al. (1997). The G-mean indicates the balance between the classification performance in both classes. This metric takes into account the sensitivity and the specificity (which is the accuracy of the negative examples). The G-mean is defined as:

$$\text{G-mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}},$$

where Sensitivity = Recall and the Specificity is defined as

$$\text{Specificity} = 1 - \frac{\text{FP}}{\text{TN + FP}}.$$

This metric tries to maximise accuracy to balance both classes at the same time. It is an evaluation measure that allows us to simultaneously maximise the accuracy in positive and negative examples with a good trade-off. Examples of this metric being used to evaluate the classifier on an imbalanced dataset can be found in Chawla (2009).

Generalized Index of Balanced Accuracy (IBA)

The Generalized Index of Balanced Accuracy was proposed by García et al. (2012). This measure is defined in terms of the performance metric M and a weight parameter α which reduces the influence on the result of the particular metric M. The IBA is defined as

$$\text{IBA}_\alpha(\mathsf{M}) = (1 - \alpha \times \text{Dom}) \times \mathsf{M},$$

where the dominance, $\text{Dom} \in [-1, 1]$, is defined as

$$\text{Dom} = TP - TN.$$

Note that the dominance is used to estimate the relationship between the two rates TP and TN. The closer the dominance is to 0, the more balanced both individual class accuracies are. The IBA metric quantifies a certain trade-off between a measure of overall accuracy and an index of how balanced the two-class accuracies are. Unlike most performance metrics, the IBA metric not only takes care of the overall accuracy but also intends to favour classifiers with better results on the minority class (usually the most important class); see López et al. (2013).

From the definition of IBA, note that if $\alpha = 0$, the quantity $\text{IBA}_\alpha(\mathsf{M})$ simplifies to the performance metric and, in practice, one should select a value of α depending on the metric used. When we compare the performance of the classifier in the imbalanced case we set the performance metric $\mathsf{M} = \text{G-Mean}^2$ and $\alpha = 0.1$.

The area under the ROC curve (AUC)

The AUC provides a single measure of a classifier's performance by summarising the performance of the classifier into a single metric. The AUC is another performance metric that can be used to evaluate the performance of an imbalanced classifier (Bradley, 1997). It is based on the Receiver Operating Characteristic (ROC) curve, which can be considered as one of the most popular metrics to measure the performance of imbalanced classifiers (Hanley and McNeil, 1982). The ROC curve plots the True Positive (TP) rate against the False Positive (FP) rate. A single run of a classifier produces a single point on the ROC plot, and a perfect classifier would be a point at the point (0,1) (100% true positives, 0% false positives), while the worst classifier that got everything wrong would be at the point (1,0). The closer to the top-left-hand corner the result of a classifier is, the better the classifier has performed and any classifier that lies on the diagonal line from (0,0) to (1,1) behaves exactly at random; we call this a random classifier. To generate a ROC curve, each point is generated by moving the decision boundary. That is, points nearer to the left in ROC space are the result of requiring a higher threshold for classifying an instance as a positive instance.

The reason why the ROC curve is suited to evaluated the performance of a classifier for imbalanced observations, according to Japkowicz (2013), is because the performance on each class is decomposed into a pair of distinct measures. For example, for the sensitivity and the specificity, the ROC curve gives an evaluation of what may happen in these diverse situations. The AUC provides a single scalar measure of a classifier's performance by summarising the performance of the classifier into a single metric. It is computed by obtaining the area of the graphic. In other words,

$$\text{AUC} = \frac{1 + TP - FP}{2},$$

where TP is the true positive rate, which is the percentage of positive instances correctly classified, FP is the false positive rate which is the percentage of negative instances misclassified. For our purpose, when dealing with imbalanced data we will consider the AUC as a metric to evaluate the performance of the imbalanced classifiers.

Note that all the above evaluation metrics can reasonably evaluate the performance of a classifier for imbalanced datasets because their formula is relative to the minority class.

4.4 Methods to deal with imbalanced data for functional data

In this section, we start by commenting on the Bayesian approach to deal with imbalanced observations.

4.4.1 A Bayesian Approach

Bayes' theorem can be used to estimate the posterior probability that the observed signed depth belongs to one of the classes. The posterior probability that the signed depth $sdp(x_i(\mathbf{t}))$ belongs to population Π_0 is given by

$$\frac{\bar{f}_0(sdp(x_i(\mathbf{t})))\pi_0}{\bar{f}_0(sdp(x_i(\mathbf{t})))\pi_0 + \bar{f}_1(sdp(x_i(\mathbf{t})))\pi_1}, \tag{4.2}$$

where $\bar{f}_1(sdp(x_i(\mathbf{t}))$ represents the density of the signed depth observations in the population Π_1, π_1 is the prior probability that the observed signed depth is in population Π_1, $\bar{f}_0(sdp(x_i(\mathbf{t}))$ represents the density of the signed depth observations in the population Π_0 and π_0 is the prior probability that the observed signed depth is in population Π_0. The densities $\bar{f}_1$ and $\bar{f}_0$ are estimated using a kernel density estimation approach.

In the case when the data is imbalanced, a Bayesian classifier is a flexible classifier that allows us to incorporate information about the sample size of the observations in each group by modifying the prior probabilities. To investigate the effect of the prior probabilities when we consider a Bayesian classifier, we study how sensitive the evaluation metrics are to the effect of the prior probabilities. We consider different values for the number of observations in the minority class and it is important to note that the prior probabilities are estimated by considering the number of curves in each group. In other words,

$$\hat{\pi}_1 = \frac{n_1}{n_0 + n_1} \quad \text{and} \quad \hat{\pi}_0 = \frac{n_0}{n_1 + n_0}, \tag{4.3}$$

where n_1 is the number of observations in the minority class and n_0 is the number of observations in the majority class. Using the Bayesian classifier, we assign a new observed signed depth $sdp(x_0(\mathbf{t}))$ to the class with the highest posterior probability. A Bayesian classifier for imbalanced data classifies a new observation to population Π_0 if

$$\bar{f}_0(sdp(x_0(\mathbf{t})))\pi_0 > \bar{f}_1(sdp(x_0(\mathbf{t})))\pi_1.$$

To demonstrate of the performance of the Bayesian classifier with a different set of priors, we start by first considering different scenarios. We briefly summarise the scenarios in Table 4.2. For each of the different scenarios, we consider the misclassification rate, the area under the curve, the generalised index of balanced accuracy with $\mathsf{M} = \text{G-Mean}^2$ and $\alpha = 0.1$, the G-Mean and the F-Measure with $\beta = 1$.

The data we consider in this example is generated using the same model introduced in Section 3.3. We consider a Gaussian process $\mathcal{GP}(m(t), K(s,t))$ where $m(t) = 80 * (1-t) * t^2$, $K(s,t) = 0.1 * \exp(-100 * (s-t)^2)$ and the number of observations in each class varies according to each scenario.

Table 4.2: Different scenarios for the Bayesian approach.

	$\mathcal{S}_1$	$\mathcal{S}_2$	$\mathcal{S}_3$	$\mathcal{S}_4$	$\mathcal{S}_5$
n_0	25	50	75	100	150
n_1	200	200	200	200	200

To measure the performance of the Bayesian classifier when the data is imbalanced we implement the Bayesian classifier for the different scenarios. Figure 4.1(A) shows the kernel density estimation for scenario $\mathcal{S}_2$, with $n_0 = 50$ and $n_1 = 200$ observations in their respective classes. The value of the bandwidth was calculated using cross validation. For the observations in both groups, and using the priors probabilities as in equation (4.3), we compute the posterior probabilities

$$\mathbb{P}(sdp(x_i(\mathbf{t})) \in y_0 \mid sdp(x_i(\mathbf{t}))) = \frac{\bar{f}_0(sdp(x_i(\mathbf{t})))\pi_0}{\bar{f}_0(sdp(x_i(\mathbf{t})))\pi_0 + \bar{f}_1(sdp(x_i(\mathbf{t})))\pi_1}, \tag{4.4}$$

and

$$\mathbb{P}(sdp(x_i(\mathbf{t})) \in y_1 \mid sdp(x_i(\mathbf{t}))) = \frac{\bar{f}_1(sdp(x_i(\mathbf{t})))\pi_1}{\bar{f}_0(sdp(x_i(\mathbf{t})))\pi_0 + \bar{f}_1(sdp(x_i(\mathbf{t})))\pi_1}. \tag{4.5}$$

The posterior probabilities against the signed depth are shown in Figure 4.1(B) and Figure 4.1(C), respectively.

To investigate the effect of the prior probabilities when dealing with imbalanced observations, we perform a small experimental study. The study consists of varying the number of

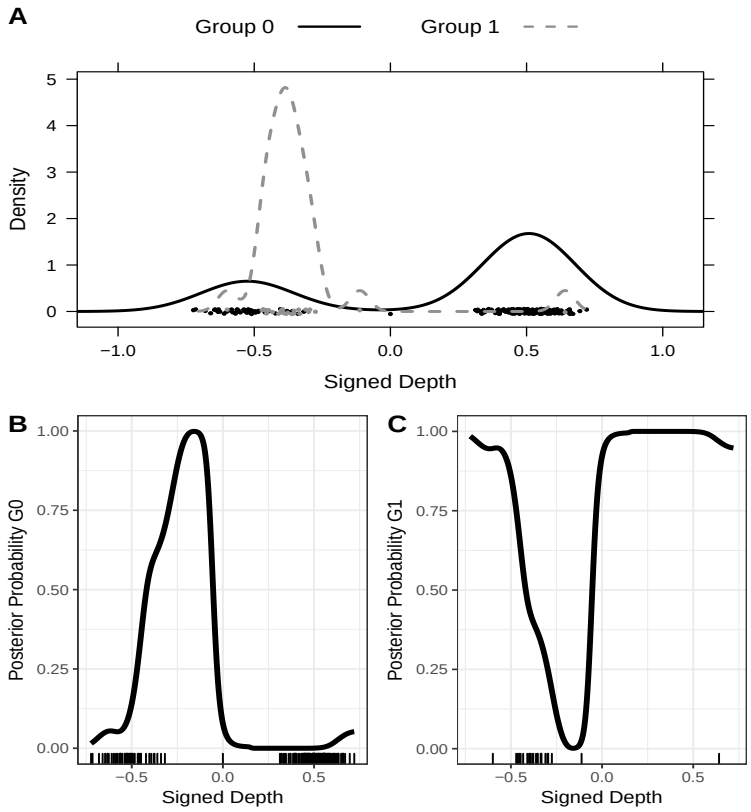


Figure 4.1: (A) Kernel density estimation for the observations in the minority and majority class with the posterior probabilities for the the signed depths to belong to (B) the minority class (B) and (C) the majority class.

observations in each group by modifying the prior probabilities. The resulting calculations of the error rate, the area under the curve, IBA, G-mean and F-measure can be seen in Table 4.3.

Table 4.3: Several performance measures for different scenarios considered.

Scenario	Measure of Performance				
	Error Rate	AUC	$IBA_{\alpha=0.1}$	G-Mean	F-Measure
$\mathcal{S}_1$	0.066	0.823	1.526	0.286	0.963
$\mathcal{S}_2$	0.108	0.813	2.651	0.401	0.933
$\mathcal{S}_3$	0.138	0.830	3.411	0.506	0.904
$\mathcal{S}_4$	0.114	0.889	3.795	0.622	0.912
$\mathcal{S}_5$	0.103	0.901	2.797	0.779	0.907

We can see that there is an increase in the G-mean for all the scenarios. While from scenario 3 onwards, both the AUC and the G-mean increases as the data become more balanced. For all the considered scenarios, the F-measure is greater than 0.90, indicating a good performance of the classifier on the minority class.

Another important observation from Table 4.3 is that the value of the index of balanced accuracy, were high for scenarios $\mathcal{S}_3$ and $\mathcal{S}_4$. However, the IBA function does not take care of the overall accuracy only, but tends to favour classifiers with better results on the minority class. Results of this small simulation study show the flexibility of the information we can incorporate about the minority class through prior probabilities.

4.4.2 Oversampling techniques for functional data

When dealing with functional data, a better approach to using only prior probabilities is to oversample the distribution of the principal component scores in the minority group. By doing this, we create as many new observations we require from the same distribution.

To oversample the distribution of the principal component scores, we start by estimating the functional component scores in the minority group and use the the fact that observed functions can be decomposed into functional principal components and their uncorrelated principal component scores.

Let $X_i(t)$ for $t \in [a,b]$, be a collection of random functions; the Karhunen-Lòeve decomposition can be written as,

$$X_i(t) = \bar{X}(t) + \sum_{p=1}^{\infty} \Xi_{ip}\psi_p(t) \quad \text{for} \quad i = 1,\ldots,n, \tag{4.6}$$

where $\bar{X}(t)$ is the mean function, Ξ_{ip} is the p^{th} principal component score of the i^{th} observation and $\psi_p(t)$ is the p^{th} functional principal component eigenfunction. Using the Karhunen-Lòeve expansion for random functions, a collection of observed curves at times $t_1,\ldots,t_M$ can be approximated by

$$x_i^{(p)}(t_j) \approx \bar{x}(t_j) + \sum_{p'=1}^{p} \hat{\Xi}_{ip'}\hat{\psi}_{p'}(t_j) \quad \text{for} \quad i = 1,\ldots,n, \quad j = 1,\ldots,M, \tag{4.7}$$

where $\bar{x}(t_j)$ is the sample mean, $\hat{\Xi}_{ip}$ is the estimated p^{th} principal component score of the i^{th} observation and $\hat{\psi}_p(t)$ is the p^{th} estimated functional principal component eigenfunction, for a fixed dimension p. Hereafter, the couple $\left(\hat{\Xi}_{ip}, \hat{\psi}_p(t_j)\right)$ will form the foundation of our subsequent methodology. The approaches we will consider in the methodology are the following:

1. Sampling with replacement from the principal component scores.

2. Sampling with replacement from the principal component scores and adding Gaussian noise in the curves.

3. Smoothed bootstrapping and randomly generation from kernel densities with Gaussian noise in the curves.

4. Synthetic Minority Oversampling Technique (SMOTE) and adding Gaussian noise in the curves.

5. A refinement of the data-driven (Taylor - Thomson) algorithm, allowing us to generate as many simulated observations we require, with Gaussian noise in the reconstructed curves.

We start by summarising the methods.

Bootstrap

In multivariate data analysis, the use of bootstrap techniques has been popularized since the work of Efron (1979), Efron (1982) and Efron and Tibshirani (1993). For functional data, there is little work on bootstrapping. For bootstrapping the functional principal component scores, Shang (2015) provides a study of the distributional properties of sample eigenvalues, while Hall and Vial (2006) use the bootstrap approach to assess the finite dimensionality for functional data.

Selection of principal components has been a recurring topic in PCA, and no consensus has emerged yet. It is an important model selection problem in most practical problems and is not an easy task. A popular method is the scree plot by Cattell (1966). The work of Li et al. (2013) discusses selecting the number of principal components based on an AIC criterion. While more sophisticated methods, in a theoretical and computational sense, are those based on cross-validation Wold (1978), a more recent approach is the minimum description length (MDL) method proposed by Poskitt and Sengarapillai (2013), where each principal component is counted as one parameter, and is considered as a parameter problem. The process to generate bootstrap samples from the principal component scores is as follows.

Given a collection of observed functions $\left\{x_i(t_j)\right\}$ for $i = 1,\ldots,n$ observed on a grid of points $t_1,\ldots,t_M$,

*B*1. Hold the sample mean $\bar{x}(t_j)$ and the principal component functions $\hat{\psi}_p(t_j)$ for $p' = 1,\ldots,p$, and $j = 1,\ldots,M$, fixed at their realised values for the observations in the minority class.

*B*2. For $i = 1,\ldots,n$, generate bootstrap replication $\hat{\Xi}^*_{ip}$ by sampling with replacement from the rows of $\hat{\Xi}_{ip}$.

*B*3. Construct the bootstrap sample $\left\{x^*_i(t_j)\right\}$ where the bootstrap realisation $x^*_i(t_j)$ for $i = 1,\ldots,n$, and $j = 1,\ldots,M$, is constructed by replacing the $\hat{\Xi}_{ip}$ by $\hat{\Xi}^*_{ip}$ and reconstructing the data functions

$$x^{(p)}_i(t_j) = \bar{x}(t_j) + \sum_{p'=1}^{p} \hat{\Xi}^*_{ip'}\hat{\psi}_{p'}(t_j) \quad \text{for} \quad i = 1,\ldots,n, \quad j = 1,\ldots,M.$$

For approach 2, we also have the additional step.

*B*4. Finally, add some Gaussian noise to the reconstructed $x^{(p)}_i(t_j)$ curves.

Smoothed Bootstrap

The smoothed bootstrap is an extension of the standard bootstrap method using kernel densities. It is based on the fact that obtaining a smoothed bootstrap sample from the data is equivalent to sampling from a kernel density estimate of the distribution and it is a modification to the bootstrap procedure to avoid samples with these properties. For an introduction to the smoothed bootstrap see Silverman and Young (1987).

Suppose that $\mathbf{x}_1,\ldots,\mathbf{x}_n \in \mathbb{R}^p$ are n observations drawn from an unknown p-variate density f. Let $\hat{f}$ be the estimated p-variate density from the observations. New realisations $\mathbf{Y}^*$ from f can be generate as follows,

1. Choose I uniformly with replacement from $\{1,\ldots,n\}$.
2. Generate ε to have probability density function $N(\mathbf{0}, I)$.
3. Set $\mathbf{Y}^* = \bar{\mathbf{X}} + (\mathbf{X}_I - \bar{\mathbf{X}} + h\varepsilon)/(1+h^2)^{1/2}$.

where $\bar{\mathbf{X}}$ is the mean vector, I is the identity matrix and the smoothing parameter h is set to be 0.5. Density estimates are explained in more detail in Silverman and Young (1987).

We can simulate different realisations from the principal component scores in the minority class using the smoothed bootstrap approach, outlined below.

*SB*1. For the observations in the minority class, hold the sample mean $\bar{x}(t_j)$ and the estimated principal component functions $\hat{\psi}_p(t_j)$ for $p' = 1,\ldots,p$, and $j = 1,\ldots,M$, fixed at their realised values.

*SB*2. For $i = 1,\ldots,n$, generate new realisations $\hat{\Xi}^*_{ip}$ by estimating a product kernel density $\hat{\Xi}_{ip}$ using Silverman's rule of thumb, Chacón et al. (2011) for bandwidth selection, and taking i.i.d. random draws from the estimated density. The random generation from the product kernel is done by drawing with replacement the rows of $\hat{\Xi}_{ip}$ and adding Gaussian noise from univariate kernels, parametrised by the corresponding bandwidth parameter of the sampled values.

*SB*3. Construct a bootstrap sample $\left\{x^*_i(t_j)\right\}$ where the bootstrap realisation $x^*_i(t_j)$ for $i = 1,\ldots,n$, and $j = 1,\ldots,M$, is constructed as

$$x^{(p)}_i(t_j) = \bar{x}(t_j) + \sum_{p'=1}^{p} \hat{\Xi}^*_{ip'}\hat{\psi}_{p'}(t_j) \quad \text{for} \quad i = 1,\ldots,n, \quad j = 1,\ldots,M.$$

where $\hat{\Xi}_{ip}$ is replaced by $\hat{\Xi}^*_{ip}$ from the estimated density with Gaussian noise. Add some Gaussian noise to the reconstructed $x^{(p)}_i(t_j)$ curves.

The density can be estimated with respect to different kernel functions. When we consider the multivariate Gaussian kernel, the bandwidth parameter matrix H is a covariance matrix, and the Gaussian noise is simulated from a multivariate normal distribution centred at the data points and parametrised by corresponding bandwidth matrix H.

SMOTE for Functional Data

The Synthetic Minority Oversampling Technique (SMOTE), proposed by Chawla et al. (2002), is one of the most popular approaches for dealing with imbalanced data. It is used mainly for multivariate data but we consider a modified version for functional data.

The main idea of the SMOTE is to create a new minority class of observations by interpolating several minority class instances in the training set, Lusa et al. (2013). Using this technique the minority class is over-sampled and new observations are introduced by interpolating rather than over-sampling with replacement. The SMOTE uses the k nearest neighbours approach and, for a fixed value of k, neighbours from the k-neighbourhood are randomly chosen and added to the minority class until both classes have the same size. This

approach effectively forces the decision region of the minority class to become more general (Chawla et al., 2002). The detailed algorithm for the SMOTE can be found in Chawla et al. (2002).

An alternative to the SMOTE is the borderline SMOTE. It was introduced by Han et al. (2005) and is a method in which only the observations near the boundary in the minority class are over-sampled. The borderline SMOTE also considers a set of nearest neighbours usually $k = 5$, with the idea being to over-sample or strengthen the observations that are near the boundary (Wang et al., 2015).

We start describing the SMOTE to increase the number of observations in the minority class for functional observations.

SMOTE 1. For the observations in the minority class, hold the sample mean $\bar{x}(t_j)$ and the principal component functions $\hat{\psi}_p(t_j)$ for $p' = 1,\ldots,p$, and $j = 1,\ldots,M$, fixed at their realised values.

SMOTE 2. For all the observations in the minority class generate a SMOTE realisations $\hat{\Xi}^S_{ip}$ by synthetically generating new observations from $\hat{\Xi}_{ip}$.

SMOTE 3. Construct a SMOTE sample $\left\{x^S_i(t_j)\right\}$ where the SMOTE realisation $x^S_i(t_j)$ for $i = 1,\ldots,n$, and $j = 1,\ldots,M$, is constructed by replacing the $\hat{\Xi}_{ip}$ by $\hat{\Xi}^S_{ip}$ and reconstructing the data functions as

$$x^{(p)}_i(t_j) = \bar{x}(t_j) + \sum_{p'=1}^{p} \hat{\Xi}^S_{ip'}\hat{\psi}_{p'}(t_j) \quad \text{for} \quad i = 1,\ldots,n, \quad j = 1,\ldots,M.$$

SMOTE 4. Finally, add some Gaussian noise to the reconstructed $x^{(p)}_i(t_j)$ curves.

A refinement of the Taylor - Thomson algorithm

The Taylor - Thompson algorithm proposed by Taylor and Thompson (1986) does not require estimation of the underlying density as the smoothed bootstrap approach. Instead, it borrows concepts from density estimation. The method uses the k nearest neighbours of a randomly selected data row ($k-1$ nearest neighbours and the original row itself) and it was originally developed for multivariate data for its computational simplicity (Demirtas and Hedeker, 2011). The value of k plays the role of the smoothing parameter as in the density estimation approach.

The original algorithm considers a matrix of dimension $n \times p$ and generates a pseudo-random matrix of simulated draws whose dimension is the same as the original data. However, the original Taylor - Thomson algorithm has the drawback that it is not possible to generate more observations from the original data but only from the original dimension itself. In other words, if the original data is of dimension $n \times p$, the Taylor - Thomson algorithm generates the same same number of observations n with same dimension p. To overcome this limitation, we consider a refinement of the Taylor - Thomson algorithm to simulate from the principal component scores. We briefly describe the algorithm below.

Let $\hat{\Xi}_{ip}$ be the matrix of size n with the p estimated principal component scores. In other words each $\{\xi_1, \ldots, \xi_n\}$ is a vector of length p denoting the i^{th} row of the matrix of the principal components scores. The steps of the refinement of the Taylor - Thomson algorithm are as follows:

*TT*1. For the observations in the minority class, hold the sample mean $\bar{x}(t_j)$, and the principal component functions $\hat{\psi}_p(t_j)$ for $p' = 1, \ldots, p$, and $j = 1, \ldots, M$, fixed at their realised values.

*TT*2. For $i = 1, \ldots, n$, generate new realisations $\hat{\Xi}_{ip}^{TT}$ in the following way:

1. Select a row from ξ_i from $\hat{\Xi}_{ip}$ at random and determine its k nearest neighbours using the Euclidean distance.
2. Centre the vectors ξ_i around the sample mean $\bar{\xi} = \dfrac{1}{k}\sum_{i=1}^{k} \xi_i$ and define $\xi_i^T = \xi_i - \bar{\xi}$.
3. Generate a random sample $u_1, \ldots, u_k$ from the uniform distribution
$$U\Big(\frac{1}{k} - 3(k-1)/k^2, \frac{1}{k} + 3(k-1)/k^2\Big).$$
4. Generate a realization forming the linear combination
$$\xi_i^{TT} = \sum_{i=1}^{k} u_i \xi_i^T + \bar{\xi}.$$
5. Add the generated realisation to the original matrix and call it $\hat{\Xi}_{ip}^{MTT}$.
6. Sample another row from the matrix $\hat{\Xi}_{ip}^{MTT}$ with replacement, and repeat the above steps until the desired number of simulations is reached. The output will be a new matrix $\hat{\Xi}_{ip}^{MTT}$ of simulated draws whose dimension is the number of simulations $n_{sim} \times p$.

*TT*3. Construct the modified Taylor-Thompson sample $\left\{x_i^{MTT}(t_j)\right\}$ where the realisation $x_i^{MTT}(t_j)$ for $i = 1,\ldots,n$, and $j = 1,\ldots,M$, is constructed by replacing the $\hat{\Xi}_{ip}$ by $\hat{\Xi}_{ip}^{MTT}$ and reconstructing the data functions

$$x_i^{(p)}(t_j) = \bar{x}(t_j) + \sum_{p'=1}^{p} \hat{\Xi}_{ip'}^{MTT} \hat{\Psi}_{p'}(t_j) \quad \text{for} \quad i = 1,\ldots,n, \quad j = 1,\ldots,M.$$

*TT*4. Finally add some Gaussian noise to the reconstructed $x_i^{(p)}(t_j)$ curves.

In the original Taylor - Thompson algorithm, the first and the second empirical moments are compatible with the original data. The proof can be found in Taylor and Thompson (1986). Note in the case where $k = 1$, the procedure is simply a classical bootstrap. As the value of k increases, we arrive at something similar to the smoothed bootstrap approach, where a small amount of (normally distributed) zero-centred Gaussian noise is added on to each re-sampled observation. An iterative procedure to select the value of k can be found in Demirtas and Hedeker (2011).

4.4.3 Performance Comparison

To compare the performance of the previously discussed methods and to evaluate the performance of generating more observations from the minority class, we implement a simulation study. We start by considering the second scenario $\mathcal{S}_2$, where the minority class n_0 consist of 50 observations and the majority class consists of 200 observations. For scenario $\mathcal{S}_2$, we compare the area under the curve before and after oversampling the data. Figure 4.2 shows the original data (before preprocessing the data) in terms of the signed depth and distance to the mode.

To compare the performance of our different methods of oversampling the data to the simulated dataset, we implemented all algorithms in R building on the **smotefamily** and **kernelboot** package for the SMOTE and the smoothed bootstrap approach. For the original data, the area under the curve applying the k-RNN classifier based on depth corresponds to 0.803. Results of the implementation can be seen in Figure 4.3 for the bootstrap method, while Figure 4.4 shows the implementation for the remaining methods.

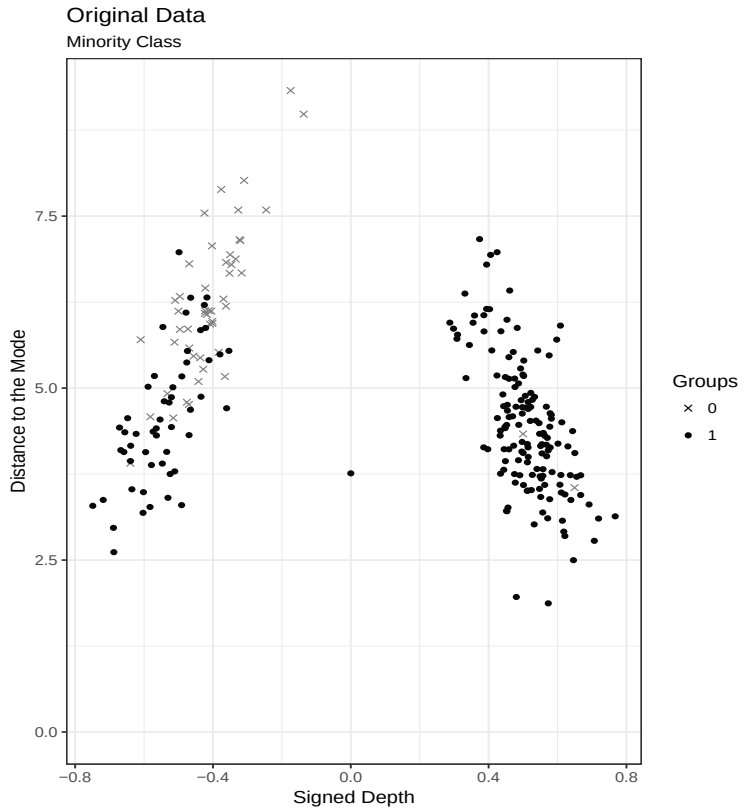


Figure 4.2: A scatterplot of signed depth against the distance to the mode for the simulated data. The number of observations in each group are $n_0 = 50$ and $n_1 = 200$.

We use $\mathcal{M}_1$ to refer to the sampling with replacement from the principal component scores approach, $\mathcal{M}_2$ to be the sampling with replacement with Guassian noise approach, $\mathcal{M}_3$ the smoothed bootstrap approach, $\mathcal{M}_4$ the SMOTE approach and $\mathcal{M}_5$ the refinement of the data-driven Taylor-Thomson algorithm approach. For all the different approaches we consider the same number of principal components $(p = 2)$ to reconstruct the curves. The variance of the added Gaussian noise to the curves is the total variation minus the variation explained by the two principal component scores, except in the first approach.

The corresponding values of the area under the curve are $AUC_{\mathcal{M}_1} = 0.928$, $AUC_{\mathcal{M}_2} = 0.927$, $AUC_{\mathcal{M}_3} = 0.908$, $AUC_{\mathcal{M}_4} = 0.921$ and $AUC_{\mathcal{M}_5} = 0.914$. Compared with the Bayesian rule incorporating prior probabilities shown in Table 4.3, all the approaches achieve a higher AUC value.

The results show that even there is even an improvement in terms of the area under the curve for the bootstrap method which generates many ties in the distance to the mode and signed depth. We observe the same behaviour in terms of the AUC in the second approach. In this case, the bootstrap oversamples too many observations and there is an overlap in the

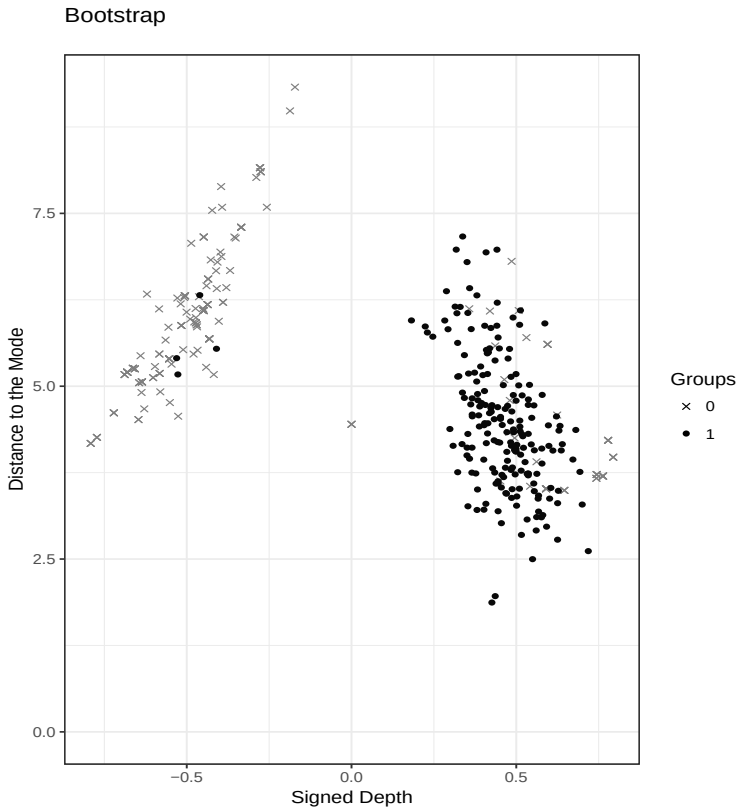


Figure 4.3: A scatterplot of signed depth against the distance to the mode after the original data is oversampled by bootstrapping the functional principal component scores. After oversampling the data both groups contain the same number of observations $n_0 = n_1 = 200$.

data.

The remaining approaches $\mathcal{M}_3$, $\mathcal{M}_4$ and $\mathcal{M}_5$ show an improvement in terms of the area under the curve and a more realistic method to sample the curves efficiently. For conciseness, we focus on only two approaches for the generation of data from the minority class; we consider the smoothed Bootstrap and Random Generation from Kernel Densities with Gaussian noise in the curves and the Synthetic Minority Oversampling Technique (SMOTE) with Gaussian noise in the curves.

4.5 Boundary observations

To achieve better prediction, we are interested in developing new methods that can be applied to observations at the boundary where the misclassification tends to occur. These are observations near to the borderline and are therefore more apt to be misclassified than the ones far from the borderline, and thus more important for classification. This set of

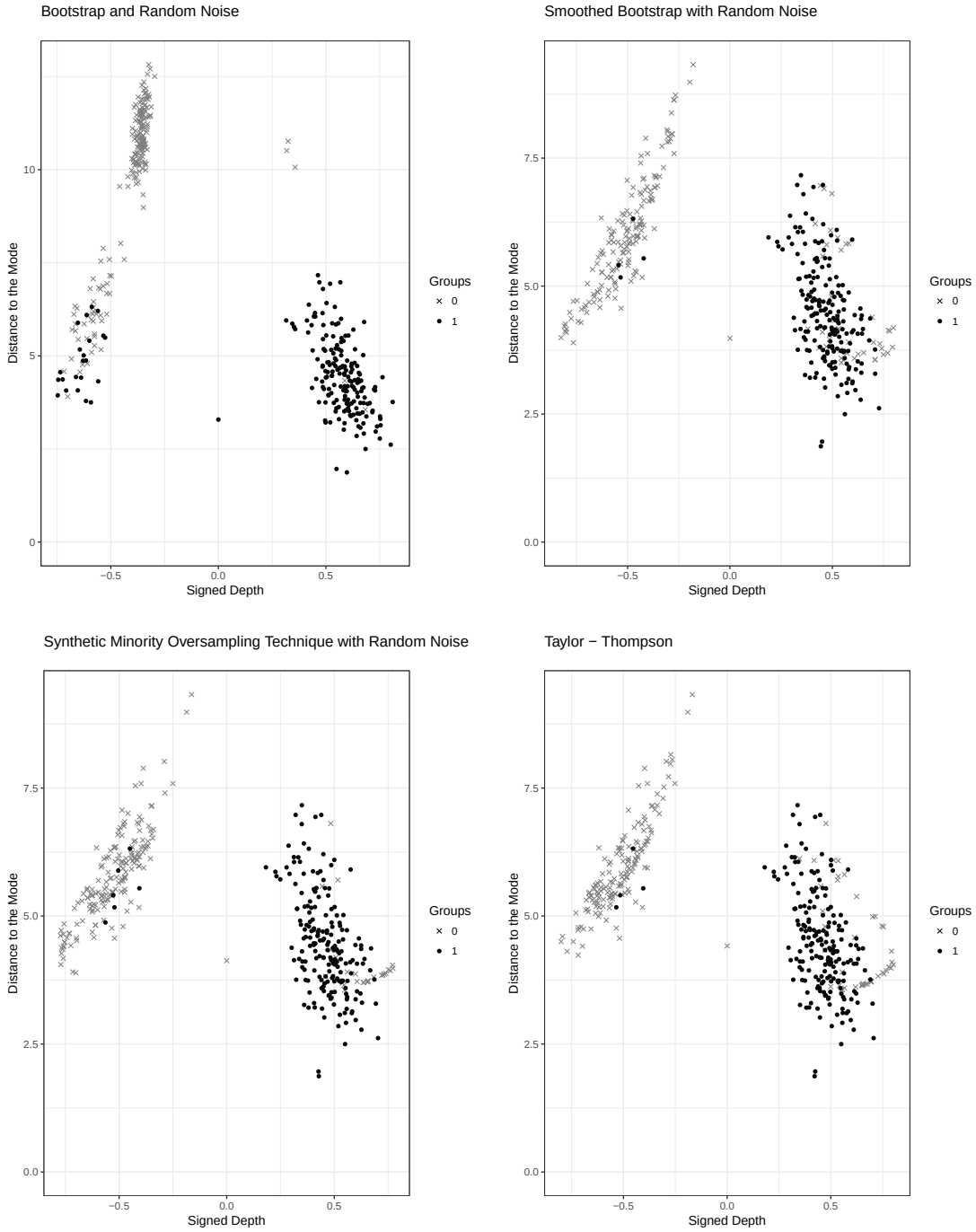


Figure 4.4: Scatterplots of the signed depth against the distance to the mode using (top-left:) Bootstrap with Gaussian noise, (top-right:) smoothed bootstrap with Gaussian noise, (bottom-left:) SMOTE with p=2 and Gaussian noise, and (bottom-right:) the Taylor-Thompson algorithm.

observations can be defined in terms of a distance function or in terms of the discriminant rule. We focus on the latter. We propose a new method to generate new observations from the minority class by oversampling the observations in the minority class and generating new curves by using a linear combination of the curves at the borderline and later we extend this method by sampling from the majority group as well. Note that this proposed method differs from the previous methods when the whole minority class is over-sampled using the principal component scores.

4.5.1 Running Example

To illustrate the borderline in terms of the discrimination rule we consider the example introduced in Section 3.3. We consider a Gaussian process $\mathcal{GP}(m(t), K(s,t))$ with mean $m(t) = 80 * (1-t) * t^2$ and $K(s,t) = 0.1 * \exp(-100 * (s-t)^2)$. We let the number of observations in each different population be imbalanced with $n_0 = 50$ and $n_1 = 250$ curves sample from Π_0 and Π_1 respectively, such that population Π_0 is the minority class.

Usually when we validate our k-RNN classifier, we have a training and test dataset. In the balanced case, a common method for choosing the appropriate number of neighbours k, is to use R-fold cross-validation. R-fold cross-validation splits the data into R sets arbitrarily, which may introduce additional biases. We use this for our proposed method which allows us to oversample the minority class and train our classifier in the original data, and then we test our classifier on the fold.

For the imbalanced case, a common approach is to perform stratified cross-validation where instead of sampling completely at random, the original class distribution is preserved in each fold; see Japkowicz (2013) and Forman and Scholz (2010) for a discussion about this approach. However, a drawback of this method is that it introduces overfitting in the data. A proposed variation of the R-fold cross-validation in the imbalanced case works as follows:

Step 1. Oversample the minority class using any of the above methods previously discussed.

Step 2. Split the original data (not oversampled) into R equally size subsets (or folds).

Step 3. Perform a R-fold cross-validation as described in Section 3.3.1.

Step 4. In each of the folds, train the classifier in the original data and test the classifier on the fold.

Step 5. Repeat this process R times and take the average of the metric.

The results of applying this method can be seen in Figure 4.5. In this case we vary the number of neighbours with $k \in \{1,\ldots,45\}$. A minimum can be seen from this cross-validation curve at $k = 9$ neighbours. For this purpose we consider the overall misclassification rate but other different metrics as F-Measure, G-mean and AUC can be used. A further investigation considering different metrics showed that the optimum number of neighbours is the same when considering overall accuracy, overall error rate and AUC.

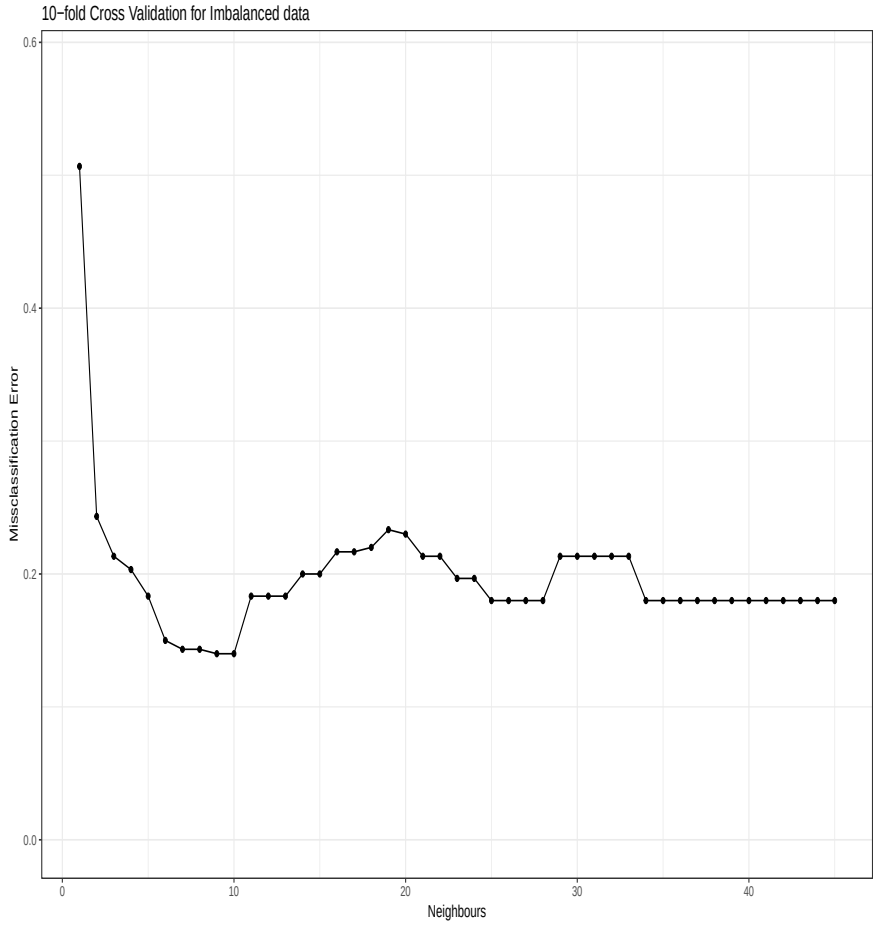


Figure 4.5: The misclassification error rated plotted against the number of neighbours for imbalanced data.

4.5.2 Border Set

Once we determine our value of k using 10-fold cross validation, we focus on the observations at the boundary. The first step is to find the borderline observations from the minority class and form a new set of observations called the *border set*.

Observations that belong to the border set are more apt to be misclassified than the ones far from the borderline. We can define *border bands*, which are determined by a fixed parameter τ and a decision threshold d_τ. When the data is balanced, the decision threshold, $d_\tau \in [0,1]$, is calculated based on equal prior probabilities, and usually the decision threshold is set to be 0.5. To form our border bands, we consider a more general version; the value of τ is added and subtracted to d_τ forming a closed interval of the form $[d_\tau - \tau, d_\tau + \tau]$.

For the first inspection of our running example, we consider a fixed value of k determined by the 10-fold cross validation previously discussed. The number of neighbours that minimised the misclassification error for the imbalanced data was $k = 9$. To start determining the border band we consider a fixed value of $\tau = 0.15$ and a decision threshold $d_\tau = 0.5$, i.e., we consider the observations from the minority group that are in the closed interval $[0.35, 0.65]$. Implementations of the border bands can be seen in Figure 4.6. Note that this value was determined for demonstration of the border bands, but later we choose the border bands in terms of the classification rule.

Another important observation is that the area under the curve changes with respect to the number of principal components used to reconstruct the curves. For this reason, we consider a fixed value of $p = 4$, explaining 98% of the variability in the data. Also, when we consider the number of observations in the border bands, for each observation in the border bands the variability explained by the principal component scores varies.

After we determine the observations in the border band, it is possible to visualise these observations in terms of the signed depth and distance to the mode. Figure 4.7 demonstrates this for observations in the majority group, minority group and the border set group as defined in Section 4.5.2. Note that the observations in the border region belong to the observations in the minority class which overlap with or are close to the observations in the majority class.

We have seen that when we want to determine the classification of the observations in the border set, it is necessary to define a threshold τ and a decision threshold $d_\tau = 0.5$. Therefore, we investigate how the classification performance varies with respect to the upper and lower bounds. For this purpose, we start varying the upper and lower bound by different distances from the decision threshold $d_\tau = 0.5$, with different values of $\tau = 0, 0.025, 0.050, 0.075, 0.100, 0.125, 0.150, 0.175, 0.200, 0.225$ and 0.250. Note that a value greater than these will result in including more than 70% of the data from the minority class. The results of the implementation can be seen in Table 4.4.

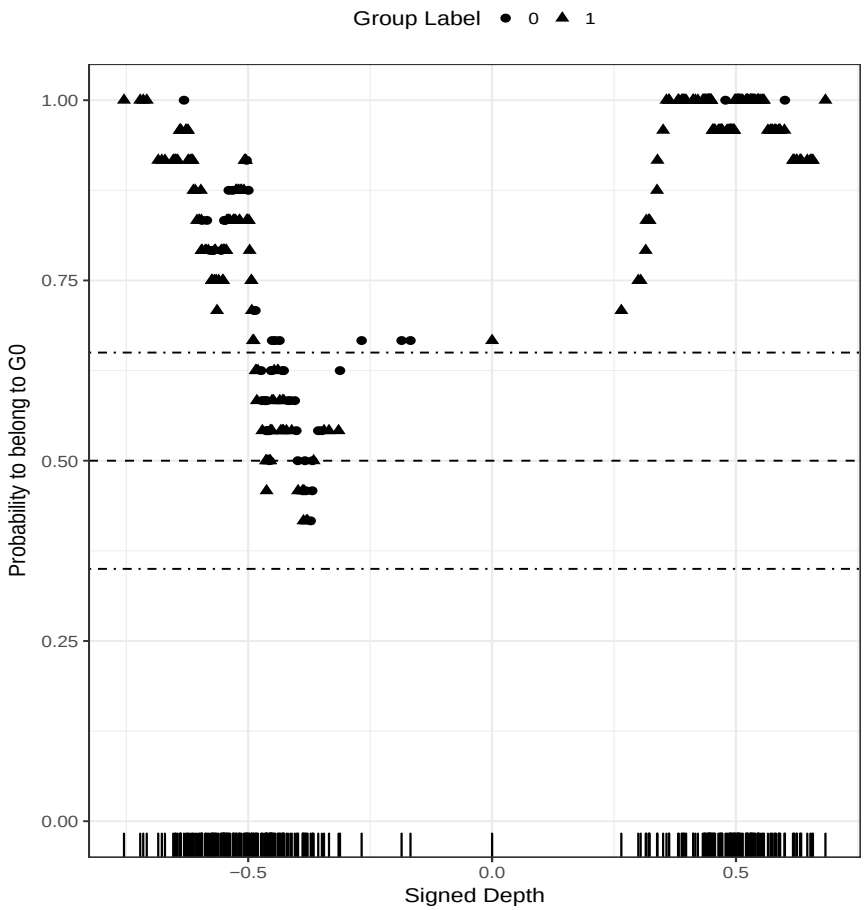


Figure 4.6: Border bands for the imbalanced data. The border observations are the observations in the minority class that fall in the closed interval of $[0.35, 0.65]$ with respect to the probability that it belongs to the population Π_0.

Table 4.4 shows how the number of observations in the border set, the area under the curve, the precision, recall F-measure and G-mean vary with respect to the threshold parameter. Results shows an improvement with respect to the AUC against the Bayesian rule incorporating prior probabilities and moreover, with respect to the proposed sampling approaches. The value of $\tau = 0$ corresponds to considering all of the observations from the minority class as observations on the borderline. Note that a maximum in terms of the area under the curve can be achieve when we consider a threshold of $\tau = 0.075$ and $\tau = 0.200$ with an AUC of 0.922.

4.5.3 Generating observations in the border set

One of the drawback of the previously proposed approach is the computation time involving generating new observations in the minority class and finding the observations in the border

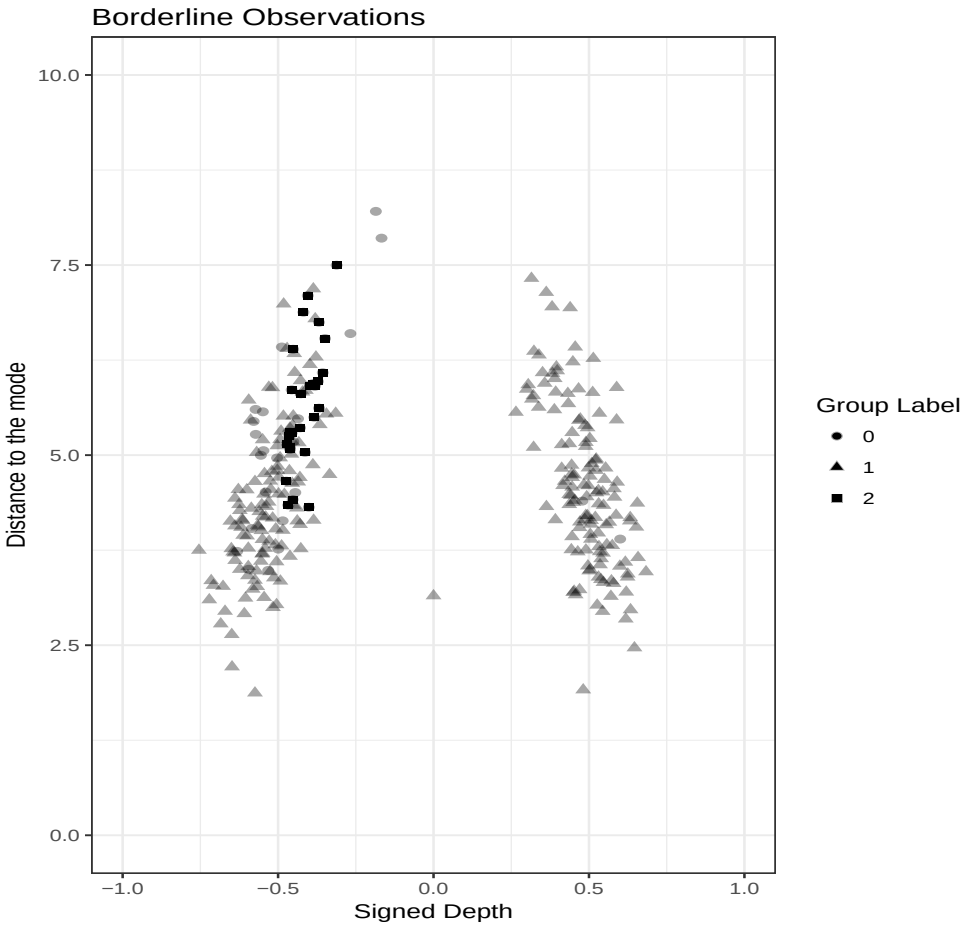


Figure 4.7: Observations in the border region for the simulated dataset. Observations in group 1 correspond to the majority class. Observations in group 0 correspond to those in the minority class and observations in group 2 correspond to those in the border set.

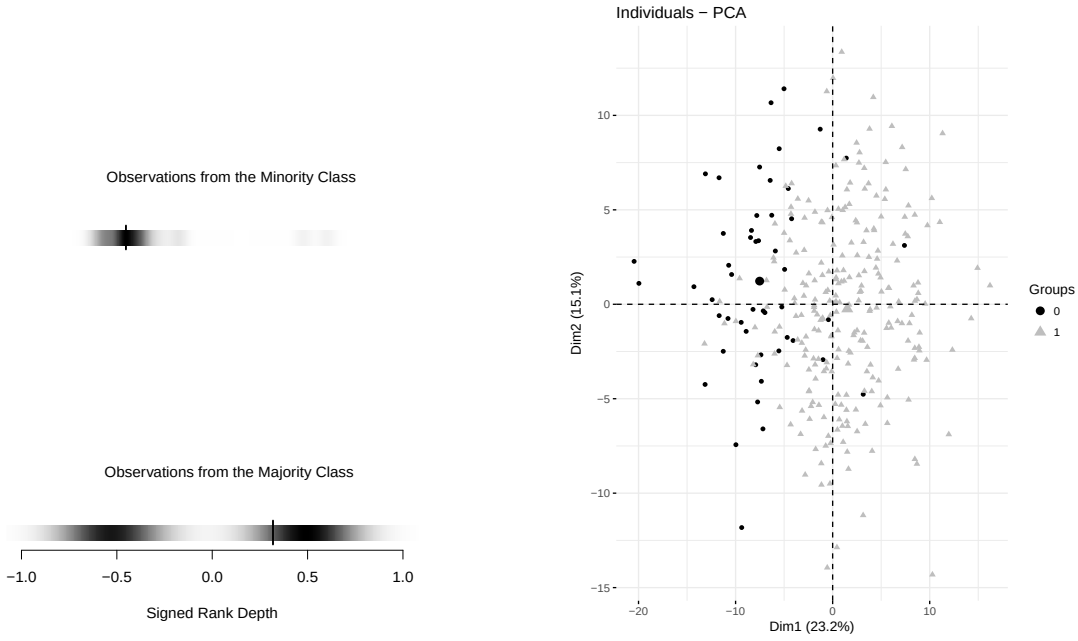
set. Another drawback of the previous approach is when that when we oversample the data some of the observations in the majority class are misclassified. To avoid the computational time and to investigate if we can avoid observations in the majority class to be misclassified, we propose a new method to generate observations in the border set which involves using k nearest neighbours and captures the *shape* of the curves in the minority group. Capturing the shape of the curves in the minority group can be achieved by introducing the *difference curve*.

To start explaining the proposed method, we consider the running example previously discussed in Section 4.5.1. Once we have obtained the data and estimated the signed depth as explained in the Algorithm 1 in Chapter 3, we obtain the ranked observations. To visualise the observations from each group we estimate the density strip for the observations in both groups. Figure 4.8(a) represents the density strip estimation for both observations in the

Table 4.4: Observations in the border set, the area under the curve, precision/recall, F-measure and G-mean for different values of the threshold parameter.

τ	Observations in the border set	AUC	Precision	Recall	F-measure	G-Mean
0	4	0.895	0.981	0.814	0.876	0.398
0.025	4	0.920	0.991	0.880	0.932	0.411
0.050	13	0.920	0.991	0.880	0.932	0.411
0.075	13	0.922	0.991	0.884	0.934	0.412
0.100	21	0.920	0.991	0.880	0.932	0.411
0.125	27	0.866	0.985	0.792	0.878	0.385
0.150	27	0.860	0.984	0.780	0.870	0.382
0.175	33	0.920	0.990	0.880	0.932	0.411
0.200	33	0.922	0.991	0.884	0.934	0.412
0.225	35	0.920	0.990	0.880	0.932	0.411
0.250	35	0.914	0.990	0.868	0.925	0.408

minority and majority class, while Figure 4.8(b) represents the first two functional principal component scores of the observations in both groups.



(a) (b)

Figure 4.8: (a) Density strip and (b) first two principal components for observations in the majority and minority classes.

The proposed algorithm consists of four different steps. The first step is to select beforehand a value of the k-nearest neighbours by a method of cross-validation, as we explained before. Once we determine the number of neighbours, the next step is to determine the observations in the border set. To determine the observations in the border we consider the

following approach.

Suppose that a functional dataset $\left\{\left(x_i(t_j), y_i\right),\ i = 1,\ldots,N,\ j = 1,\ldots,M\right\}$ is available consisting of two different populations and containing an imbalanced number of observations in each group. We represent the majority class by the set of curves

$$\mathcal{M}ajority = \left\{x_1^{Maj}(t_j),\ldots,x_{n_{\mathcal{M}ajority}}^{Maj}(t_j)\right\},$$

and the minority class by the set of curves

$$\mathcal{M}inority = \left\{x_1^{Min}(t_j),\ldots,x_{n_{\mathcal{M}inority}}^{Min}(t_j)\right\},$$

where $n_{\mathcal{M}ajority}$ represents the number of curves in the majority set and $n_{\mathcal{M}inority}$ represents the number of curves in the minority set, respectively.

Step 1. For every curve $x_j^{Min}(t_j) \in \mathcal{M}inority$ class, calculate the k nearest neighbours of the signed depth for the observations in the training set, where the value of k is even. The number of majority examples among the k nearest neighbours is denoted by $k^{'}$ with $0 \leq k^{'} \leq k$.

Step 2. If $k/2 \leq k^{'} < k$, i.e., the number of majority nearest neighbours is larger than the number of the minority ones, then $x_j^{Min}(t_j) \in \mathcal{M}inority$ is considered to be misclassified and we put it into the border set. If $0 \leq k^{'} < k/2$, then the observation is safe and we do not consider it in the next steps. Finally, if $k^{'} = k$, then all the k nearest neighbours of $x_j^{Min}(t_j) \in \mathcal{M}inority$ are from the minority class thus we do not consider it to be in the border set.

Step 3. The border set is a subset of the minority class and we denote the border set by

$$\mathcal{B}\text{order - set} = \left\{x_1^{Bor}(t_j),\ldots,x_{n_{\mathcal{B}order}}^{Bor}(t_j)\right\},$$

where $n_{\mathcal{B}order}$ denotes the number of curves in the border set, with the restriction $0 \leq n_{\mathcal{B}order} \leq n_{\mathcal{M}inority}$. For each curve in the border set we calculate its k nearest neighbours in signed depth from the minority set.

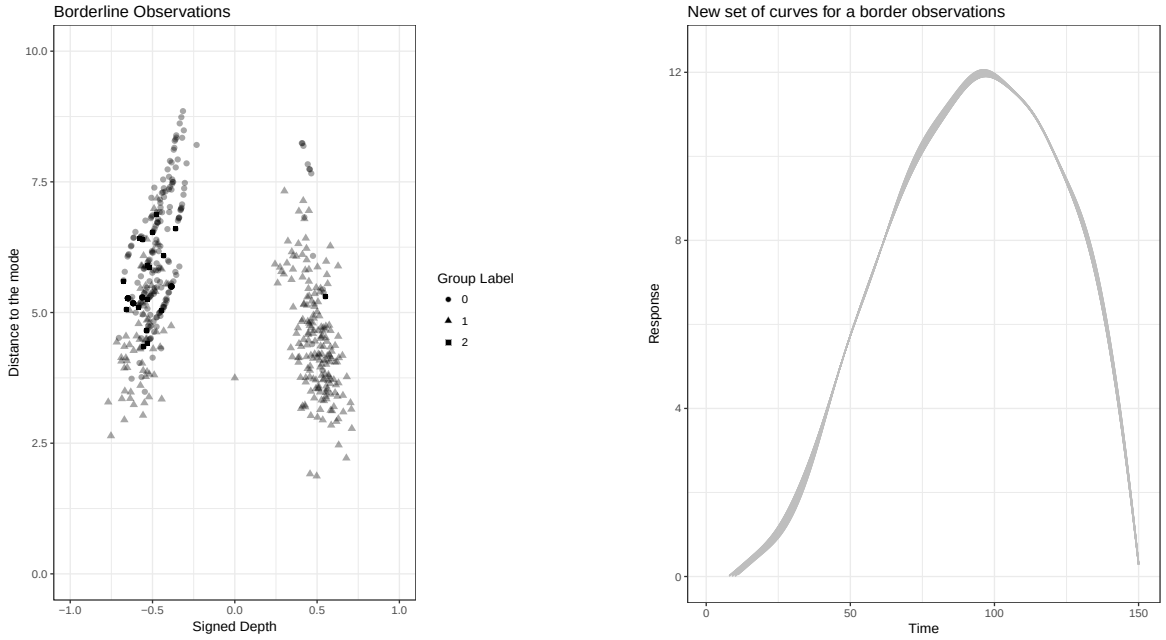
Step 4. Generate new curves from the data in the border set. For each observation in the border set, calculate its k nearest neighbours from the curves in the minority set and call this curve $x_j^k(t_j)$. Then, generate a new curve, called the difference curve, denoted

$Diff$, to be the difference between $x_j^{Bor}(t_j) \in \mathcal{B}$order - set and its k nearest curve, in Euclidean distance, from the minority set. Then multiply the difference curve by a random number $\delta \in Unif[0,1]$ and generate a new sample as

$$\begin{aligned} x_{new}(t_j) &= x_j^{Bor}(t_j) + (\mathcal{D}iff) \times \delta \\ &= x_j^{Bor}(t_j) + (x_j^k(t_j) - x_j^{Bor}(t_j)) \times \delta \\ &= x_j^{Bor}(t_j) \times (1-\delta) + \delta \times (x_j^k(t_j)), \end{aligned} \tag{4.8}$$

and repeat for each $x_j^{Bor}(t_i)$ in the border set.

Note from equation (4.8), that when $\delta = 1$, we are only generating curves according to k nearest neighbours from the curves in the minority set $x_j^k(t_j)$, while when $\delta = 0$, we only generate curves from the border set.



(a) (b)

Figure 4.9: For (A) Observations in the border region for the simulated dataset and (B) new set of generated curves in the border region.

Recall that in our running example, the observations in class Π_0 are the observations in the minority class, with $n_0 = 50$ and $n_1 = 250$. We implement our methodology and the new set of curves in the border set are shown in Figure 4.9. From this we can see that out method only oversamples the observations in the border set which are curves from the minority class.

We implement our methodology to generate a fixed number of observations from the border set (30 new observations for each curve in the borderset) and results are shown in Table 4.5. They show an improvement in terms of the AUC, the Precision and the G-Mean. However, for other metrics like the accuracy, there is a small decrease.

Table 4.5: The value of the metrics before and after we oversample the observations in the border set.

Metrics	Before	After
Error Rate	0.1933	0.1966
Accuracy	0.8066	0.8033
AUC	0.5800	0.7940
Precision	0.8582	0.9483
Recall	0.9200	0.8080
F-measure	0.8880	0.8725
Sensitivity	0.9200	0.8080
G-Mean	0.2101	0.3550
IBA	1.0068	2.1806

Note that the random variable δ shrinks the observations to be closer to the original curve in the border set. Therefore, considering smaller values for δ will generate observations closer to the curves in the border set. Combined with the fact that the curves in the border set can be generated at random, we can generate observations from the border observations as close as we need. For a fixed number of iterations (*nsim*), we start simulating a different set of $Diff$ curves:

Step 1. First, identify the observations in the border set and for an even fixed value of $k = k_{fix}$, calculate the observations closest in depth by looking $k/2$ above and $k/2$ observations below.

Step 2. For a random number δ, generate k new samples following equation (4.8). For this first iteration, classify all the observations, and record all the saved observations.

Step 3. Repeat these steps for the total number of iterations.

By repeating the above procedure, we have a distribution over the labels of the observations in the border set. This allow us to classify the observations using a majority voting approach.

We investigated the performance of this procedure with different values of δ. We considered $\delta = 0.5$ and $\delta = 1$ to generate observations closer to the original curves in the border set and the value of the number of simulations we consider was $nsim = 100$. The results are given in Table 4.6 and show an improvement in terms of the AUC, the precision and the G-Mean. We also observe that for $\delta = 0.5$, the AUC and the G-Mean is the highest value achieved.

Table 4.6: The value of different metrics before and after oversampling the data.

Metrics	Before	$\delta = 0.5$	$\delta = 1$
Error Rate	0.1933	0.1833	0.1900
Accuracy	0.8066	0.8166	0.8100
AUC	0.5800	0.8180	0.8140
Precision	0.9558	0.9577	0.9573
Recall	0.9200	0.8160	0.8080
F-measure	0.8880	0.8812	0.8763
Sensitivity	0.9200	0.8160	0.8080
G-Mean	0.2101	0.3658	0.3640
IBA	1.0068	2.315	2.2659

In terms of the confusion matrix, we can compare the matrices before oversampling the observations in the border set and after oversampling with $\delta = 0.5$. These are shown in Table 4.7. Before oversampling the data, the classifier correctly classifies the observations in the majority class. However, when we oversample the data some of the observations in the majority class are misclassified. This seems to be the penalty when we use oversample to improve the accuracy of classification for the minority class.

Table 4.7: Confusion matrix for the simulated dataset before and after oversampling.

	Before Oversampling		Oversampling ($\delta = 0.5$)	
	Predicted Class 0	Predicted Class 1	Predicted Class 0	Predicted Class 1
True Class 0	12	38	41	9
True Class 1	20	230	48	202

4.5.4 Improving the observations from the majority set as well as the minority set

As we saw before, oversampling observations in the border set only increases the correctly classified observations in the minority set but not the observations in the majority set. To overcome such a difficulty, we propose a new algorithm that not only oversamples the minority class samples but also the observations from the majority set that are near the border. To determine the observations from the majority set we want to oversample, we consider k nearest neighbour observations from the majority set that are closest in distance and signed depth. Our proposed algorithm works as follows.

Step 1. For every curve $x_j(t_j) \in \mathcal{M}ajority$ class we determine the curves in the border set and we define a new border set called the Majority border set, which is a subset of the majority class, as

$$\mathcal{M}\text{ajority}\mathcal{B}\text{order - set} = \left\{x_1^{MajBor}(t_j), \ldots, x_{n_{\mathcal{M}\text{ajority}\mathcal{B}order}}^{MajBor}(t_j)\right\}$$

Step 2. For a fixed value of k, let $k^{'}$ be the number of majority examples among the k nearest neighbours. If $k/2 \leq k^{'} < k$, i.e., the number of minority observations is larger than the number of the majority ones, then $x_j(t_j) \in \mathcal{M}ajority$ is considered to be misclassified and we put it into the majority border set.

Step 3. For each $x_j(t_j) \in \mathcal{M}\text{ajority}\mathcal{B}\text{order - set}$, calculate its k nearest neighbour, in signed depth, by looking $k/2$ observations above and $k/2$ observations below the signed depth of the curve and reduce the observations in the $\mathcal{M}\text{ajority}\mathcal{B}\text{order - set}$.

Step 4. Finally, for each $x_j^{MajBor}(t_j)$ in the reduced $\mathcal{M}ajority\ \ \mathcal{B}\text{order - set}$, we generate k new curves by calculating its k nearest neighbours from the observations in $\mathcal{M}ajority$ and generating a new curve, called the difference curve as defined in equation (4.8).

For ease of comparison, we revisit our running example with $\delta = 0.5$ and $\delta = 1$. The experiment consists of generating observations from both the majority and the minority group in the border set. We perform a simulation study for a total of 100 simulations. The results are given in Table 4.8 and show an improvement in terms of the AUC, the misclassification error rate, G-Mean and F-measure. For the value of $\delta = 0.5$, we achieve the lowest misclassification error rate.

Table 4.8: The value of different metrics before and after oversampling the data, using nine nearest neighbours and the difference curve and sampling from both the majority and minority group.

Metrics	Before	$\delta = 0.5$	$\delta = 1$
Error Rate	0.1933	0.1100	0.1133
Accuracy	0.8066	0.8900	0.8866
AUC	0.5800	0.7980	0.7880
Precision	0.9558	0.9322	0.9285
Recall	0.9200	0.9360	0.9360
F-measure	0.8880	0.9341	0.9322
Sensitivity	0.9200	0.9360	0.9360
G-Mean	0.2101	0.3514	0.34610
IBA	1.0068	2.6069	2.5399

Tables 4.9 and 4.10 show the confusion matrix before oversampling the observations in the border set and after oversampling with $\delta = 1$ and $\delta = 0.5$, respectively. We can see an improvement in the correct classification in both groups, for both values of δ. These results illustrate that by oversampling the minority class and also observations from the majority set that are near the border, we can gain an improvement in terms of the AUC, the misclassification error rate, G-Mean and F-measure. The proposed approach avoids the overlapping produced by only sampling the observations in the minority set, by considering oversampling observations from the majority set.

Table 4.9: Confusion matrix for the simulated dataset before and after oversampling, using using nine nearest neighbours and the difference curve and sampling from both the majority and minority group with a value of $\delta = 1$.

	Before Oversampling		Oversampling ($\delta = 1$)	
	Predicted Class 0	Predicted Class 1	Predicted Class 0	Predicted Class 1
True Class 0	12	38	32	18
True Class 1	20	230	16	234

Table 4.10: Confusion matrix for the simulated dataset before and after oversampling using using nine nearest neighbours and the difference curve and sampling from both the majority and minority group with a value of $\delta = 0.5$.

	Before Oversampling		Oversampling ($\delta = 0.5$)	
	Predicted Class 0	Predicted Class 1	Predicted Class 0	Predicted Class 1
True Class 0	12	38	33	17
True Class 1	20	230	16	234

4.6 Simulation Setup

The goal of this section is to investigate the performance of the k-RNN and the proposed method to generate observations in the border when the ratio of observations in the classes is varied. We investigate, under different scenarios, how oversampling the minority class and also observations from the majority set that are in the border set improves different metrics like the AUC, the misclassification error rate, G-Mean and F-measure. In order to investigate this, we make the following considerations:

1. **Datasets:** We simulate different functional data using the Gaussian process approach discussed in Chapter 2. We start by simulating functional data using a Gaussian Process (GP). All the datasets we consider are two class problems and are observed in the same period of time. Here, the minority class label will be represented by 0, and the majority class label will be represented by 1. Along with the data, we vary the imbalance ratio, which determines the number of observations in the majority class to the number of observations in the minority class. For the simulated datasets we consider three different scenarios $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3$, respectively. In each scenario we keep fixed the minority sample size (n_0 =50), while we vary the number of observations in the majority class. This can be described in terms of a percentage of the majority class, corresponding to 12.5%, 25.0% and 50.0%, respectively.

2. **Resampling Strategies:** Only the oversampling techniques that represent an improvement in terms of the area under the curve are considered. These oversampling techniques are the smoothed bootstrap and random generation from kernel densities. However, we consider only the smoothed bootstrap approach, because of its good performance. We also consider adding Gaussian noise to the curves.

3. **Borderline:** To determine the observations at the borderline, we consider using the k observations closest in depth. For the first scenario and both Gaussian processes, we use k=6, while for the second and third scenario, the choice was k=12, due to a major variability in the data.

4. **Performance metrics:** The following unweighted metrics were chosen for the evaluation.

    - Accuracy and Missclassification Rate.

- AUC.
- Precision / Recall.
- Sensitivity.
- Generalized Index of Balanced Accuracy with $\mathsf{M} = \text{G-Mean}^2$ and $\alpha = 0.1$.
- G-Mean.
- F-Measure with $\beta = 1$.

4.7 Simulations Results

Databases were divided into three different collections according to the imbalance ratio. We refer to the first group as a strongly imbalanced group, which is represented by $\mathcal{S}_1$, the second group as moderately imbalanced, represented by $\mathcal{S}_2$ and the last group as a weakly imbalanced denoted by $\mathcal{S}_3$. We present our results according to the different simulation models. For both scenarios, we start summarising different statistics. We first consider the misclassification error rate obtained in the form of a matrix. Results of applying our proposed classifier with $\delta = 0.5$ and $\delta = 1$ can be seen in Figure 4.10. From this, we can observe that by considering oversampling both classes in the border set with $\delta = 0.5$, the value of the misclassification error rate decreases. For the first scenario, the error of classifying without oversampling the observations is 0.2, while when oversampling both classes is considered and we let $\delta = 0.5$, the misclassification error drops to 0.155. In the second scenario, there is a bigger decrease; the misclassification error without oversampling the observations in the border is 0.188 but applying the oversampling method decreases the misclassification error to 0.092 with $\delta = 0.5$ and 0.096 with $\delta = 1$. In this case, there is a major gain in terms of the classification. Finally, the third scenario considered also shows a decrease in terms of the misclassification error. Before oversampling the data, the misclassification error was 0.1289 and after considering both oversampling using a value of $\delta = 0.5$, we obtain 0.0956. While a higher misclassification error is obtained with $\delta = 1$.

The results when the second Gaussian Process is applied can be found in Figure 4.11. Under the second Gaussian process, which generates smoother curves, we can see a similar pattern; in the first scenario, the misclassification error without oversampling the observations in the border line is 0.28, as opposed to 0.112 found using the oversampling method and

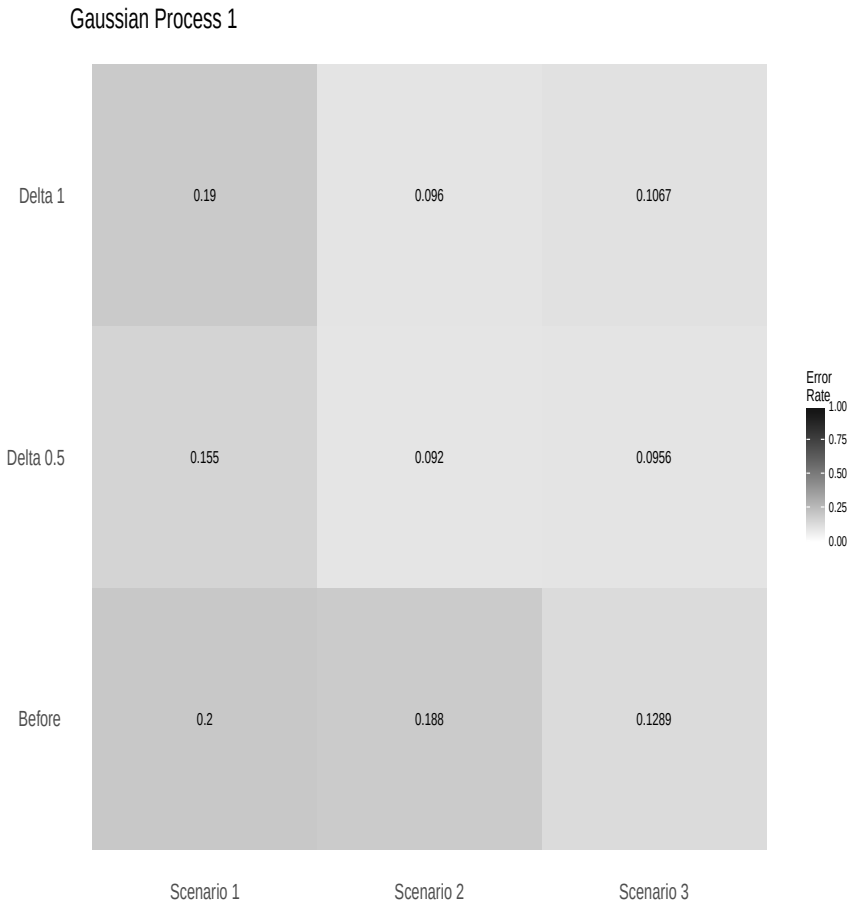


Figure 4.10: Misclassification error rates for the proposed methods, applied to the first Gaussian Process, across the different scenarios and considering an oversampling of both classes near the border.

$\delta = 0.5$, which decreases further when we consider a value of $\delta = 1$. For the second scenario, the misclassification error decreases by half when we oversample the observations using a value of $\delta = 0.5$ and the same occurs when we oversample using a value of $\delta = 0.5$ in scenario 3.

Along with the misclassification error rate, we also investigate the performance of the proposed method using different evaluation metrics. We consider the same metrics as before, the missclassification error rate, the accuracy, the area under the curve, precision, F-measure, sensitivity, G-mean and $\text{IBA}_{\alpha=0.1}$. The values of these metrics can be seen in Table 4.11.

These show an improvement in terms of the misclassification error rate. After we oversample the observations in the border set from both classes, we also observe that there is an improvement in different metrics like the pair precision/recall. For the first Gaussian process with respect to scenario one, we observe that there is an improvement in terms of

Table 4.11: Different metric values for the proposed methods, across the different scenarios. The metrics are evaluated before and after oversampling the observations using a value of $\delta = 0.5$ and $\delta = 1$.

	Scenario 1			Scenario 2			Scenario 3		
	Before	$\delta = 0.5$	$\delta = 1$	Before	$\delta = 0.5$	$\delta = 1$	Before	$\delta = 0.5$	$\delta = 1$
Gaussian Process 1									
Error Rate	0.2000	0.0956	0.1067	0.1880	0.0920	0.0960	0.1289	0.1550	0.1900
Accuracy	0.8000	0.9044	0.8933	0.8120	0.9080	0.9040	0.8711	0.8450	0.8100
AUC	0.5688	0.7300	0.7012	0.6650	0.8800	0.8825	0.7600	0.7300	0.7200
Precision	0.9033	0.9400	0.9322	0.8626	0.9583	0.9585	0.8873	0.8521	0.8544
Recall	0.9575	0.9625	0.9400	0.9100	0.9200	0.9250	0.8400	0.9600	0.9000
F-Measure	0.9296	0.9471	0.9400	0.8856	0.9388	0.9415	0.8630	0.9028	0.8766
Sensitivity	0.9575	0.9625	0.9400	0.9100	0.9200	0.9250	0.8400	0.9600	0.9000
G-Mean	0.1468	0.2301	0.2472	0.3091	0.4395	0.4407	0.4363	0.4000	0.4025
$\text{IBA}_{\alpha=0.1}$	0.8273	1.9746	2.1996	1.6339	2.9366	2.9720	1.9421	2.0640	1.9116
Gaussian Process 2									
Error Rate	0.2800	0.1120	0.1960	0.2450	0.1220	0.2040	0.2700	0.1111	0.2160
Accuracy	0.7200	0.8880	0.8040	0.7550	0.8780	0.7960	0.7300	0.8889	0.7200
AUC	0.5000	0.8762	0.8600	0.7000	0.5500	0.5625	0.4867	0.7650	0.6067
Precision	0.8785	0.8951	0.8991	0.7967	0.8170	0.8213	0.7449	0.7919	0.8013
Recall	0.1250	0.9825	0.9925	0.9800	0.9600	0.9650	0.9733	0.9133	0.8333
F-Measure	0.1176	0.9478	0.9408	0.8789	0.8828	0.8874	0.8439	0.8483	0.8170
Sensitivity	0.1250	0.9625	0.9925	0.9800	0.9600	0.9650	0.9733	0.9133	0.8333
G-Mean	0.1250	0.2303	0.2478	0.0000	0.1833	0.1965	0.0000	0.2920	0.3249
$\text{IBA}_{\alpha=0.1}$	0.8158	0.3007	0.3007	0.0000	0.6552	0.7527	0.0000	1.1338	1.2244

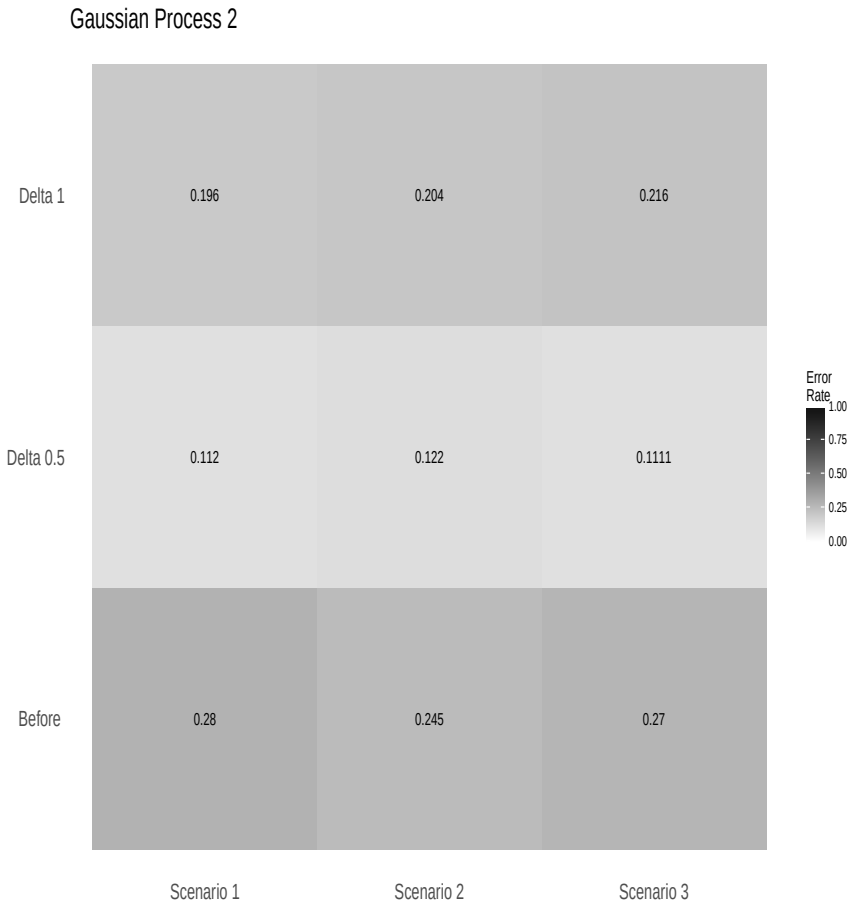


Figure 4.11: Misclassification error rates for the proposed methods, applied to the second Gaussian Process, across the different scenarios and considering an oversampling of both classes near the border.

the missclassfication error, which decreases. However, when we consider different metrics like the AUC, we observe that it is lower when $\delta = 0.5$ than if $\delta = 1$. Both methods show a bigger AUC after we oversample the observations in the border set. This can be because when we improve the observations from the majority set and we repeat the experiment 100 times, some of the observations from the majority group can be misclassified giving a lower value of the AUC. This behaviour does not happen under scenario 3 and the same Gaussian Process because of the number of observations we consider is $n_0 = 50$ and $n_1 = 100$.

For the second Gaussian process and the first scenario, we observe that when we oversample the data with $\delta = 0.5$, the misclassification error rate decreases and it is even lower for $\delta = 1$. In terms of the AUC, there is also an improvement; the AUC increases from 0.5 to 0.87 when $\delta = 0.5$ and 0.86 when $\delta = 1$. We also observe that there is an improvement in terms of the of the precision and recall, which increase for our oversampling method. Similarly, for

scenario 2 and scenario 3, there is an improvement in terms of the misclassification error rate. However, in the same scenarios, a higher value for the AUC can be achieved considering $\delta = 1$, rather than $\delta = 0.5$.

Overall, from our extensive simulation study, we have seen that our proposed algorithm shows an improvement in terms of the different metrics to evaluate our imbalanced classifier. We showed that for the two different Gaussian Processes, the value of the missclassification error decreases after oversampling the observations in both groups.

4.8 Application to some Real imbalanced data sets

In this section we apply our methods to two different real imbalanced datasets. The creation of the imbalanced datasets is discussed in Section 2.4.1. The datasets we analyse are the orange juice dataset and the NIR gasoline spectra for binary classification. We start analysing the orange juice dataset; the first population Π_0 is a population of size $n_0 = 48$ (sucrose < 30) and the second population Π_1 is formed by $n_1 = 170$ curves with (sucrose > 30).

The second real dataset is the NIR spectra of gasoline dataset. The first population Π_0 is formed by $n_0 = 15$ curves with (octane < 86) and the second population Π_1 is formed by $n_1 = 45$ curves with (octane > 86).

We start by classifying the real data before and after we oversample the observations from the majority and minority classes in the border set. We summarise the results of the imbalance in Table 4.12. Here, we can observe that there is a gain in terms of different metrics; for the first real dataset, there is an improvement in terms of the misclassification error and the AUC. Other metrics also improve like the the accuracy of the classifier.

Table 4.12: Different metrics values for the proposed methods, across the two different real datasets and considering an oversampling of both classes near the border.

	Real Dataset 1			Real Dataset 2		
	Before	$\delta = 1$	$\delta = 0.5$	Before	$\delta = 1$	$\delta = 0.5$
Metrics						
Error Rate	0.1743	0.1422	0.1330	0.3000	0.1667	0.1833
Accuracy	0.8257	0.8578	0.8670	0.7000	0.8333	0.8167
AUC	0.6565	0.7743	0.7801	0.4667	0.6667	0.6333
Precision	0.8402	0.8971	0.8983	0.7368	0.8182	0.8036
Recall	0.9588	0.9235	0.9353	0.9333	1.0000	1.0000
F-Measure	0.8956	0.9101	0.9164	0.8235	0.9000	0.8911
Sensitivity	0.9588	0.9235	0.9353	0.9333	1.0000	1.0000
G-Mean	0.3096	0.4037	0.4063	0.0000	0.3333	0.2981
$IBA_{\alpha=0.1}$	-1.3040	2.2328	2.2942	0.0000	0.5556	0.4533

4.9 Conclusions

Throughout this chapter, we have introduced an oversampling approach which involves oversampling the distribution of the principal component scores. When it comes to classifying observations which are close to the border, we propose a new method that shrinks the observations that are closer to the original curve in the border set. This proposed method has been applied to simulated data and real data. When dealing with imbalanced classifiers, we considered different metrics to evaluate the performance. By means of simulations we showed that oversampling the observations in the border set and setting a value of $\delta = 0.5$ outperforms the standard method in terms of the misclassification error. Therefore, our approach should be used for classifying imbalanced datasets, since it is better in terms of the misclassification error rate.

Even though the smoothed bootstrap approach is the standard method we compared method to, there are other methods available. It could be useful to consider evaluating the performance of our approach in comparison to these. In addition, we have limited ourselves to the case of imbalanced observations, we could also consider the performance of our approach when this is not the case.

Chapter 5

Principal Component Analysis for Functional Data

5.1 Introduction

The main goal in supervised classification for functional data is to predict the true group label for each of the observed curves. The main challenge for classifiers based on the density for functional data is that the probability density function does not strictly exist (Delaigle and Hall, 2010). Hence, classifiers based on density estimation need to be modified. Yet, functional component scores capture most of the variability of the data, it is possible to define a meaningful concept of density through representation of principal component scores for a specific scale which is linked to a particular dimension.

In high dimensions, the larger the dimension the sparser the data. So, usually, nonparametric density estimators with a fixed bandwidth procedure are not effective for high dimensional problems (Liu et al., 2007). To tackle this problem we want to rely on a method which adapts the amount of smoothing to the local density of the data. The adaptive kernel method is a method that works well in this situation. The estimate is similar to the nonparametric density estimators with fixed bandwidth, but with the bandwidth proportional to a function that allows to vary from one data point to another (Silverman, 1986).

To start introducing the concept of semimetrics, we first define a norm. A norm, defined on a vector space $\mathbf{F}$, is a real valued function $\|\cdot\|$ satisfying: $\forall \;\; \upsilon_1, \upsilon_2 \in \mathbf{F}$; $\|\upsilon_1\| \geq 0$, $\|\upsilon_1\| = 0$ if and only if $\upsilon_1 = 0$, for $c \in \mathbb{R}$, $\|c\upsilon_1\| = |c| \, \|\upsilon_1\|$ and $\|\upsilon_1 + \upsilon_2\| \leq \|\upsilon_1\| + \|\upsilon_2\|$. And a seminorm is a norm satisfying the above properties except $\|\upsilon_1\| = 0$ if and only if

$\upsilon_1 = 0$. In multivariate data, the most popular approach for measuring closeness is to select the Euclidean norm. If we apply the norm to the distance between $\upsilon_1, \upsilon_2 \in \mathbf{F}$ then we can define a metric space.

A space $\mathcal{D}$ is called a metric space if there is a real valued function $d(\upsilon_1, \upsilon_2)$ between $\upsilon_1, \upsilon_2 \in \mathcal{D}$, such that for a pair $\forall\ \upsilon_1, \upsilon_2 \in \mathcal{D}$; $d(\upsilon_1, \upsilon_2) = d(\upsilon_2, \upsilon_1) \geq 0$ provided that $\upsilon_1 \neq \upsilon_2$, $d(\upsilon_1, \upsilon_2) = 0$ if and only if $\upsilon_1 = \upsilon_2$ and for every $\upsilon_1, \upsilon_2, \upsilon_3 \in \mathcal{D}$; $d(\upsilon_1, \upsilon_2) \leq d(\upsilon_1, \upsilon_3) + d(\upsilon_3, \upsilon_2)$. Semimetrics can be defined on this metric space $\mathcal{D}$ satisfying the above properties except $d(\upsilon_1, \upsilon_2) = 0$ if and only if $\upsilon_1 = \upsilon_2$.

In the functional case, and when we are dealing with function, we can not define a vector norm for functions. But what is possible is to have semimetrics such that the rate of convergence in the functional case is similar to the one of the finite dimensional one and is usual to consider semimetrics based on seminorms (Ferraty and Vieu, 2006).

A commonly used measure of proximity between two different functions is to use the semimetrics based on the principal component scores and multivariate partial least squares. Such semimetrics form fundamental part of our study in this chapter allowing a density estimate to be constructed in a p dimensional space. We use semimetrics to study a nonparametric adaptive density Bayesian classifier using density of log ratios of functional principal component scores based on different semimetrics linked to a particular dimension.

The structure of the chapter is as follows. Section 5.2 introduces the problem formulation and the Bayes classifier for multivariate data. Section 5.3 introduces a Cross-Validation (CV) procedure to select the number of principal components for functional data based on the Integrated Square Error (ISE). Section 5.4 discusses in detail the role of semimetrics for functional data. Section 5.5 introduces the adaptive variable kernel method and the selection of the smoothing parameters. In Section 5.6, we describe the Bayesian classifier based on the density of the functional principal component scores and we extend the problem to more than two classes. In Section 5.7, the performance of our proposed classifier is investigated numerically. In Section 5.8, we apply our simulations to real datasets. Finally, conclusions are stated in Section 5.9.

5.2 Problem formulation & Background

In this section, we start by discussing multivariate density and we introduce the Bayes classification rule for multivariate data. Later, we extend this idea to the functional case, where we consider a p-dimensional representation of each function.

Suppose that we have an observed vector, $\mathbf{x}$, of a multivariate random p-vector. The observed vector $\mathbf{x}$ is known to belong to one of the G classes or groups, which we can denote by $\Pi_0, \Pi_1, \ldots, \Pi_{G-1}$ but the true class for $\mathbf{x}$ is unknown. It is of interest to estimate the posterior probability $\mathbb{P}(\Pi_i \mid \mathbf{x})$ which is the conditional probability that $\mathbf{x}$ comes from class Π_i.

Let $\pi_i = \mathbb{P}(\Pi_i)$ be the probability that a random selected observation is in class Π_i. The π_i's may be regarded as the prior probabilities of a population membership which are formulated without regard to the values in $\mathbf{x}$. Let $f_i(\mathbf{x} \mid \Pi_i)$ be the conditional probability density function of $\mathbf{x}$ given that it is an observation from Π_i. Using Bayes' rule we have that

$$\begin{aligned} \mathbb{P}(\Pi_i|\mathbf{x}) &= \frac{\mathbb{P}(\text{ Member of } \Pi_i \text{ and we observe } \mathbf{x})}{\mathbb{P}(\text{We observe } \mathbf{x})} \\ &= \frac{\mathbb{P}(\pi_i)\,\mathbb{P}(\mathbf{x} \mid \Pi_i)}{\mathbb{P}(\mathbf{x})} \\ &= \frac{\pi_i f_i(\mathbf{x} \mid \Pi_i)}{\sum_{j=0}^{G-1} \pi_j f_j(\mathbf{x} \mid \Pi_j)}, \end{aligned} \tag{5.1}$$

which gives the posterior probability. The numerator is the probability that a randomly selected observation $\mathbf{x}$ occurs given that it comes from class Π_i multiplied by the probability that a random selected observation is in class Π_i. While the denominator is the sum of the probabilities of observing $\mathbf{x}$ given that the true class is Π_i, multiplied by the corresponding prior probabilities for each class. The classification rule assigns $\mathbf{x}$ to the class Π_i for which $\mathbb{P}(\Pi_i|\mathbf{x})$ is the largest. This procedure is known as the Bayes classification rule.

In the case when we have two populations we can consider the ratio of the two posterior probabilities. We will assign $\mathbf{x}$ to Π_1 if

$$\frac{\pi_1 f_1(\mathbf{x} \mid \Pi_1)}{\pi_2 f_2(\mathbf{x} \mid \Pi_2)} > 1, \tag{5.2}$$

i.e., if

$$\frac{f_1(\mathbf{x} \mid \Pi_1)}{f_2(\mathbf{x} \mid \Pi_i)} > \frac{\pi_2}{\pi_1}. \tag{5.3}$$

Note that the denominator in equation (5.1) is the same for all values of i so the decision rule is to assign $\mathbf{x}$ to the class for which the value of $\pi_i f_i(\mathbf{x} \mid \Pi_i)$ is the largest.

We can extend the Bayes classification rule to the functional case. Suppose that $x(t)$ is an observed curve on an interval $\mathcal{T}$. The Bayesian classifier that minimises the Bayesian risk assigns $x(t)$ to population Π_1 if

$$\frac{\pi_1 f_1(x(t) \mid \Pi_1)}{\pi_2 f_2(x(t) \mid \Pi_2)} > 1, \tag{5.4}$$

where $f_1(x(t) \mid \Pi_1)$ is the conditional probability density function of $x(t)$ given that it is an observation from Π_1 and $f_2(x(t) \mid \Pi_2)$ is the conditional probability density function of $x(t)$ given that it is an observation from Π_2. In the functional case, a drawback of this approach is that the probability density function does not strictly exist. A meaningful alternative is to define a density estimation based on the functional component scores which capture most of the variability of the data (Delaigle and Hall, 2010).

In the finite dimensional case, if $\mathbf{X}$ is a random vector of finite length, then there exists a probability density function $f(\mathbf{X} \mid \Pi_1)$ which is defined as the limit when $\varepsilon \to 0$ of the probability that $\mathbf{X}$ is inside a ball of radius ε and divided over the Lebesgue measure of the ball. More precisely,

$$f(\mathbf{X} \mid \Pi_1) = \lim_{\varepsilon \to 0} \frac{\mathbb{P}(\|\mathbf{X} - \mathbf{x}\| \leq \varepsilon)}{\varepsilon^p v_p}, \tag{5.5}$$

where v_p represents the volume of the p-dimensional unit sphere. In the functional case, we might expect the same, but instead of being divided over the volume it is expected to be divided over a function that depends on ε. Delaigle and Hall (2010) proved that the density for functional data strictly does not exist but it is meaningful to define a density estimator based on the functional component scores, which are linked to the probability mass contained in a small ball around a given fixed function and this property can be used to define a simple, easily estimable surrogate for the density.

Let $x^{(p)}(\mathbf{t})$ be a p-dimensional representation of the curve $x(\mathbf{t})$, observed at a grid of time points $t_1, \ldots, t_M$, Delaigle and Hall (2010) propose an estimator of the log density of $x^{(p)}(\mathbf{t})$ conducted via the product of the densities f_j of the principal component scores, given by

$$\hat{l}(x^{(p)}(\mathbf{t})|p) = \frac{1}{p}\sum_{j=1}^{p}\log\hat{f}_j(\xi_j), \tag{5.6}$$

where ξ_j is the version of the score for the function $x(\mathbf{t})$. More precisely,

$$\xi_j = \theta_j^{-1/2}\int_{\mathcal{T}}(x_j(t)-\bar{x}(t))\psi_j(t)\ dt \quad \text{for} \quad j=1,\ldots,p, \tag{5.7}$$

where $\bar{x}(t)$ is the mean function and $\theta_1 \geq \theta_2 \geq \ldots \theta_p \geq 0$ are the eigenvalues with respective orthonormal functions $\psi_j(\mathbf{t})$. To find estimators of the quantities θ_j and $\psi_j(\mathbf{t})$, three different strategies are suggested by Ramsay et al. (2009). These strategies are: the discretisation - interpolation approach, the basis function expansion approach and the numerical quadrature approach. We do not discuss these in detail, but each method aims to approximate the solutions of the functional eigen-equation:

$$\int_{\mathcal{T}}\widehat{Cov}(\mathbf{s},\mathbf{t})\psi(\mathbf{t})dt = \theta_j\psi_j(\mathbf{s}), \tag{5.8}$$

for positive eigenvalue/eigenfunction pairs, $\big(\theta_j,\psi_j(\mathbf{t})\big)$ and an estimator $\widehat{Cov}(\mathbf{s},\mathbf{t})$ of the covariance function.

Having found the estimators using any of the strategies suggested, $\hat{\theta}$ and $\hat{\psi}_j(\mathbf{t})$ represent the estimators of the eigenvalues and the eigenfunctions, respectively. If $x_i^{(p)}(\mathbf{t}),\ldots,x_n^{(p)}(\mathbf{t})$ represents some sample curves in a p-dimensional space, then the decomposition of the curves can be expressed via a linear combination of the principal component scores and an orthonormal basis. More precisely,

$$x_i^{(p)}(\mathbf{t}) = \sum_{j=1}^{p}\hat{\Xi}_{ij}\hat{\psi}_j(\mathbf{t}) \quad \text{for} \quad i=1,\ldots,n, \tag{5.9}$$

where $\hat{\Xi}_{ij}$ is given by

$$\hat{\Xi}_{ij} = \hat{\theta}_j^{-1/2}\int_{\mathcal{T}}(x_i(t)-\bar{x}(t))\hat{\psi}_j(t)\ dt \quad \text{for} \quad i=1,\ldots,n,\ \ j=1,2,\ldots,p. \tag{5.10}$$

Then the density f_j of the principal component scores is estimated using kernel density estimators. More precisely

$$\hat{f}_j(\hat{\xi}_j) = \frac{1}{nh}\sum_{i=1}^{n}K\left(\frac{\hat{\Xi}_{ij}-\hat{\xi}_j}{h}\right) \quad \text{for} \quad j=1,2,\ldots,p, \tag{5.11}$$

where $K(\cdot)$ is a positive function, also known as the kernel function, and h is the bandwidth. For an introduction to kernel density estimation see Wand and Jones (1994) and Bowman and Azzalini (1997).

An attractive feature of the estimator in equation (5.6), as pointed out by Delaigle and Hall (2010), is that it can be computed for a range of values of p. To obtain estimators of $\hat{\theta}_j$ and $\hat{\psi}_j(\mathbf{t})$, we followed the basis function expansion approach implemented via the R package fda package (Ramsay et al., 2014).

5.3 Determining a finite dimensional space

When we deal with an infinite dimensional space, in practice it is not possible to compute infinite sums or infinite integrals, so we need to determine a fixed dimension p where we want to work. Perhaps the most popular method to determine a fixed dimension p is the scree plot by Cattell (1966). This method consists of plotting the eigenvalues in decreasing order of magnitude and looking for an *elbow* in the graph. The method is simple and useful when it works, but in many cases, there is no recognisable *elbow* in the plot.

In this section, we present a Cross-Validation (CV) procedure to select the number of principal components for functional data using the definition of the Integrated Squared Error (ISE). Different procedures that maximise or minimises different criterion can be proposed. In the functional case, we saw that this criterion performs well. The 10-fold Cross-Validation approach to select the number of principal components is based on the ISE, which is given by,

$$ISE\left(x^{(p)}(\mathbf{t})\right) = \int_{\mathcal{T}} \left(x^{(p)}(t) - x(t)\right)^2 dt. \tag{5.12}$$

Here $x(t)$ represents the true curve and $x^{(p)}(\mathbf{t})$ the curve projected using the first p scores, curves are observed over an interval $\mathcal{T}$. The Mean Integrated Squared Error (MISE) is given by the expectation of the integrated square error, i.e.,

$$MISE = \mathbb{E}\left[ISE\left(x^{(p)}(\mathbf{t})\right)\right]. \tag{5.13}$$

To select the number of principal components in the functions, the 10-fold cross-validation approach can be summarised as follows:

Step 1. For $p = 1, \ldots, p_{max}$ and a fixed $p_{max} \in \mathbb{Z}^+$, split the data, from both groups, into a randomly selected training set and omit 10% of the data from class Π_1 to generate a test set called $\mathcal{T}est$.

Step 2. Use all of the data from from class Π_0 and the reduced set of Π_1 data to estimate the functional principal component.

Step 3. Using the fitted model to estimate each of the omitted curves from class Π_1 calculate the ISE,

$$ISE(x^{(p)}(\mathbf{t})) = \int_{\mathcal{T}} \left(x^{(p)}(t) - x(t)\right)^2 dt,$$

between the estimate and actual curve each time.

Step 4. Average the ISE's calculated in the previous step.

Step 5. Repeat steps (2) to (4) above for each remaining 10% subset of data in Π_1.

Step 6. Repeat steps (2) to (5) above now keeping all the Π_1 data and omitting 10% of the Π_0 data each time.

Step 7. Average the 20 average ISE values you have obtained in steps (2) to (6) above to get a cross-validation score.

The above cross-validation algorithm chooses the value of p by omitting and keeping data from both groups.

Different Cross-Validation (CV) procedures with different criteria can be used, e.g., Yao et al. (2005) which includes an AIC criterion for selecting the number of principal components in sparse functional data and Hall and Vial (2006) which used bootstrap methods. More recent contributions to select the number of principal components includes Poskitt and Sengarapillai (2013). A possibility is to compare the numerical performance of the different methods, however, due to time limitations this was not possible. In practice and for classification purposes, the Cross-Validation (CV) procedure using the Integrated Squared Error works well.

5.4 The Role of semimetrics in Functional Data

In this section we discuss the role of semimetrics in representing a function $x(t)$. We introduce two main semimetrics that are the core of our simulation study. The first is the semimetric based in principal component analysis and the second one is the semimetric based in partial least squares.

Proximities between two different functions play an important role in functional data, e.g., Ferraty and Vieu (2003) and Ferraty and Vieu (2006). They can be used for different purposes such as classification and we see that the choice of the norm becomes crucial and have a strong impact on the results. We start by introducing closeness notions for functional data.

In the functional case, if $x_i(\mathbf{t})$ and $x_j(\mathbf{t})$ are two functions observed on the closed interval $\mathcal{T}$, we can define the square distance between two functions by

$$d(x_i(\mathbf{t}),x_j(\mathbf{t})) = \int_{\mathcal{T}} \big(x_i(t) - x_j(t))\big)^2 dt. \tag{5.14}$$

and more general

$$\big\|d(x_i(\mathbf{t}),x_j(\mathbf{t}))\big\|_q = \left(\int_{\mathcal{T}} \big(x_i(t) - x_j(t))\big)^q dt\right)^{1/q}. \tag{5.15}$$

Let $x^{(p)}(\mathbf{t})$ be a p-dimensional representation of the curves constructed using equation (5.9), where the dimension p is usually determined using Cross-Validation procedures. Thus, the curves in equation (5.14) and equation (5.15) are replaced by their p-dimensional representation. In a p-dimensional space, the $\mathcal{L}^2$-norm can be written as

$$\left\|d(x_i^{(p)}(\mathbf{t}),x_j^{(p)}(\mathbf{t}))\right\|_2 = \left(\int_{\mathcal{T}} \left(x_i^{(p)}(t) - x_j^{(p)}(t))\right)^2 dt\right)^{1/2}. \tag{5.16}$$

A natural extension is to consider the $\mathcal{L}^q$-norm. More precisely,

$$\left\|d(x_i^{(p)}(\mathbf{t}),x_j^{(p)}(\mathbf{t}))\right\|_q = \left(\int_{\mathcal{T}} \left(x_i^{(p)}(t) - x_j^{(p)}(t))\right)^q dt\right)^{1/q}. \tag{5.17}$$

As an alternative to equation (5.14), we can consider the Karhunen-Lòeve expansion for random functions and we can express a curve in terms of a linear combination of the principal component scores and a complete orthonormal basis functions. To achieve this, we followed an approach similar to Section 3.5. Let $\left\{x_i^{(p)}(t_j), i = 1,\ldots,n, j = 1,\ldots,M\right\}$ be a collection

of observed curves constructed using equation (5.9). The expansion for each curve in terms of the principal component scores is given by

$$x_i^{(p)}(\mathbf{t}) = \sum_{j=1}^{p} \hat{\Xi}_{ij}\hat{\psi}_j(\mathbf{t}) \quad \text{for} \quad i = 1,\ldots,n, \tag{5.18}$$

where the matrix $\hat{\Xi}_{ij}$ for $j = 1,\ldots,p$ consists of the functional principal component scores. Thus, $x_j^{(p)}(\mathbf{t})$ admits an expansion of the following form

$$x_j^{(p)}(\mathbf{t}) = \sum_{j=1}^{p} \hat{\xi}_j\psi_j(\mathbf{t}), \tag{5.19}$$

where $\hat{\xi}$ is the version of the score for the function $x_j(\mathbf{t})$. An expression for the square distance difference defined in equation (5.14) between two observed functions in a p-dimensional space constructed using equation (5.9) is given by

$$\begin{aligned}
\int_{\mathcal{T}} \left(x_i^{(p)}(t) - x_j^{(p)}(t)\right) dt &= \int_{\mathcal{T}} \Big(\sum_{j=1}^{p} \hat{\Xi}_{ij}\hat{\psi}_j(t) - \hat{\xi}_j\hat{\psi}_j(t) \Big) dt \\
&= \int_{\mathcal{T}} \Big(\sum_{j=1}^{p} \left(\hat{\Xi}_{ij} - \hat{\xi}_j\right) \hat{\psi}_j(t) \Big) dt \\
&= \sum_{j=1}^{p} \left(\hat{\Xi}_{ij} - \hat{\xi}_j\right) \int_{\mathcal{T}} \hat{\psi}_j(t) dt \\
&= \sum_{j=1}^{p} \left(\hat{\Xi}_{ij} - \hat{\xi}_j\right),
\end{aligned} \tag{5.20}$$

for $i = 1,\ldots,n$ and the integral $\int_{\mathcal{T}} \hat{\psi}_j(t)dt$ integrates to one. This can be used to rewrite the $\mathcal{L}^2$-norm in equation (5.16) as

$$\left\| d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t})) \right\|_2 = \left(\sum_{j=1}^{p} \left(\hat{\Xi}_{ij} - \hat{\xi}_j\right)^2 \right)^{1/2}. \tag{5.21}$$

And similar, the $\mathcal{L}^q$-norm as in equation (5.17) as

$$\left\| d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t})) \right\|_q = \left(\sum_{j=1}^{p} \left(\hat{\Xi}_{ij} - \hat{\xi}_j\right)^q \right)^{1/q}. \tag{5.22}$$

Equation (5.21) represents a finite dimensional representation of the curves and can be used to compute the distance between the principal component scores.

Principal Components Using Mercer's theorem, a decomposition of the curves can be expressed via a linear combination of the principal component scores and a complete orthonormal basis. The semimetric based on principal components scores uses a truncated version of the linear combination. It is defined as

$$d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t}))^{PCA} = \sum_{k=1}^{p} \left(\int_{\mathcal{T}} \left(x_{ik}^{(p)}(t) - x_{jk}^{(p)}(t) \right)^2 dt \right) \psi_k, \tag{5.23}$$

where $\psi_1, \dots \psi_p$ are the orthonormal eigenvectors of the covariance matrix associated with their corresponding eigenvalues. More generally, the $\mathcal{L}^q$-norm has the following form

$$\left\| d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t}))^{PCA} \right\|_q = \left(\sum_{k=1}^{p} \left(\int_{\mathcal{T}} \left(x_{ik}^{(p)}(t) - x_{jk}^{(p)}(t) \right) dt \right)^q \psi_k \right)^{1/q}. \tag{5.24}$$

From a numerical perspective, integrals in equation (5.23) and equation (5.24) can be approximated using quadrature weights. Thus, for equation (5.24), the semimetric based on principal component scores can be approximated, on a grid of times, using its empirical version:

$$d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t}))^{PCA} = \sum_{k=1}^{p} \left(\sum_{l=1}^{M} \omega_l \left(x_{ik}(t_l) - x_{jk}(t_l) \right) \psi_k \right)^2, \tag{5.25}$$

where $\omega_1, \dots, \omega_M$ are the quadrature weights.

5.4.1 Multivariate Partial Least Square Regression (MPLSR)

The Multivariate Partial Least Square Regression (MPLSR) is a statistical method for regressing a multivariate response on a multivariate predictor. It was originally developed in economic science and is extensively used in image processing and the chemometrics community. It was developed to predict a multivariate response from independent variables when there is a high degree of collinearity among the predictors and when the number of predictors is large with respect the number of observations.

Multivariate Partial Least Square Regression is used to compress a matrix of observed predictors $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_p]$ that contains the values of p predictors for n samples, into a vector of factors $\left[\psi_k^p\right]$. This is usually determined sequentially using the nonlinear iterative partial least squares (NIPALS) algorithm (Wold, 1966). Then the orthogonal factors scores are used to fit a set of n observations to p dependent variables. An attractiveness feature of this

method is that it finds a parsimonious model even when the predictors are highly correlated or linearly dependent. In the functional case, the semimetric based on partial least squares is defined as

$$d(x_i^{(p)}(\mathbf{t}),x_j^{(p)}(\mathbf{t}))^{PLS} = \sum_{k=1}^{p}\left(\sum_{l=2}^{M}\omega_l\left(x_{ik}(t_l)-x_{jk}(t_l)\right)\left[\psi_k^p\right]\right)^2, \tag{5.26}$$

where $\left[\psi_k^p\right]$ is a vector performed by MPLSR on the curves, and $\omega_1,\dots\omega_M$ are the quadrature weights. Similar to the PCA, the semimetric based on partial least squares can be applied with the $\mathcal{L}^q$-norm in the following way

$$\left\|d(x_i^{(p)}(\mathbf{t}),x_j^{(p)}(\mathbf{t}))^{PLS}\right\|_q = \left(\sum_{k=1}^{p}\left(\sum_{l=2}^{M}\omega_l\left(x_{ik}(t_l)-x_{jk}(t_l)\right)\left[\psi_k^p\right]\right)^q\right)^{1/q}. \tag{5.27}$$

The number of factors plays a similar role to the number of dimensions considered in PCA. The main difference with functional principal components comes from the fact that the components performed with PCA explain only the predictors, whereas in the PLS approach, the components are also relevant for the multivariate response (Preda et al., 2007; Escabias et al., 2007).

Additionally to the semimetric based on partial least squares and PCA, we have the semimetric based on derivatives. However, the computation of semimetrics based on derivatives is numerically intensive and involves a numerical stability problem (Ferraty and Vieu, 2006) due to the fact that the approximation of the curves is obtained using B-spline expansion for each curve. Thus, successive derivatives lead to numerical stability problems.

Motivated by Hall et al. (2001), semimetrics based on principal component scores and partial least squares allows a density estimate to be constructed in a p-dimensional space. Using kernel density estimation techniques, we can define a semimetric kernel density estimation (SKDE) given by,

$$\hat{f}\left(x^{(p)}(\mathbf{t})\right) = \frac{1}{nh}\sum_{i=1}^{n}K\left(h^{-1}\left\|d(x_i^{(p)}(\mathbf{t}),x^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.28}$$

where h is the bandwidth and $K(\cdot):\mathbb{R}^p\to\mathbb{R}$ is a positive compact function defined on $[0,\infty)$ that integrates to 1, known as the kernel function. Observe that the semimetrics map p-dimensional representations of curves into the real positive line. Thus, a suitable kernel function is the half normal kernel defined as

$$K(x) = \sqrt{\frac{2}{\pi}} e^{x^2/2}.$$

5.5 Variable Kernel Density Estimation

When the value of p is large, fixed kernel density estimators are not effective for high dimensional problems. This is not only because of the scarcity of the data over the estimation space, but also because of the computational costs of cross validation when bandwidths need to be selected for each dimension, and the slow rate of convergence (Liu et al., 2007). For example, if we consider observations following a Gaussian distribution, in the univariate case, the mass is concentrated around the mean. However, as the dimension increases, the probability mass is heavy around the tails. A fixed bandwidth procedure will have difficulties with densities that exhibit large changes in magnitudes (Cacoullos, 1966).

Note that from equation (5.28), the most common approach is to fix h, but two different approaches have been put forth to vary h. In this section, we briefly discuss both approaches. The first proposal is the k-nearest neighbour by Loftsgaarden et al. (1965) and the second proposal is the adaptive kernel density estimate of Breiman et al. (1977).

5.5.1 Nearest Neighbour Methods

The nearest neighbour method is widely used in nonparametric discriminant analysis. Some examples can be found in Mack and Rosenblatt (1979). It takes its name from the fact that the density depends on near neighbours rather than nearest neighbours to a particular point and it has its roots in univariate and multivariate data.

To start describing the nearest neighbour kernel method, let us assume that $x_1^{(p)}(\mathbf{t}), \ldots, x_n^{(p)}(\mathbf{t})$ is a collection of p-dimensional representation of the curves observed in the interval $t \in [a,b]$ and let

$$d\left(x_1^{(p)}(\mathbf{t}), x_2^{(p)}(\mathbf{t})\right) = \left|x_1^{(p)}(\mathbf{t}) - x_2^{(p)}(\mathbf{t})\right|,$$

be the distance between $x_1^{(p)}(\mathbf{t})$ and $x_2^{(p)}(\mathbf{t})$. For each $x^{(p)}(\mathbf{t})$ in the sample, we define the distances $d_k(x^{(p)}(\mathbf{t})) = \left|x^{(p)}(\mathbf{t}) - x_k^{(p)}(\mathbf{t})\right|$ and arrange them in an ascending order from $x^{(p)}(\mathbf{t})$ to the k^{th} nearest curve in the sample. In other words,

$$d_1(x^{(p)}(\mathbf{t})) \leq d_2(x^{(p)}(\mathbf{t})) \leq \ldots \leq d_n(x^{(p)}(\mathbf{t})).$$

Then, the nearest neighbour density estimation is defined by

$$\hat{f}_k(x^{(p)}(\mathbf{t})) = \frac{k/n}{2d_k(x^{(p)}(\mathbf{t}))}, \tag{5.29}$$

with $d_k(x^{(p)}(\mathbf{t})) = \left|x^{(p)}(\mathbf{t}) - x_k^{(p)}(\mathbf{t})\right|$.

A drawback of the nearest neighbour density estimation is that it is not a smooth curve and the function $d_k(x(\mathbf{t}))$ has discontinuity derivatives. Another drawback about the nearest neighbour density estimation is that it is not a probability density, since it fails to integrate to unity (Silverman, 1986). Thus, it is not recommended when we try to estimate the entire density.

The nearest neighbour method can be generalised as a kernel based method. For a semimetric, $d(\cdot,\cdot)$, the generalised k^{th} nearest neighbour density estimation is then defined by

$$\hat{f}_k\left(x^{(p)}(\mathbf{t})\right) = \frac{1}{nh}\sum_{i=1}^{n} K\left((d_k(x(\mathbf{t})))^{-1}\left\|d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.30}$$

where $K(\cdot)$ which is a kernel function integrating to one and $d_k(x)(\mathbf{t})$ is the distance between the observation $x^{(p)}(\mathbf{t})$ and the k^{th} nearest observation.

5.5.2 The Adaptive Kernel Method

The adaptive kernel methods is another method that proposes to vary the smoothing parameter. As opposed to the Nearest Neighbour Methods, this method relates the k-nearest distances to the amount of smoothing to the local density of the data. To relate both quantities, we estimate the density similarly to the local density estimation in the data as in equation (5.28). However, the scale parameter is placed on the data points to allow them to vary from one data point to another. Let $d_i^k(x^{(p)}(\mathbf{t}))$ be the Euclidean distance between the curve $x_i^{(p)}(\mathbf{t})$ and the k^{th} nearest curve in the set of $n-1$ observations. Then we can define the adaptive kernel density estimation with a smoothing parameter h and a kernel function $K(\cdot)$, given by,

$$\hat{f}_k\left(x^{(p)}(\mathbf{t})\right) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{hd_i^k(x^{(p)}(\mathbf{t}))} K\left((hd_i^k(x^{(p)}(\mathbf{t})))^{-1}\left\|d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t}))\right\|_q\right). \tag{5.31}$$

Observe now that the window width is proportional to $d_i^k(x(\mathbf{t}))$, therefore, regions where the data is sparse will have flat kernels associated with them (Silverman, 1986). For a fixed k, the overall degree of smoothing will depend on the smoothing parameter h and the choice of k will determine how responsive the window width is to very local detail. In contrast with the generalised k^{th} nearest neighbour kernel based method defined in equation (5.30), the adaptive kernel density estimation integrates to unity provided the kernel $K(\cdot)$ integrates to 1.

5.5.3 Choosing the Smoothing parameter h and k

Bandwidth selection is the main problem of kernel density estimation. In order to construct a density estimate from observed data it is necessary to choose a value for the smoothing parameter h and the number of nearest observations k. When the kernel density estimation only depends on the smoothing parameter h, procedures based on minimising the mean integrated square error or the asymptotic mean integrated square error are common, but for high dimensions, these methods are computationally intensive.

Clearly equation (5.31) depends on two different parameters: the k nearest points and a smoothing parameter h. To select both parameters, we propose an algorithm which, for a fixed value of k, maximises the accuracy. It explores a range of values of h and finds a value which tries to maximise the accuracy of the classifier. Algorithm 3 explains in detail this method. Thereafter, this algorithm will form part of the training of the classification rule based on the original sample data. Note that we select a global smoothing parameter by selecting the smoothing parameter h to be the same in both groups.

Algorithm 3: Optimal value of k and h that minimises the misclassification error.

Choosing (h,k) **Input :** Functional Principal Components Scores in a finite dimensional space $x^{(p)}(\mathbf{t})$, a set of labels for each curve, a norm $||\cdot||_q$, a semimetric, $d(\cdot,\cdot)$, an initial smoothing parameter range $h \in [h_1, h_2]$ and a fixed $k_{max} \in \mathbb{Z}^+$.

Output : A pair of (h^*, k^*) that maximises the empirical accuracy.

Fix an initial value of $h_0 > 0 \in [h_1, h_2]$ and

foreach $k = 1, \ldots, k_{max}$ **do**

- Let $x_{ij}^{(p)}(\mathbf{t})$ be an observed curve belonging to the j^{th} group and let $x_0^{(p)}(\mathbf{t})$ be a p dimensional representation of a new curve we are interested in classifying.
- Compute the distance based on the semimetric $\left\| d(x_i^{(p)}(\mathbf{t}), x_j^{(p)}(\mathbf{t})) \right\|_q$ between the curve we want to classify and the principal component scores in the j^{th} group.
- Assign the new curve $x_0^{(p)}(\mathbf{t})$ to the the class Π_1 if $\hat{f}_1(x_0^{(p)}(\mathbf{t})) > \hat{f}_2(x_0^{(p)}(\mathbf{t}))$, using the adaptive kernel method with $k = 1, \ldots, k_{max}$ and $h = h_0$.
- Form a confusion matrix and calculate the misclassification error rate $\mathcal{E}$, defined as $\mathcal{E} = \sum_{i=1}^{n} \frac{1}{n} \mathbb{1}_{\{y_i \neq \hat{y}_i\}}$ where $\hat{y}_i$ are the predicted labels.
- Choose k such that it has the lowest misclassification error rate, i.e., $k^* = \min\{\mathcal{E}\}$.

For k^*, generate a sequence of values for h with step size $\delta = \frac{1}{n}$.

foreach *h in the sequence* **do**

- Reclassify the data according to the adaptive discrimination approach with $k = k^*$.
- Calculate the confusion matrix and calculate the misclassification error rate.
- Select h which achieves the lowest misclassification error rate, i.e., $h^* = \arg\min\{\mathcal{E}\}$.

5.6 Statistical discrimination for functional data

In this section, we describe the Bayesian classifier based on the densities of the functional principal component scores. We start by considering a binary classification problem.

Let $x_0^{(p)}(\mathbf{t})$ be a new curve is known to belong to one of the classes Π_0 or Π_1, but the true class for the curve is unknown. A classification rule will assign the curve $x_0^{(p)}(\mathbf{t})$ to the class for which $\hat{\mathbb{P}}(\Pi_0 \mid x_0^{(p)}(\mathbf{t}))$ or $\hat{\mathbb{P}}(\Pi_1 \mid x_0^{(p)}(\mathbf{t}))$ is the greatest. Let $\hat{\pi}_0 = \hat{\mathbb{P}}(\Pi_0)$ be the estimated probability that a random selected observation belong to class Π_0 and let $\hat{f}_j\left(x_0^{(p)}(\mathbf{t}) \mid \Pi_j\right)$ be the estimated conditional probability density function of $x_0^{(p)}(\mathbf{t})$ surrogated by the principal components scores in a p dimensional space conditional on the class j. Using Bayes' rule we can estimate the posterior probability for the class Π_0, given by

$$\hat{\mathbb{P}}(\Pi_0|x_0^{(p)}(\mathbf{t})) = \frac{\hat{\pi}_0\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{\pi}_0\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_0) + \hat{\pi}_1\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}, \tag{5.32}$$

and similarly for Π_1

$$\hat{\mathbb{P}}(\Pi_1|x_0^{(p)}(\mathbf{t})) = \frac{\hat{\pi}_1\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{\pi}_0\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_0) + \hat{\pi}_1\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}, \tag{5.33}$$

where

$$\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_0}\sum_{i=1}^{n_0}\frac{1}{hd^k(x_0^{(p)}(\mathbf{t}))}K\left((hd^{k0}(x_0^{(p)}(\mathbf{t})))^{-1}\left\|d(x_i^{(p)}(\mathbf{t}),x_0^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.34}$$

and

$$\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_1}\sum_{i=1}^{n_1}\frac{1}{hd^k(x_0^{(p)}(\mathbf{t}))}K\left((hd^{k1}(x_0^{(p)}(\mathbf{t})))^{-1}\left\|d(x_i^{(p)}(\mathbf{t}),x_0^{(p)}(\mathbf{t}))\right\|_q\right). \tag{5.35}$$

The quantities n_0 and n_1 denote the sample sizes of the observations in the classes Π_0 and Π_1, respectively, h is the global smoothing parameter and $x_{i0}^{(p)}(\mathbf{t})$ and $x_{i1}^{(p)}(\mathbf{t})$ are i.i.d. samples drawn belonging to population Π_0 and Π_1, respectively. The quantity $d^{k0}(x_0^{(p)}(\mathbf{t}))$ is Euclidean distance between the $x_0^{(p)}(\mathbf{t})$ curve and the k^{th} nearest curve in the set of $n_0 - 1$ observations, and $d^{k1}(x_0^{(p)}(\mathbf{t}))$ is the Euclidean distances between the $x_0^{(p)}(\mathbf{t})$ curve and the k^{th} nearest curve in the set of $n_1 - 1$ observations. In practice the number of k^{th} nearest neighbours is set to be the same in both populations Π_0 and Π_1. When comparing two classes,

it is sufficient to look at the ratio (or log-ratio) and we can classify an observation belonging to class Π_0 if

$$\log\left(\frac{\hat{\mathbb{P}}(\Pi_0|x^{(p)}(\mathbf{t}))}{\hat{\mathbb{P}}(\Pi_1|x^{(p)}(\mathbf{t}))}\right) > 0. \tag{5.36}$$

More precisely, if

$$\log\left(\frac{\hat{\mathbb{P}}(\Pi_0|x^{(p)}(\mathbf{t}))}{\hat{\mathbb{P}}(\Pi_1|x^{(p)}(\mathbf{t}))}\right) = \log\left(\frac{\hat{f}_0(x^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{f}_1(x^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) + \log\left(\frac{\hat{\pi}_0}{\hat{\pi}_1}\right) > 0. \tag{5.37}$$

The prior probabilities are estimated by the number of curves in each group, i.e.,

$$\hat{\pi}_0 = \frac{n_0}{n_1 + n_0}, \tag{5.38}$$

and

$$\hat{\pi}_1 = \frac{n_1}{n_1 + n_0}. \tag{5.39}$$

If all the classes have the same prior probabilities, i.e., $\pi_0 = \pi_1$, equation (5.37) reduces to

$$\log\left(\frac{\hat{\mathbb{P}}(\Pi_0|x^{(p)}(\mathbf{t}))}{\hat{\mathbb{P}}(\Pi_1|x^{(p)}(\mathbf{t}))}\right) = \log\left(\frac{\hat{f}_0(x^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{f}_1(x^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) > 0, \tag{5.40}$$

and we classify an observation to class Π_0 if and only if

$$\log\left(\frac{\hat{\mathbb{P}}(\Pi_0|x^{(p)}(\mathbf{t}))}{\hat{\mathbb{P}}(\Pi_1|x^{(p)}(\mathbf{t}))}\right) > 0.$$

5.6.1 Statistical discrimination for more than two classes

Here, we extend the Bayesian classifier to scenarios with more than 2 classes. Consider G classes, and assume that the densities of the functional principal component scores are available, i.e., $\hat{f}_0, \ldots, \hat{f}_{G-1}$. Let $\hat{\pi}_g = \hat{\mathbb{P}}(\Pi_g)$ be the estimated prior probability that a random selected observation is in population Π_g. The resulting posterior probability that the observed curve $x^{(p)}(\mathbf{t})$ belongs to the g^{th} class is given by

$$\hat{\mathbb{P}}(\Pi_g|x^{(p)}(\mathbf{t})) = \frac{\hat{\pi}_g \hat{f}_g(x^{(p)}(\mathbf{t}) \mid \Pi_g)}{\sum_{g=0}^{G-1} \hat{\pi}_g \hat{f}_g(x^{(p)}(\mathbf{t}) \mid \Pi_g)}. \tag{5.41}$$

The Bayesian classifier for G classes assigns the curve to that class with the highest posterior probability. Because the denominator in equation (5.41) is the same for all π_g, we assign a curve to the g^{th} group, with the highest posterior probability

$$\max_{0\leq g\leq G-1}\left\{\hat{\pi}_g\hat{f}_g(x^{(p)}(\mathbf{t})\mid\Pi_g)\right\}. \tag{5.42}$$

Observe that random assignment can be used (where needed) to break the tie between the appropriate classes. In other words, a new curve $x^{(p)}(\mathbf{t})$ is assigned to the g^{th} population if

$$\hat{\pi}_g\hat{f}_g(x^{(p)}(\mathbf{t})\mid\Pi_g)>\hat{\pi}_l\hat{f}_l(x^{(p)}(\mathbf{t})\mid\Pi_l)\quad\forall\ g\neq l. \tag{5.43}$$

We can define the Bayesian classifier in an equivalent form by pairwise comparisons of posterior probabilities. We define the *log-odds* as

$$L_{g,l}(x^{(p)}(\mathbf{t}))=\log\left(\frac{\hat{\pi}_g\hat{f}_g(x^{(p)}(\mathbf{t})\mid\Pi_g)}{\hat{\pi}_l\hat{f}_l(x^{(p)}(\mathbf{t})\mid\Pi_l)}\right). \tag{5.44}$$

Then, we assign a new curve $x_0^{(p)}(\mathbf{t})$ to the g^{th} population if

$$L_{g,l}(x_0^{(p)}(\mathbf{t}))>0\quad\forall\ g\neq l. \tag{5.45}$$

5.7 Simulations

In this section we propose three main practical implementations of the Bayesian classifier with simulated data and real data. The goal of this section is to compare the proposed adaptive method based on two main semimetrics: a semimetric based on the principal component scores and the semimetric based on partial least squares. For both semimetrics, we compare the methods: the fixed kernel density estimate, the proposed adaptive approach, the k-NN density estimate, the product kernel density method, and finally a regression approach. All the methods we consider here involve the choice of tuning parameters (mainly bandwidths and number of principal components) and we describe how these parameters are specified.

For conciseness, we restrict our simulation study to generating data according to two main methods which vary the amount of the variability explained by the numbers of components. The simulations are structured with the data, i.e., we train and test on the same data, which

may be over optimistic. Even though we train and test on the same data the simulation methods are still comparable.

For the first scenario, more that 90% of the variability is explained by the first 5 principal components. While, for the second scenario, 15 principal components were needed to explain more than 90% of the total variability. For each of the two main models, we consider a modest sample size scenario, with $n_1 = n_2 = 50$ and a larger sample size scenario with $n_1 = n_2 = 200$. In terms of atypicals in the data, we generate samples containing atypicals observations as we described in Chapter 2. For each simulation scenario a total of 100 simulations were run. We sampled the curves on 100 equidistant points over the compact interval $[0, 100]$. We perform the comparison among methods in terms of misclassification percentages and we report their means and standard deviations. To start with we consider two different scenarios in which we vary the basis coefficients previously introduced in Section 2.2.

Scenario 1. The basis coefficients ξ_{ij} are chosen to be independent and normally distributed with mean 0 and decaying variance $\sigma_j^2 = 2.5e^{-j/2}$, $j = 0, \ldots, 2M$; $i = 1, \ldots, n$.

Scenario 2. The basis coefficients ξ_{ij} are chosen to be independent and normally distributed with mean 0 and decaying variance $\sigma_j^2 = 2.5e^{-j/20}$, $j = 0, \ldots, 2M$; $i = 1, \ldots, n$.

The general mean function in both scenarios is considered to be the same $\bar{x}(t) = 50 \times (1-t) \times t^2$ and we consider the following set of sub-models: a no contamination model, an asymmetric contamination model, a linear peak contamination model and a shape contamination model.

The simulations have been conducted in R using the functions from the package fda.usc (Febrero-Bande and Oviedo de la Fuente, 2012) and fda (Ramsay et al., 2014) to obtain estimates of the functional principal component scores. We also use the packages RSpectra and caret for creation of the folds, and the package fds for the use of real datasets. To compute the semimetrics, we use the functions from the `https://www.math.univ-toulouse.fr/~ferraty/SOFTWARES/NPFDA/`, the first function was the **semimetric.mplsr** which computes the distance between curves based on the partial least squares method. The second was the **semimetric.pca**, which computes the distance between curves based on the functional principal components analysis method. For the method based on regression approach, we choose the smoothing parameter using a global choice from the routine **funopare.knn.gcv**

and a quadratic kernel. To compare the classifiers, we use the misclassification error rate, sensitivity and specificity.

5.7.1 Competing methods

We start by describing the fixed kernel density estimate, followed by the adaptive method. We also describe the k-NN kernel density estimation method, the product kernel method and methods based on a regression.

In our simulations, we consider an optimisation routine for the parameter selection. For the kernel methods that depend on only one parameter, we started by exploring a range of values of the smoothing parameter h, to minimise the misclassification error. Except for the regression methods where the optimal value of the smoothing parameter is chosen using a CV approach.

1. **Fixed Kernel Rule Method**

   The fixed kernel rule method is similar to the proposed adaptive kernel estimator, except that the bandwidth is fixed. A fixed bandwidth can result in under-smoothing areas with only sparse observations while over-smoothing in others. Let $\hat{\pi}_1 = \hat{\pi}_2$ be the same prior probabilities in both groups. The fixed kernel rule assigns a new observation $x_0^{(p)}(\mathbf{t})$ to class Π_0 if and only if

$$\log\left(\frac{\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) > 0. \tag{5.46}$$

   We estimate the density according to

$$\hat{f}_0(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_0}\sum_{i=1}^{n_0}\frac{1}{h}K\left((h)^{-1}\left\|d(x_{0i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.47}$$

   and

$$\hat{f}_1(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_1}\sum_{i=1}^{n_1}\frac{1}{h}K\left((h)^{-1}\left\|d(x_{1i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right). \tag{5.48}$$

   The quantities n_0 and n_1 denote the sample sizes of the observations in the classes Π_0 and Π_1, respectively, and $d(x_{0i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))$ is the semimetric based on PCA or PLSE with $x_{0i}^{(p)}(\mathbf{t})$ to be the i^{th} observation of the curve in population Π_0. The semimetric

based on PCA or PLSE is given by $d(x_{1i}^{(p)}(\mathbf{t}), x_1^{(p)}(\mathbf{t}))$, where $x_{1i}^{(p)}(\mathbf{t})$ is the i^{th} observation of the curve in population Π_1. The global smoothing parameter h is optimised such that it minimises the misclassification error.

2. **Proposed Adaptive Method**

The adaptive approach is the method we proposed in Section 5.5.2. Using this approach and letting the prior probabilities be the same, i.e., $\hat{\pi}_1 = \hat{\pi}_2$ we classify a new observation, $x_0^{(p)}(\mathbf{t})$, to class Π_0 if and only if

$$\log\left(\frac{\hat{f}_{k0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{f}_{k1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) > 0.$$

Here $\hat{f}_{k0}(\cdot)$ and $\hat{f}_{k1}(\cdot)$ are estimated using the adaptive kernel density estimation with a smoothing parameter h and a kernel function $K(\cdot)$. More precisely,

$$\hat{f}_{k0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_0}\sum_{i=1}^{n_0}\frac{1}{hd^{k0}(x_0(\mathbf{t}))}K\left((hd^{k0}(x_0(\mathbf{t})))^{-1}\left\|d(x_{0i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.49}$$

and

$$\hat{f}_{k1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_1}\sum_{i=1}^{n_0}\frac{1}{hd^{k1}(x_0(\mathbf{t}))}K\left((hd^{k1}(x_0(\mathbf{t})))^{-1}\left\|d(x_{1i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right). \tag{5.50}$$

The quantities n_0 and n_1 denote the sample sizes of the observations in the classes Π_0 and Π_1, respectively, and $d^{k0}(x_0^{(p)}(\mathbf{t}))$ is Euclidean distance between the $x_0^{(p)}(\mathbf{t})$ curve and the k^{th} nearest curve in the set of $n_0 - 1$ observations. The Euclidean distance between $x_0^{(p)}(\mathbf{t})$ curve and the k^{th} nearest curve in the set of $n_1 - 1$ observations is given by $d^{k1}(x_0^{(p)}(\mathbf{t}))$, and the semimetrics based on PCA or PLS are written $d(x_{gi}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))$ with $x_{ji}^{(p)}(\mathbf{t})$ being the i^{th} observation of the curve in j^{th} group.

The adaptive approach we considered requires that the optimisation of the smoothing parameter h and the optimal number of k nearest neighbours is the same in both populations. In our simulations we consider the PCA and the PLS semimetrics. The optimisation procedure is described in the Algorithm 3, using a grid of proposed values.

The distance is multiplied by the global smoothing parameter when the density is estimated.

3. ***k*-NN Kernel Method**

As before, the k-NN kernel rule assigns a new observation, $x_0^{(p)}(\mathbf{t})$, to class Π_0 if and only if

$$\log\left(\frac{\hat{\hat{f}}_{k0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{\hat{f}}_{k1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) > 0. \tag{5.51}$$

For a given semimetric, the generalised k^{th} nearest neighbour density estimation for the g^{th} group is then defined by

$$\hat{\hat{f}}_{k0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_0 h}\sum_{i=1}^{n_0} K\left((d_{k0}(x_0(\mathbf{t})))^{-1}\left\|d(x_{0i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right), \tag{5.52}$$

and

$$\hat{\hat{f}}_{k1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_1 h}\sum_{i=1}^{n_1} K\left((d_{k1}(x_0(\mathbf{t})))^{-1}\left\|d(x_{1i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q\right). \tag{5.53}$$

The quantities n_0 and n_1 denote the sample sizes of the observations in the classes Π_0 and Π_1, respectively, $d_{kj}(x)_0(\mathbf{t})$ is the Euclidean distance between the new observation and the k^{th} nearest observation in the j^{th} group and $K(\cdot)$ is kernel function integrating to one.

4. **Product Kernel Method**

The product kernel method assigns a new observation, $x_0^{(p)}(\mathbf{t})$, to class Π_0 if and only if

$$\log\left(\frac{\hat{\hat{f}}_{\mathbf{H}0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_0)}{\hat{\hat{f}}_{\mathbf{H}1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1)}\right) > 0. \tag{5.54}$$

In its most general form, the product kernel method (Wand and Jones, 1994) is a multivariate kernel density estimation given by

$$\hat{\hat{f}}_{\mathbf{H}0}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_0 \prod_{j=i}^{p} h_{0j}}\sum_{i=1}^{n_0}\prod_{j=1}^{p} K\left(\frac{\left\|d(x_{0i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t}))\right\|_q}{h_{0j}}\right), \tag{5.55}$$

and

$$\hat{\hat{f}}_{\mathbf{H}1}(x_0^{(p)}(\mathbf{t}) \mid \Pi_1) = \frac{1}{n_1 \prod_{j=i}^{p} h_{1j}} \sum_{i=1}^{n_0} \prod_{j=1}^{p} K\left(\frac{\left\| d(x_{1i}^{(p)}(\mathbf{t}), x_0^{(p)}(\mathbf{t})) \right\|_q}{h_{1j}} \right), \tag{5.56}$$

for each group, respectively. As before, the function $K(\cdot)$ is a kernel function integrating to one, n_0 and n_1 denote the sample sizes of the observations in the classes Π_0 and Π_1, the semimetrics based on PCA or PLS are written $d(x_{gi}^{(p)}(t), x_0^{(p)}(t))$ with $x_{ji}^{(p)}(t)$ being the i^{th} observation of the curve in g^{th} group, and the quantities h_{0j} and h_{1j} are the entries of the diagonal matrix $\mathbf{H}_0$ and $\mathbf{H}_1$, chosen in the same way for both groups:

$$\mathbf{H}_0 = \text{diag}\left\{h_{10} \ldots h_{p0}\right\} = 0.9 A_0 n_0^{-1/5}, \quad \mathbf{H}_1 = \text{diag}\left\{h_{11} \ldots h_{p1}\right\} = 0.9 A_1 n_1^{-1/5}.$$

Let $SD(\hat{\xi}_j)_g$ for $j = 1, \ldots, p$, be the standard deviation of the j^{th} principal component score in the g^{th} group, as in equation (5.7). Let $A_0 = \min\left\{SD(\hat{\xi}_j)_0, IQR/1.34\right\}$ being the minimum of the standard deviations of the j^{th} principal component scores and the interquartile range of the observations in the population Π_0, and let $A_1 = \min\left\{SD(\hat{\xi}_j)_1, IQR/1.34\right\}$ be the standard deviation and interquartile range of the observations in population Π_1. The asymptotically optimal bandwidth, which is defined so that the squared distance between the density estimator and the true density has the same convergence rate, it is of the order $n^{-1/5}$ in one dimension. While in higher dimensions usually asymptotically optimal bandwidth suggest to use a power $n^{-(1/p+4)}$, where p is the dimension. If the j^{th} principal component score is normal distributed with mean zero and standard deviation σ, then expected value of the IQR of the samples is $2q_{0.75}\sigma \approx 1.34$ where $q_{0.75}$ is the 75% percentile of a standard normal variable.

5. **Regression Methods**

In functional data, it is common to consider regression estimates based on Nadaraya - Watson estimators (Ferraty and Vieu, 2003). The use of a Nadaraya-Watson type estimator is to ensure that it converges in functional regression estimates and the classifier is consistent (Devroye, 1978). We consider a comparison between kernel methods (Nadaraya, 1964; Watson, 1964) and an appropriate way to estimate the

regression function is using the Nadaraya-Watson estimators. More precisely, we consider estimators of the form

$$\mathbb{E}(\Pi_0|x^{(p)}(\mathbf{t})) = \mathbb{P}(\Pi_0|x^{(p)}(\mathbf{t})) = \frac{\sum_{i=1}^{n_0} \mathbb{1}_{\{y_i\in\Pi_0\}} K(h^{-1}d(x_i^{(p)}(\mathbf{t}),x^{(p)}(\mathbf{t})))}{\sum_{i=1}^{n_0} K(h^{-1}d(x_i^{(p)}(\mathbf{t}),x^{(p)}(\mathbf{t})))}, \tag{5.57}$$

where $K(\cdot)$ is the defined kernel $K(u) = (1-u^2)\mathbb{1}_{[0,1]}(u)$. To classify a new observation $x_0^{(p)}(\mathbf{t})$, based on ratios. We consider the following:

$$\frac{\mathbb{P}(\Pi_0|x_0^{(p)}(\mathbf{t}))}{\mathbb{P}(\Pi_1|x_0^{(p)}(\mathbf{t}))} = \frac{\mathbb{E}(\Pi_1|x_0^{(p)}(\mathbf{t}))}{\left[1-\mathbb{E}(\Pi_0|x_0^{(p)}(\mathbf{t}))\right]}, \tag{5.58}$$

with

$$\mathbb{E}(\Pi_1|x_0^{(p)}(\mathbf{t})) = \mathbb{P}(\Pi_1|x_0^{(p)}(\mathbf{t})) = \frac{\sum_{i=1}^{n_1} \mathbb{1}_{\{y_i\in\Pi_1\}} K(h^{-1}d(x_i^{(p)}(\mathbf{t}),x_0^{(p)}(\mathbf{t})))}{\sum_{i=1}^{n_1} K(h^{-1}d(x_i^{(p)}(\mathbf{t}),x_0^{(p)}(\mathbf{t})))}. \tag{5.59}$$

To select the bandwidth, h, we followed a global cross-validation approach (Ferraty and Vieu, 2003). This approach selects the bandwidth, h, regarding the global choice of the number of neighbours and selects the k_{opt} in $h_{k_{opt}}^{-1}$ using a cross-validation approach. The main goal is to compute the quantity

$$R_{CV}^{KNN} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{y_i\in\Pi_0\}} K(h_{k_{opt}}^{-1}d(x_i^{(p)}(\mathbf{t}),x^{(p)}(\mathbf{t})))}{\sum_{i=1}^{N} K(h^{-1}d(x_i^{(p)}(\mathbf{t}),x^{(p)}(\mathbf{t})))}, \tag{5.60}$$

where k_{opt} is the bandwidth which corresponds to the optimal number of neighbours obtained by a cross-validation procedure, i.e., the k_{opt} is chosen such that it minimises the CV function

$$k_{opt} = \arg\min_k \sum_{i=1}^{N} \left(y_i - R_{-i}^{KNN}\left(x_i^{(p)}(\mathbf{t})\right)\right), \tag{5.61}$$

and $R_{-i}^{KNN}\left(x_i^{(p)}(\mathbf{t})\right)$ is the term with the i^{th} curve removed, for all $j \neq i$. The global term means that the same number of neighbours are used for all curves. For implementation purposes, the routine **funopare.knn.gcv** computes the quantities

$$R_{CV}^{KNN}(x_1^{(p)}(t_j)),\ldots,R_{CV}^{KNN}(x_N^{(p)}(t_j)). \tag{5.62}$$

for a set of discretised curves over $t_j = t_1,\ldots,t_M$.

5.7.2 Simulation Results

For both scenarios, we summarise the misclassification error rate in terms of boxplots. We also report the mean specificity, mean sensitivity and mean error rate along with the standard deviation in a table form.

In each of the simulations, the number of principal components is selected in a way that minimises the mean integrate square error (MISE) as defined in Section 5.3 using the $\mathcal{L}^2$-norm. In the adaptive kernel method, the values of the smoothing parameter h and k are selected such that the misclassification error rate is minimised and we use the Euclidean distance for both semimetrics.

We report the results of the proposed adaptive method with the two different semimetrics, except for the larger sample size, where only the results of the semimetric based on principal components is reported. We start by summarising our results in a boxplot form against all the different classifiers.

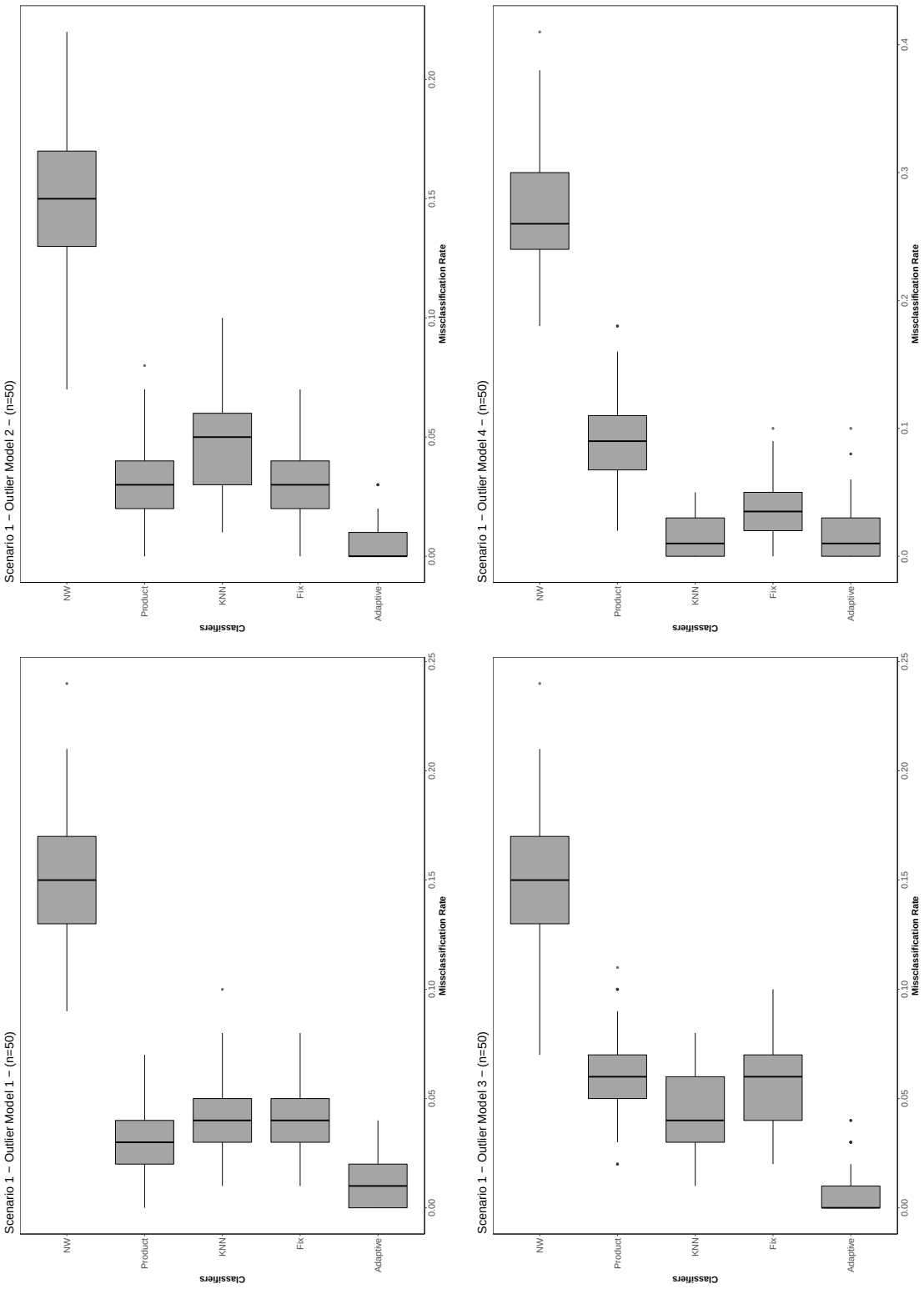


Figure 5.1: Boxplots of misclassification error rates obtained for the first scenario and across the different contamination models with the PC semimetric and a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

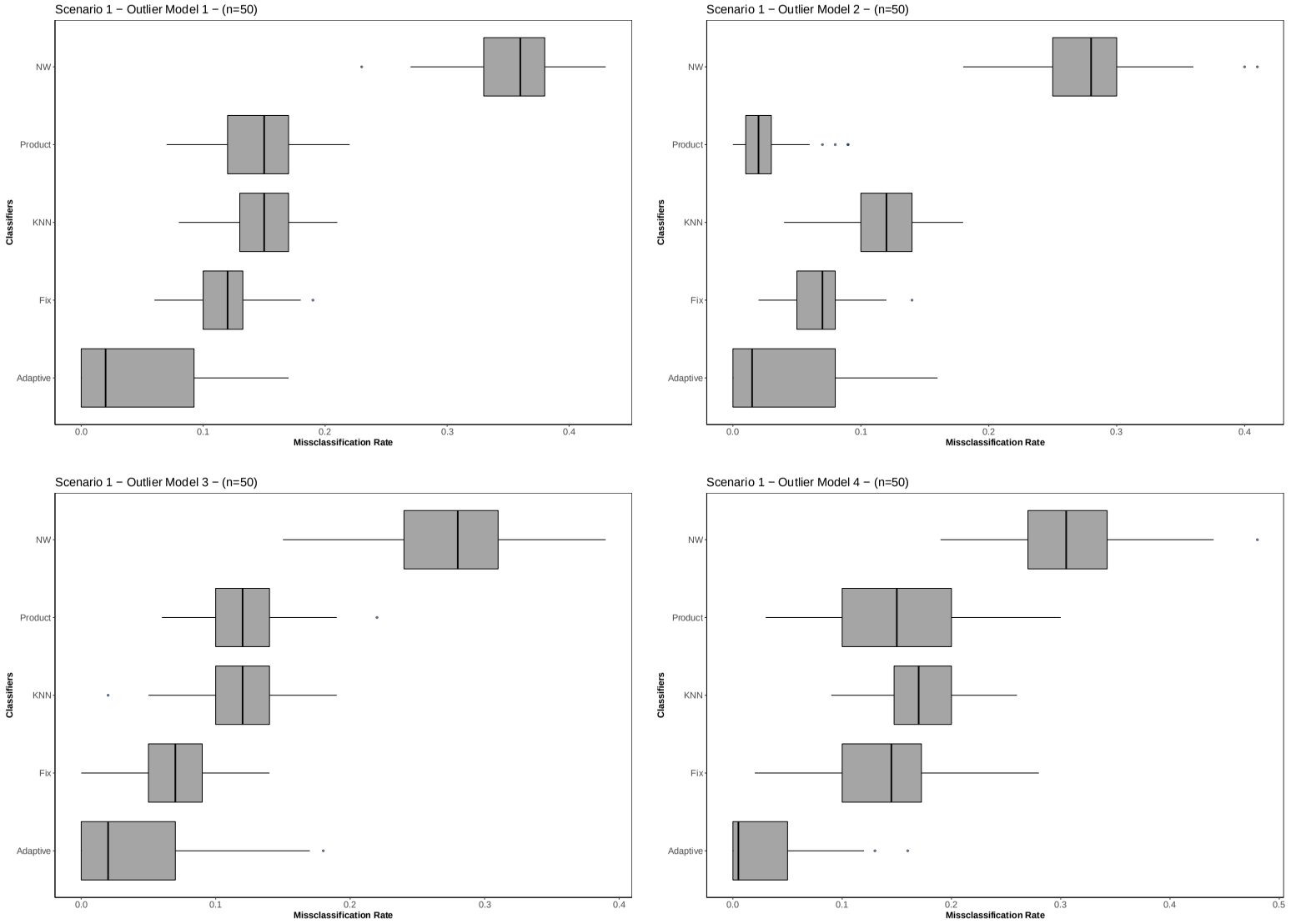


Figure 5.2: Boxplots of misclassification error rates obtained for the second scenario with the PC semimetric and different contamination model with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

Figure 5.1 and Figure 5.2 show the boxplots of the misclassification error rates over the different classification scenarios for the different outlier models using the PC semimetric. In Figure 5.1 we observe that in most scenarios the adaptive approach outperforms all other methods. Howevever, when the data is contaminated by introducing atypicals, the k-NN kernel approach performs similarly to the adaptive approach, but still, the adaptive outperforms the remaining different methods.

Results of the simulations for the second scenario can be seen in Figure 5.2. In this case, we can observe that the adaptive approach outperforms all the different methods under most of the different scenarios. However, the variability is higher in this scenario. This suggests that by increasing the number of principal components in the data, we obtain more variability for the adaptive classifier. This can be explained because the adaptive approach now requires exploration of the data in a higher dimensional space. In the case where there is asymmetric contamination, the product rule perform as well as the adaptive approach.

In all examples, the adaptive method performs better than the methods based on Nadaraya-Watson estimates and outperforms classifiers based on density estimation. However, in the second scenario we see that when the number of principal components increases, this tends to increase the variability of the classifier.

We also investigate the performance when we increase the sample size to $n_1 = n_2 = 200$ observations in each population. Results of considering a different sample size can be seen in Figure 5.3 for the first scenario, and Figure 5.4 for the second scenario.

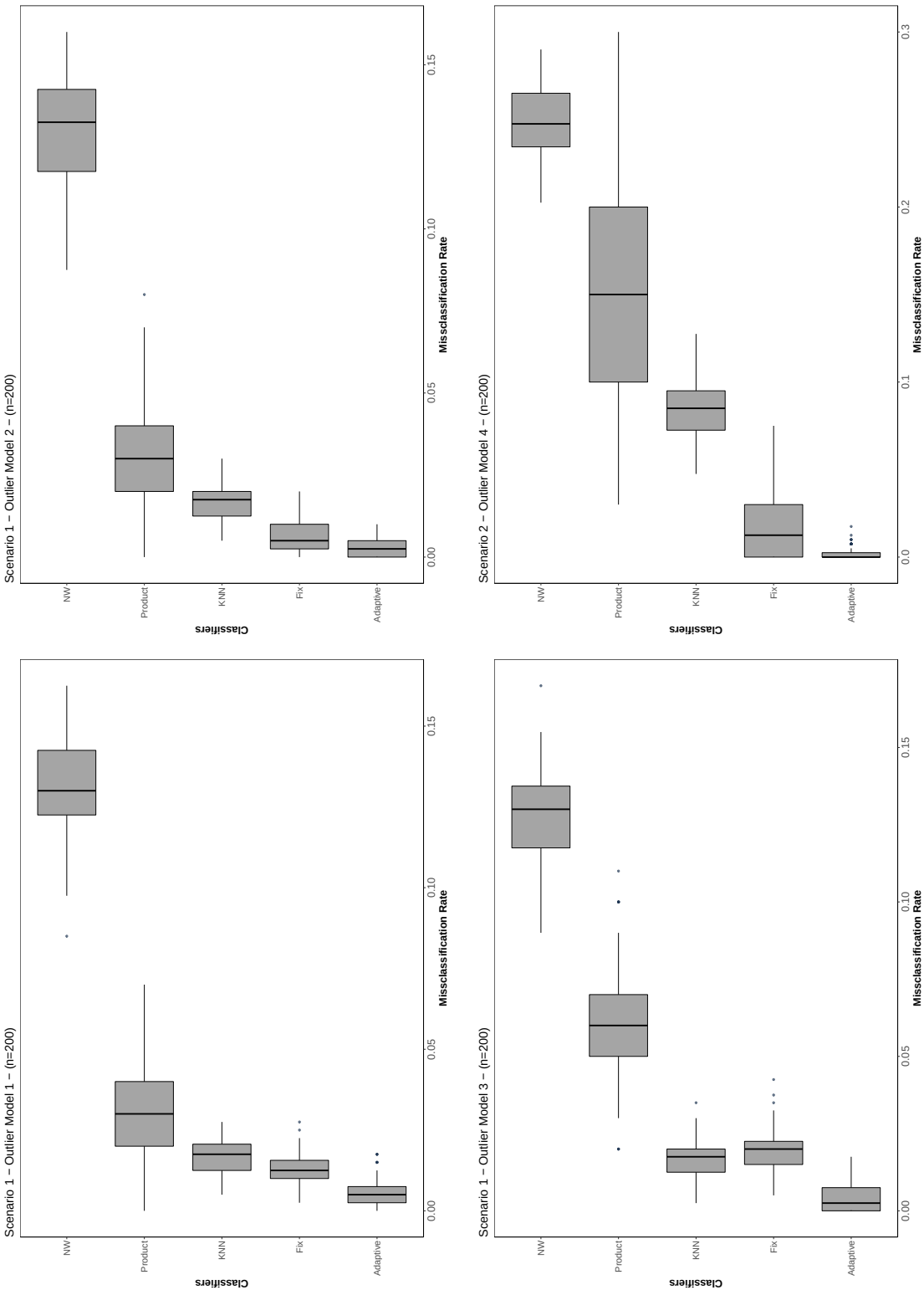


Figure 5.3: Boxplots of misclassification error rates obtained for the first scenario and different contamination model with a sample size $n_1 = n_2 = 200$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

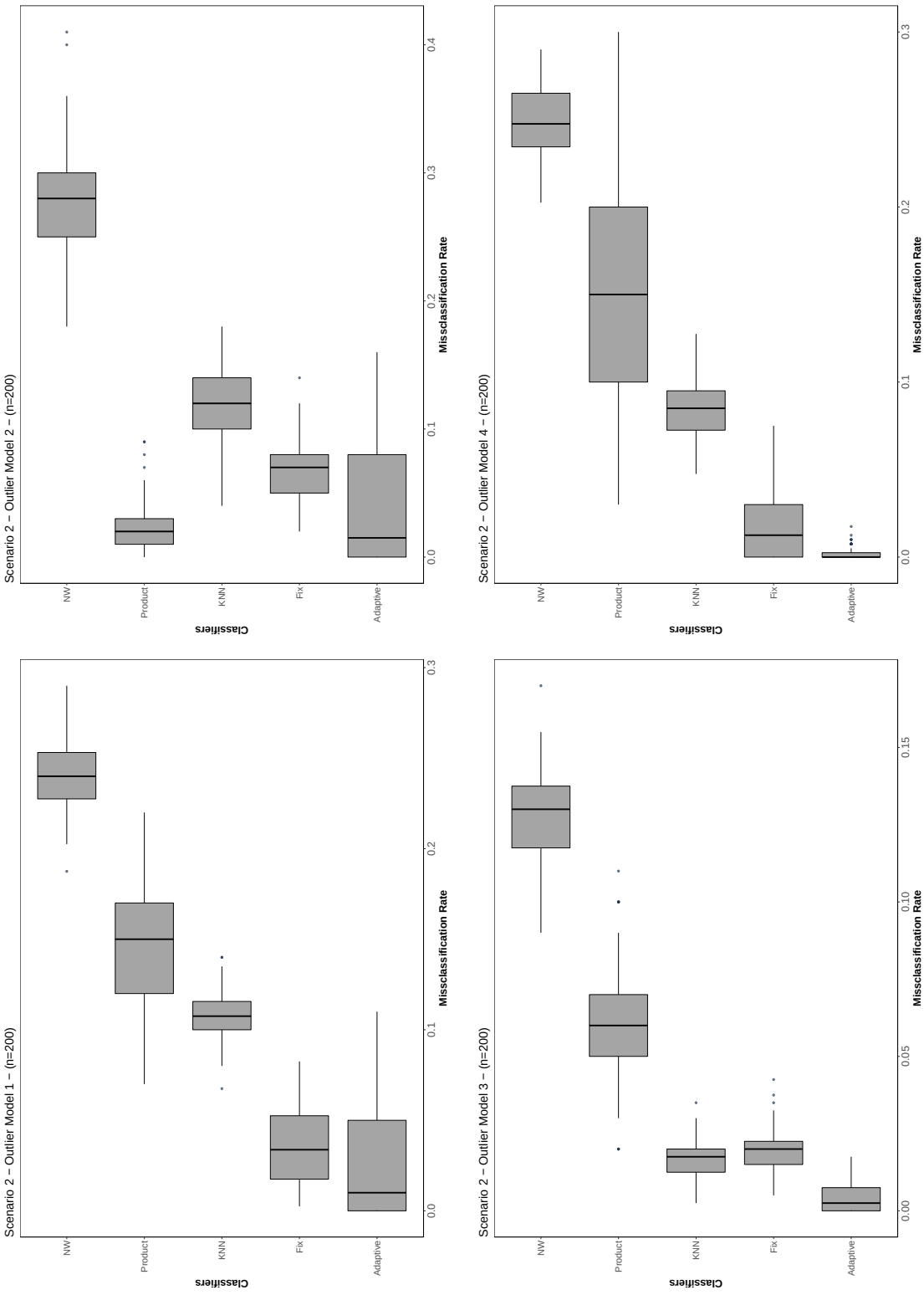


Figure 5.4: Boxplots of misclassification error rates obtained for the first scenario and different contamination models with a sample size $n_1 = n_2 = 200$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

Figure 5.3 shows the misclassification error rate of the adaptive approach against different methods. In Figure 5.3 we observe that in all the different scenarios the adaptive approach performs better than the other competitor methods. In all the different scenarios, we can observe that the adaptive approach, along with the other kernel methods approach (the fix kernel method, the k-NN and the product kernel method) perform better than the regression approach based on Nadaraya-Watson type estimates.

In Figure 5.4 we observe that the adaptive approach often performs as good as the other competitors but the variance of the classifier greatly increases. For the uncontaminated model, the adaptive approach performs better than the other methods. While when we consider an Asymmetric Contaminated Model the variability increases but the product kernel method perform as good as the adaptive approach. For the other contaminated models, the adaptive approach performs better even when we consider a Shape Contamination Model.

When increasing the sample size in both groups we can conclude that the not only the misclassification error rate decreases in all the different scenarios but also the other methods performs better in different scenarios. For the first scenario when there is a low variance, the adaptive approach performs better than the other competitor methods except for the Shape Contamination Model. For the second scenario, the adaptive approach outperform all the different methods.

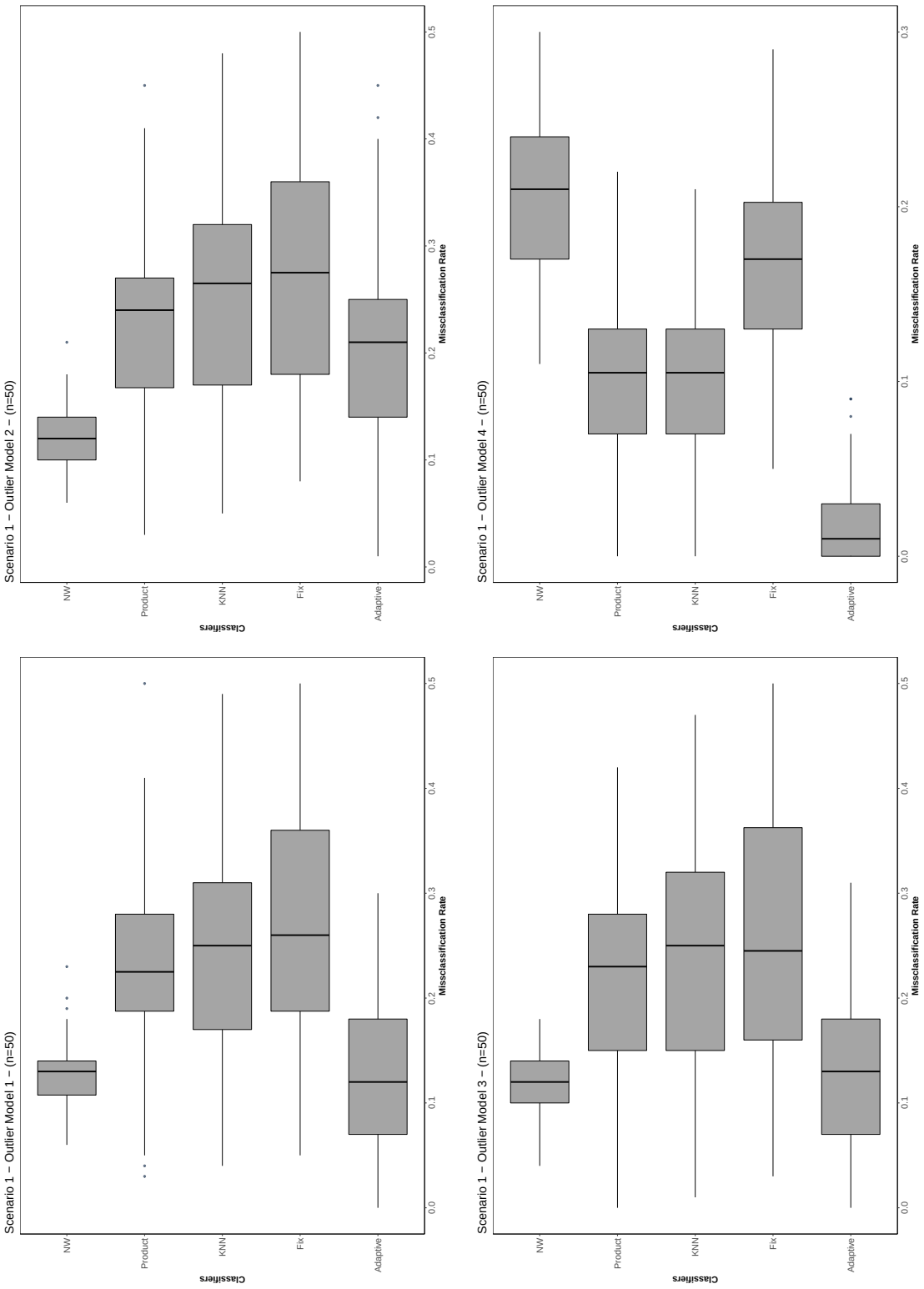


Figure 5.5: Boxplots of misclassification error rates across different contamination models obtained for the first scenario using the PLS semimetric with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

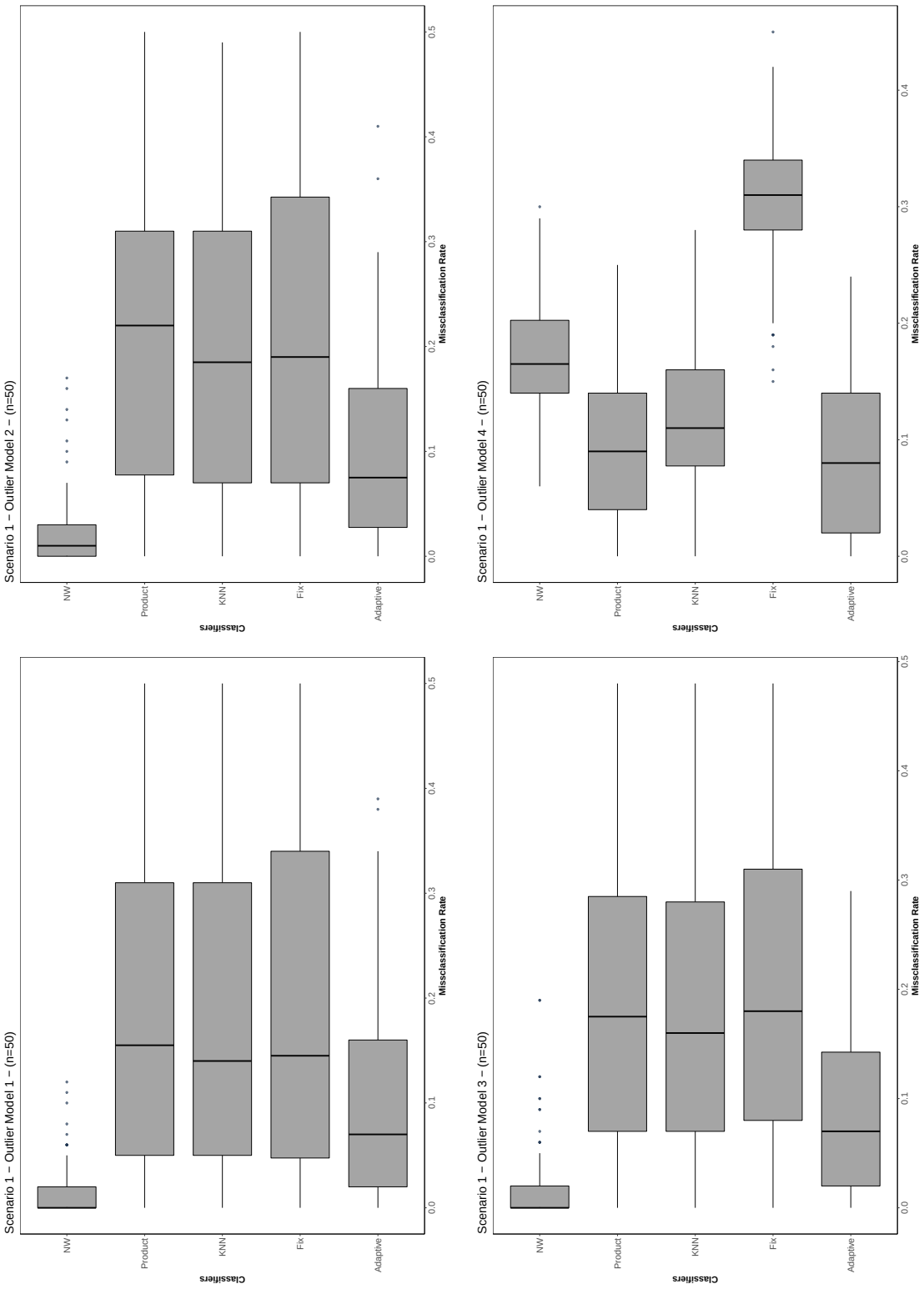


Figure 5.6: Boxplots of misclassification error rates obtained for the second scenario using the PLS semimetric and different contamination model with a sample size $n_1 = n_2 = 50$ for a (top-left:) Uncontaminated Model, (top-right:) The Asymmetric Contaminated Model, (bottom-left:) The Linear Peak Contaminated Model and (bottom-right:) Shape Contamination Model

To investigate the performance of different metrics, we conduct a simulation experiment in which we consider the partial least square semimetric. Figure 5.5 and Figure 5.6 show the boxplots of the misclassification error rates over the different outlier models under the first scenario and using the partial least square semimetric. As in the previous scenarios, the dimension is selected in a way that minimises the mean integrated square error (MISE) using the $\mathcal{L}^2$-norm. The smoothing parameters h and k are selected such that they minimise the misclassification error rate. For partial least squares, the number of factors in each simulated dataset is selected in a way that minimises the misclassification error rate.

When there is no contamination in the, data the Nadaraya-Watson type estimates perform as good as the adaptive approach. However, when we consider an asymmetric contamination model, the Nadaraya-Watson regression estimate performs better than the other methods; we can observe the same behaviour under the third model but not under the atypicals observations, where the adaptive approach outperforms the different methods.

For the second scenario, under the no contamination model, the methods based on Nadaraya-Watson estimate perform better than all the other methods. However, the variability of the misclassification error rate in the others classifiers increases. This can be explained because the adaptive approach now requires us to explore the data in a higher dimensional space. Under the asymmetric and linear peak contamination models the Nadaraya-Watson estimate performs best but not under the shape contamination model where the adaptive approach perform best. In all the examples, the variability of the misclassification error is higher.

From the second scenario, we conclude that the use of a different semimetric influences the behaviour of the classifier along with the number of factors we consider. The variability of the misclassification error rate is higher than when we consider a semimetric based on principal components. Under this semimetric the adaptive approach does not perform as well as using principal components and the variability of the misclassification error rate is higher.

Table 5.1: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PCA semimetric.

		Scenario 1 - PCA Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
$n_1 = n_2 = 50$					
Adaptive	Error	0.0113 (0.0001)	0.0057 (0.0001)	0.0077 (0.0001)	0.0193 (0.0004)
	Sensitivity	0.9874 (0.0002)	0.9951 (0.0001)	0.9922 (0.0002)	0.9793 (0.0007)
	Specificity	0.9903 (0.0002)	0.9937 (0.0001)	0.9928 (0.0002)	0.9827 (0.0004)
Fix	Error	0.0389 (0.0002)	0.0267 (0.0002)	0.0570 (0.0003)	0.0355 (0.0006)
	Sensitivity	0.9609 (0.0005)	0.9726 (0.0005)	0.9435 (0.0007)	0.9688 (0.0007)
	Specificity	0.9626 (0.0005)	0.9750 (0.0004)	0.9442 (0.0007)	0.9612 (0.0008)
KNN	Error	0.0413 (0.0003)	0.0457 (0.0003)	0.0446 (0.0003)	0.0157 (0.0002)
	Sensitivity	0.9578 (0.0005)	0.9532 (0.0006)	0.9564 (0.0006)	0.9834 (0.0004)
	Specificity	0.9607 (0.0006)	0.9567 (0.0006)	0.9556 (0.0006)	0.9857 (0.0003)
Product	Error	0.0276 (0.0002)	0.0305 (0.0003)	0.0611 (0.0004)	0.0908 (0.0011)
	Sensitivity	0.9715 (0.0005)	0.9688 (0.0006)	0.9394 (0.0009)	0.9190 (0.0015)
	Specificity	0.9745 (0.0005)	0.9712 (0.0005)	0.9401 (0.0007)	0.9011 (0.0013)
NW	Error	0.1492 (0.0008)	0.1500 (0.0009)	0.1478 (0.0011)	0.2679 (0.0023)
	Sensitivity	0.8558 (0.0016)	0.8529 (0.0018)	0.8559 (0.0022)	0.7728 (0.0054)
	Specificity	0.8505 (0.0018)	0.8527 (0.0020)	0.8538 (0.0019)	0.7109 (0.0029)

Table 5.2: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PCA semimetric.

		Scenario 2 - PCA Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
$n_1 = n_2 = 50$					
Adaptive	Error	0.0483 (0.0028)	0.0424 (0.0025)	0.0401 (0.0024)	0.0285 (0.0015)
	Sensitivity	0.9524 (0.0031)	0.9555 (0.0029)	0.9594 (0.0024)	0.9723 (0.0016)
	Specificity	0.9520 (0.0028)	0.9611 (0.0026)	0.9613 (0.0026)	0.9718 (0.0017)
Fix	Error	0.1189 (0.0008)	0.0698 (0.0005)	0.0676 (0.0007)	0.1408 (0.0033)
	Sensitivity	0.8872 (0.0011)	0.9330 (0.0010)	0.9350 (0.0009)	0.8677 (0.0034)
	Specificity	0.8775 (0.0015)	0.9296 (0.0010)	0.9319 (0.0013)	0.8527 (0.0037)
KNN	Error	0.1505 (0.0007)	0.1176 (0.0008)	0.1167 (0.0010)	0.1720 (0.0014)
	Sensitivity	0.8533 (0.0012)	0.8790 (0.0011)	0.8807 (0.0016)	0.8381 (0.0022)
	Specificity	0.8484 (0.0012)	0.8880 (0.0012)	0.8885 (0.0014)	0.8221 (0.0020)
Product	Error	0.1457 (0.0009)	0.0225 (0.0004)	0.1235 (0.0009)	0.1509 (0.0036)
	Sensitivity	0.8590 (0.0013)	0.9785 (0.0006)	0.8789 (0.0014)	0.8526 (0.0032)
	Specificity	0.8512 (0.0011)	0.9772 (0.0006)	0.8761 (0.0011)	0.8474 (0.0047)
NW	Error	0.3550 (0.0015)	0.2779 (0.0020)	0.2742 (0.0026)	0.3085 (0.0032)
	Sensitivity	0.6479 (0.0017)	0.7248 (0.0026)	0.7254 (0.0024)	0.7271 (0.0059)
	Specificity	0.6438 (0.0016)	0.7232 (0.0025)	0.7291 (0.0035)	0.6732 (0.0033)

Table 5.3: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 200$ and using the PCA semimetric.

		Scenario 1 - PCA Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
$n_1 = n_2 = 200$					
Adaptive	Error	0.0055750 (0.00002119)	0.0023250 (0.00000710)	0.0038000 (0.00002165)	0.0127250 (0.00006428)
	Sensitivity	0.9944189 (0.00003096)	0.9975131 (0.00001416)	0.9964598 (0.00002611)	0.9889960 (0.00007408)
	Specificity	0.9944658 (0.00002841)	0.9978574 (0.00001013)	0.9959682 (0.00003069)	0.9856616 (0.00010057)
Fix	Error	0.0134250 (0.00002483)	0.0064500 (0.00002022)	0.0197750 (0.00004496)	0.0075750 (0.00003137)
	Sensitivity	0.9860838 (0.00006155)	0.9935878 (0.00004157)	0.9801720 (0.00007400)	0.9940536 (0.00003489)
	Specificity	0.9871877 (0.00004661)	0.9935765 (0.00003031)	0.9804268 (0.00008765)	0.9908831 (0.00006469)
KNN	Error	0.0172750 (0.00003032)	0.0166250 (0.00003136)	0.0165750 (0.00004339)	0.0013500 (0.00000687)
	Sensitivity	0.9827077 (0.00005663)	0.9835072 (0.00006223)	0.9829104 (0.00006186)	0.9989557 (0.00001005)
	Specificity	0.9828539 (0.00005831)	0.9833543 (0.00005439)	0.9840270 (0.00006647)	0.9983567 (0.00000954)
Product	Error	0.0276000 (0.00019418)	0.0305000 (0.00031389)	0.0611000 (0.00040383)	0.0908000 (0.00111855)
	Sensitivity	0.9714718 (0.00046248)	0.9688323 (0.00059022)	0.9394239 (0.00086531)	0.9190054 (0.00150487)
	Specificity	0.9745045 (0.00046624)	0.9711658 (0.00049015)	0.9401139 (0.00070317)	0.9010961 (0.00128018)
NW	Error	0.1309750 (0.00021476)	0.1299250 (0.00023225)	0.1290250 (0.00019948)	0.2014500 (0.00053474)
	Sensitivity	0.8710554 (0.00045562)	0.8690441 (0.00055595)	0.8708578 (0.00037800)	0.8470179 (0.00164171)
	Specificity	0.8683985 (0.00048686)	0.8728521 (0.00055227)	0.8722698 (0.00045815)	0.7656601 (0.00100189)

Table 5.4: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 200$ and using the PCA semimetric.

		Scenario 2 - PCA Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
$n_1 = n_2 = 200$					
Adaptive	Error	0.0253000 (0.00089221)	0.0424000 (0.00253156)	0.0038000 (0.00002165)	0.0020250 (0.00001208)
	Sensitivity	0.9742197 (0.00093824)	0.9555407 (0.00288047)	0.9964598 (0.00002611)	0.9980069 (0.00001548)
	Specificity	0.9753072 (0.00089311)	0.9610702 (0.00257790)	0.9959682 (0.00003069)	0.9979571 (0.00001553)
Fix	Error	0.0351500 (0.00043937)	0.0698000 (0.00047875)	0.0197750 (0.00004496)	0.0176250 (0.00039298)
	Sensitivity	0.9647313 (0.00049022)	0.9330186 (0.00096557)	0.9801720 (0.00007400)	0.9842548 (0.00037650)
	Specificity	0.9651832 (0.00048275)	0.9295762 (0.00095311)	0.9804268 (0.00008765)	0.9806314 (0.00046056)
KNN	Error	0.1080000 (0.00016780)	0.1176000 (0.00075580)	0.0165750 (0.00004339)	0.0839000 (0.00021784)
	Sensitivity	0.8919534 (0.00022027)	0.8789700 (0.00109849)	0.9829104 (0.00006186)	0.9181026 (0.00033420)
	Specificity	0.8925373 (0.00031203)	0.8879713 (0.00120695)	0.9840270 (0.00006647)	0.9145312 (0.00027226)
Product	Error	0.1457000 (0.00093991)	0.0225000 (0.00042096)	0.0611000 (0.00040383)	0.1509000 (0.00360625)
	Sensitivity	0.8589650 (0.00133268)	0.9785460 (0.00058394)	0.9394239 (0.00086531)	0.8526347 (0.00318199)
	Specificity	0.8511947 (0.00108299)	0.9772352 (0.00059924)	0.9401139 (0.00070317)	0.8474109 (0.00470302)
NW	Error	0.2409750 (0.00035226)	0.2779000 (0.00201878)	0.1290250 (0.00019948)	0.2495500 (0.00038755)
	Sensitivity	0.7573740 (0.00047423)	0.7248364 (0.00257061)	0.8708578 (0.00037800)	0.7840387 (0.00197260)
	Specificity	0.7615804 (0.00046515)	0.7231674 (0.00252790)	0.8722698 (0.00045815)	0.7306080 (0.00111609)

Table 5.5: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PLS semimetric.

		Scenario 1 - PLS Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
Adaptive	Error	0.1256 (0.0051)	0.1995 (0.0084)	0.1299 (0.0054)	0.0169 (0.0005)
	Sensitivity	0.8783 (0.0052)	0.8008 (0.0090)	0.8693 (0.0060)	0.9844 (0.0006)
	Specificity	0.8731 (0.0059)	0.8048 (0.0090)	0.8737 (0.0057)	0.9827 (0.0007)
Fix	Error	0.2707 (0.0145)	0.2774 (0.0133)	0.2665 (0.0153)	0.1711 (0.0032)
	Sensitivity	0.7295 (0.0142)	0.7239 (0.0135)	0.7337 (0.0148)	0.8412 (0.0061)
	Specificity	0.7305 (0.0150)	0.7226 (0.0135)	0.7344 (0.0160)	0.8308 (0.0052)
KNN	Error	0.2467 (0.0103)	0.2494 (0.0078)	0.2419 (0.0104)	0.1040 (0.0019)
	Sensitivity	0.7509 (0.0103)	0.7516 (0.0086)	0.7607 (0.0111)	0.8963 (0.0031)
	Specificity	0.7611 (0.0116)	0.7558 (0.0091)	0.7619 (0.0113)	0.9012 (0.0029)
Product	Error	0.2253 (0.0077)	0.2264 (0.0056)	0.2191 (0.0072)	0.1032 (0.0022)
	Sensitivity	0.7724 (0.0077)	0.7755 (0.0070)	0.7818 (0.0082)	0.8947 (0.0036)
	Specificity	0.7816 (0.0091)	0.7801 (0.0069)	0.7866 (0.0082)	0.9040 (0.0025)
NW	Error	0.1256 (0.0010)	0.1207 (0.0009)	0.1230 (0.0008)	0.2055 (0.0020)
	Sensitivity	0.8780 (0.0025)	0.8835 (0.0021)	0.8757 (0.0017)	0.8033 (0.0041)
	Specificity	0.8771 (0.0018)	0.8796 (0.0014)	0.8829 (0.0016)	0.7947 (0.0028)

Table 5.6: Mean error rate, sensitivity rate, mean specificity and their variance in parentheses, for the second scenario over 100 simulations with sample sizes $n_1 = n_2 = 50$ and using the PLS semimetric.

		Scenario 2 - PLS Semimetric			
		Uncontaminated	Asymmetric	Linear Peak	Shape
Adaptive	Error	0.1047 (0.0099)	0.1008 (0.0081)	0.0913 (0.0072)	0.0861 (0.0045)
	Sensitivity	0.8967 (0.0100)	0.9026 (0.0083)	0.9124 (0.0069)	0.9282 (0.0059)
	Specificity	0.8954 (0.0103)	0.8982 (0.0085)	0.9070 (0.0079)	0.9106 (0.0064)
Fix	Error	0.1951 (0.0250)	0.2120 (0.0219)	0.1989 (0.0205)	0.3062 (0.0038)
	Sensitivity	0.8059 (0.0255)	0.7849 (0.0220)	0.8018 (0.0211)	0.6994 (0.0050)
	Specificity	0.8056 (0.0249)	0.7930 (0.0223)	0.8021 (0.0205)	0.6990 (0.0047)
KNN	Error	0.1814 (0.0211)	0.2014 (0.0187)	0.1775 (0.0167)	0.1156 (0.0033)
	Sensitivity	0.8233 (0.0218)	0.8009 (0.0188)	0.8244 (0.0176)	0.8922 (0.0054)
	Specificity	0.8176 (0.0211)	0.8016 (0.0193)	0.8251 (0.0169)	0.8890 (0.0057)
Product	Error	0.1876 (0.0213)	0.2095 (0.0194)	0.1848 (0.0172)	0.0944 (0.0037)
	Sensitivity	0.8160 (0.0220)	0.7917 (0.0200)	0.8166 (0.0183)	0.9056 (0.0041)
	Specificity	0.8119 (0.0212)	0.7937 (0.0196)	0.8178 (0.0169)	0.9081 (0.0043)
NW	Error	0.0139 (0.0006)	0.0219 (0.0013)	0.0193 (0.0014)	0.1707 (0.0025)
	Sensitivity	0.9870 (0.0005)	0.9776 (0.0013)	0.9812 (0.0016)	0.8367 (0.0037)
	Specificity	0.9856 (0.0008)	0.9796 (0.0015)	0.9813 (0.0014)	0.8299 (0.0036)

We summarise the misclassification error rate, the sensitivity and the specificity of the simulations for the first scenario and using the principal component semi-norm in Table 5.3 and similarly for the second scenario in Table 5.4. In this scenario, it can be seen that the adaptive approach outperforms other competitors across different contamination models.

For the partial least squares semimetric, the simulation results of the misclassification error rate, the sensitivity and the specificity are reported in Table 5.5 and Table 5.6 for the first and second scenario, respectively. The performance of the adaptive approach deteriorates somewhat under this semimetric but still performs better under the shape contamination model.

To conclude this simulation experiment, we can observe that the use of the different semimetrics influences the behaviour of the classifier, not only in the misclassification error rate but also in the variability of the misclassification error rate in all classifiers. The adaptive approach outperforms all the other competitors under a principal component semimetric and in both scenarios. When there are differences between the semimetric and the contamination model, the adaptive approach deteriorates rapidly exhibiting a large amount of variability and a higher misclassification error rate. In practice, a classifier with a lower variability and lower misclassification error rate is preferred.

5.8 Real Data Study

In this section, we present three different real datasets previously introduced in Section 2.4, to illustrate the performance of the proposed adaptive kernel method under the same semimetric but using different norms.

The first data example concerns classifying the orange juice spectra data. As we mentioned before, we form the groups to be two equally balanced groups using the sucrose content. Defining the first population Π_0, of size $n_0 = 109$ (sucrose > 40) and the second population Π_1, formed by $n_1 = 109$ curves with (sucrose < 40) and the group membership as the binary response to be predicted. For this dataset, 99.96% of the variability can be explained with five principal components scores. We showed in Figure 5.7 the kernel density estimates of the first four projection scores; the densities are not normal and the third principal component score appears to be bimodal. For this particular example, we can observe that the difference between each pair of densities is not only limited by the location and scale, but

also to the shape of the densities.

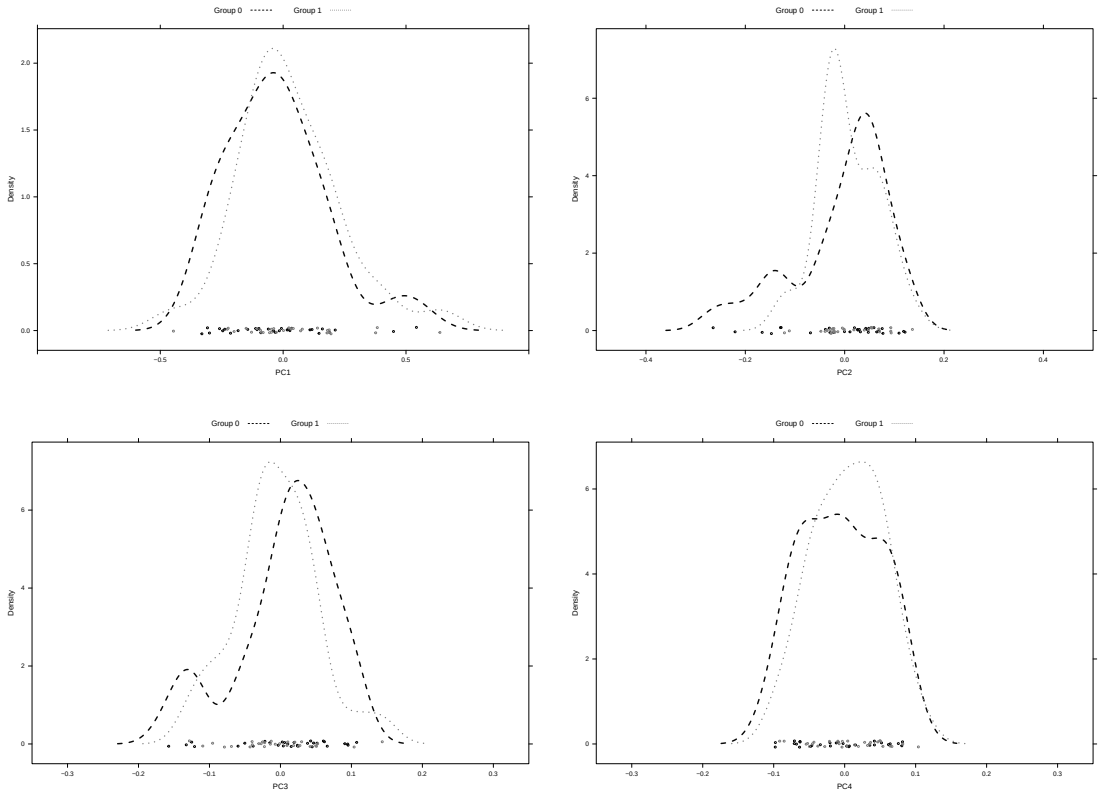


Figure 5.7: Kernel density estimates for the first four principal component scores for the orange juice dataset.

To illustrate the performance of applying the adaptive approach under different semimetrics, we consider the adaptive kernel method where the values of the smoothing parameter h and k are selected such that they minimise the misclassification error rate (detailed in Algorithm 3). In terms of the norm, we vary q over a set $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2\}$.

In the second example, we analyse the NIR spectra of gasoline dataset. The dataset contains two equally balanced groups according to the octane content and the predictors are diffuse reflectance observed at from 900 to 1700nm in 2nm intervals giving 401 wavelengths. The first population Π_0 is formed by $n_0 = 30$ curves with (octane < 88) and the second population Π_1 is formed by $n_1 = 30$ curves with (octane > 88). For the NIR spectra gasoline dataset a total of 10 principal component scores explain 99.21% of the variability in the data. To investigate the densities of the principal component scores, density estimates of the first four principal component scores can be seen in Figure 5.8. From Figure 5.8 the kernel density estimate is far from normal and the first, second and third projection of the principal component scores appear to be bimodal.

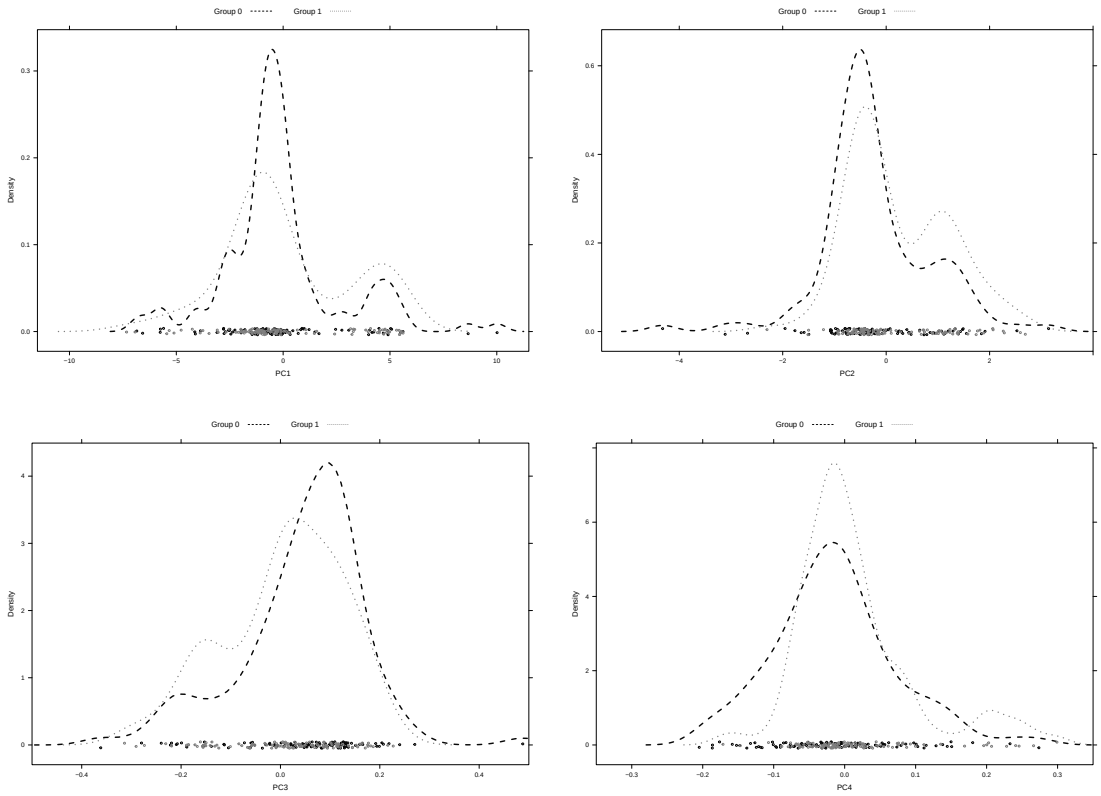


Figure 5.8: Kernel density estimates for the first four principal component scores for the NIR gasoline spectra.

As in the first real dataset, we classify the observations using the adaptive kernel method, where the values of the smoothing parameter h and k are selected such that minimises the misclassification error rate. Additionally, we investigate the performance of different fractional norms. For such proposer we consider to vary the norm starting from $q = 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1$ and 2. This is to see if fractional norms also minimises the misclassification error rate.

In Section 5.6.1 we extended the adaptive approach to more than two groups. The third real example consists of more than two groups and is the phoneme dataset introduced in Section 2.4. The phoneme dataset consist of 400 curves in each group. From each speech frame, we compute the log-periodogram, which is the one suitable for speech recognition. Thus the data consists of 1200 log-periodograms of length 256, with known class memberships. For this real dataset, we construct an adaptive approach based on all the observations, for each of these examples, we applied the proposed adaptive approach across different norms and present the results in Table 5.7.

Table 5.7: Missclassification error rate, sensitivity and specificity rates for the three real datasets and different metrics.

	Real Dataset 1				Real Dataset 2				Real Dataset 3	
	Error Rate	Accuracy	Sensitivity	Specificity	Error Rate	Accuracy	Sensitivity	Specificity	Error Rate	Accuracy
Norms										
0.1	0.1167	0.8833	0.8710	0.8966	0.0275	0.9725	0.9640	0.9813	0.0167	0.9833
0.2	0.1167	0.8833	0.8710	0.8966	0.0321	0.9679	0.9636	0.9722	0.0183	0.9817
0.3	0.1167	0.8833	0.8710	0.8966	0.0413	0.9587	0.9464	0.9717	0.0125	0.9875
0.4	0.0333	0.9667	0.9667	0.9667	0.0550	0.9450	0.9450	0.9450	0.0158	0.9842
0.5	0.0167	0.9833	0.9677	1.0000	0.1055	0.8945	0.8772	0.9135	0.0133	0.9867
0.6	0.0500	0.9500	0.9355	0.9655	0.0642	0.9358	0.9358	0.9358	0.0175	0.9825
0.7	0.9833	0.9833	0.9677	1.0000	0.1055	0.8945	0.8772	0.9135	0.0075	0.9925
0.8	0.0167	0.8833	0.8710	0.8966	0.0550	0.9450	0.9533	0.9369	0.0033	0.9967
0.9	0.0500	0.9500	0.9355	0.9655	0.0275	0.9725	0.9640	0.9813	0.0058	0.9942
1	0.0167	0.9833	0.9677	1.0000	0.0367	0.9633	0.9633	0.9633	0.0142	0.9858
2	0.0500	0.9500	0.9355	0.9655	0.1055	0.8945	0.8772	0.9135	0.0133	0.9867

As we can see from Table 5.7, the proposed adaptive approach works particularly well for functional classification and performs better when we consider using different norms. Also, the proposed adaptive approach perform better than using the traditional $\mathcal{L}^2$ norm. Thus, a higher accuracy can be obtained with a fractional norm.

Overall, this demonstrates how fractional norms combined with the adaptive approach allow us to achieve better results in terms of classification. As we saw for all the datasets, the use of the fractional norm increases the accuracy of the classifier.

5.9 Conclusions

Principal Component Analysis plays a very important role in functional data. We saw that in general it is not possible to define a density for functional data, but is possible to define a meaningful concept of density for a specific scale which is linked to a particular dimension. Throughout this chapter we have introduced an adaptive density Bayesian classifier using density of log ratios of functional principal component scores.

We studied the performance of different semimetrics linked to a particular dimension. Techniques such as R-fold Cross-Validation are very useful to determine the number of principal components when combined with the definition of Integrated Squared Error. By means of a simulation study, we saw the performance of the proposed classifier under two semimetrics and different contamination models. We explore two different semimetrics important in the study of functional data: the semimetric based on principal components scores and the semimetric based on partial least squares. Simulation results showed that under different scenarios the adaptive approach outperforms other competitors. Fractional norms can be combined with the adaptive approach to achieve better results in classification. Using real datasets we showed how fractional norms can increase the accuracy of the adaptive classifier.

Even though we consider the semimetric based on principal components it could be useful to consider a semimetric based on a robust approach and compare our approach. We limited ourselves to the case of balanced observations but we could also consider the performance of our approach when this is not the case.

Chapter 6

Summary and Further Research

Functional data analysis has an inherent nonparametric approach and most of the methodologies we developed do not assume any parametric distribution. Functional data possesses multiple challenges and we have tackled a number of them. This thesis deal with supervised classification problems for functional data and proposes two main contributions. The first contribution is based on Nearest Neighbours methods and the second contribution is based on densities ratios and different semimetrics. We summarise our contributions by chapters.

In Chapter 2, by using a Gaussian Process we explored the flexibility to generate different functions and we utilised the form of the covariance function to generate rough or smooth curves according to its parametric form. We explored a method to simulate functional data based on Fourier basis. We controlled the basis coefficients to be chosen randomly from a Normal distribution with a decaying variance as the the number of coefficients increases an we simulate functional data using two different scenarios. We also included different methods to generate atypicals for functional data. Finally, we introduced functional real (balance and imbalanced) datasets which are later used for testing our proposed methodologies. Overall, the chapter provides a basis for the simulations and applications of methodologies throughout the thesis.

In Chapter 3, we study the k-RNN as an alternative of the k-NN for functional data and with the use of a running example, we explore different features of the classifier including choosing the value of k and the exploration with different functional depths. Motivated by the simplicity of the k-RNN classifier, we investigated how this classifier can be interpreted in terms of conditional probabilities and as a moving average. We also saw how, by considering conditional probabilities that the signed depth of a curve belongs to a particular group, we can

construct point-wise confidence intervals for the estimated conditional probability. To study the k-RNN more in detail, we followed a generalized additive model approach based on the signed depth and the signed distance to the mode. We extended this classifier to more than two classes and by means of a numerical simulation study, we observe an advantage of the proposed classifier under different contamination scenarios and against different competitors, with respect to the misclassification rate for both real and simulated dataset.

Within Chapter 4, we explored the imbalanced problem in which the number of observations in the majority class exceeds the number of observations in the minority class. We focused on borderline observations in terms of the nature of the classification rule and proposed a new methodology to strengthen the borderline minority observations in terms of the functional principal component scores. We also proposed generating new curves by considering a linear combination of the observations in the border and the observations closest in depth. We then applied our proposed methodology to simulated and real datasets and we saw that the proposed method outperforms the standard methods in terms of the misclassification error.

In Chapter 5, we proposed a nonparametric adaptive density Bayesian classifier using log density ratios of functional principal component scores based on different semimetrics for a fixed dimension. Selecting the number of principal components is an important model selection problem in almost all practical contexts of functional data analysis; for the Bayesian classifier we selected a particular dimension using Cross Validation. Thus, working in a finite dimensional space allows us to estimate a density linked to a particular dimension. By means of a simulation study, we investigated the performance of the proposed classifier under two semimetrics: PCA and PLS. We saw that the choice of the semimetric plays an important role in classification. We then tested our classifier in real and simulated datasets. For the real datasets, the use of the fractional norm increased the accuracy of the classifier.

6.1 Limitations and Further work

In this section, we provide a discussion of several issues that are not addressed in the thesis and worth investigating in the near future. A common limitation is computational power, however recent advances mean that computations of multiple integrals are less burdensome.

These types of calculations are needed to compute the signed depth, hence apply to our methodology.

6.1.1 *k*- Ranked Nearest Neighbours for functional data

The concept of ranks in functional data provides a monotonic ordering for the data. In Chapter 3, we proposed a generalized additive model based on the signed depth and the distance to the mode. However, in the *k*-RNN we restricted ourselves to the use the Fraiman and Muniz (FM) depth. In recent years, new measures of depth for functional data have been developed, for example we can mentioned the modified band depth by Arribas-Gil and Romo (2014) and the modified epigraph index by López-Pintado and Romo (2009). It is important to explore the behaviour of the proposed classifier under different depth measures to those explored in the thesis.

Although, we ran our simulations with a particular degree of separation in the data, for example $\delta = 0.35$ in our running example, an alternatives is to explore how effective the methods become when there is a different degree of separation in the data.

When the signed depth and the distance to the mode is visualised on the plane, the proposed method can help us to identify clusters and outlier observations. As a further work, we can explore the problem of clustering. For instance, functional depth as the FM depth previously introduced in Chapter 3 is the basis of robust clustering methods such as K-means.

6.1.2 Dealing with Imbalanced Observations for functional data

In Chapter 4 we proposed an approach which involves oversampling the distribution of the principal component scores. We focused on observations that are close to the border and we proposed a new method that shrinks the observations that are closer to the original curve in the border set. We also explored the borderline observations in terms of the classification rule where the borderline observations can be defined in terms of a distance function. Thus, we can explore different distance functions in a the near future and compare the performance of our approach with these methods.

Even though the bootstrap approach is the standard method we compared to, there are other sampling methods and it could be useful to consider and evaluate the performance of our approach in comparison to these. In addition, we have limited ourselves to the case of

imbalanced observations; we could also consider the performance of our approach when this is not the case.

6.1.3 Principal Component Analysis for Functional Data

In the presence of outliers, the covariance operator is not robust. As a result, the estimated functional principal components extracted from the covariance operator can lead to wrong estimates. In Chapter 5, we proposed a nonparametric adaptive density Bayesian classifier using a density of log ratios of functional principal component scores, based on different semimetrics for a fixed dimension.

For further research, we can consider a robust approach of the functional principal components of which there is little discussion about in the literature. Although the proposed Bayesian classifier was tested on balanced datasets, we can use the flexibility to study its performance under the imbalanced case, by modifying the prior probabilities. We could also investigate the performance of our approach when this is not the case and with a suitable semimetric for classification.

The simulations in Chapter 5 are structured with the data and this may be overoptimistic. However, to mimic what happens in practice and to incorporate a test set in the simulations, we explored two different approaches. The first approach started by considering an independent test set of size $n_{\mathcal{T}est} = 100$ generated from the same model and containing equally balanced of observations from each group. In this approach, the dimension of the data is obtained by minimising the MISE. We trained our classifiers in our simulated data and we saved the optimised parameters, which we then used in the simulated test set to predict the new 100 observations. The second approach consisted of having an independent test set of size $n_{\mathcal{T}est} = 100$ containing equally balanced observations from each group but instead of determining the dimension using MISE, we used a fix the dimension. Again, we saved the optimised parameters for the competing methods and the proposed adaptive approach. As a preliminary numerical study, we implemented both approaches. For the first approach we implemented the uncontaminated case as we described in Section 5.7. Table 6.1 shows the distribution of the number of principal components of the simulated test data and the number of times that the correct dimension is selected. In this case we can observe that the 96 of the times it selects the principal component that counts for most of the variability and minimises the MISE.

Table 6.1: Distribution of the number of principal components in the simulations including a test set and minimising the mean integrate square error.

PCs	3	4	5	6
	2/100	1/100	96/100	1/100

In terms of the misclassification error rate, we summarise the results of both approaches in Table 6.2 where we show the mean error rate, sensitivity rate, specificity and their variances. In the first approach, we can observe that the mean misclassification error for the proposed adaptive is 0.0144, for the fix kernel rule method it is 0.0406, for the k-NN kernel method it is 0.0431, for the product kernel method it is 0.0276 and for the NW regression approach it is 0.1517. All these exhibit a higher misclassification error rate than that in Section 5.7.2.

The second approach, which does not require us to select the dimension by minimising the MISE, exhibits the mean misclassification value for the proposed adaptive as 0.0136, for the fix kernel rule method classifier it is 0.0402, for the k-NN kernel method it is 0.0425, for the product kernel method it is 0.0276 and for the NW regression approach it is 0.1513. We saw that the first approach shows a higher variability when applied to the new data. In general, we observe that both approaches may be much more variable when applied to new data compared to reclassifying the training data.

As further work, we can investigate the behaviour of the proposed adaptive approach under different scenarios and when a simulated test dataset is available. Finally, we can explore fractional norms combined with the adaptive approach in our simulations, since we saw that the use of fractional norms of the adaptive with real data achieve better results in terms of classification.

Table 6.2: Mean error rate, sensitivity rate, mean specificity and their variances in parentheses, for the first scenario testing over 100 simulated test dataset $n_1 = n_2 = 50$ and using the PCA semimetric.

		Scenario 1 (Uncontaminated) - PCA Semimetric	
		Approach 1	Approach 2
$n_1 = n_2 = 50$			
Adaptive	Error	0.0144 (0.0002)	0.0136 (0.0002)
	Sensitivity	0.9852 (0.0003)	0.9879 (0.0002)
	Specificity	0.9863 (0.0002)	0.9853 (0.0003)
Fix	Error	0.0406 (0.0003)	0.0402 (0.0002)
	Sensitivity	0.9608 (0.0007)	0.9600 (0.0005)
	Specificity	0.9592 (0.0005)	0.9607 (0.0005)
KNN	Error	0.0431 (0.0004)	0.0425 (0.0003)
	Sensitivity	0.9577 (0.0007)	0.9567 (0.0006)
	Specificity	0.9585 (0.0006)	0.9583 (0.0006)
Product	Error	0.0276 (0.0003)	0.0268 (0.0003)
	Sensitivity	0.9744 (0.0005)	0.9721 (0.0007)
	Specificity	0.9729 (0.0005)	0.9738 (0.0005)
NW	Error	0.1517 (0.0007)	0.1513 (0.0010)
	Sensitivity	0.8522 (0.0018)	0.8488 (0.0017)
	Specificity	0.8507 (0.0020)	0.8506 (0.0020)

Bibliography

Akbani, R., Kwek, S., and Japkowicz, N. Applying support vector machines to imbalanced datasets. In *European conference on machine learning*, pages 39–50. Springer, 2004.

Akhiezer, N. I. and Glazman, I. M. *Theory of linear operators in Hilbert space*. Courier Corporation, 2013.

Anderson, T. Some nonparametric multivariate procedures based on statistically equivalent blocks. *Multivariate Analysis*, 1:5–27, 1966.

Arribas-Gil, A. and Romo, J. Shape outlier detection and visualization for functional data: the outliergram. *Biostatistics*, 15(4):603–619, 2014.

Bagui, S. C. and Pal, N. R. A multistage generalization of the rank nearest neighbor classification rule. *Pattern Recognition Letters*, 16(6):601–614, 1995.

Bagui, S. C., Mehra, K., and Rao, M. A nearest neighbour classification rule for multiple observations based on a sub-sample approach. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 316–332, 1995.

Bagui, S. C., Bagui, S., Pal, K., and Pal, N. R. Breast cancer detection using rank nearest neighbor classification rules. *Pattern recognition*, 36(1):25–34, 2003.

Baillo, A. and Cuevas, A. Supervised functional classification: A theoretical remark and some comparisons. *arXiv preprint arXiv:0806.2831*, 2008.

Bowman, A. W. and Azzalini, A. *Applied smoothing techniques for data analysis: the kernel approach with S-Plus illustrations*, volume 18. OUP Oxford, 1997.

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

Breiman, L., Meisel, W., and Purcell, E. Variable kernel estimates of multivariate densities. *Technometrics*, 19(2):135–144, 1977.

Breiman, L., Friedman, J., Stone, C. J., and Olshen, R. A. *Classification and regression trees*. CRC press, 1984.

Buja, A., Hastie, T., and Tibshirani, R. Linear smoothers and additive models. *The Annals of Statistics*, pages 453–510, 1989.

Cacoullos, T. Estimation of a multivariate density. *Annals of the Institute of Statistical Mathematics*, 18(1):179–189, 1966.

Caruana, R. and Niculescu-Mizil, A. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM, 2004.

Cattell, R. B. The scree test for the number of factors. *Multivariate behavioral research*, 1 (2):245–276, 1966.

Cérou, F. and Guyader, A. Nearest neighbor classification in infinite dimension. *ESAIM: Probability and Statistics*, 10:340–355, 2006.

Chacón, J. E., Duong, T., and Wand, M. Asymptotics for general multivariate kernel density derivative estimators. *Statistica Sinica*, pages 807–840, 2011.

Chawla, N. V. Data mining for imbalanced datasets: An overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

Chawla, N. V., Lazarevic, A., Hall, L. O., and Bowyer, K. W. Smoteboost: Improving prediction of the minority class in boosting. In *European conference on principles of data mining and knowledge discovery*, pages 107–119. Springer, 2003.

Chawla, N. V., Japkowicz, N., and Kotcz, A. Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6, 2004.

Chen, S., He, H., and Garcia, E. A. Ramoboost: ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10):1624–1642, 2010.

Conover, W. J. and Conover, W. J. Practical nonparametric statistics. 1980.

Copas, J. B. Regression, prediction and shrinkage. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 311–354, 1983.

Cressie, N. A. Statistics for spatial data: Wiley series in probability and mathematical statistics. *Find this article online*, 1993.

Cuevas, A., Febrero, M., and Fraiman, R. On the use of the bootstrap for estimating functions with functional data. *Computational statistics & data analysis*, 51(2):1063–1074, 2006.

Cuevas, A., Febrero, M., and Fraiman, R. Robust estimation and classification for functional data via projection-based depth notions. *Computational Statistics*, 22(3):481–496, 2007.

Dauxois, J., Pousse, A., and Romain, Y. Asymptotic theory for the principal component analysis of a vector random function: some applications to statistical inference. *Journal of multivariate analysis*, 12(1):136–154, 1982.

Delaigle, A. and Hall, P. Defining probability density for a distribution of random functions. *The Annals of Statistics*, pages 1171–1193, 2010.

Delaigle, A. and Hall, P. Achieving near perfect classification for functional data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):267–286, 2012.

Demirtas, H. and Hedeker, D. Generating multivariate continuous data via the notion of nearest neighbors. *Journal of Applied Statistics*, 38(1):47–55, 2011.

Demmler, A. and Reinsch, C. Oscillation matrices with spline smoothing. *Numerische Mathematik*, 24(5):375–382, 1975.

Devroye, L. P. The uniform convergence of the nadaraya-watson regression function estimate. *Canadian Journal of Statistics*, 6(2):179–191, 1978.

Donoho, D. L. Breakdown properties of multivariate location estimators. Technical report, Technical report, Harvard University, Boston. URL http://www-stat. stanford. edu/˜ donoho/Reports/Oldies/BPMLE. pdf, 1982.

Donoho, D. L. and Gasko, M. Breakdown properties of location estimates based on halfspace depth and projected outlyingness. *The Annals of Statistics*, 20(4):1803–1827, 1992.

Efron, B. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

Efron, B. *The jackknife, the bootstrap and other resampling plans*. SIAM, 1982.

Efron, B. and Tibshirani, R. J. The jackknife. In *An introduction to the bootstrap*, pages 141–152. Springer, 1993.

Escabias, M., Aguilera, A. M., and Valderrama, M. J. Functional pls logit regression model. *Computational Statistics & Data Analysis*, 51(10):4891–4902, 2007.

Estabrooks, A., Jo, T., and Japkowicz, N. A multiple resampling method for learning from imbalanced data sets. *Computational intelligence*, 20(1):18–36, 2004.

Eubank, R. L. *Nonparametric regression and spline smoothing*. CRC press, 1999.

Febrero-Bande, M. and Oviedo de la Fuente, M. Statistical computing in functional data analysis: The R package fda.usc. *Journal of Statistical Software*, 51(4):1–28, 2012. URL `http://www.jstatsoft.org/v51/i04/`.

Ferraty, F. and Vieu, P. Curves discrimination: a nonparametric functional approach. *Computational Statistics & Data Analysis*, 44(1):161–173, 2003.

Ferraty, F. and Vieu, P. *Nonparametric functional data analysis: theory and practice*. Springer Science & Business Media, 2006.

Ferraty, F., Sued, M., and Vieu, P. Mean estimation with data missing at random for functional covariables. *Statistics*, 47(4):688–706, 2013.

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

Fix, E. and Hodges Jr, J. L. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

Forman, G. and Scholz, M. Apples-to-apples in cross-validation studies: pitfalls in classifier performance measurement. *ACM SIGKDD Explorations Newsletter*, 12(1):49–57, 2010.

Fraiman, R. and Muniz, G. Trimmed means for functional data. *Test*, 10(2):419–440, 2001.

Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Fubini, G. Sugli integrali multipli. *Rend. Acc. Naz. Lincei*, 16:608–614, 1907.

García, V., Sánchez, J. S., and Mollineda, R. A. On the effectiveness of preprocessing methods when dealing with different levels of class imbalance. *Knowledge-Based Systems*, 25(1):13–21, 2012.

Gasser, T., Hall, P., and Presnell, B. Nonparametric estimation of the mode of a distribution of random curves. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 60(4):681–691, 1998.

Gervini, D. Robust functional estimation using the median and spherical principal components. *Biometrika*, 95(3):587–600, 2008.

Ghosh, A. K. and Chaudhuri, P. On maximum depth and related classifiers. *Scandinavian Journal of Statistics*, 32(2):327–350, 2005.

Green, P. J. and Silverman, B. W. *Nonparametric regression and generalized linear models: a roughness penalty approach*. CRC Press, 1993.

Guo, X., Yin, Y., Dong, C., Yang, G., and Zhou, G. On the class imbalance problem. In *Natural Computation, 2008. ICNC'08. Fourth International Conference on*, volume 4, pages 192–201. IEEE, 2008.

Hall, P. and Heckman, N. E. Estimating and depicting the structure of a distribution of random functions. *Biometrika*, 89(1):145–158, 2002.

Hall, P. and Vial, C. Assessing the finite dimensionality of functional data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(4):689–705, 2006.

Hall, P., Poskitt, D. S., and Presnell, B. A functional data—analytic approach to signal discrimination. *Technometrics*, 43(1):1–9, 2001.

Han, H., Wang, W.-Y., and Mao, B.-H. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005.

Hand, D. J. and others, . Classifier technology and the illusion of progress. *Statistical science*, 21(1):1–14, 2006.

Hanley, J. A. and McNeil, B. J. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

Hastie, T. J. Pseudosplines. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 379–396, 1996.

Hastie, T. J. Generalized additive models. In *Statistical models in S*, pages 249–307. Routledge, 2017.

Hastie, T. J. and Tibshirani, R. J. Generalized additive models: some applications. *Journal of the American Statistical Association*, 82(398):371–386, 1987.

Hastie, T. J. and Tibshirani, R. J. Generalized additive models, volume 43 of monographs on statistics and applied probability, 1990.

Hastie, T. J., Buja, A., and Tibshirani, R. J. Penalized discriminant analysis. *The Annals of Statistics*, pages 73–102, 1995.

He, H. and Garcia, E. A. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009.

He, S., Mazumdar, S., and Arena, V. C. A comparative study of the use of gam and glm in air pollution research. *Environmetrics*, 17(1):81–93, 2006.

Hosmer, D. W. and Lemesbow, S. Goodness of fit tests for the multiple logistic regression model. *Communications in statistics-Theory and Methods*, 9(10):1043–1069, 1980.

Hosmer Jr, D. W., Lemeshow, S., and Sturdivant, R. X. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

Huber, P. J. Robust estimation of a location parameter. *The annals of mathematical statistics*, pages 73–101, 1964.

Huber, P. J. Robust regression: asymptotics, conjectures and monte carlo. *The Annals of Statistics*, pages 799–821, 1973.

Hubert, M., Rousseeuw, P. J., and Segaert, P. Multivariate functional outlier detection. *Statistical Methods & Applications*, 24(2):177–202, 2015.

Japkowicz, N. Assessment metrics for imbalanced learning. *Imbalanced learning: Foundations, algorithms, and applications*, pages 187–206, 2013.

Japkowicz, N. and others, . Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA, 2000.

Kalivas, J. H. Two data sets of near infrared spectra. *Chemometrics and Intelligent Laboratory Systems*, 37(2):255–259, 1997.

Kohavi, R. and others, . A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Stanford, CA, 1995.

Koshevoy, G. and Mosler, K. Zonoid trimming for multivariate distributions. *The Annals of Statistics*, pages 1998–2017, 1997.

Kubat, M., Matwin, S., and others, . Addressing the curse of imbalanced training sets: one-sided selection. In *Icml*, volume 97, pages 179–186. Nashville, USA, 1997.

Kwon, A. M., Ouyang, M., and Cheng, A. Resampling-based classification using depth for functional curves. *Communications in Statistics-Simulation and Computation*, 45(9): 3329–3338, 2016.

Lemeshow, S. and Hosmer Jr, D. W. A review of goodness of fit statistics for use in the development of logistic regression models. *American journal of epidemiology*, 115(1): 92–106, 1982.

Li, W., Goovaerts, P., and Meurens, M. Quantitative analysis of individual sugars and acids in orange juices by near-infrared spectroscopy of dry extract. *Journal of agricultural and food chemistry*, 44(8):2252–2259, 1996.

Li, Y., Wang, N., and Carroll, R. J. Selecting the number of principal components in functional data. *Journal of the American Statistical Association*, 108(504):1284–1294, 2013.

Lin, W.-J. and Chen, J. J. Class-imbalanced classifiers for high-dimensional data. *Briefings in bioinformatics*, 14(1):13–26, 2012.

Liu, H., Lafferty, J., and Wasserman, L. Sparse nonparametric density estimation in high dimensions using the rodeo. In *Artificial Intelligence and Statistics*, pages 283–290, 2007.

Liu, R. Y., Parelius, J. M., Singh, K., and others, . Multivariate analysis by data depth: descriptive statistics, graphics and inference,(with discussion and a rejoinder by liu and singh). *The annals of statistics*, 27(3):783–858, 1999.

Liu, R. Y. and others, . On a notion of data depth based on random simplices. *The Annals of Statistics*, 18(1):405–414, 1990.

Loftsgaarden, D. O., Quesenberry, C. P., and others, . A nonparametric estimate of a multivariate density function. *The Annals of Mathematical Statistics*, 36(3):1049–1051, 1965.

López, V., Fernández, A., García, S., Palade, V., and Herrera, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, 250:113–141, 2013.

López-Pintado, S. and Romo, J. Depth-based classification for functional data. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 72:103, 2006.

López-Pintado, S. and Romo, J. On the concept of depth for functional data. *Journal of the American Statistical Association*, 104(486):718–734, 2009.

López-Pintado, S., Romo, J., and Torrente, A. Robust depth-based tools for the analysis of gene expression data. *Biostatistics*, 11(2):254–264, 2010.

Lusa, L. and others, . Smote for high-dimensional class-imbalanced data. *BMC bioinformatics*, 14(1):106, 2013.

Mack, Y. and Rosenblatt, M. Multivariate k-nearest neighbor density estimates. *Journal of Multivariate Analysis*, 9(1):1–15, 1979.

Mahalanobis, P. C. On the generalized distance in statistics. *Proceedings of the National Institute of Sciences (Calcutta)*, 2:49–55, 1936.

Maronna, R., Martin, R. D., and Yohai, V. *Robust statistics*, volume 1. John Wiley & Sons, Chichester. ISBN, 2006.

Mathew, J., Luo, M., Pang, C. K., and Chan, H. L. Kernel-based smote for svm classification of imbalanced datasets. In *Industrial Electronics Society, IECON 2015-41st Annual Conference of the IEEE*, pages 001127–001132. IEEE, 2015.

Mercer, J. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209:415–446, 1909.

Nadaraya, E. A. On estimating regression. *Theory of Probability & Its Applications*, 9(1): 141–142, 1964.

Nychka, D. Bayesian confidence intervals for smoothing splines. *Journal of the American Statistical Association*, 83(404):1134–1143, 1988.

Pearson, K. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

Poskitt, D. and Sengarapillai, A. Description length and dimensionality reduction in functional data analysis. *Computational Statistics & Data Analysis*, 58:98–113, 2013.

Preda, C., Saporta, G., and Lévéder, C. Pls classification of functional data. *Computational Statistics*, 22(2):223–235, 2007.

Ramsay, J. O., Wickham, H., Graves, S., and Hooker, G. *fda: Functional Data Analysis*, 2014. URL `http://CRAN.R-project.org/package=fda`. R package version 2.4.4.

Ramsay, J. O. *Functional data analysis*. Wiley Online Library, 2006.

Ramsay, J. O., Hooker, G., and Graves, S. *Functional data analysis with R and MATLAB*. Springer Science & Business Media, 2009.

Rasmussen, C. E. Gaussian processes for machine learning. 2006.

Shang, H. L. Resampling techniques for estimating the distribution of descriptive statistics of functional data. *Communications in Statistics-Simulation and Computation*, 44(3): 614–635, 2015.

Silverman, B. W. *Density estimation for statistics and data analysis*, volume 26. CRC press, 1986.

Silverman, B. and Young, G. The bootstrap: To smooth or not to smooth? *Biometrika*, 74(3): 469–479, 1987.

Stein, M. L. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

Stone, C. J. Consistent nonparametric regression. *The annals of statistics*, pages 595–620, 1977a.

Stone, M. Asymptotics for and against cross-validation. *Biometrika*, pages 29–35, 1977b.

Taylor, M. S. and Thompson, J. R. A data based algorithm for the generation of random vectors. *Computational Statistics & Data Analysis*, 4(2):93–101, 1986.

Tibshirani, R. J. and Hastie, T. J. Local likelihood estimation. *Journal of the American Statistical Association*, 82(398):559–567, 1987.

Tibshirani, R. J., Wainwright, M., and Hastie, T. J. *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC, 2015.

Tocher, K. D. The art of simulation. 1967.

Trexler, J. C. and Travis, J. Nontraditional regression analyses. *Ecology*, 74(6):1629–1637, 1993.

Tukey, J. W. Mathematics and the picturing of data. In *Proceedings of the international congress of mathematicians*, volume 2, pages 523–531, 1975.

Tyler, D. E. Robust statistics: Theory and methods, 2008.

Wahba, G. Bayesian" confidence intervals" for the cross-validated smoothing spline. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 133–150, 1983.

Wahba, G. *Spline models for observational data*, volume 59. Siam, 1990.

Wand, M. P. and Jones, M. C. *Kernel smoothing*. Chapman and Hall/CRC, 1994.

Wang, K.-J., Adrian, A. M., Chen, K.-H., and Wang, K.-M. A hybrid classifier combining borderline-smote with airs algorithm for estimating brain metastasis from lung cancer: A case study in taiwan. *Computer methods and programs in biomedicine*, 119(2):63–76, 2015.

Watson, G. S. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 359–372, 1964.

Wold, H. Estimation of principal components and related models by iterative least squares. *Multivariate analysis*, pages 391–420, 1966.

Wold, S. Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics*, 20(4):397–405, 1978.

Wood, A. T. and Chan, G. Simulation of stationary gaussian processes in [0, 1] d. *Journal of computational and graphical statistics*, 3(4):409–432, 1994.

Wood, S. N. *Generalized additive models: an introduction with R*. Chapman and Hall/CRC, 2006.

Yao, F., Müller, H.-G., and Wang, J.-L. Functional data analysis for sparse longitudinal data. *Journal of the American Statistical Association*, 100(470):577–590, 2005.

Zivot, E. and Wang, J. Rolling analysis of time series. In *Modeling Financial Time Series with S-Plus®*, pages 299–346. Springer, 2003.

Zuo, Y. and Serfling, R. General notions of statistical depth function. *Annals of statistics*, pages 461–482, 2000.

Appendix A

In this appendix we present the code of the submitted file to use the High Throughput Computing (HTC) system at the University of Manchester to interact with the EPS Condor cluster.

High Throughput Computing (HTC) - Code

```
1
2  Universe = vanilla
3
4  Requirements = (Target.Opsys == "LINUX" &&
5  Target.Arch == "X86_64" && HAS_R_3_4=?=True)
6  Request_Memory = 1000
7  Request_CPU = 4
8
9  Log = KRNNSimulations.log
10 Output = KRNNSimulations.out
11 Error = KRNNSimulations.error
12 Notification = Error
13
14 Should_Transfer_Files = Yes
15 When_To_Transfer_Output = ON_EXIT
16
17 # GetEnv=True is required for the gcc compiler to work on Condor *clients*
18 GetEnv = True
19 Environment = R_LIBS_USER=.
20 Executable = /opt/R-3.4.3/bin/Rscript
21 Transfer_Executable = False
22 Arguments = --no-save M1K2N50.R
```

```
23 Transfer_Input_Files = M1K2N50.R
24 Transfer_Output_files = M1K2N50Results.rds
25
26 Queue
27
```