Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Estatística

Dissertação de Mestrado

Métodos Estatísticos em Aprendizado de Máquinas para problemas de Classificação

por

Luana Lúcia Alves de Azevêdo

Orientadora: Prof.ª Dr.ª Cibele Queiroz da Silva
Coorientador: Prof. Dr. Ernest Fokoué

Junho de 2018

Luana Lúcia Alves de Azevêdo

Métodos Estatísticos em Aprendizado de Máquinas para problemas de Classificação

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da Universidadede de Brasília como requisito parcial à obtenção do título de Mestre em Estatística.

Universidade de Brasília
Brasília, Junho de 2018

Termo de Aprovação

Luana Lúcia Alves de Azevêdo

Métodos Estatísticos em aprendizado de Máquinas para problemas de Classificação

Dissertação apresentada ao Departamento de Estatística do Instituto de Ciências Exatas da Universidadede de Brasília como requisito parcial à obtenção do título de Mestre em Estatística.

Data da defesa: 29 de Junho de 2018

Orientadora:

Prof.ª Dr.ª Cibele Queiroz da Silva
Departamento de Estatística, UnB

Coorientador:

Prof. Dr. Ernest Fokoué
Rochester University-USA

Comissão Examinadora:

Prof. Dr. Pedro Henrique Melo Albuquerque
Departamento de Administração, UnB

Prof. Dr. André Luiz Fernandes Cançado
Departamento de Estatística, UnB

Prof. Dr. Antônio Eduardo Gomes
Departamento de Estatística, UnB

Brasília, Junho de 2018

Ao meu pai (in memorian),
que me acompanhou no início desta caminhada...
Que sempre me dizia:
"Calma, vai dar tudo certo"...
Obrigada por sempre acreditar em mim!
Onde quer que esteja,
sei que esteve sempre presente...
Saudades eternas!

Agradecimentos

Agradeço primeiramente a Deus pela vida, por tudo que Ele me proporcionou e por dar-me força nos momentos de decisão.

Aos meus pais, Waldemar (in memorian) e Maria, pela base, educação, incentivo e apoio que sempre souberam me dar em todos os momentos de minha vida. A vocês que, muitas vezes, renunciaram aos seus sonhos para que eu pudesse realizar o meu.

A minha orientadora, Profa. Dra. Cibele Queiroz, exemplo de pessoa e profissional, pelos ensinamentos, correções, incentivo, paciência e por toda dedicação nesta dissertação. Quando "crescer", eu quero ser como você. Muito obrigada por tudo!

Ao meu coorientador, Prof. Dr. Ernest Fokoué, por ter sugerido o tema, ter enviado várias referências importantes e pelas valiosas sugestões que contribuíram para o resultado final do trabalho.

À minha irmã, Laila, e aos demais familiares, irmãos, avós, tios, primos e sobrinhos, pelo apoio incondicional. Não citarei nomes, para não me esquecer de ninguém.

À uma grande amiga, Helen, que esteve sempre presente me dando forças e ajudando a seguir em frente, principalmente, no momento em que mais precisei. Não tenho palavras para agradecer toda ajuda e consolo. Espero que perdure a amizade que construímos.

Ao Alonso, que dividiu comigo muitos momentos de estudos na biblioteca, ouvinte atento de algumas dúvidas, inquietações, desânimos e sucessos. Muito obrigada pelo apoio e confiança de sempre!

Ao Júlio, por ter ajudado no início da parte computacional da pesquisa. Obrigada pela troca de conhecimentos e pela pronta disposição em ajudar.

Aos demais colegas, em especial, Bruna e Alex, que contribuíram para o meu aprendizado durante as disciplinas, pelos conhecimentos e materiais compartilhados, muito obrigada!

Ao Programa de Pós-Graduação em Estatística - PGEST/UnB - pela oportunidade de realização do meu mestrado. Aos docentes, em geral, pelos ensinamentos e grandes contribuições à minha formação profissional e aquisição de novos conhecimentos.

A Universidade Federal de Rondônia (UNIR) pelo apoio financeiro, sem o qual, seria difícil a estadia em Brasília. Aos docentes do Departamento de Matemática e Estatística - DME, pelo aprendizado durante a graduação.

Ao Prof. Cristóvão Abrantes, pelas dicas, paciência e disposição que sempre teve ao longo dos projetos realizados na UNIR e, sobretudo, pela amizade. Obrigada por tudo!

A todos os professores desde o ensino fundamental ao mestrado, obrigada por todos os ensinamentos!

Resumo

As técnicas de aprendizado de máquina são amplamente utilizadas em tarefas de classificação de dados. Neste trabalho, são apresentados três métodos de aprendizagem supervisionadas que são adequadas à classificação de indivíduos. Estes métodos foram aplicados a dois conjuntos de dados, com características distintas, e realizados estudos de simulação para comparações entre os resultados. O método RDA destacou-se por obter o melhor desempenho de classificação em dados massivos e caso de $n \lll p$. Por sua vez, as técnicas FA e SVM obtiveram o melhor desempenho quando aplicadas ao conjunto de dados em que $n \ggg p$. As técnicas de validação cruzada (VC) são úteis para a definição dos valores ótimos dos hiper-parâmetros dos modelos. Neste trabalho utilizou-se três técnicas de VC: *Stratified Cross Validate (SCV), Leave-One-Out Cross Validation (LOOCV) e Shuffle and Split (SS)*. Para as comparações entre os resultados foram realizadas diversas análises, dentre elas, gráficos das curvas ROC, taxas de má classificação e EQMs. A avaliação final, utilizada para a escolha do melhor método de classificação, deu-se por meio do Erro Médio de Teste (*Average Test Error - AVTE*). As simulações e análises foram realizadas utilizando o *software* R.

Palavras-chave: Aprendizado de Máquina; SVM; Análise Discriminate Regularizada; Florestas Aleatórias; Validação Cruzada; Erro Médio de Teste.

Abstract

Machine learning techniques are widely used in data classification tasks. In this paper, we present three supervised learning methods that are suitable for the classification of individuals. These methods were applied to two sets of data, with different characteristics, and simulation studies were carried out to compare the results. The RDA method was distinguished by obtaining the best performance of classification in massive data and case of $n \lll p$. On the other hand, the techniques FA and SVM obtained the best performance when applied to the dataset where $n \ggg p$. Cross-validation (VC) techniques are useful for defining the optimum values of the hyper-parameters of the models. In this work three VC techniques were used: Stratified Cross Validate (SCV), Leave-One-Out Cross Validation (LOOCV) and Shuffle and Split (SS). For the comparisons between the results, several analyzes were carried out, among them, graphs of ROC curves, misclassification rates and EQMs. The final evaluation, used to choose the best classification method, was done through the Average Test Error (AVTE). Simulations and analyzes were performed using software R.

key words: Machine Learning; SVM; Regularized Discriminate Analysis; Random Forests; Cross-validation; Average Test Error.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

1.1 Apresentação

Aprendizado de Máquinas (AM) (*Machine Learning*) é considerado como um subcampo da Inteligência Artificial e está relacionado ao desenvolvimento de técnicas e métodos que permitem que o computador aprenda. Em termos simples, nesta área do conhecimento dedica-se ao desenvolvimento de algoritmos que permitem que máquinas aprendam e executem tarefas e atividades. O aprendizado de máquinas apresenta grande intersecção com a área de Estatística.

Um tópico importante no Aprendizado de Máquinas é a classificação de objetos. Nessa abordagem, o usuário gera ou separa um conjunto de *dados de treinamento*, que é um subconjunto dos dados completos, e que inclui exemplares representativos do problema em questão com classes predefinidas ou conhecidas. O algoritmo de aprendizagem mecânica infere automaticamente as regras para discriminar as classes que podem, então, ser aplicadas ao conjunto completo dos dados. Este tipo de aprendizagem é denominada por *supervisionada*. Na aprendizagem de máquinas o objetivo principal é inferir propriedades gerais da distribuição de dados a partir de alguns dados de treinamento (Hastie et al., 2001; Bishop, 2006; Tarca et al., 2007).

Nesta dissertação pretende-se estudar alguns métodos de AM que são adequados à classificação de indivíduos ou objetos. Entre estes o interesse reside no estudo e aplicações dos métodos SVM (Support Vector Machine), Florestas Aleatórias (FA) e Análise de Discriminante Regularizada (RDA). Dentre as principais referências utilizadas destacam-se os trabalhos de Bill (2015); Fokoué e Bill (2014); James et al. (2013); Hastie et al. (2001); Clarke et al. (2009) e Fokoué (2014).

O método *Support Vector Machine* (SVM) foi desenvolvido por Cortes e Vapnik (1995) para problemas de classificação binária. Segundo Sommer e Gerlich (2013), a técnica visa encontrar um hiperplano de decisão que separa pontos de dados de diferentes classes com uma margem maximal (ou seja, a distância máxima aos pontos de treinamento mais próximos). Como os pontos de dados de diferentes classes podem não ser sempre completamente separáveis para um hiperplano, a maioria das implementações SVM são baseados em uma margem suave, o que permite classificações incorretas a um certo valor de custo. Os SVM são essencialmente classificadores lineares, mas estes permitem a geração de limites de decisão não-lineares se os pontos de dados forem previamente transformados para dimensões mais elevadas (como um Kernel gaussiano) usando uma função de mapeamento (Vapnik, 2000). SVMs são relativamente robustos a características que apresentam ruídos e são computacionalmente eficientes. Além disso, as implementações computacionais dos SVMs estão disponíveis em vários pacotes computacionais de bioimagem (Held et al., 2010; Conrad et al., 2011; Horvath et al., 2011).

As Florestas Aleatórias (FA) (*random forests*, Breiman, 2001) treinam um conjunto de árvores de decisão (Breiman et al., 1983) sob influência aleatória para a média dos resultados. Ao tomar a média das predições de um conjunto de árvores de decisão reduz-se a variância global

mas mantém-se sob controle o viés, que é típico em árvores de decisão. As FAs são robustas para dados em altas dimensões, por causa da implícita seleção de características (*feature selection*), que constitui a base para distinguir os objetos por um algoritmo classificador. Além disso, são computacionalmente eficientes.

A Análise de Discriminante Linear (LDA) é um método estatístico bem conhecido e popular no reconhecimento de padrões e classificação. A ideia básica é otimizar os critérios discriminantes, em que a razão entre as distâncias intra e entre classes é maximizada. Essa abordagem é teoricamente bem estabelecida e há uma vasta bibliografia demonstrando sua utilidade e desempenho. No entanto, essa abordagem apresenta limitações para pequenos tamanhos n de amostra, principalmente quando o número de covariáveis ou características p é tal que $n \lll p$. Algo semelhante ocorre para o caso da Análise de Discriminante Quadrática (QDA). Há também um método intermediário entre a LDA e a QDA, que é uma versão *regularizada* da análise discriminante (RDA). Este método foi proposto por Friedman (1989). A RDA apresenta melhor desempenho relativo à minimização de classificações incorretas quando comparada à LDA e a QDA, especialmente quando as matrizes de covariâncias associadas às populações em análise não são próximas e/ou o tamanho da amostra é muito pequeno para que a LDA/QDA possam ser ajustadas.

1.2 Objetivos

Neste trabalho de dissertação objetiva-se estudar técnicas supervisionadas de Aprendizado de Máquinas que sejam úteis para problemas de classificação. As técnicas que são alvo deste trabalho relacionam os métodos SVM (Support Vector Machine), Florestas Aleatórias (FA) e Análise de Discriminante Regularizada (RDA). Os conjuntos de dados a serem trabalhados nesta dissertação incluem, mas não estão restritos, ao caso em que $n \lll p$.

No Brasil, as técnicas de Aprendizado de Máquinas têm sido tradicionalmente estudadas e aplicadas nas áreas da Engenharia Elétrica e Computação, no entanto, com as visões próprias destas áreas. Na área da Estatística praticamente não há trabalhos relativos a este tema. No entanto, a Comunidade Estatística já está ciente da importância da temática e da urgência em formar capital humano junto aos alunos de suas graduações, mestrados e doutorados, que sejam capazes de atuar neste novo e desafiador campo de atuação.

Diante da quase inexistência de material escrito em Português sobre o tema, um dos objetivos deste trabalho é fornecer um texto que sirva como ponto de partida para aqueles que se interessem pelo tema de Aprendizado de Máquinas, em especial no que concerne às técnicas de classificação. Além disso, há poucos exemplos de códigos escritos na linguagem R que ajudem os interessados a se desenvolverem neste assunto.

No intuito de alcançar o objetivo geral, estabelecem-se os seguintes objetivos específicos:

i) Escrever um texto que sirva de referência para aqueles que desejem iniciar seus estudos em Aprendizado Supervisionado de Máquinas com ênfase em técnicas de classificação.

ii) Aplicar as técnicas citadas a dois bancos de dados reais, com características distintas.

iii) Estabelecer análises comparativas entre as técnicas.

iv) Apresentar os códigos `R` que forem utilizados na dissertação.

Conjuntos de dados a serem explorados neste trabalho:

1. Colon Cancer Microarray Data (Alon, et al.1999). Este conjunto de dados está disponível na biblioteca `rda` do software `R`. Também está disponível no site http://microarray.princeto

n.edu/oncology/. Este conjunto de dados é composto por uma amostra contendo 62 observações das quais 44 destas são de pacientes com câncer no intestino grosso e 22 amostras são de pacientes sem a doença. Os dados são compostos por informações colhidas a partir de tecido do intestino grosso, sendo que dados genéticos de 2000 genes, que são supostos serem importantes na detecção da doença, foram extraídos. Este conjunto de dados é tratado como um conjunto de dados binário. Utilizando as técnicas descritas objetiva-se classificar os indivíduos em doentes ou não-doentes de acordo com as informações genéticas.

2. Voice Gender Data (https://www.kaggle.com/primaryobjects/voicegender). Este banco de dados foi criado para identificar uma voz como homem ou mulher, com base nas propriedades acústicas da voz e da fala. O conjunto de dados consiste em 3.168 amostras de voz gravadas, coletadas de falantes masculinos e femininos. As amostras de voz são pré-processadas por análise acústica em R usando os pacotes `seewave` e `tuneR`, com uma faixa de freqüência analisada de 0hz-280hz (faixa vocal humana). Um total de 21 propriedades acústicas de cada voz foram medidas. Determinar o sexo de uma pessoa como homem ou mulher, com base em uma amostra de sua voz parece ser inicialmente uma tarefa fácil. Muitas vezes, o ouvido humano pode facilmente detectar a diferença entre uma voz masculina ou feminina já nas primeiras palavras faladas. Objetiva-se classificar os indivíduos, segundo o gênero, de acordo com as propriedades acústicas de suas vozes.

1.3 Estrutura da dissertação

O trabalho está organizado em seis capítulos. No Capítulo 2, foi feita uma revisão de conceitos básicos necessários para os próximos capítulos, tais como: os tipos de aprendizagem de máquina (Seção 2.1), o problema da dimensionalidade dos dados (Seção 2.2), problemas de regressão e problemas de classificação (Seção 2.3) e validação cruzada (Seção 2.4).

No Capítulo 3 discorre-se sobre os métodos de classificação supervisionados: Regularização (Seção 3.1), Kernelização (Seção 3.2) e Técnicas de Conjuntos Aleatórios (Seção 3.3). Dentre os métodos abordados, ressaltam-se as técnicas de Análise Discriminante Regularizada, Máquinas de Vetor de Suporte (SVM) e Florestas Aleatórias (FA), que são alvos deste trabalho.

O Capítulo 4 descreve o processo de implementação computacional de cada uma das técnicas no *software* R. Inicialmente, são apresentadas algumas etapas importantes a serem seguidas para atingir-se a unicidade dos parâmetros ótimos para análise de modo a atingir-se o mínimo absoluto para a Validação Cruzada (VC). O Capítulo enfatizou ainda, algumas medidas utilizadas para a avaliação do método de classificação empregado, tais como, a Matriz de Confusão (Seção 4.2) e os gráficos ROC (Seção 4.3). Adicionalmente, no intuito de calcular o Erro Médio de Teste, descrito na Seção 4.1, também utilizou-se o processo de Bagging (Seção 4.4). Maiores detalhes sobre cada técnica de classificação, como os pacotes computacionais que foram utilizados e os parâmetros estimados, são apresentados na Seção 4.5.

O Capítulo 5 apresenta os resultados das aplicações, de cada técnica, em dados reais. As simulações foram realizadas comparando-se cada método de classificação de dados, bem como as técnicas de validação cruzada. Inicialmente, apresentou-se os resultados obtidos usando os dados de câncer de cólon (Seção 5.1), em seguida, os resultados usando-se os dados de voz (Seção 5.2). A Seção 5.4 apresentou algumas comparações entre os resultados obtidos.

Por fim, o Capítulo 6 trata das conclusões obtidas nesta dissertação, bem como os trabalhos futuros.

Capítulo 2

Introdução à Aprendizagem Estatística

A teoria da aprendizagem estatística foi estabelecida no final da década de 60. Até meados dos anos 90 referia-se somente a uma análise teórica do problema de estimação de funções a partir de um conjunto de dados. Nas últimas décadas, com a crescente complexidade dos problemas a serem tratados computacionalmente, o processo de aprendizagem estatística tem sido marcado pela crescente utilização de softwares. Além disso, os desafios em aprender com os dados levaram à evolução do conjunto de ferramentas e técnicas estatísticas utilizadas.

A aprendizagem estatística possui um papel fundamental em diversas áreas, tais como: ciência, finanças e indústria. Alguns exemplos são (Hastie et al., 2001):

- Prever se um paciente hospitalizado, devido a um ataque cardíaco, terá um segundo ataque cardíaco. A previsão deve ser baseada em medidas clínicas e demográficas, para esse paciente;
- Identificar os números de um código postal manuscrito, a partir de uma imagem digitalizada;
- Identificar os fatores de risco para câncer de próstata, com base em variáveis clínicas e demográficas.

Em geral, suponha uma resposta quantitativa Y (variável de saída - *output*) e p diferentes preditores, $X_1, X_2, \ldots, X_p$ (variáveis de entrada - *input*). Assuma que há relação entre Y e $X = (X_1, X_2, \ldots, X_p)$ e que possa ser escrita na forma geral

$$Y = f(X) + \epsilon,$$

na qual f é alguma função fixa desconhecida e ϵ é um termo de erro aleatório, independente de X e com média 0.

A função f pode envolver várias *variáveis de entrada* e deve ser estimada com base nos dados observados. De acordo com James et al. (2013), a aprendizagem estatística refere-se a um conjunto de ferramentas usadas para a compreensão dos dados. Tais ferramentas podem ser divididas em duas grandes áreas: métodos supervisionadas e métodos não supervisionadas.

2.1 Aprendizagem Supervisionada versus Aprendizagem não Supervisionada

Na aprendizagem supervisionada, para cada observação das variáveis preditoras ($X_{1i}, X_{2i}, \ldots, X_{pi}$, $i = 1, 2, \ldots, n$) há uma variável resposta Y_i associada. Deseja-se ajustar um modelo que relaciona a variável resposta aos preditores, com o objetivo de predizer, acuradamente, as

respostas às futuras observações (predições), assim como entender o relacionamento entre a resposta e os preditores (inferência).

Vários métodos estatísticos conhecidos podem ser incluídos ao domínio da aprendizagem supervisionada, tais como as regressões linear e logística, os Modelos Aditivos Generalizados (GAM), originalmente introduzidos por Hastie e Tibshirani (1990), os métodos de *boosting*, destinados à redução de vício e variância nos problemas de classificação (Breiman, 1998) e os métodos denominados por *Support Vector Machine (SVM)* relativos aos problemas de classificação em dois grupos (Vapinik, 1995).

Na aprendizagem supervisionada o termo "supervisionar" refere-se à presença ou disponibilidade de uma variável resposta (outcome) que possa guiar ou balizar o processo de aprendizado.

Os problemas de aprendizagem não supervisionada caracterizam-se pela ausência de medidas de variáveis resposta (output), e o objetivo destes métodos é descrever as associações e padrões existentes entre o conjunto de medidas de entrada (input).

Nesta dissertação trabalharemos com métodos de aprendizado supervisionado.

2.2 O problema da dimensionalidade: n pequeno, p grande

Nos problemas de aprendizado supervisionado, que estudaremos nesta dissertação, faremos uso de dados massivos (também conhecidos, popularmente, como Big Data). No entanto, a característica principal dos dados com os quais iremos trabalhar é a alta dimensionalidade p dos vetores de variáveis explanatórias ou preditores quando o tamanho da amostra n é extremamente baixo, isto é, $n \lll p$.

Os conjuntos de dados com os quais trabalharemos podem ser representados por

$$D = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n),$$

em que

$$x_i^T = (x_{i1}, x_{i2}, \ldots, x_{ip}), i = 1, 2, \ldots, n,$$

sendo x_i^T um vetor p-dimensional de características ou variáveis de entrada (input) ou preditoras, pertencentes ao espaço $\mathcal{X}$ de "entradas" e y_i representa o valor correspondente da resposta (output) pertencente ao espaço $\mathcal{Y}$ de "saídas". Como a dimensão p é muito elevada comparativamente ao tamanho amostral n disponível para análise (High-Dimensional Low Sample Size - HDLSS) os dados são esparsos, implicando em serem pouco informativos.

O efeito da esparsidade dos dados nas análises estatísticas dos mesmos é muito bem descrito no artigo de Fokoué (2013). O autor exemplifica o problema utilizando-se dos modelos de Regressão Linear Múltipla (RLM) com p preditores e resíduo Gaussiano. Nessa classe de modelos, para p variáveis explanatórias há um total de $2^p - 1$ modelos e submodelos. Se $p = 20$, o espaço μ dos modelos lineares inclui mais de 1 bilhão de possibilidades. Dessa forma, teoricamente, necessita-se buscar nesse espaço μ, dos modelos lineares possíveis, aquele que proporciona melhor ajuste.

O efeito nas análises do aumento exponencial da dimensionalidade do espaço μ aliado a pequenos tamanhos de n foi denominado por Bellman (1961) como "maldição da dimensionalidade" (*curse of dimensionality*).

Fokoué (2013) estipula que qualquer problema com um conjunto de entradas contendo mais de 50 variáveis é um problema de dados massivos, porque procedimentos computacionais de busca em espaços de 1000 trilhões de possibilidades são, claramente, extremamente desafiadores, mesmo para os potentes computadores da atualidade. Na realidade ocorre, rotineiramente, em dados genéticos, do tipo microarray, em que p para este tipo de dados pode estar na casa dos

Tabela 2.1: Massividade em função de $\frac{n}{p}$. Os esquemas A e D são os mais complexos.

	$\frac{n}{p} < 1$ Informação pobre $(n \lll p)$	$1 \leq \frac{n}{p} < 10$ Escassêz de informações	$\frac{n}{p} \geq 10$ Abundância de informação $(n \ggg p)$
$n > 1000$	p grande, n grande A	p pequeno, n grande B	p muito pequeno, n grande C
$n \leq 1000$	p grande, n pequeno D	p pequeno, n pequeno E	p muito pequeno, n pequeno F

milhares. Adicionalmente, se o problema requer a estimação de matrizes de covariância e sua inversão, facilmente pode-se deparar com problemas computacionalmente insolúveis.

Fokoué (2013) apresenta uma tabela que auxilia na classificação da massividade dos dados em função de n e p. O autor observa que, na verdade, a razão n/p é que é determinante na especificação da qualidade da informação disponível de um problema, e não n e p separadamente.

De acordo com a Tabela 1.1, uma vez que $n > 10p$ para os casos em que há abundância de informações, há que se ter pelo menos 10 vezes mais observações do que variáveis. Dessa forma, para cada variável explanatória em estudo são necessárias 10 observações para que se tenha uma análise que apresente boa precisão e acurácia em suas inferências.

2.3 Problemas de regressão e problemas de classificação

Quanto à abordagem, a aprendizagem supervisionada de máquina pode ser dividida em duas grandes áreas: problemas de classificação e problemas de regressão. Quando a variável resposta é quantitativa os problemas são de regressão e quando a resposta é qualitativa, os problemas são de classificação.

As variáveis quantitativas são características que podem ser medidas e apresentadas por números. Exemplo: altura, idade, tempo (relógio) e número de filhos. As variáveis qualitativas são características que não podem ser medidas fisicamente, porque não possuem valores quantitativos. Entretanto, podem ser categorizadas e representar uma classificação dos indivíduos. Exemplo: sexo, cor dos olhos, escolaridade (1º, 2º e 3º graus), presença ou ausência de dado atributo, isto é, $y \in \{0, 1\}$.

De acordo com Clarke et al. (2009), quase todas as técnicas de regressão proporcionam um classificador se forem aplicadas a respostas discretas, e cada procedimento de classificação corresponde a um problema de regressão, apesar de y poder assumir apenas dois valores, como é o caso da regressão logística.

Por exemplo, uma regressão logística pode ser usada para obter-se um classificador. No caso de trabalharmos com um classificador para apenas duas classes, sejam

$$P(Y = 0|X = x) = \frac{e^{\beta_0 + \beta^T x}}{1 + e^{\beta_0 + \beta^T x}} \tag{2.1}$$

e

$$P(Y = 1|X = x) = \frac{1}{1 + e^{\beta_0 + \beta^T x}}, \tag{2.2}$$

as funções discriminantes para as duas classes. Então, a transformação monotônica dada pelo logito, isto é, $p \longrightarrow \log(1/(1-p))$, resulta em

$$\log \frac{P(Y = 0|X = x)}{P(Y = 1|X = x)} = \beta_0 + \beta^T x, \tag{2.3}$$

e obtém-se que a reta $\beta_0 + \beta^T x = 0$ estabelece a fronteira entre as classes 0 e 1. Outros métodos úteis para particionar o espaço destinado a cada classe nos problemas de discriminação incluem árvores de regressão, redes neurais e SVMs.

Nesta dissertação, trabalha-se com classificações pela metodologia de aprendizagem supervisionada, em que objetiva-se classificar determinadas observações, x_i, para sua classe correta, y_i, com erros mínimos. Isto é realizado removendo ou mantendo os rótulos de classe y_i, de um conjunto de observações e passando os vetores de características correspondentes, x_i, para um algoritmo escolhido com o intuito de prever a classe y_i para cada vetor de características ou observação. Os $\hat{y}_i$ preditos são então comparados com os correspondentes verdadeiros $y's$ de cada observação. Para cada $\hat{y} \neq y$, uma penalidade ou custo é incorrido. Isto define a função perda $0-1$, que é uma das funções perda mais utilizadas na aprendizagem de máquina,

$$l(Y, f(X)) = \begin{cases} 1, & \text{se} \quad Y \neq f(X), \\ 0, & \text{se} \quad Y = f(X). \end{cases} \tag{2.4}$$

A classificação tem como objetivo realizar uma predição acurada para um dado conjunto de dados. Para atingir esse objetivo, deve-se realizar um processo que inclui uma série de etapas que inicia com a validação cruzada para que sejam determinados valores ótimos para os parâmetros do modelo, e finaliza com a estimativa do erro médio na comparação entre os classificadores. Entretanto, primeiramente é importante compreender o conceito de erro de treinamento, erro de teste e o dilema entre viés e variância.

2.3.1 Erro de treinamento e erro de teste

Suponha que f é estimado com base nas observações $\{(x_1, y_1), \ldots, (x_n, y_n)\}$, em que $y_1, \ldots, y_n$ são qualitativos. Uma abordagem para quantificar a precisão da estimativa $\hat{f} = \hat{f}(x) = \hat{y}$, é a taxa de erro de treinamento, que é a proporção de erros que foram produzidos por um classificador quando aplicado a um certo conjunto de dados contendo ambos os valores de covariáveis e suas respostas, isto é,

$$\frac{1}{n} \sum_{i=1}^{n} I(y_i \neq \hat{y}_i), \tag{2.5}$$

em que $I(y_i \neq \hat{y}_i)$ é um indicador de variável, que é igual a um se $y_i \neq \hat{y}_i$, e igual a zero se $y_i = \hat{y}_i$.

A expressão (2.5) refere-se à taxa de erro de treinamento porque é computada com base nos dados que foram utilizados para treinar o classificador. Na aprendizagem de máquina, o treinamento de um modelo preditivo consiste em encontrar uma função que leva um conjunto de valores x a um valor y. Se o modelo for aplicado aos dados desconhecidos na fase de treinamento, o que deseja-se calcular é o *erro na testagem*.

O erro na testagem é o erro médio que resulta do uso de um método de aprendizagem estatística para prever uma resposta para uma nova observação, ou seja, que ainda não foi utilizada no treinamento do modelo. A taxa de erro na testagem associada a um conjunto de observações de teste da forma (x_0, y_0) é dada por

$$\text{Média}(I(y_0 \neq \hat{y}_0)), \tag{2.6}$$

onde y_0 é o rótulo da classe prevista, que resulta da aplicação do classificador à observação de teste com o preditor x_0. Um bom classificador é aquele para o qual o erro na testagem (2.6) é pequeno.

2.3.2 Medindo a qualidade do ajuste

Para analisar o desempenho de um método de aprendizagem estatística em um conjunto de dados é necessário medir o quanto as previsões para determinadas observações estão próximas de seus respectivos valores verdadeiros. A medida geralmente utilizada no ajuste de regressão é o Erro Quadrático Médio - EQM (*Mean Squared Error - MSE*), dado por

$$EQM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2, \tag{2.7}$$

onde $\hat{y}$ é a previsão de y para a i-ésima observação.

Se as respostas previstas estiverem próximas das respostas verdadeiras, o EQM será pequeno. Porém, se as respostas previstas diferem substancialmente das respostas verdadeiras, o EQM será grande.

O EQM em 2.7 é calculado utilizando-se os dados de treinamento (*training data*) que foram usados para ajustar o modelo, de modo que deve ser mais precisamente referido como *EQM de treinamento*, EQM_{TR}. No entanto, o interesse maior não está em se o modelo/método se comporta bem nos dados de treinamento. Na verdade, o interesse reside na avaliação da acurácia das predições que se obtém quando aplica-se o método a dados de testagem (*test data*) que não foram analisados previamente.

No caso dos dados de treinamento, já se conhecem todos os detalhes a respeito do mesmo. Por exemplo, num estudo sobre diabetes, já se sabe quais são os pacientes que tem diabetes. No caso dos dados de testagem, deseja-se predizer quais indivíduos têm diabetes com base em suas características individuais e o modelo treinado em questão.

Em termos matemáticos, suponha que no método de aprendizado estatístico foram utilizadas as observações de treinamento $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ e obtida a estimativa $\hat{f}$. Então calcula-se $\hat{f}(x_1), \hat{f}(x_2), \ldots, \hat{f}(x_n)$. Se estas estimativas estão próximas de $y_1, y_2, \ldots, y_n$, então o EQM_{TR} será pequeno. No entanto, o que realmente importa é saber se $\hat{f}(x_0)$ é aproximadamente igual a y_0, sendo (x_0, y_0) uma observação de testagem, que não foi observada anteriormente e que não foi utilizada para treinar o método de aprendizagem estatística. Objetiva-se selecionar o método que minimize o EQM associado aos dados de testagem, EQM_{TS}, ao invés do EQM_{TR}. Se tivermos um número grande de observações de testagem n_{TS} pode-se calcular o erro quadrático médio dos erros de predição, EQM_{TS}, em que

$$EQM_{TS} = \frac{1}{n_{TS}}\sum_{j=1}^{n_{TS}}(\hat{f}(x_{0j}) - y_{0j})^2 \tag{2.8}$$

para as observações de testagem $(x_{01}, y_{01}), \ldots, (x_{0n_{TS}}, y_{0n_{TS}})$. Desta forma, deseja-se selecionar um modelo que miniminize o EQM_{TS}.

No entanto, como selecionar um método que minimize o EQM_{TS}? Em alguns casos há dados de testagem disponíveis, isto é, tem-se acesso a observações que não foram utilizadas para treinar os dados. Dessa forma, pode-se simplesmente computar (2.8) com base nestas observações, e selecionar o método de aprendizado para o qual o EQM_{TS} seja mínimo.

O que fazer quando não há dados de testagem disponíveis? Uma solução plausível seria selecionar o método de aprendizagem cujo EQM_{TR} seja mínimo. No entanto, não há garantia de que o método que minimiza o EQM_{TR} seja o mesmo que minimiza EQM_{TS}. A maioria dos métodos estatísticos disponíveis são tais que os coeficientes estimados minimizam o EQM_{TR}. Para estes métodos o EQM_{TR} pode ser pequeno, mas o EQM_{TS} é usualmente muito grande.

Independentemente do conjunto de dados utilizado, quando um método produz pequeno EQM_{TR} mas grande EQM_{TS}, diz-se que está ocorrendo um problema de "sobreajustamento" (overfitting). Isto ocorre porque o procedimento de aprendizagem estatística está muito sensível a qualquer mudança, mesmo as que ocorrem meramente ao acaso, e as classifica \ entende

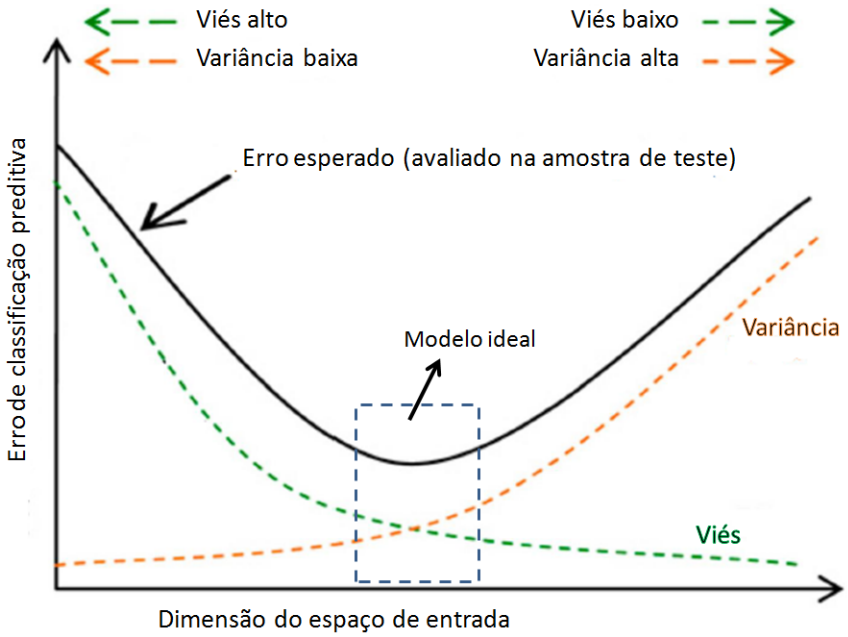


Figura 2.1: O dilema entre viés e variância.

incorretamente como se fossem "padrões". Quando há sobreajustamento nos dados de treinamento, o EQM_{TS} será muito grande, porque os supostos "padrões" não existem nos dados de testagem. De qualquer forma, independentemente de ter ocorrido sobreajustamento, quase sempre espera-se que o EQM_{TR} seja menor do que o EQM_{TS}, por causa do já mencionado fato dos métodos estatísticos em geral serem projetados visando minimizar o EQM_{TR}.

O sobreajustamento ocorre muito frequentemente nos casos em que o modelo em estudo é pouco flexível.

2.3.3 O Dilema entre viés de variância

O valor esperado do EQM_{TS} (vide expressão 2.8) é dado a seguir

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) + [Bias(\hat{f}(x_0))]^2 + Var(\epsilon), \tag{2.9}$$

onde $Var(\hat{f}(x_0))$ é a variância de $\hat{f}(x_0)$ e $[Bias(\hat{f}(x_0))]^2$ é o viés quadrático de $\hat{f}(x_0)$.

Para minimizar o valor esperado do EQM_{TS} e encontrar um bom desempenho do modelo na fase de testagem, deve-se selecionar um método de aprendizagem estatística que proporcione, simultaneamente, baixa variação e o baixo viés. A relação entre o viés, a variância e o EQM_{TS} dada pela expressão (2.8) é comumente chamada de *Dilema entre viés e variância*, isso porque é fácil obter um método com viés muito baixo, mas com alta variância ou um método com viés alto, mas variância pequena. O desafio reside em encontrar um método que minimize ambos, viés e variância. Estes conceitos serão discutidos em mais detalhes a seguir.

As Figuras 2.1 e 2.2 ilustram a relação entre o viés e a variância, bem como a relação entre o erro de treinamento e o erro de teste. Observa-se que a medida que o modelo torna-se mais complexo, o erro de treinamento diminui, diminuindo o viés e aumentando a variância. Uma vez que a amostra de testagem é introduzida, contendo dados que o algoritmo nunca viu antes, o erro de testagem aumenta rapidamente. Como já mencionado, isto é referido como sobreajustamento e resulta numa quantidade proporcionalmente maior de variabilidade em resposta à diminuição do viés fazendo com que o modelo não se generalize bem, ou seja, o modelo ajustado com dados de testagem apresente ajuste ruim. Por outro lado, quando

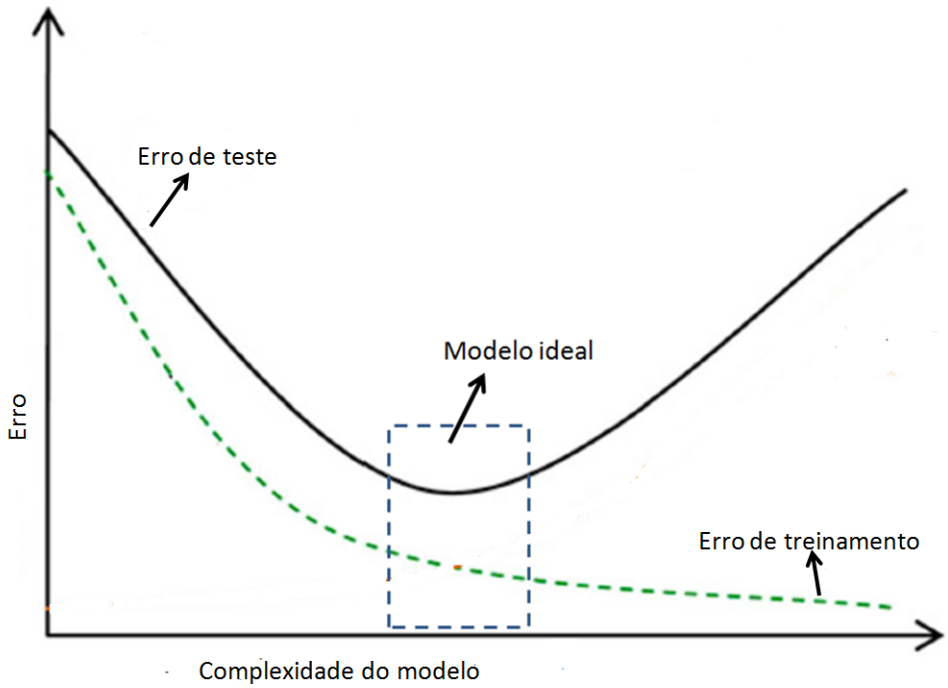


Figura 2.2: Erro de testagem e erro de treinamento.

a variância é muito baixa, em geral, há uma grande quantidade de viés presente no modelo, fazendo com que o modelo fique mal ajustado, o que resultará novamente em um modelo que não generaliza bem. O desafio é encontrar um ponto de equilíbrio ótimo entre o viés e a variância que, proporcionalmente fornece o menor erro de generalização ou erro de testagem (Bill, 2015). Este processo é discutido em detalhes na próxima seção, que trata de validação cruzada.

2.4 Validação Cruzada

A Validação Cruzada (VC) é um técnica amplamente utilizada para estimar o erro de teste associado ao ajuste de um determinado método de aprendizagem estatística em um conjunto de observações. É útil na avaliação do modelo e ajuda a garantir que este não esteja superajustado (*overfitting*).

Há diversos tipos de VC, um deles corresponde em dividir aleatoriamente o conjunto de observações em duas partes: um conjunto de treinamento e um conjunto de teste. O modelo é definido no conjunto de treinamento e utilizado para prever as respostas para as observações no conjunto de teste.

Para a realização da VC, basicamente, adota-se uma das seguintes técnicas para divisão dos dados: *holdout, k-fold* ou *leave-one-out*. Esta pesquisa se baseia no estudo de dois tipos de VC k-Fold: *Shuffle and Split (SS) e Stratified Cross Validate (SCV)*, bem como no estudo da *Leave-one-out Cross Validation (LOOCV)*.

2.4.1 Validação *holdout*

O método *holdout*, representado na Figura 2.3, consiste em dividir o total dos dados em dois subconjuntos mutuamente exclusivos: um para treinamento e o outro para teste, sem a alternação que ocorre com o k-Fold, que será visto mais adiante.

O subconjunto usado para o treino é utilizado para estimação dos parâmetros e, usualmente, é uma fração de $\frac{2}{3}$ do total dos dados. O outro subconjunto, com $\frac{1}{3}$ do total dos dados, é usado com a finalidade de verificar a acurácia do modelo (Schreiber et al., 2017; Cunha, 2016).

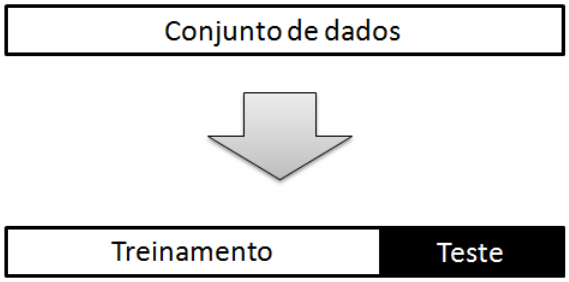


Figura 2.3: Validação *Holdout*.

2.4.2 Validação Cruzada K-Fold

A Validação Cruzada k-Fold, que será denominada VC k-Fold, é utilizada para encontrar os melhores valores de hiper-parâmetros que produzem um desempenho de generalização satisfatório. Uma vez encontrados os valores dos hiper-parâmetros ideais, pode-se treinar o modelo no conjunto de treinamento e obter uma estimativa final de desempenho usando o conjunto de teste.

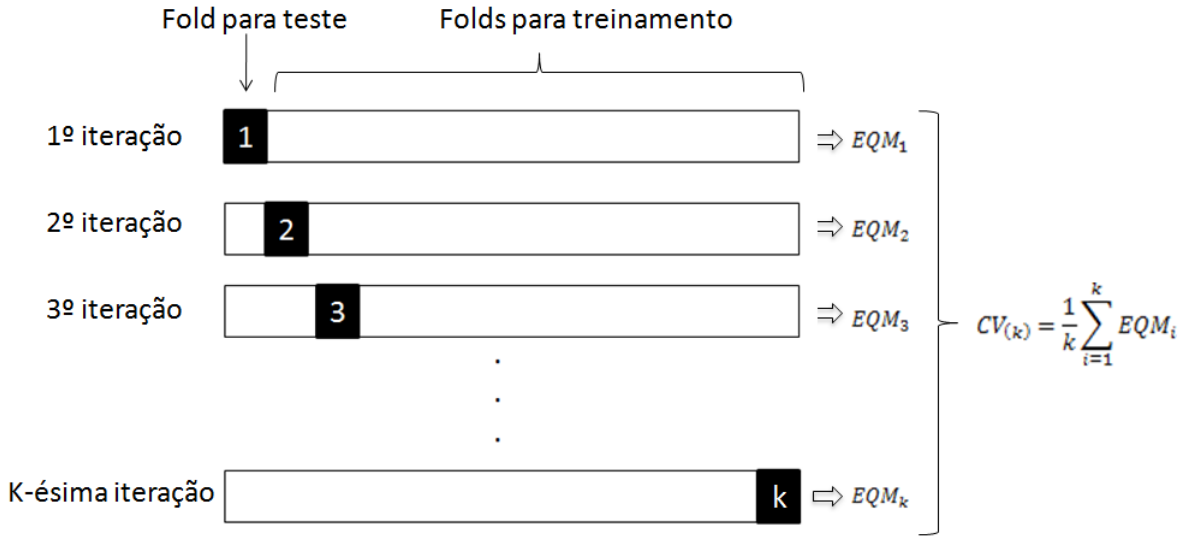


Figura 2.4: O método VC K-Fold implica em dividir aleatoriamente o conjunto de observações em k folds sem substituição, onde o primeiro *fold* é tratado como o conjunto de teste e os $k-1$ *folds* restantes formam o conjunto de treinamento.

A Figura 2.4 apresenta o método VC K-Fold. Este método implica em dividir aleatoriamente o conjunto de observações em k grupos (ou *folds*) sem substituição e de tamanho aproximadamente igual. O primeiro *fold* é tratado como o conjunto de teste e os $k-1$ *folds* restantes formam o conjunto de treinamento.

O modelo é ajustado utilizando o conjunto de treinamento. Então, é medida a precisão preditiva dos modelos através do cálculo do erro quadrático médio, EQM_1, no conjunto de testes. O processo de validação cruzada é repetido k vezes, de modo que cada um dos k grupos sejam utilizados exatamente uma vez como dado de teste para validação do modelo. Esse processo resultará em k estimativas do erro de teste, $EQM_1, \ldots, EQM_k$. A estimativa de VC

k-Folds é calculada pela média destes valores. (James et al., 2013),

$$CV_{(k)} = \frac{1}{k}\sum_{i=1}^{k} EQM_i. \tag{2.10}$$

O resultado final desse processo é o desempenho médio do classificador nos k testes e o objetivo de repetir os testes múltiplas vezes é aumentar a confiabilidade da estimativa da precisão do classificador.

Os dois tipos de VC k-Folds (SS e SCV), que serão apresentadas a seguir, realizam exatamente a mesma validação cruzada, a diferença reside na forma como as amostras são divididas em folds. Ao dividir ou particionar os dados, a taxa de má classificação, para um determinado parâmetro ou conjunto de parâmetros, é avaliada no final de k iterações.

Bill (2015), págs 38 a 40, apresenta algoritmos para os tipos de VC k-Fold. Alguns algoritmos foram reescritos a seguir: Algoritmo 1 - K-Fold Cross-Validation, Algoritmo 2 - SS, Algoritmo 3 - SCV.

Algoritmo 1 - K-Fold Cross-Validation

For $i = 1$ to n Loop
Desconsidere temporariamente o i-ésimo fold das observação.
n_i = número de observações no i-ésimo fold.
Construa um classificador para uma determinada técnica usando as observações nos k − i-ésimos folds restantes.
Média das i-ésima classificações erradas = $\sum(\hat{y}_i \neq y_i)/n_i$.
Total das médias de classificações erradas = total das médias de classificações erradas + média das i-ésimas classificações erradas
End i Loop.
Erro médio da CV para o(s) parâmetro(s) = total das médias de classificações erradas / k.

Shuffle and Split (SS)

Esta é a forma de validação cruzada utilizada por padrão nos softwares de aprendizagem de máquinas R e Matlab. Neste processo, os dados são, primeiramente, embaralhados e, em seguida, simplesmente divididos em k folds sem estratificação.

Algoritmo 2 - SS

For $i = 1$ to K
Misture as observações em ordem aleatória.
Divida as observações em partições k-fold.
Dado um parâmetro específico ou conjunto de parâmetros, execute
K-fold Cross-Validation Pseudocode.

Stratified Cross Validate (SCV)

De acordo com Bill (2015) este método difere do método SS, na medida em que esta técnica utiliza a estratificação ao dividir as amostras em k folds. Na SCV, as proporções de classe são preservadas em cada fold para garantir que cada fold seja representativo das proporções de classe no conjunto de dados original. Este processo é explicado no Algoritmo 3.

Algoritmo 3 - SCV

For $j = 1$ to Number of classes Loop
Para o j-ésimo label da classe, misture as observações em ordem aleatória.
Divida as amostras aleatórias para o j-ésimo label de classe em partições K-fold.
Quando as observações para a j-ésima classe não se dividem uniformemente nos folds, as suas observações extras são distribuídas, uma por vez, a partir do primeiro fold, em sequência até serem esgotadas.
Concatenar a divisão k fold da j-ésima classe, juntamente com as divisões da classe k fold da j-ésima -1 classe anterior, de modo que elas se acumulam.
End i Loop
Dado um parâmetro específico ou conjunto de parâmetros executados
K-fold Cross-Validation

Uma importante escolha ao utilizar este método é justamente o valor de k. De acordo com Clarke et al. (2009), k pode ser escolhido entre 5 e 15 dependendo de n e de outros aspectos da modelagem (ver Clarke et al., 2009, pg. 28). James et al. (2013) afirmam que existe um dilema entre viés e variância associado à escolha de k na VC k-Fold e que, devido a este fato, normalmente realiza-se a VC k-fold utilizando $k = 5$ ou $k = 10$. Esses valores foram sugeridos de forma empírica para produzir estimativas da taxa de erro de teste que não são de um viés excessivamente alto nem de uma variância muito alta (para maiores detalhes, ver James et al., 2013, pg. 184). Quando $k = n$, esse método é chamado de "leave one-out", ou "loo" VC.

2.4.3 Leave-One-Out Cross Validation (LOOCV)

A validação cruzada *leave-one-out* (Leave-One-Out Cross Validation - LOOCV) envolve dividir o conjunto de observações em duas partes, conforme figura 2.5. No entanto, ao invés de criar dois subconjuntos de tamanho comparável, uma única observação (x_1, y_1) é utilizada para o conjunto de teste e as observações restantes $\{(x_2, y_2), \ldots, (x_n, y_n)\}$ formam o conjunto de treinamento. O método de aprendizagem estatística é ajustado nas $n - 1$ observações de treinamento e uma predição y_1 é feita para a observação excluída, usando o seu valor x_1. Como (x_1, y_1) não foi utilizado no processo de ajustamento, $EQM_1 = (y_1 - \hat{y}_1)^2$ fornece uma estimativa não tendenciosa para o erro de teste.

Esse processo pode ser repetido selecionando (x_2, y_2) para validação, treinando o procedimento estatístico de aprendizagem nas observações $\{(x_1, y_1), (x_3, y_3) \ldots, (x_n, y_n)\}$ e computando $EQM_2 = (y_2 - \hat{y}_2)^2$. Repetindo isso n vezes produz-se n erros quadráticos, $EQM_1, \ldots,$ EQM_n. A estimativa LOOCV é a média destes n erros de teste (James et al., 2013),

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} EQM_i. \tag{2.11}$$

A LOOCV tem algumas vantagens importantes. Em primeiro lugar, apresenta um pequeno viés. O método de aprendizagem estatística é aplicado repetidamente usando conjuntos de treinamento que contêm $n - 1$ observações, que são quase todas as que estão no conjunto de dados. Logo, a LOOCV tende a não sobrestimar tanto a taxa de erro de teste. Em segundo lugar, a realização da LOOCV múltiplas vezes sempre produzirá os mesmos resultados: não há aleatoriedade no conjunto de treinamento/teste.

A desvantagem é que a LOOCV pode se tornar um processo muito demorado se n for grande, uma vez que o modelo tem que ser ajustado n vezes. Através de um atalho realizado

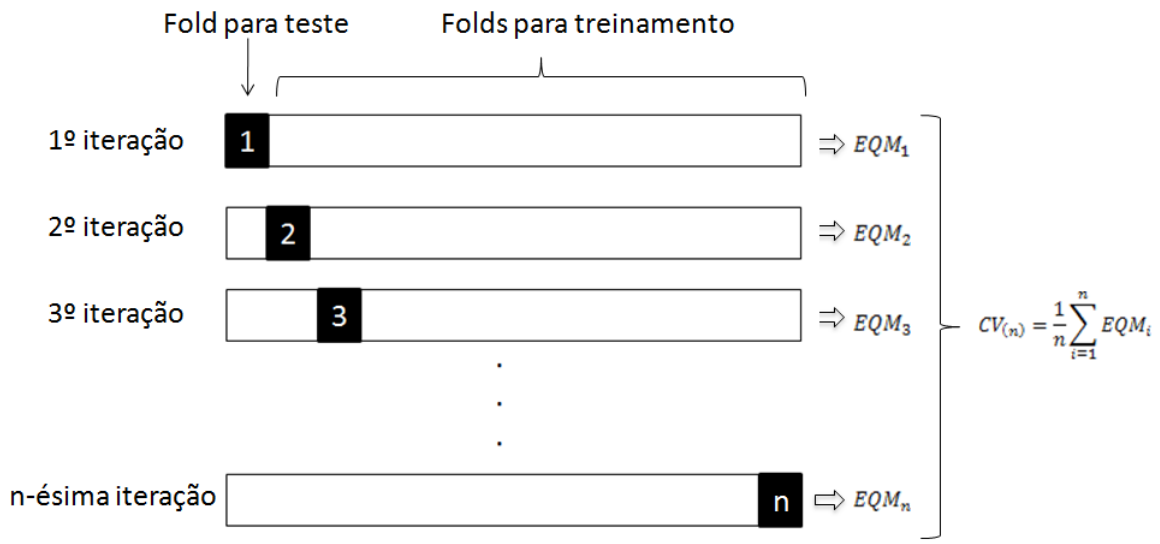


Figura 2.5: Exibição esquemática da LOOCV. Um conjunto de n pontos de dados é repetidamente dividido em um conjunto de treinamento (parte branca do retângulo) contendo todas as observações, exceto uma, que é o conjunto de validação que contém apenas essa observação (representada pela parte preta do retângulo). Adaptado de James et al. (2013).

com regressão linear ou polinomial de mínimos quadrados, a seguinte fórmula torna-se válida:

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - h_i}\right)^2, \tag{2.12}$$

em que $\hat{y}_i$ é o i-ésimo valor ajustado pelo método dos mínimos quadrados e h_i é a estatística de alavancagem definida por

$$h_i = \frac{1}{n} + \frac{(x_i - \overline{x})^2}{\sum_{i'=1}^{n}(x_{i'} - \overline{x})^2}.$$

A alavancagem está entre $1/n$ e 1, e reflete o quanto uma observação influi em sua própria capacidade.

LOOCV é um método muito geral e pode ser usado com qualquer tipo de modelagem preditiva, por exemplo, na regressão logística ou na análise discriminante linear. A fórmula (2.12) não é válida para o caso geral, caso em que o modelo deve ser redefinido n vezes.

A validação cruzada pode ser realizada em vários métodos de aprendizagem estatística, ou em um único método, a fim de identificar o método que resulta no menor erro de teste. Se for levada em consideração a redução do viés, o LOOCV é preferível em relação ao VC k-fold. Todavia, o LOOCV tem variação maior que o VC k-fold com $k < n$.

A validação cruzada também pode ser utilizada na classificação, caso em que a resposta Y é qualitativa. Nesse contexto, a validação cruzada funciona da mesma forma que no ajuste da regressão, exceto que ao invés de usar o EQM para quantificar o erro do teste, usa-se o número de observações mal classificadas. Na taxa de erro LOOCV tem-se (James et al., 2013):

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i, \tag{2.13}$$

onde $Err_i = I(y_i \neq \hat{y}_i)$. A taxa de erro VC k-fold é definida de forma análoga.

Capítulo 3

Técnicas de Classificação

Em diversas situações há necessidade de se trabalhar com variável resposta qualitativa. Quando o objetivo é realizar previsões de respostas qualitativas, são utilizadas técnicas de classificação ou classificadores. Os classificadores são avaliados quase que exclusivamente em dados futuros.

A classificação tem como intuito alocar objetos de uma população em uma, duas ou mais categorias, tomando como base um conjunto de características em cada objeto. Como por exemplo, a classificação de pacientes em grupos de baixo, médio e alto risco.

Assim como na regressão, na classificação, os dados compreendem n pares $(X_i, Y_i), i = 1, 2, \ldots, n$ em que $x_i \in \mathbb{R}^P$. É importante utilizar os dados para definir quais componentes do vetor de covariáveis X são necessários para determinar a qual categoria Y_i, a i-ésima observação pertence. Essas informações podem ser usadas para buscar uma função de variáveis explicativas que identificarão a classe para um dado X (James et al., 2013).

Os problemas de classificação mais simples separam uma população em duas classes, rotuladas de 1 e 2. Os problemas de classificação binária quase sempre podem ser generalizados para problemas de classificação com múltiplas classes. A tarefa é encontrar uma função de decisão para discriminação entre dados de k diferentes classes, onde $k \geq 2$ (ver Clarke et al., 2009). O conjunto de treinamento de um classificador f consiste de amostras (x_i, y_i) para $i = 1, \ldots, n$ onde $x_i \in \mathbb{R}^p$ são vetores de características e $y_i \in \{1, \ldots, K\}$ é o rótulo da classe para a i-ésima amostra. Com base no conjunto de treinamento, o objetivo principal é o aprendizado da regra de decisão,

$$f(x) : \mathbb{R}^p \longrightarrow \{1, \ldots, K\},$$

usada para separar as K classes e prever o rótulo da classe para uma nova entrada $x = x_{new}$.

Geralmente, um multiclassificador previamente treinado é associado a uma função K-dimensional

$$D(x) = (d_1(x), \ldots, d_K(x)),$$

em que $d_k(x)$ representa a força da evidência com que x pertence à classe k. O Classificador ϕ induzido a partir de f é definido como

$$f(x) = \arg \max_{k=1,\ldots,K} d_k(x). \tag{3.1}$$

A fronteira de decisão entre as classes k e l é descrita pelo conjunto

$$\{x \in \mathbb{R}^P : d_k(x) = d_l(x)\} \forall k \neq l.$$

Se K não é demasiadamente grande, uma forma de simplificar problemas multiclasse é transformá-los em uma série de problemas binários. Cada $d_k(x)$ é treinado para separar a

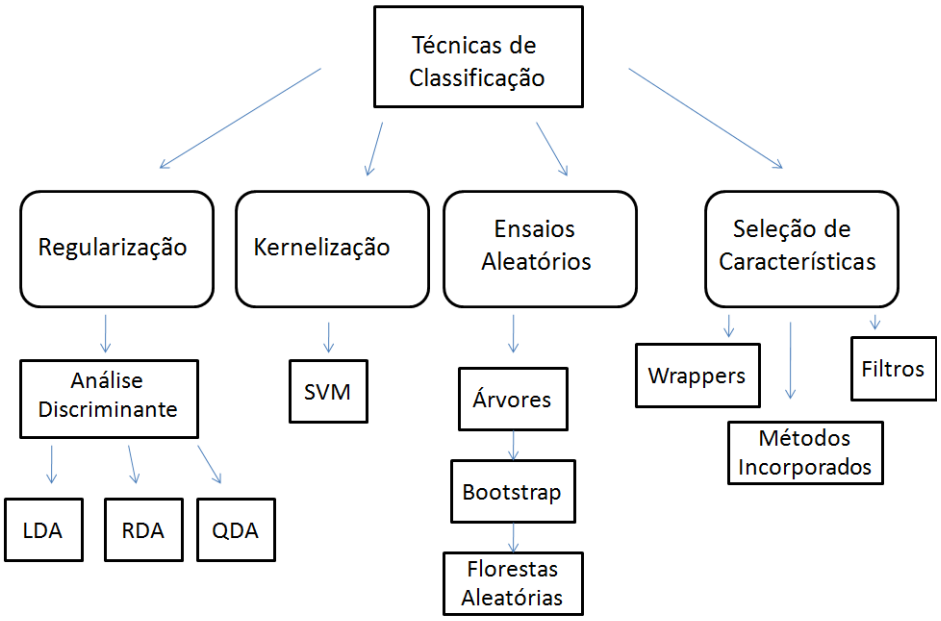


Figura 3.1: Técnicas de Classificação e suas técnicas específicas. Fonte: Adaptado de Bill (2015).

classe k das restantes. Estes K classificadores binários são, então, combinados para dar uma classificação final

$$\hat{f}(x) = \arg \max_{k=1,\dots,K} \hat{d}_k(x). \tag{3.2}$$

Contudo, no caso geral, os classificadores mais utilizados são: a regressão logística, a análise discriminante linear e os k - vizinhos mais próximos, bem como os seus métodos de uso intensivo, tais como árvores, florestas e máquinas de vetores e suporte. Os métodos discriminantes e SVMs são mais utilizados para o caso binário ($k = 2$).

A Figura 3.1 apresenta quatro tipos de técnicas de classificação, entre elas: Regularização, Kernelização, Ensaios Aleatórios e Seleção de Característica. No entanto, este trabalho aborda apenas três dessas técnicas, a saber: Regularização, Kernelização e Ensaios Aleatórios. Conforme pode ser observado na Figura 3.1, na técnica de regularização discutiremos sobre a Análise Discriminate, que se subdivide em Análise Discriminante Linear (LDA), Análise Discriminante Quadrática (QDA) e Análise Discriminante Regularizada (RDA). A técnica de Kernelização abrange as Máquinas de Vetores de Suporte (SVM). Por sua vez, nos Ensaios Aleatórios serão apresentadas as árvores e as Florestas Aleatórias.

3.1 Técnica de Regularização

3.1.1 Análise Discriminante

A análise discriminante tem como objetivo construir uma função classificadora F que atribui pontos a uma classe de variáveis ou rótulos $y \in \{1, 2, \dots, k, \dots, K\}$ com base num conjunto de medições $X = (x_1, x_2, \dots, x_p)$ de modo que o erro de classificação é tão pequeno quanto possível (Bill, 2015; Fokoué e Bill, 2014). O *custo* ou *perda* associado ao erro do classificador é definido como:

$$L(k, \hat{k}) \quad 1 \leq k, \hat{k} \leq K, \tag{3.3}$$

em que $L(k,k) = 0$ e $L(k,\hat{k}) \geq 0$, sendo k o grupo correto na atribuição da classe e $\hat{k}$ a atribuição que foi realmente feita.

De acordo com Friedman (1989), os valores do vetor de medição associadas com todos os membros de cada classe k compreendem uma distribuição de valores caracterizada por uma densidade de probabilidade $f_k(\underline{X})$. O objetivo usual é minimizar o risco de erro de classificação, que é definido como a perda esperada de classificação incorreta, $E[L(k,\hat{k})]$ sobre a amostra a ser classificada. O risco (perda esperada) incorrido na classificação de um objeto com o vetor de medida $\underline{X}$ na categoria $\hat{k}$ é

$$R(\hat{k}|\underline{X}) = E[L(k,\hat{k})] = \frac{\sum_{k=1}^{K} L(k,\hat{k}) f_k(\underline{X})\pi_k}{\sum_{k=1}^{K} f_k(\underline{X})\pi_k}, \tag{3.4}$$

em que π_k é a probabilidade incondicional a priori de observar um membro de classe k. O risco $R(\hat{k}|\underline{X})$ é minimizado ao escolher k que minimiza o numerador da equação (3.4). Para o caso especial da perda $0-1$,

$$L(k,\hat{k}) = 1 - \delta(k,\hat{k}), \tag{3.5}$$

em que

$$\delta(k,\hat{k}) = \begin{cases} 1, & \text{se} \quad k = \hat{k}, \\ 0, & \text{se} \quad k \neq \hat{k}, \end{cases}$$

com base em (3.4) e (3.5) o risco $R(\hat{k}|\underline{X})$ é minimizado por

$$\hat{y} = f_{\hat{k}}(\underline{X})\pi_{\hat{k}} = \max_{1 \leq k \leq K} f_k(\underline{X})\pi_k. \tag{3.6}$$

A matriz de perdas associadas à equação (3.5) associa uma perda de uma unidade para cada erro, independente do tipo do mesmo. O risco de reclassificação é simplesmente a fração de atribuições que são incorretas. A regra resultante de se escolher $\hat{k}$ que minimize $R(\hat{k}|\underline{X})$ é conhecida como *Regra de Bayes* e esta minimiza o risco de má classificação entre todas as regras possíveis.

Considerando a perda $0-1$, a expressão reduz-se a

$$d_k = P(Y = k|\underline{X}) = \frac{f_k(\underline{X})\pi_k}{\sum_{l=1}^{k} f_l(x)\pi_l}, \tag{3.7}$$

onde $f_k(\underline{X}) = p_k(x)$ é a densidade condicional de $X = x$ dado o valor predito k para aquela observação, enquanto $\pi_k = P(Y = k)$, baseado na informação a priori e d_k tem a funcionalidade explicitada na expressão (3.1).

A classe $f_k(\underline{X})$ raramente é conhecida. Na maioria das vezes obtém-se uma amostra de observações de cada classe que tenham sido corretamente classificadas por algum mecanismo externo. O objetivo é usar estas observações como "amostra de treinamento" (training sample) para construir uma regra de classificação ao obter-se estimativas adequadas de $f_k(\underline{X})$, para cada rótulo k de classes.

Considerando a perda $0-1$ e a priori π_j para a população, Π_j com densidade f_j, $j = 1, \ldots, k$, a regra de classificação ótima, também chamada de Regra de Bayes, é dada por: Classifique uma observação X_i em Π_j se e somente se $\pi_j f_j(X_i) \geq \pi_l f_l(X_i), l = 1, 2, \ldots, k$.

Quando as probabilidades a priori das classes são também desconhecidas, pode-se juntar todos os dados que, desta forma, podem ser usados como sendo uma amostra aleatória a partir da população dos dados agregados. Dessa forma, as probabilidades a priori de cada classe podem ser estimadas pela fração de cada classe na amostra agregada.

Para estimar as probabilidades a priori de cada classe na amostra agregada, as seguintes equações são usadas:

$$Z_{ik} = \begin{cases} 1, & \text{se } y_i = k \\ 0, & \text{caso cotrário.} \end{cases}$$

$$\hat{\pi}_k = \frac{n_k}{n} = \frac{\text{número de observações } i \text{ na classe } k}{\text{tamanho da amostra}},$$

onde

$$n_k = \sum_{i=1}^{n} Z_{ik}.$$

O estimador da média da classe é dado por

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i=1}^{n} Z_{ik} X_i = \overline{X}_k. \tag{3.8}$$

A matriz de variância-covariância estimada comum ou agregada é dada por

$$\hat{\Sigma} = S = \frac{\sum_{l=1}^{k} (n_l - 1) S_k}{\sum_{l=1}^{k} (n_l - 1)}. \tag{3.9}$$

Análise Discriminante Linear (LDA)

Suponha que a regra de classificação de cada categoria k é baseada em uma distribuição normal p-variada com vetor de médias μ_k e matriz de covariâncias Σ_k, isto é $x_k \sim N_p(\mu_k, \Sigma_k)$. Adicionalmente, assume-se igualdade das matrizes de covariância, $\Sigma_k = \Sigma$. Desta forma,

$$f_k(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k)}. \tag{3.10}$$

Considerando (3.6) e (3.10),

$$\begin{aligned} \hat{y} = \arg\max_k [P(y = k | X = x)] &= \arg\max_k [f_k(x)\pi_k] = \arg\max_k [\log(f_k(x))\pi_k] \\ &= \arg\max_k \left[-\log((2\pi)^{p\backslash 2} |\Sigma|^{1\backslash 2}) - \frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(\pi_k) \right] \\ &= \arg\max_k \left[-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) + \log(\pi_k) \right]. \end{aligned} \tag{3.11}$$

Note que

$$-\frac{1}{2}(x - \mu_k)^T \Sigma^{-1} (x - \mu_k) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k - \frac{1}{2} x^T \Sigma^{-1} x,$$

desse modo, para cada classe k a função discriminante linear é definida por

$$d_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k.$$

Portanto,

$$\hat{y} = \arg\max_k d_k(x).$$

A *fronteira de decisão* entre as classes k e l é escrita por

$$\{x : d_k(x) = d_l(x)\},$$

ou, equivalentemente, a fronteira de decisão entre as classes k e l é definida quando

$$0 = \log\left(\frac{d_k(x)}{d_l(x)}\right) = \log\left(\frac{\pi_k}{\pi_l}\right) - \frac{1}{2}(\mu_k + \mu_l)^T \Sigma^{-1}(\mu_k - \mu_l) + x^T \Sigma^{-1}(\mu_k - \mu_l) = 0.$$

Os parâmetros das distribuições Gaussianas são estimados utilizando os dados de treinamento, onde: $\hat{\pi}_k = N_k/N$, sendo N_k o número de observações da classe k; $\hat{\mu}_k = \sum_{l=1}^{N} x_l I(g_l = k)$; $\hat{\Sigma} = \sum_{k=1}^{K} \sum_{g_i=k} (x_i - \hat{\mu}_k)(\mu_i - \hat{\mu}_k)^T/(N - K)$ e g_i é uma variável categórica que identifica a população.

De acordo com Clarke et al. (2009), essas estimativas podem ser instáveis na presença de outliers. Portanto, muitas vezes são preferidos estimadores mais robustos.

Exemplo de classificação Binária: Sejam $k = 1$ e $l = 2$, defina:

$$a_0 = \log\left(\frac{\pi_1}{\pi_2}\right) - \frac{1}{2}(\mu_1 + \mu_2)^T \Sigma^{-1}(\mu_1 - \mu_2),$$

e

$$(a_1, a_2, \ldots, a_p)^T = \Sigma^{-1}(\mu_1 - \mu_2).$$

Regra de classificação: Classifique a i-ésima observação na classe 1 se $a_0 + \sum_{i=j}^{p} a_j x_j > 0$ e na classe 2, caso contrário.

Para o caso de $\pi_1 = \pi_2 = 0.5$, $\mu_1 = (0, 0)^T$, $\mu_2 = (2, -2)^T$ e $\Sigma = \begin{pmatrix} 1.0 & 0.0 \\ 0.0 & 0.5625 \end{pmatrix}$, a fronteira de decisão é dada por: $5,56 - 2.00x_1 + 3.56x_2 = 0.0$.
Logo, a regra de decisão é dada por:

$$\hat{y}(x) = \begin{cases} 1, & \text{se} \quad 5,56 - 2.00x_1 + 3.56x_2 \geq 0 \\ 2, & \text{caso contrário.} \end{cases}$$

A LDA pressupõe que a diferença entre as classes está relacionada apenas com o vetor médio, assumindo que a matriz de covariância é igual para todas as classes. Deste modo, supõe-se que as duas classes possuem matrizes de covariância idênticas, mas distintos vetores média e a fronteira de decisão é definida por uma função linear.

A Figura 3.2 mostra as duas classes linearmente separáveis pela fronteira: $5.56 - 2.00x_1 + 3.56x_2 = 0.0$. Nota-se que as classes são linearmente separáveis porque não há sobreposição das classes identificadas nas fronteiras de decisão. A LDA tenta encontrar uma transformação linear através da maximização da distância entre-classes e minimização da distância intra-classe. Desta forma, tenta encontrar também, a melhor direção, de tal forma que quando os dados são projetados em um plano, as classes podem ser separadas.

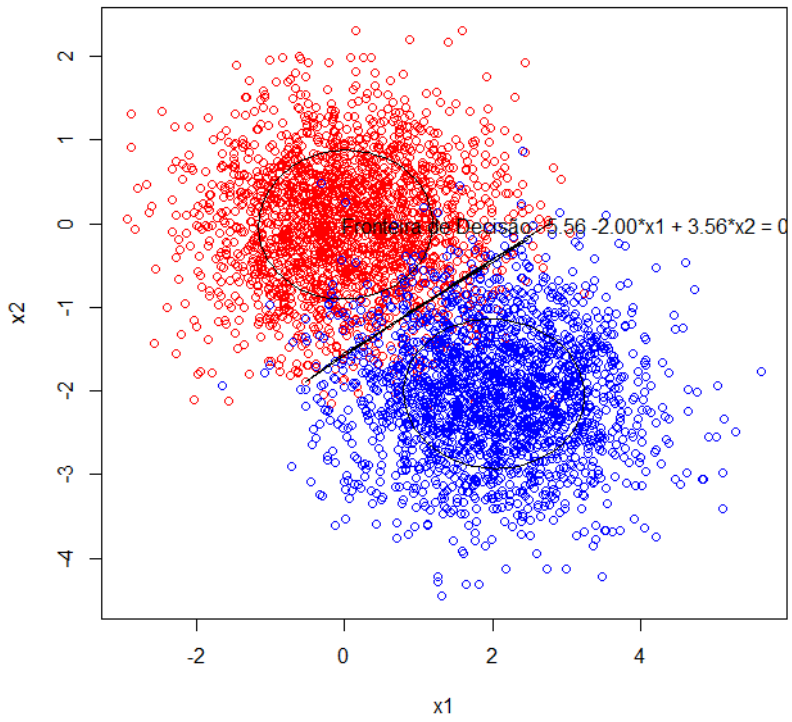


Figura 3.2: Gráfico de dispersão de objetos com hiperplano de separação de classes obtidos por LDA.

Para programar a figura 3.2, foram utilizados os comandos no `R`:

```
install.packages("MASS")
library(MASS)
install.packages("ellipse")
library(ellipse)
Sigma <- matrix(c(1,0,0,0.5625),2,2)
pop1<-mvrnorm(n = 2000, c(0, 0), Sigma)
pop2<-mvrnorm(n = 2000, c(2, -2), Sigma)
##FRONTEIRA DE DECISÃO
pi1=pi2=0.5
aux1<-t(solve(Sigma)%*%(c(2,-2)-c(0,0)))
aux2<- (c(2,-2)-c(0,0))%*%solve(Sigma)%*%(c(2,-2)-c(0,0))/2 +log(0.5/0.5)
fronteira<-function(xx){
yy<- -(aux1[1]*xx[,1]+aux1[2]*xx[,2]-aux2)
return(yy)
}
fr1<-fronteira(rbind(pop1,pop2))
pop1.1<-cbind(pop1,rep(2,nrow(pop1)))
pop2.1<-cbind(pop2,rep(4,nrow(pop2)))
##SEPARANDO OS PARES PRÓXIMOS À FRONTEIRA DE DECISÃO
mat1<-cbind(rbind(pop1,pop2),fr1)
mat2<-mat1[abs(mat1[,3])<=0.08,]
mat.full<-rbind(pop1.1,pop2.1)
win.graph()
plot(mat.full[,c(1,2)],col=mat.full[,3],xlab="x1",ylab="x2")
lines(mat2[,1],mat2[,2],type="l",col=1)
text(3.0,0, paste("Fronteira de Decisão:  5.56 -2.00*x1 + 3.56*x2 = 0"))
centre1<-c(mean(pop1[,1]),mean(pop1[,2]))
centre2<-c(mean(pop2[,1]),mean(pop2[,2]))
win.graph()
plot(mat.full[,c(1,2)],col=mat.full[,3],xlab="x1",ylab="x2")
lines(ellipse(cov(pop1),centre=centre1,level=0.5))
lines(ellipse(cov(pop2),centre=centre2,level=0.5))
lines(mat2[,1],mat2[,2],type="l",col=1)
text(3.0,-0.005, paste("Fronteira de Decisão:  5.56 -2.00*x1 + 3.56*x2 = 0"))
```

Análise Discriminate Quadrática (QDA)

Assim como na LDA, na Análise Discriminante Quadrática (QDA) as observações dentro de cada classe K seguem uma distribuição normal multivariada e as estimativas para os parâmetros são obtidas através do teorema de Bayes.

Todavia, na QDA cada classe tem sua própria matriz de covariância, ou seja, uma observação pertencente à k-ésima classe é da forma $X \sim N(\mu_k, \Sigma_k)$, onde Σ_k é uma matriz de covariância para a K-ésima classe. Diante disso, o classificador de Bayes atribui uma observação $X = x$ para a classe:

$$\delta_k(x) = -\frac{1}{2}(x-\mu_k)^T\Sigma_k^{-1}(x-\mu_k) + \log\pi_k = -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma_k^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k + \log\pi_k. \tag{3.12}$$

A quantidade x aparece como um termo quadrático na equação, o que explica de onde surgiu o nome QDA.

A regra de classificação ótima pode ser descrita como:

Classifique X em Π_j se e somente se

$$\frac{1}{2}\log(|\Sigma_j|) - \frac{1}{2}\{(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\} \geq \frac{1}{2}\log(|\Sigma_l|) - \frac{1}{2}\{(x-\mu_l)^T\Sigma_l^{-1}(x-\mu_l)\}, l = 1, 2, \ldots, k. \tag{3.13}$$

Os estimadores dos parâmetros μ_j e Σ_j são dados por:

$$\hat{\mu}_j = \Sigma_{g_i=j}\frac{x_i}{N_j}$$

e

$$\hat{\Sigma}_j = \frac{1}{N_j - 1}\sum_{l=1}^{N}(x_l - \hat{\mu}_j)(x_l - \hat{\mu}_j)^T I(g_l = j), j = 1, \ldots, k,$$

e a regra quadrática (3.13) é escrita em função dessas quantidades.

A QDA é recomendada quando o conjunto de observações for muito grande, ou se não for sensato supor matriz de covariância comum para as classes k. Por sua vez, a LDA é uma alternativa a ser utilizada caso houver poucas observações e no caso de análises reais parsimoniosas.

Thomas (2017) realizou algumas análises comparativas entre a LDA e a QDA. Para isso, utilizou a biblioteca `MASS` do software `R` com a função discriminante linear. Também foram utilizados, o conjunto de dados `Mathlevel` do pacote `Ecdat` para prever o sexo de uma pessoa entrevistada, com base na pontuação em algumas disciplinas e no número de aulas que entrevistado assistiu. O código inical para a análise é dado por:

```
install.packages("Ecdat")
library(MASS)
library(Ecdat)
data("Mathlevel")
```

Primeiramente, o autor removeu todos os dados faltantes para executar o modelo. Em seguida, criou um conjunto de dados chamado "math" que contém o conjunto de dados `Mathlevel`, conforme pode ser observado a seguir:

```
math<-na.omit(Mathlevel)
set.seed(123)
math.ind<-sample(2,nrow(math),replace=T,prob = c(0.7,0.3))
math.train<-math[math.ind==1,]
math.test<-math[math.ind==2,]
```

Observa-se também que o próximo passo foi especificar a semente que gera os dados por meio da função `set.seed()`. Os dados foram divididos utilizando uma divisão 70/30. Então, os dados do conjunto de treinamento foram chamados de `math.train` e os do conjunto de teste, de `math.test`.

Para realizar a LDA é utilizada a função `lda` do pacote `MASS`. O modelo foi chamado de `lda.math` e inclui todas as variáveis do conjunto de dados `math.train`,

```
lda.math<-lda(sex~.,math.train)
lda.math

# Call:
## lda(sex ~ ., data = math.train)
##
## Prior probabilities of groups:
##      male    female
## 0.5986079 0.4013921
##
## Group means:
##        mathlevel.L mathlevel.Q mathlevel.C mathlevel^4 mathlevel^5
## male   -0.10767593  0.01141838 -0.05854724   0.2070778  0.05032544
## female -0.05571153  0.05360844 -0.08967303   0.2030860 -0.01072169
##        mathlevel^6      sat languageyes  majoreco  majoross   majorns
## male    -0.2214849 632.9457  0.07751938 0.3914729 0.1472868 0.1782946
## female  -0.2226767 613.6416  0.19653179 0.2601156 0.1907514 0.2485549
##          majorhum mathcourse physiccourse chemistcourse
## male   0.05426357   1.441860    0.7441860      1.046512
## female 0.07514451   1.421965    0.6531792      1.040462
##
## Coefficients of linear discriminants:
##                       LD1
## mathlevel.L    1.38456344
## mathlevel.Q    0.24285832
## mathlevel.C   -0.53326543
## mathlevel^4    0.11292817
## mathlevel^5   -1.24162715
## mathlevel^6   -0.06374548
## sat           -0.01043648
## languageyes    1.50558721
## majoreco      -0.54528930
## majoross       0.61129797
## majorns        0.41574298
## majorhum       0.33469586
## mathcourse    -0.07973960
## physiccourse  -0.53174168
## chemistcourse  0.16124610
```

O comando `lda.math` apresenta as probabilidades de um entrevistado ser do sexo masculino ou feminino, a média de cada variável por sexo e os coeficientes de discriminação linear. Conforme o autor, a saída mostra que há uma grande quantidade de sobreposição entre os sexos masculino e feminino no modelo, o que indica uma grande quantidade de erros de classificação, porque os dois grupos não estão claramente separados.

Desta forma, o próximo passo é usar a função `predict` nos dados do conjunto de treinamento para observar como o modelo está classificando os respondentes por gênero. A predição do modelo será comparada com a classificação real.

```
math.lda.predict<-predict(lda.math)
math.train$lda<-math.lda.predict$class
table(math.train$lda,math.train$sex)

##
##          male female
##   male    219    100
##   female   39     73
mean(math.train$lda==math.train$sex)
## [1] 0.6774942
```

Conforme pode ser observado, existem muitos erros de classificação, provavelmente, muitos respondentes do sexo masculino estão sendo classificados como do sexo feminino.

A mesma análise será realizada no conjunto de dados de teste:

```
lda.math.test<-predict(lda.math,math.test)
math.test$lda<-lda.math.test$class
table(math.test$lda,math.test$sex)
##
##          male female
##   male     92     43
##   female   23     20
mean(math.test$lda==math.test$sex)
## [1] 0.6292135
```

Os resultados para a análise no conjunto de teste foram semelhantes. Logo, o modelo não parece ser muito adequado. O motivo principal é que há pouca distinção entre machos e fêmeas. Desta forma, Thomas (2017) sugere a utilização da QDA a fim de verificar a possibilidade um melhor desempenho em comparação à LDA. Uma vez que a QDA permite que cada classe na variável dependente tenha sua própria covariância ao invés de uma covariância compartilhada, como no LDA. Para realizar a QDA é necesário utilizar a função `qda` do pacote `MASS`. Logo, o código para os dado do conjunto de treinamento ficará da seguinda forma:

```
math.qda.fit<-qda(sex~.,math.train)
math.qda.predict<-predict(math.qda.fit)
math.train$qda<-math.qda.predict$class
table(math.train$qda,math.train$sex)
##
##          male female
##   male    215     84
##   female   43     89
mean(math.train$qda==math.train$sex)
## [1] 0.7053364
```

Observa-se que quase não há diferença entre os resultados. O código para a análise realizada no conjunto de dados de teste também é apresentado:

```
math.qda.test<-predict(math.qda.fit,math.t
math.test$qda<-math.qda.test$class
table(math.test$qda,math.test$sex)
##
##          male female
##   male     91     43
##   female   24     20
mean(math.test$qda==math.test$sex)
## [1] 0.6235955
```

Como pode ser observado, não há quase nenhuma diferença entre os resultados da LDA e da QDA. No entanto, independente dos resultados, a intenção deste exemplo é proporcionar uma visão geral de aplicação dessas duas técnicas.

Análise Discriminante Regularizada (RDA)

Considere o caso da LDA em que estima-se uma única matriz de covariâncias Σ através da matriz de variância e covariância combinada (*pooled*) das diversas classes. Já no caso do QDA estima-se uma matriz de covariância, $\hat{\Sigma}_i$ para cada população i. Com a RDA objetiva-se reduzir o efeito do número de parâmetros a estimar ao interpolar os casos que conduzem à LDA e QDA.

Seja a mistura de matrizes de covariâncias dada por

$$\hat{\Sigma}_i(\alpha) = (1-\alpha)\hat{\Sigma}_i + \alpha\hat{\Sigma}, \tag{3.14}$$

em que $\alpha \in [0,1]$. O valor α controla o grau de complexidade do modelo ou o grau da aproximação (shrinkage) de $\hat{\Sigma}_i(\alpha)$ com respeito à estimativa combinada $\hat{\Sigma}$. Quando $\alpha = 1$, o problema reduz-se a LDA. Quando $\alpha = 0$, o problema reduz-se ao QDA. Valores entre os limites de α representam graus de regularização menos severos do que a LDA, representando um meio termo entre a LDA e a QDA.

De acordo com Friedman (1989), frequentemente o uso de uma pequena regularização é capaz de eliminar grande parte da instabilidade do problema. Quando as matrizes $\hat{\Sigma}_i$ diferem substancialmente, o uso de valores menores de α (menores do que $\alpha = 1$) melhoram consideravelmente o desempenho do problema.

Quando $n \lll p$, nem a QDA e nem a LDA podem ser utilizadas devido ao mal condicionamento das matrizes de covariância $\hat{\Sigma}_k$. Friedman mostrou que este problema pode ser contornado pela aplicação da regularização para estabilizar a matriz de covariância $\hat{\Sigma}_k$.

Guo et al. (2005) observam que a regularização das estimativas dos parâmetros é um protocolo comum para introduzir uma pequena quantidade de viés na matriz de covariância em troca do ganho na estabilização da variância e, assim, reduzir o erro de generalização.

A RDA utiliza a regularização do método LDA para resolver o problema da singularidade que acontece quando a matriz de covariância amostral é singular e não pode ser invertida. É muito comum a ocorrência de matrizes de covariância singulares para conjuntos de dados em que $n \lll p$.

Seja $\alpha \in (0,1)$, a versão regularizada de $\hat{\Sigma}$ é dada por

$$\tilde{\Sigma} = \alpha\hat{\Sigma} + (1-\alpha)I_p, \qquad 0 \leq \alpha < 1. \tag{3.15}$$

em que I_p é a matriz identidade.

O hiperparâmetro de regularização α é utilizado para resolver o problema da singularidade e estabilizar a estimativa de covariância, $\hat{\Sigma}$. O parâmetro α controla a aproximação de $\tilde{\Sigma}$ em relação à matriz identidade I_p.

Este processo de regularização tem o efeito de atenuar grandes valores de autovalores e de aumentar a influência de autovalores pequenos. Isso tem impacto positivo na inversa da matriz de covariância $\tilde{\Sigma}$, que é utilizada para definir uma função de discriminação quadrática.

A função discriminante regularizada tem a forma

$$d_k^*(x) = (x - \overline{x}_k)^T \tilde{\Sigma}_k^{-1}(x - \overline{x}_k) + ln|\tilde{\Sigma}| - 2ln\pi_k, \tag{3.16}$$

em que $\tilde{\Sigma}_k$ é definido de acordo com a expressão 3.14.

A estimativa RDA de y é dada por

$$\hat{y} = \hat{f}_{RDA}(x) = \arg \max_{k \in \{1,\dots,k\}} \{d_k^*(x)\}. \tag{3.17}$$

Existem várias variações da análise discriminante regularizada. Nesta pesquisa optou-se por utilizar o método SCRDA (Shrunken Centroid Regularized Discriminant Analysis) de Guo et al. (2005). Os métodos apresentados por estes autores são muito úteis para problemas de regularização em dados de microarray, que têm a característica de $n \lll p$. Adicionalmente, Guo, Hastie e Tibshirani (2015) propuseram o pacote R {`rda`}, ampliando, consideravelmente, a utilização das técnicas propostas em Guo et al. (2005). A seguir detalha-se o métado SCRDA de Guo et al. (2005).

O método SCRDA

Como mencionado, a expressão (3.15) é útil para resolver problemas de singularidade de matrizes de covariância. Uma outra possibilidade é a regularização da matriz de covariância amostral

$$\hat{R} = \hat{D}^{-1/2}\hat{\Sigma}\hat{D}^{-1/2}, \tag{3.18}$$

em forma similar:

$$\tilde{R} = \alpha\hat{R} + (1 - \alpha)I_p, \tag{3.19}$$

em que $\hat{D}$ é a matriz diagonal obtida ao tomar os elementos da diagonal de $\hat{\Sigma}$.

Posteriormente calcula-se a matriz de covariância regularizada através de

$$\tilde{\Sigma} = \hat{D}^{1/2}\tilde{R}\hat{D}^{1/2}. \tag{3.20}$$

Com base em 3.15 ou 3.20, a correspondente função de discriminação regularizada é dada por

$$\tilde{d}_g(x) = x^T\tilde{\Sigma}^{-1}\overline{x}_g - \frac{1}{2}\overline{x}_g^T\tilde{\Sigma}^{-1}\overline{x}_g + \log \pi_g, \tag{3.21}$$

sendo

- $\hat{\mu}_g = \overline{x}_g = \frac{1}{n_g}\sum_{i=1}^{n_g} x_{g,i}$;
- $x_{g,i}$ é a i-ésima observação da população g, com $x_{g,i} \in \mathbb{R}^P$;
- $x_{g,i}$ segue distribuição $N_p(\mu_g, \Sigma), 1 \leq g \leq G, 1 \leq i \leq n_g$;

- $n = n_1 + n_2 + \ldots + n_G$;
- $\hat{\Sigma} = \frac{1}{n}(X - \overline{X})(X - \overline{X})^T$;
- $X = X_{p\times n} = [x_{1,1}, \ldots, x_{1,n_1}, \ldots, x_{G,1}, \ldots, x_{G,n_G}]$;
- $X_{p\times n} = [\hat{\mu}_1, \ldots, \hat{\mu}_G]$.

O método SCRDA é baseado no método NSC (Nearest Shrunken Centroids) desenvolvido por Tibshirani et al. (2003). No método NSC, o centróide do grupo ao qual uma dada observação pertence, é ajustado (shrunken) individualmente.

Tibshirani et al. (2003) formalizaram sua teoria utilizando aplicações a dados de microarrays. No entanto, a mesma é válida para qualquer tipo de dado em que $p \ggg n$.

Seja $x_{g,i,j}$ a j-ésima componente do vetor p-dimensional associado à i-ésima observação da população g. Dessa forma,

$$x_{g,i} = (x_{g,i,1}, x_{g,i,2}, \ldots, x_{g,i,p})^T.$$

Tibshirani et al (2003) designaram $x_{g,i,p}$ como sendo a "expressão" do gene j na amostra i da população g. A j-ésima componente do centroide para a classe k é

$$\overline{x}_{j,k} = \sum_{l=1}^{n_k} x_{k,l,j}/n_k, \tag{3.22}$$

que representa a "expressão média" da classe k para a coordenada j (gene j).

A j-ésima componente do centroide global é dada por

$$\overline{x}_j = \sum_{k=1}^{G}\sum_{l=1}^{n_k} x_{k,l,j}/n. \tag{3.23}$$

Os autores trabalham com as estimativas dos centroides associados a cada classe ou população e aplicam transformações, combinando a informação advinda de $\overline{x}_j$, de modo a aperfeiçoar a estimativa proporcionada por $\overline{x}_{jk}$. Tibsharani et al (2003) denominam tal procedimento por "ajustar" (to shrink) os centroides das classes em relação ao centroide global ("to shrink the class centroids toward the overall centroid"). Previamente a este procedimento os elementos $\overline{x}_{jk}$ são normalizados utilizando o desvio padrão interclasse combinado, S_j, da j-ésima componente. Dessa forma, seja

$$d_{jk} = \frac{\overline{x}_{jk} - \overline{x}_j}{m_k \times S_j}, \tag{3.24}$$

em que a variância interclasse combinada da classe j é dada por

$$S_j^2 = \frac{1}{n-G}\sum_{k=1}^{G}\sum_{l=1}^{n_k}(x_{k,l,j} - \overline{x}_{jk})^2, \tag{3.25}$$

e $m_k = \sqrt{1/n_k - y_n}$ possibilita associar a distribuição t-student a d_{ik}.

Com base na expressão (3.24) obtém-se

$$\overline{x}_{jk} = \overline{x}_j + m_k \times S_j \times d_{jk}, \tag{3.26}$$

com $\overline{x}_k = (\overline{x}_{1k}, \ldots, \overline{x}_{pk})^T$.

Na proposta dos autores, cada d_{jk} é "ajustado"(shrunk) com respeito ao valor nulo. Para tanto, aplica-se novo "ajuste" de modo a obter-se

$$\overline{x}_{jk}^{'} = \overline{x}_j + m_k \times S_j \times d_{jk}^{'}, \tag{3.27}$$

em que

$$d_{jk}^{'} = \text{sinal}(d_{jk})(|d_{jk}| - \Delta)_+, \tag{3.28}$$

sendo que o subscrito "+" indica a parte positiva ($t_+ = t$ se $t > 0$ e zero caso contrário) e $|\cdot|$ representa o valor absoluto do argumento.

Na expressão (3.28) o valor absoluto de cada d_{jk} é reduzido pelo valor Δ (desconta-se o valor Δ). No caso de $(|d_{jk}| - \Delta) < 0$, convenciona-se fixar tal valor como sendo zero. Na expressão (3.27) $\overline{x}_{jk}^{'}$ representa o novo centroide ajustado da classe k (new shrunken centroid). O método NSC de ajuste (shrinkage) é denominado pelos autores por *soft thresholding*. Guo et al. (2005) afirmam que o método NSC remove ruídos nos dados devido a flutuações aleatórias.

Utilizando as expressões (3.25) a (3.28), Tibsharani et al. (2003) calculam o escore, dado pela expressão 3.29, a ser usado no processo de classificação de uma dada observação x^* baseado no NSC:

$$d_{j,k}(x_j^*) = \frac{(x_j^* - \overline{x}_{jk}^{'})^2}{2S_j^2}. \tag{3.29}$$

A observação x^* é classificada no grupo k se k minimiza a soma dos escores para todas as p dimensões, isto é,

$$x^* \in \text{grupo k} = \arg\min_{k'}, \sum_{j=1}^{p} d_{j,k'}(x_j^*) - \log \pi_{k'},$$

que é equivalente a

$$x^* \in \text{grupo k} = \arg\min_{k'}, (x^* - \overline{x}_{k'}^{'})^T \hat{D}^{-1}(x^* - \overline{x}_{k'}^{'} - \log \pi_{k'}),$$

em que $\hat{D} = \text{diag}(S_1^2, \ldots, S_p^2)$.

3.2 Kernelização

3.2.1 Máquina de Vetor de Suporte (SVM)

A Máquina de Vetor de Suporte, mais conhecida por sua denominação em inglês *Support Vector Machine* (SVM), é um método de aprendizagem supervisionada originada da Teoria do Aprendizado Estatístico (TAE), também conhecida como teoria VC (Vapnik e Chervonenkis).

Conforme Lorena e Carvalho (2007), a TAE permite controlar a possibilidade de overfitting através do controle das medidas da margem do hiperplano. Essa teoria também busca a capacidade de generalização, que é a capacidade da máquina realizar uma classificação eficiente perante um conjunto de dados.

Um dos principais problemas relacionados à TAE consiste no aprendizado de uma função, ou classificador, capaz de classificar uma observação com base em qual lado da margem ela se encontra. Isso é conhecido como classificador de margem máxima. Espera-se que um classificador que tenha uma margem máxima de classificação nos dados de treinamento, também tenha uma margem máxima de classificação nos dados de testagem. Embora o classificador de

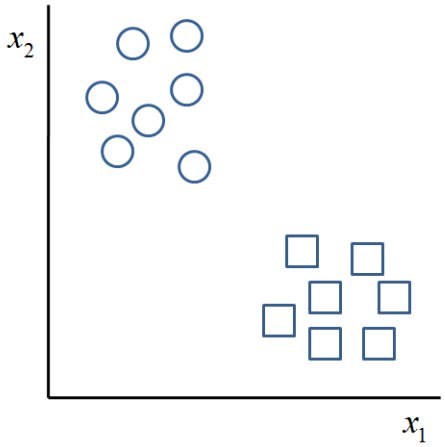


Figura 3.3: Plano $X = (X_1, X_2)$ descrevendo a região onde estão dispostos exemplares de duas características, círculos e quadrados.

margem máxima seja frequentemente bem sucedido, ele pode levar a sobrecarga quando p é grande (Lorena e Carvalho, 2007).

As SVMs são utilizadas para classificação de dados em duas classes, em que a ideia principal é definir uma fronteira (hiperplano) de separação máxima dos pontos mais próximos entre essas duas classes. Deste modo, supõe-se que se deseja classificar objetos m-dimensionais $\mathrm{x} = (x_1, x_2, \ldots, x_m)$ nas classes $+1$ e -1. Portanto, o conjunto de treinamento consiste de n observações $\mathrm{x}_i \in \mathbb{R}^m$, com suas respectivas classificações binárias.

Esta dissertação apresenta o uso da SVM na obtenção de fronteiras lineares que separam os dados pertencentes a duas classes. Em uma primeira abordagem, tem-se a SVM para o caso linearmente separável. Em seguida, para o caso não linearmente separável.

SVMs com Margens Rígidas

Seja T um conjunto de treinamento com n dados $\mathrm{x}_i \in X$ e seus respectivos rótulos $y_i \in Y$, em que X constitui o espaço dos dados e $Y = \{-1, +1\}$. T é linearmente separável se é possível separar os dados das classes $+1$ e -1 por um hiperplano. As SVMs utilizadas para classificação de conjuntos de treinamento linearmente separáveis são também denominadas SVMs com margens rígidas (Lorena e Carvalho, 2007).

Körting (2014), em sua videoaula, explica de forma bem didática o funcionamento do SVM para conjuntos binários linearmente separáveis. Considere o caso bidimensional e seja a Figura 3.3 em que se considera um plano descrito por $X = (X_1, X_2)$ e duas classes de objetos ou características (círculos e quadrados). O vetor X descreve um conjunto de covariáveis ou *inputs*. Com o SVM objetiva-se descrever um hiperplano de separação (vide linha vermelha na Figura 3.4) que possibilite classificar todo e qualquer objeto em uma das duas classes.

Na Figura 3.5 apresenta-se dois hiperplanos distintos, descritos pelas linhas verde e vermelha. Ambos os hiperplanos possibilitam a correta classificação de cada um dos objetos (círculos ou quadrados). No entanto, o "melhor hiperplano" (hiperplano ótimo) de separação é o que apresenta a *margem* máxima para ambas as classes. A margem é descrita pela distância entre um hiperplano e os elementos (das classes distintas) mais próximos a este hiperplano.

A Figura 3.6 descreve dois hiperplanos, Z_1 e Z_2, e suas respectivas margens (linhas tracejadas). Observa-se que as margens associadas ao hiperplano vermelho (Z_2) são maiores do que aquelas associadas ao hiperplano verde (Z_1). Nesse caso, o hiperplano vermelho é preferível ao verde.

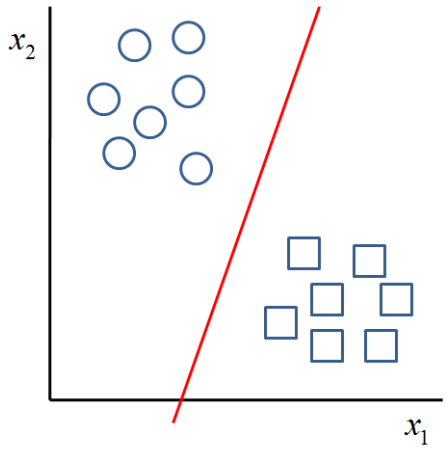


Figura 3.4: Hiperplano de separação entre as classes “círculos” e “quadrados”.

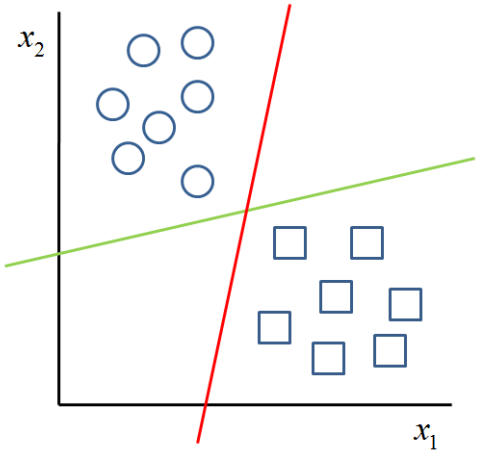


Figura 3.5: Hiperplanos de separação dos objetos (círculos ou quadrados). Linhas verde e vermelha.

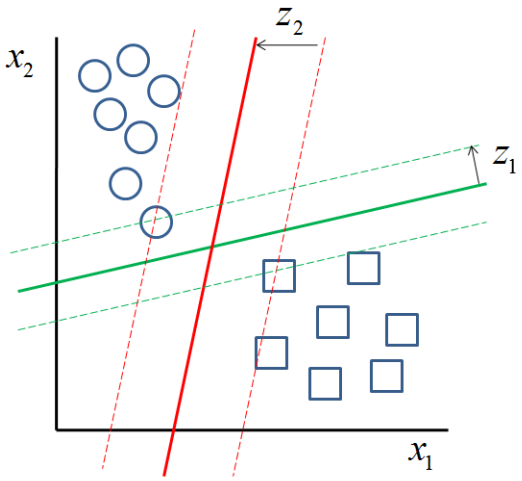


Figura 3.6: Hiperplanos de separação dos objetos (círculos ou quadrados). Linhas verde e vermelha.

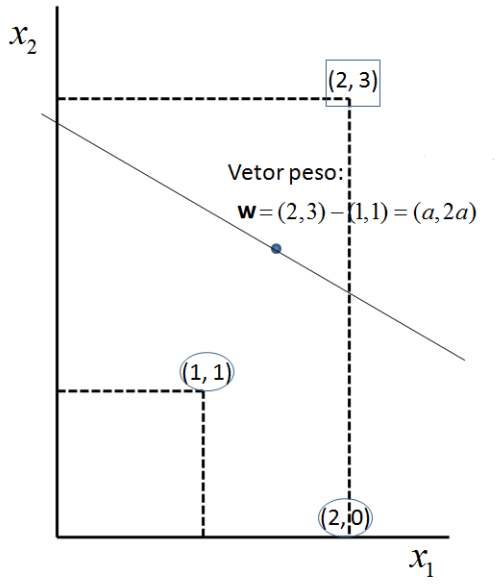


Figura 3.7: Exemplo de SVMs para Margens Rígidas.

Considere que o hiperplano vermelho é descrito pela equação $f(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x} + b$, em que $\mathbf{w}$ é um vetor de "pesos" e b é um valor constante, que denota um viés. De acordo com as escolhas de $\mathbf{w}$ e b esta equação pode ser construída de modo a fornecer valores maiores ou iguais a 1 sempre que um objeto pertence à *classe 1* (círculos) e valores menores ou iguais a -1 sempre que um objeto pertence à *classe 2* (quadrados), ou seja,

$$f(\mathbf{x}) \geq 1, \forall \mathbf{x} \in \text{classe } 1 \quad \text{e} \quad f(\mathbf{x}) \leq -1, \forall \mathbf{x} \in \text{classe } 2.$$

Um exemplo simples e de fácil compreensão é muito bem explicado por Körting (2014). Primeiramente, suponha duas características, x_1 e x_2, e três valores, conforme a Figura 3.7. O objetivo é encontrar o melhor hiperplano que irá dividir as duas classes representadas no gráfico. Uma classe está representada pelos círculos, que estão nos pontos $(1, 1)$ e $(2, 0)$, a outra classe está representada por um retângulo no ponto $(2, 3)$. A melhor linha de divisão será uma linha paralela que liga os pontos $(1, 1)$ e $(2, 3)$.

O próximo passo é definir o vetor peso $\mathbf{w}$, o qual também pode ser observado na Figura 3.7. Uma vez que $z = (2, 3) - (1, 1) = (a, 2a)$, tem-se: $\mathbf{w} = (a, 2a)$.

Desta forma, pode-se resolver o vetor peso e criar a equação do hiperplano considerando $\mathbf{w} = (a, 2a)$:

$$\begin{cases} a + 2a + b = -1, \text{usando o ponto } (1,\ 1) \\ 2a + 6a + b = 1, \text{usando o ponto } (2,\ 3). \end{cases}$$

Primeiramente, a função foi aplicada considerando o ponto $(1, 1)$. Então, foi atribuído o valor -1 para a equação, porque refere-se à classe dos círculos. Na próxima equação foi utilizado o ponto $(2, 3)$ e foi atribuído o valor 1, uma vez que está sendo considerada a classe formada pelo retângulo.

A primeira equação, $a + 2a + b = -1$, pode ser simplificada da seguinte forma:

$$3a + b = -1. \tag{3.30}$$

Isolando o valor de b na segunda equação:

$$\begin{aligned} &2a + 6a + b = 1, \\ &b = 1 - 8a. \end{aligned} \tag{3.31}$$

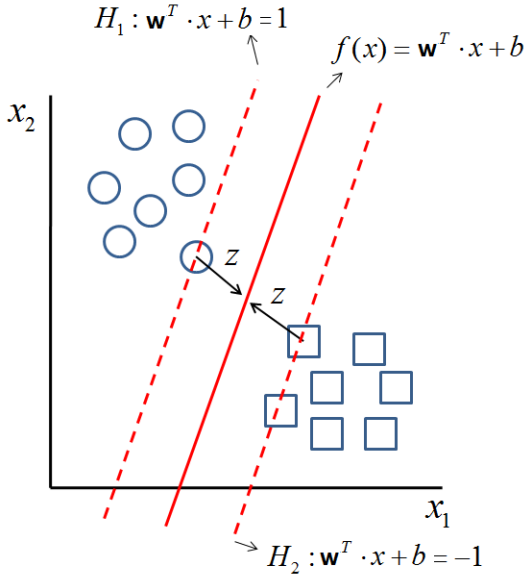


Figura 3.8: Distância z entre o hiperplano e o elemento mais próximo ao hiperplano.

Substituindo (3.31) em (3.30), obtém-se o valor de a:

$$\begin{aligned} &3a + (1 - 8a) = -1, \\ &a = \tfrac{2}{5}. \end{aligned} \tag{3.32}$$

Substituindo (3.32) em (3.31) obtém-se o valor de b:

$$b = 1 - 8a = 1 - 8\left(\frac{2}{5}\right) = -\frac{11}{5}.$$

Logo, foi obtido o vetor de pesos: $\mathbf{w} = (a, 2a) = \left(\frac{2}{5}, \frac{4}{5}\right)$, e encontrada a equação final que define o hiperplano da Figura 3.7:

$$f(\mathbf{x}) = \frac{2}{5}\mathbf{x}_1 - \frac{4}{5}\mathbf{x}_2 - \frac{11}{5},$$

simplificando,

$$f(\mathbf{x}) = \mathbf{x}_1 + 2\mathbf{x}_2 - 5,5. \tag{3.33}$$

Baseado na definição de $f(\mathbf{x})$, na Figura 3.8 indica-se a distância z dos objetos mais próximos ao hiperplano ótimo. Estas distâncias serão de, pelo menos 1 (módulo 1 - $|f(\mathbf{x})| = 1$). Da geometria sabe-se que a distância entre um ponto $\mathbf{x}$ e o hiperplano $f(\mathbf{x})$ é dada por:

$$z = \frac{|f(\mathbf{x})|}{||\mathbf{w}||}.$$

A distância total entre a margem e o hiperplano é $2z$ e, supondo que $|f(\mathbf{x})| = 1$, então $\frac{1}{||\mathbf{w}||} + \frac{1}{||\mathbf{w}||} = \frac{2}{||\mathbf{w}||}$. Ao minimizar o denominador desta última expressão, maximiza-se a margem entre as duas classes, portanto, maximiza-se a separabilidade do hiperplano $f(\mathbf{x})$.

Suponha um conjunto de n vetores de treinamento $(\mathbf{x}_i, \mathbf{y}_i)$ tais que $\mathbf{x}_i \in R^k$ e $\mathbf{y}_i \in \{-1, 1\}$. No processo de minimização de $||\mathbf{w}||$ faz-se necessário garantir a não ocorrência de pontos entre os hiperplanos H_1 e H_2. Esta condição é satisfeita ao estabelecer as seguintes desigualdades: $\mathbf{w}^T\mathbf{x}_i + b \geq +1$ quando $y_i = +1$ e $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ quando $y_i = -1$. Estas últimas expressões podem ser combinadas como:

$$y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 \quad \forall \quad i.$$

Portanto, maximiza-se a margem ao resolver-se o seguinte problema:

$$\text{Minimizar} : \frac{1}{2}||\mathbf{w}||^2, \tag{3.34}$$

com as restrições:

$$y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) - 1 \geq 0, \forall_i = 1, \ldots, n. \tag{3.35}$$

Lorena e Carvalho (2007) afirmam que as restrições são impostas de maneira a assegurar que não haja dados de treinamento entre as margens de separação das classes. Por esse motivo, a SVM obtida possui também a nomenclatura de SVM com margens rígidas.

Este problema de otimização sujeito a restrição pode ser resolvido utilizando-se multiplicadores de Lagrange. Desta forma, constrói-se e minimiza-se o Lagrangeano Primal

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i(y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) - 1), \tag{3.36}$$

em que os α_i são denominados multiplicadores de Lagrange, com $\alpha_i \geq 0$ para todo i. Os α_i´s são conhecidos como variáveis duais.

Na minimização de $L(\mathbf{w}, b, \alpha)$ com respeito a $\mathbf{w}$ e b obtém-se as derivadas parciais com respeito a estes parâmetros e iguala-se a zero as respectivas expressões, isto é:

$$\begin{array}{l} \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ \frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = -\sum_{i=1}^{n} \alpha_i y_i. \end{array} \tag{3.37}$$

Tem-se que:

$$\begin{array}{l} \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0 \\ \frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0. \end{array} \tag{3.38}$$

Resolvendo $\nabla L(\mathbf{w}, b, \alpha) = 0$ para $\mathbf{w}$ e b, um mínimo local deve satisfazer:

$$\begin{array}{l} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i. \end{array} \tag{3.39}$$

Dessa forma, $\mathbf{w}$ deve satisfazer as duas relações em (3.39). Os pesos $\mathbf{w}$ descrevem combinações lineares dos *inputs* de treinamento y_i e $\mathbf{x}_i$ e dos valores de α_i.

Com base neste mínimo local, as condições Karush-Kunh Tucker (KKT) (ver Clarke et al., 2009, Teorema pág. 275) implicam que existe um α^* tal que $\alpha_i^* = 0$ para todo $\mathbf{x}_i$ satisfazendo $y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1$. Assim, para todo $i \in \{1, 2, \ldots, n\}$, segue-se que

$$\alpha_i^* = 0 \quad \text{quando} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) > 1$$

e

$$\alpha_i^* > 0 \quad \text{quando} \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1.$$

Os vetores $\mathbf{x}_i$ para os quais $\alpha_i > 0$ (isto é, a solução tem peso estritamente positivo) são o suporte da solução e, portanto, são chamados de vetores de suporte (Clarke et al., 2009).

Ao substituir as relações (3.39) na equação (3.36) deixa-se de ter dependência entre $\mathbf{w}$ e b e obtem-se o que denomina-se por "Formulação Dual", ou seja:

$$\text{Maximizar} : \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \cdot \mathbf{x}_j), \tag{3.40}$$

com as restrições:

$$\begin{aligned} &\alpha_i \geq 0, \forall i = 1, \ldots, n, \\ &\sum_{i=1}^{n} \alpha_i y_i = 0. \end{aligned} \tag{3.41}$$

Na Formulação Dual os $\mathbf{x}_i$'s entram somente na forma de produto escalar e este fato é extremamente importante na solução de problemas de classificação que são não-linearmente separáveis.

Na Formulação Dual deriva-se (3.40) com respeito a cada α_i e iguala-se o resultado a zero. Os valores dos α_i são obtidos através de algoritmos de maximização. Uma vez determinados os valores dos α_i's obtém-se o vetor de pesos $\mathbf{w}$ do hiperplano de separação maximal através de

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i.$$

A maioria dos α_i' s e, portanto, dos $\mathbf{w}$'s será nula, e isto diminui a dimensionalidade do problema. A constante b pode ser calculada por:

$$\hat{b} = \frac{1}{2} \left(\min_{y_i=+1} \{\hat{\mathbf{w}}^T \mathbf{x}_i\} + \max_{y_i=-1} \{\hat{\mathbf{w}}^T \mathbf{x}_i\} \right).$$

Uma maneira mais simples de encontrar b é observar que as condições KKT fornecem (Clarke et al., 2009):

$$\alpha_i (y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1) = 0, \quad \forall i = 1, \ldots, n.$$

Para os vetores de suporte $\alpha_i \neq 0$, é visto que:

$$\hat{b} = y_i - \hat{\mathbf{w}}^T \mathbf{x}_i.$$

Uma vez estimados os parâmetros que definem o hiperplano ótimo, constrói-se a função de decisão, ou classificador linear SVM escrito como:

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn} \left(\sum_{\mathbf{x}_i \in SV} \hat{\alpha}_i y_i \mathbf{x}_i^T \cdot \mathbf{x} + \hat{b} \right), \tag{3.42}$$

onde

$$\text{sgn}(f(\mathbf{x})) = \begin{cases} +1 & \text{se} \quad f(\mathbf{x}) > 0 \\ -1 & \text{se} \quad f(\mathbf{x}) > 0. \end{cases} \tag{3.43}$$

Recorde que somente os SV´s terão $\alpha_i > 0$.

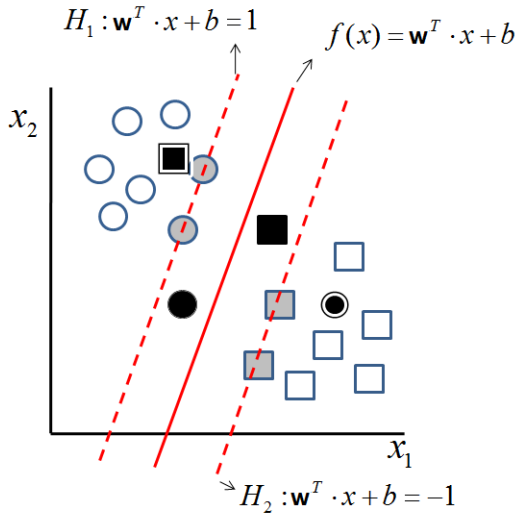


Figura 3.9: Tipos de vetores de suporte (SVs).

SVMs com Margens Suaves

Em situações reais é difícil encontrar aplicações cujos dados sejam linearmente separáveis. Isso ocorre devido a diversos fatores, tais como a presença de ruídos e outliers nos dados ou à própria natureza do problema, que pode ser não linear (Lorena e Carvalho, 2007). O caso das classes não-linearmente separáveis é tratado de forma idêntica ao caso das classes linearmente separáveis, sendo, porém, necessário introduzir uma penalização às observações que se encontram na parte incorreta do hiperplano (ver Figura 3.9). Os hiperplanos que introduzem essa penalização são conhecidos como hiperplanos de margem suave (Bonesso, 2013).

As SVMs lineares de margens rígidas podem ser estendidas para lidar com conjuntos de treinamento mais gerais para os quais alguns dados podem violar a restrição da Equação 3.35. A extensão é feita com a introdução de variáveis de folga $\xi_i, \forall i = 1, \ldots, n$. Essas variáveis "relaxam" as restrições impostas ao problema de otimização primal, que se tornam:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i^T + b) \geq 1 - \xi_i, \quad \xi \geq 0, \forall i = 1, \ldots, n. \tag{3.44}$$

A aplicação desse procedimento suaviza as margens do classificador linear, permitindo que alguns dados de treinamento permaneçam entre os hiperplanos H_1 e H_2 (Figura 3.9) e também permitindo a ocorrência de alguns erros de classificação. Por esse motivo, as SVMs obtidas neste caso também podem ser referenciadas como SVMs com margens suaves (Lorena e Carvalho, 2007).

Um erro no conjunto de treinamento é indicado por um valor de ξ_i maior que 1. Então, $\sum_i \xi_i$ é o limite superior para o número de erros de treinamento. De modo a considerar esse termo na minimização do erro nos dados de treinamento, a Equação 3.34 é reformulada como:

$$\text{Minimizar} : \frac{1}{2}||\mathbf{w}||^2 + C\left(\sum_{i=1}^{n} \xi_i\right), \tag{3.45}$$

onde C, também conhecido como "parâmetro de margem", é uma quantidade a ser escolhida pelo usuário. Quanto maior o valor do parâmetro C, maior será a penalização associada aos erros cometidos. Clake et al. (2009) relata o dilema existente entre a complexidade do modelo e a tolerância do erro controlado por C. Grandes valores de C penalizam o termo de erro, enquanto que valores pequenos de C penalizam a complexidade do modelo.

O problema de otimização gerado é quadrático com as restrições dadas na Equação 3.44. A sua solução envolve a introdução de uma função Lagrangiana, tomando suas derivadas

parciais nulas. O resultado é o *problema dual*:

$$\text{Maximizar} : \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \cdot \mathbf{x}_j), \tag{3.46}$$

com as restrições:

$$\begin{cases} 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, n, \\ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{3.47}$$

Neste caso, as condições KKT (vide texto próximo a Equação (3.39)) são equivalentes a:
$\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \text{e} \quad (C - \alpha_i)\xi_i = 0 \quad \text{e} \quad \alpha_i(y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) - 1 + \xi_i) = 0.$

Vapnik (1998) mostra que as condições KKT no caso não linearmente separável se reduzem às três condições seguintes:
1) $\alpha_i = 0 \Rightarrow y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) \geq 1$ e $\xi_i = 0$,
2) $0 < \alpha_i < C \Rightarrow y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) = 1$ e $\xi_i = 0$,
3) $\alpha_i = C \Rightarrow y_i(\mathbf{w}^T \cdot \mathbf{x}_i + b) \leq 1$ e $\xi_i \geq 0$.

Assim como nas SVMs de margens rígidas, os pontos $\mathbf{x}_i$ para os quais $\alpha_i > 0$ são denominados vetores de suporte (SVs), sendo os dados que participam da formação do hiperplano separador. No entanto, de acordo com Lorena & Carvalho (2007) e Clarke et al. (2009), existem diferentes tipos de vetores de suporte. Analisando as condições KKT de Vapnik (1998) observa-se que:

- Condição 1 do KKT: Esta condição inclui todos os SVs em branco da Figura 3.9, que são classificados corretamente e encontram-se fora das margens, uma vez que $\alpha_i = 0$ e $\xi_i = 0$;
- Condição 2 do KKT: Se $0 < \alpha_i < C$ e $\xi_i = 0$, os SVs encontram-se sobre as margens. Na Figura 3.9 tais SVs são representados pelos pontos na cor cinza;
- Condição 3 do KK: Esta última condição contém o caso em que $\alpha_i = C$ e $0 \leq \xi_i \leq 1$. Os pontos que satisfazem estas condições são classificados corretamente, porém entre as margens, e correspondem aos pontos pretos. De fato, a terceira condição também implica que os pontos que satisfazem $\alpha_i = C$ e $\xi_i > 1$ são classificados incorretamente e correspondem a erros. Na Figura 3.9, estes são pontos pretos com bordas extras.

A única diferença entre o SVM de margem suave e o SVM de margem rígida é que α_i não pode exceder C. A função de decisão é a mesma para o caso de margem rígida (3.42), porém neste caso, as variáveis α_i são determinadas pela solução da Expressão (3.46) com as restrições (3.47).

SVMs Não Lineares

As SVMs não lineares são utilizadas para os casos em que não é possível dividir satisfatoriamente os dados de treinamento por um hiperplano. Para lidar com tais situações, faz-se necessário a generalização das SMVs lineares.

As SVMs lidam com problemas não lineares mapeando o conjunto de treinamento de seu espaço original, referenciado como espaço de entradas, para um novo espaço de maior dimensão, denominado *espaço de características* (feature space) (Lorena e Carvalho, 2007).

Seja o conjunto de entrada S representado pelos pares $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, com $y_i, i = 1, 2, \ldots, n$ o rótulo de cada padrão i, o conjunto de dados de treinamento. O espaço de características é um espaço de mais alta dimensionalidade no qual será mapeado o conjunto de

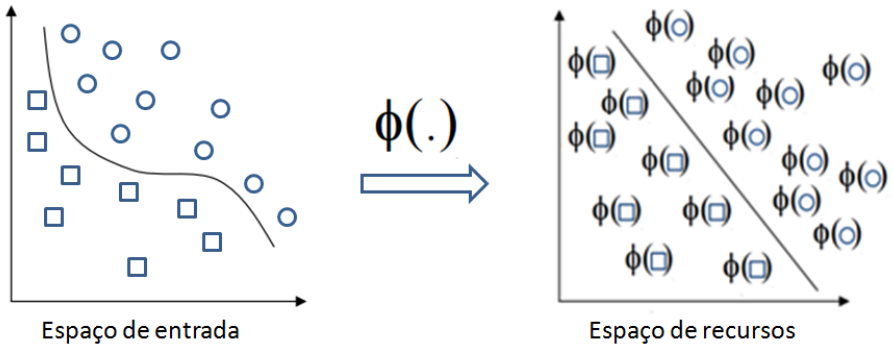


Figura 3.10: Transformação do conjunto de dados no espaço de entrada para o espaço de característica. Adaptado de Koerich (2012).

entrada S, por meio de uma função Φ, a fim de obter um novo conjunto de dados S' linearmente separável, representado por $\{(\Phi(\mathbf{x}_1), y_1), \ldots, (\Phi(\mathbf{x}_n), y_n)\}$ (Gonçalves, 2012).

Dadas as funções reais $\Phi_1, \ldots, \Phi_N$ no domínio de entrada, a escolha apropriada de Φ faz com que o conjunto de treinamento mapeado possa ser separado por uma SVM linear. As SVMs lineares anteriormente apresentadas podem então ser utilizadas sobre o conjunto de treinamento mapeado nesse espaço (este fato é ilustrado na Figura 3.10). Logo, os conceitos apresentados anteriormente podem ser estendidos para o caso não linear, através da definição de uma função de mapeamento Φ adequada.

Desta forma, para construir o hiperplano ótimo, é necessário encontrar a solução $\hat{\alpha}$ do seguinte problema:

$$\text{Maximizar} : \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i)^T \cdot \Phi(\mathbf{x}_j), \tag{3.48}$$

com as restrições:

$$\begin{cases} 0 \leq \alpha_i \leq C, \forall i = 1, \ldots, n \\ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{3.49}$$

Através da determinação dos coeficientes de Lagrange, α_i, pode-se deduzir $\mathbf{w}$ e encontrar o valor de b:

$$\hat{b} = \frac{1}{2} \left(\min_{y_i=+1} \{\hat{\mathbf{w}}^T \Phi \mathbf{x}_i\} + \max_{y_i=-1} \{\hat{\mathbf{w}}^T \Phi \mathbf{x}_i\} \right). \tag{3.50}$$

A ideia central na classificação SVM para problemas não-lineares reside na substituição do produto interno euclidiano $\mathbf{x}_i^T \cdot \mathbf{x}$ em (3.42). A expressão para o classificador SVM, com $\Phi(\mathbf{x}_i)^T \cdot \Phi(\mathbf{x})$, é dada por:

$$g(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn} \left(\sum_{x_i \in SV} \hat{\alpha}_i y_i \Phi(\mathbf{x}_i)^T \cdot \Phi(\mathbf{x}) + \hat{b} \right). \tag{3.51}$$

onde

$$\text{sgn}(f(\mathbf{x})) = \begin{cases} +1 & \text{se} \quad f(\mathbf{x}) > 0 \\ -1 & \text{se} \quad f(\mathbf{x}) > 0. \end{cases} \tag{3.52}$$

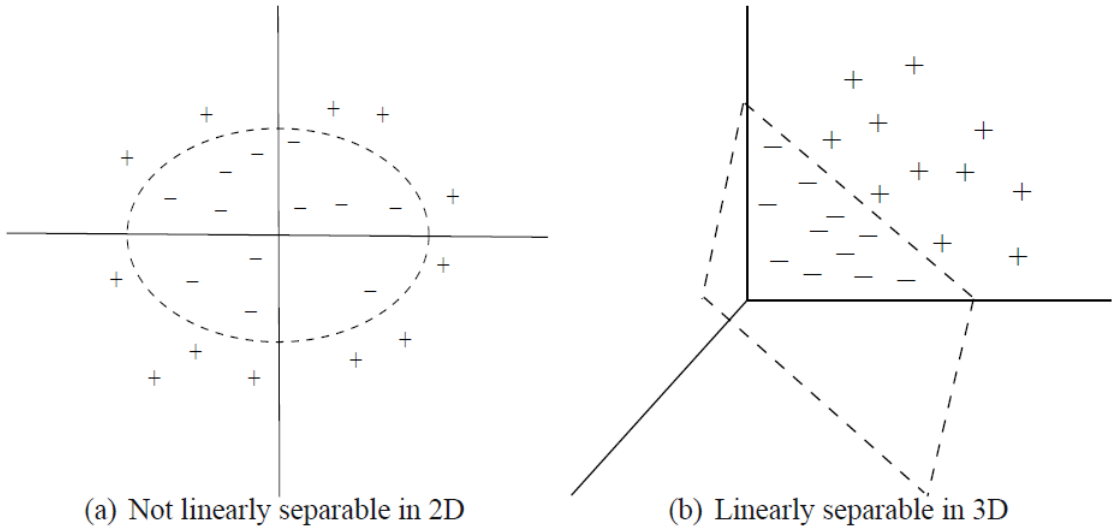


Figura 3.11: A figura (a) mostra os dados originais no plano. Eles não podem ser separados linearmente. No entanto, uma transformação pode ser usada de modo que uma representação de maior dimensão das vantagens e desvios se torne linearmente separável. Fonte: Clarke et al. (2009).

Segundo Clarke et al. (2009), as considerações de KKT, apresentadas para SVM linear, se mantêm para o caso não linear. O produto interno euclidiano é calculado no espaço de entrada do problema primal e a generalização (3.51) usa uma transformação Φ que converte um vetor de entrada $\mathbf{x}$ em um ponto no espaço de características de dimensão superior. O uso de Φ permite a inclusão de mais recursos nos vetores, tornando-os mais fáceis de separar com os hiperplanos.

A Figura 3.11 mostra o resultado de uma transformação Φ adequada. Desta forma, sejam $\mathbf{x}^T = (x_1, x_2)$ e os vetores de recursos $\mathbf{z}^T = (z_1, z_2, z_3)$ no espaço de recurso Euclidiano $\mathbb{R}^3$. Define-se Φ:$\mathbb{R}^2 \longrightarrow \mathbb{R}^3$ por (Clarke et al., 2009):

$$\Phi(\mathbf{x}) = \Phi(x_1, x_2) = (x_1^2, \sqrt{2}x_1x_2, x_2^2) = \mathbf{z}^T.$$

Com isso, um difícil problema de classificação não-linear em 2D é convertido em uma tarefa de classificação linear em 3D. Em geral, $\Phi : \mathcal{X} \longrightarrow \mathcal{F}$ transforma um espaço de entrada $\mathcal{X}$ em um espaço de recurso $\mathcal{F}$ de uma dimensão muito maior, de modo que a inclusão de mais recursos torna os dados em $\mathcal{F}$ linearmente separáveis (Clarke et al., 2009).

Os autores ainda afirmam que o problema central na implementação desta estratégia é saber se alguma transformação Φ fará com que os dados sejam separáveis no espaço de recursos. Na próxima seção será apresentada uma maneira sistemática de determinar a transformação correta para "linearizar" uma determinada tarefa de classificação não-linear.

Linearização por Kernel

De acordo com Clarke et al. (2009), para se obter uma solução linear para o problema de classificação, a imagem de Φ deve ser de maior dimensão do que suas entradas. Caso contrário, a transformação é apenas a imagem contínua de $\mathbb{R}^p$ sendo improvável que seja mais linearmente separável do que suas entradas. Por outro lado, se Φ constrói um vetor de característica de uma dimensão muito maior do que o vetor de entrada poderá haver um problema de dimensionalidade.

Na verdade, esses fatos costumam ser ignorados pelo truque do kernel. Um Kernel K é uma função que recebe dois pontos $\mathbf{x}$ e $\mathbf{y}$ do espaço de entradas e computa o produto escalar

$\Phi(\mathbf{x})^T \cdot \Phi(\mathbf{y})$ no espaço de características, como descrito em:

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^T \cdot \Phi(\mathbf{y}). \tag{3.53}$$

Clarke et al. (2009) mostram que, dados $\Phi(\mathbf{x})$ e $\Phi(\mathbf{y})$, dois vetores de características gerados por $\mathbf{x}$ e $\mathbf{y}$, o produto interno $\Phi(\mathbf{x})^T \cdot \Phi(\mathbf{y})$ no espaço de características é:

$$\Phi(\mathbf{x})^T \cdot \Phi(\mathbf{y}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2)^T = (x_1y_1 + x_2y_2)^2 = (\mathbf{x}^T\mathbf{y})^2 = K(\mathbf{x}, \mathbf{y}). \tag{3.54}$$

A computação direta dos produtos internos do espaço de características sem manipular explicitamente os próprios vetores do espaço de recursos é conhecida como o truque do kernel.

A função Kernel não altera muito o problema de SVM, pelo menos não explicitamente. Com a introdução da função Kernel, para encontrar os coeficientes α_i é necessário agora resolver o seguinte problema (Santos, 2002):

$$\text{Maximizar} : \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j), \tag{3.55}$$

Com as restrições:

$$0 \leq \alpha_i \leq C, i = 1, \ldots, n$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0.$$

Desta forma, pode-se dizer que a Kernelização é a abordagem padrão para lidar com a não-linearidade da decisão na classificação de SVM. A versão do kernel do classificador SVM é dada pela função de decisão:

$$f(\mathbf{x}) = \text{sgn}(f(\mathbf{x})) = \text{sgn}\left(\sum_{i=1}^{n} y_i \hat{\alpha}_i K(\mathbf{x}_i, \mathbf{x}) + \hat{b} \right). \tag{3.56}$$

Essas expressões são iguais às anteriores, exceto que $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i)^T \cdot \Phi(\mathbf{x}_j)$ substitui o produto interno $\mathbf{x}_i^T \cdot \mathbf{x}_j$ em (3.42).

O uso de diferentes funções Kernel $K(x_i, x)$ permite a construção de máquinas de aprendizagem com diferentes tipos de superfícies de decisão não-linear no espaço de entrada (Santos, 2002). Dentre as funções kernel mais usadas destacam-se:

- Polinomial de grau d:
- Função de Base Radial (RBF) Gaussiana;
- Função de Base Radial (RBF) Laplaciana;
- ANOVA Base Radial;
- Sigmoidal.

3.3 Técnicas de Conjuntos Aleatórios

3.3.1 Árvores

O conjunto de regras para a construção de modelos de predição a partir de dados podem ser resumidos em uma árvore. Os métodos baseados em árvore são simples e de fácil compreensão, além de serem um dos poucos modelos interpretáveis. Esse método de aprendizagem está entre os mais populares dos algoritmos de inferência indutivos e tem sido amplamente aplicado, por exemplo, em análise de crédito, investigação biomédica e genética, reconhecimento de fala, problemas em equipamentos mecânicos e elétricos, marketing, ciência política e outras ciências aplicadas.

As árvores de decisão também podem ser representadas como conjuntos de regras SE-ENTÃO (IF-THEN) para melhorar a legibilidade humana (Mitchell, 1997). Os algoritmos mais famosos de aprendizagem de árvore de decisão são: ID3, ASSISTANT, CART, C4.5 e J48. Atualmente, também existem diversos pacotes estatísticos, tais como: Splus, Statistica, SPSS, e Microsoft SQL Server, que incorporam funções que implementam árvores para problemas de classificação e regressão.

Os modelos em árvores são denominados árvores de decisão no caso de problemas de classificação e árvores de regressão no caso de problemas de regressão. Porém, tanto em árvores de decisão como em árvores de regressão, a interpretação dos modelos e dos algoritmos das árvores são muito semelhantes. Uma árvore de regressão é idêntica a uma árvore de decisão porque também é formada por um conjunto de nós de decisão, ou perguntas. A diferença é que o resultado, ao invés de ser uma categoria, é um escalar.

As árvores de classificação são projetadas para variáveis dependentes que tomam um número finito de valores não ordenados, com erro de previsão medido em termos de custo da classificação errada. As árvores de regressão são projetadas para variáveis dependentes que tomam valores discretos, contínuos ou ordenados, com erro de previsão tipicamente medido pela diferença quadrática entre os valores observados e preditos (Loh, 2011). As árvores de classificação e de regressão serão discutidas com maiores detalhes. Primeiramente, será realizada uma visão geral sobre árvores de decisão. Para isso, é importante ter em mente algumas definições:

- Exemplo: caso, dado ou registro;
- Atributo: descreve uma característica ou aspecto de um exemplo. Além disso, são variáveis observáveis e independentes que costumam ser classificadas em contínuas, categóricas ordinais e categóricas não-ordinais (Zuben, 2010);
- Classe: é uma variável dependente cujo valor é definido a partir das variáveis independentes, ou seja, a partir dos atributos;
- Instâncias: são representadas através de pares atributo-valor. Instâncias são descritas por um conjunto fixo de atributos (por exemplo: Temperatura) e seus respectivos valores (por exemplo: Quente) (Guarda, 2013).
- Nós de Decisão: são os nós internos, usados para se caminhar na árvore até atingir as folhas. Estes nós possuem condicionantes que fazem com que um caminho seja seguido, conforme os dados vão sendo aplicados. O nó raiz, que é o primeiro nó da árvore, é considerado um nó de decisão;

Tabela 3.1: Exemplo de treinos baseado em Mitchell (1997)

Dia	Aspecto	Temperatura	Umidade	Vento	Joga
D1	Sol	Quente	Alta	Fraco	Não
D2	Sol	Quente	Alta	Forte	Não
D3	Nuvens	Quente	Alta	Fraco	Sim
D4	Chuva	Ameno	Alta	Fraco	Sim
D5	Chuva	Fresco	Normal	Fraco	Sim
D6	Chuva	Fresco	Normal	Forte	Não
D7	Nuvens	Fresco	Normal	Forte	Sim
D8	Sol	Ameno	Alta	Fraco	Não
D9	Sol	Fresco	Normal	Fraco	Sim
D10	Chuva	Ameno	Normal	Fraco	Sim
D11	Sol	Ameno	Normal	Forte	Sim
D12	Nuvens	Ameno	Alta	Forte	Sim
D13	Nuvens	Quente	Normal	Fraco	Sim
D14	Chuva	Ameno	Alta	Forte	Não

- Nós Folha: são os nós externos ou nós objetivo. No caminhamento da árvore através dos nós de decisão, os nós folhas representam os resultados da predição. Desta forma, quando é realizada a divisão dos nós internos, ao atingir um nó folha, a árvore não irá mais ramificar.

2.3.1.1 Representação das árvores de decisão

A fim de proporcionar melhor compreensão sobre as árvores de decisão, a Figura 3.12 apresenta um exemplo citado por Mitchell (1997) onde foi criada uma árvore para auxiliar na decisão de jogar ou não tênis. Para isso, deve-se levar em conta certos parâmetros do ambiente, como o Aspecto do céu, a Temperatura, a Umidade e o Vento. Conforme a Tabela 3.1, cada um destes atributos tem vários valores. Por exemplo, a temperatura pode estar Amena, Fresca ou Quente. A decisão Sim (ir jogar ténis) ou Não (não ir jogar ténis) é o resultado da classificação.

Observa-se que a árvore da Figura 3.12 possui cinco folhas, logo haverá cinco regras de decisão para essa árvore. Ao aplicar a regra SE-ENTÃO lê-se a árvore da seguinte forma:

1) Se o “aspecto” do tempo for um dia ensolarado, ou seja, “sol” e a variável “umidade” é “elevada”, então, “não” jogue tênis.

2) Ou então, se o “aspecto” do tempo for “sol” e a variável “umidade” for “normal”, então, “sim”, jogue tênis.

3) Ou ainda, se o “aspecto” do tempo for “nuvens”, então, “sim”, jogue tênis.

4) Ou ainda, se o “aspecto” do tempo for formado por “chuva” e o “vento” for “forte”, então, “não” jogue tênis.

5) Ou então, se o “aspecto” do tempo for formado por “chuva” e o “vento” for “fraco”, então, “sim” jogue tênis.

Em outra palavras, tem-se:

1) Se (aspecto=sol) e (umidade=alta) então (joga=não);

2) Se (aspecto=sol) e (umidade=normal) então (joga=sim);

3) Se (aspecto=nuvem) então (joga=sim);

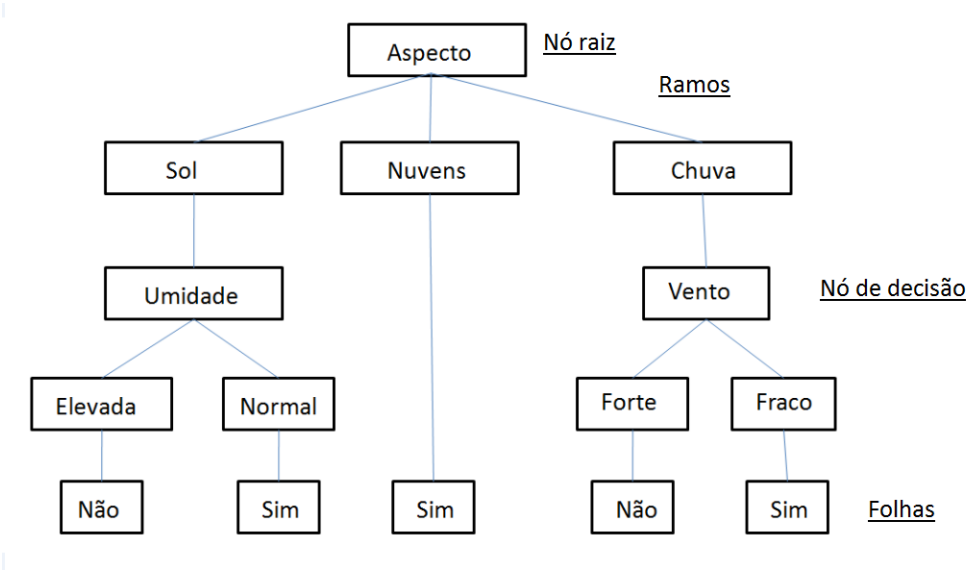


Figura 3.12: Árvore de decisão para jogar tênis baseda no exemplo de Mitchell (1997).

4) Se (aspecto=chuva) e (vento=forte) então (joga=não);

5) Se (aspecto=chuva) e (vento=fraco) então (joga=sim).

As árvores de decisão representam a conjunção e a disjunção de atributos. Cada caminho da raiz da árvore para uma folha corresponde a uma conjunção de testes de atributo e a própria árvore a uma disjunção dessas conjunções. Por exemplo, a árvore de decisão mostrada, corresponde à expressão (Mitchell, 1997):
(Aspecto = Sol $\wedge$ Umidade = Normal)
$\vee$ (Aspecto = Nuvens)
$\vee$ (Aspecto = Chuva $\wedge$ Vento = Fraco)

Um outro exemplo seria considerar a instância (Aspecto = Sol, Temperatura = Quente, Umidade = Alta, Vento = Forte). A instância seria classificada no ramo mais à esquerda dessa árvore de decisão e seria, portanto, classificada como uma instância negativa. Isto é, a árvore iria prever que a decisão é não ir jogar Tennis.

2.3.1.2 Árvores de Regressão

Para desenvolver uma árvore de regressão suponha que um conjunto de dados seja constituído por p inputs e uma resposta, para cada uma das N observações. Segundo James et al. (2013), há duas etapas para a construção de uma ávore de regressão:

1) Dividir o espaço preditor - ou seja, o conjunto dos possíveis valores para $X_1, X_2, \ldots, X_p$ em J regiões distintas e não sobrepostas, $R_1, R_2, \ldots, R_J$;

2) Para cada observação que ocorre na região R_j é feita a mesma predição, que é a média dos valores resposta para as observações de treinamento em R_j.

James et al. (2013) supõem que na etapa 1 obtém-se duas regiões R_1 e R_2 e que a média de resposta das observações de treinamento na primeira região é 10, enquanto que a média de resposta das observações de treinamento na segunda região é 20. Então, para uma dada observação $X = x$, se $x \in R_1$ será previsto o valor 10 e se $x \in R_2$ será previsto o valor 20.

Passamos agora à questão de como elaborar a etapa 1 acima. Na teoria, as regiões podem ter qualquer forma. No entanto, o espaço preditor é dividido em retângulos ou caixas, devido à simplicidade e facilidade de interpretação do modelo preditivo resultante. O objetivo

é encontrar as caixas $R_1, \ldots, R_J$ que minimizam a soma dos quadrados, dada por (James et al., 2013):

$$\sum_{j=1}^{J} \sum_{i:x_i \in R_j} (y_i - \hat{y}_{R_j})^2, \tag{3.57}$$

em que $\hat{y}_{R_j}$ representa a resposta média para as observações de treinamento dentro da j-ésima caixa.

Previamente, é computacionalmente inviável considerar todas as partições possíveis do espaço de características em J caixas. Por isso, procedemos com uma abordagem gulosa *de cima para baixo*, ou *top-down*, que é conhecida como divisão binária recursiva. Top-down porque começa no topo da árvore e gulosa porque a cada passo do processo de construção da árvore, a melhor divisão é feita em um passo particular, sem levar em consideração o passo à frente. Desta forma, começando com todos os dados, considere uma variável de divisão j e pontos de divisão s, e defina o par de "meios-planos"(James et al., 2013):

$$R_1(j, s) = \{X | X_j < s\} \quad e \quad R_2(j, s) = \{X | X_j \geq s\}, \tag{3.58}$$

em que $\{X|X_j < s\}$ significa a região do espaço preditor na qual X_j assume um valor menor que s.

Deve-se buscar pela variável j e pelo ponto s que miniminizam a equação (James et al., 2013):

$$\sum_{x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2, \tag{3.59}$$

em que $\hat{y}_{R_1}$ é a resposta média para as observações de treinamento em $R_1(j, s)$ e $\hat{y}_{R_2}$ é a resposta média para as observações de treinamento em $R_2(j, s)$

Encontrar os valores de j e s que minimizam a expressão pode ser feito de maneira rápida, especialmente quando o número de características p não é muito grande. Uma vez encontrada a melhor divisão, os dados são divididos nas duas regiões resultantes e o processo de divisão é repetido em cada uma das duas regiões. Então, este processo descrito é repetido em todas as regiões resultantes buscando o melhor preditor e o melhor ponto de corte, a fim de dividir os dados de forma a minimizar a soma dos quadrados dentro de cada uma das regiões resultantes. Esse processo continua até que um critério seja atingido.

Criadas as regiões $R_1, R_2, \ldots, R_J$, a variável resposta é prevista para uma determinada observação de teste usando a média das observações de treinamento da região que a observação de teste pertence (James et al., 2013).

O processo descrito pode gerar boas previsões no conjunto de treinamento, mas é capaz de sobrecarregar os dados, ocasionando um fraco desempenho para o conjunto de teste. Isso ocorre quando a árvore resultante é muito complexa. Desta forma, uma árvore menor e com menos divisões (menos regiões $R_1, R_2, \ldots, R_J$) pode gerar uma menor variância e melhor interpretação ao custo de um pequeno viés.

Uma possível alternativa ao processo acima seria dividir os nós da árvore apenas se a diminuição da soma dos quadrados devido a cada divisão exceder um certo limiar. Esta estratégia resultará em árvores menores, mas é muito míope, uma vez que uma separação aparentemente desnecessária no início da árvore poderia ser seguida por uma divisão muito boa, isto é, uma divisão que leva a uma grande redução na soma dos quadrados (James et al, 2013; Hastie et al., 2001).

Portanto uma melhor estratégia é crescer muito uma árvore T_0, interrompendo o processo de divisão somente quando algum número mínimo de nós é alcançado. Essa árvore grande é

podada utilizando a *poda de complexidade de custos*, também denominada de *poda de ligação mais fraca*.

O objetivo é selecionar uma subárvore que leva a uma baixa taxa de erro de testagem. O erro pode ser estimado usando validação cruzada para cada possível subárvore. No entanto, estimar o erro de validação cruzada para cada subárvore seria muito custoso, uma vez que existe um número extremamente grande destas. A poda de complexidade de custos fornece uma maneira de selecionar um pequeno conjunto dessas subárvores, porque ao invés de considerar cada uma, considera-se uma sequência de árvores indexada por um parâmetro de ajuste não negativo α.

Uma sub-árvore $T \subset T_0$ é definida como sendo qualquer árvore que pode ser obtida pela poda de T_0. Os nós terminais são indexados em m, com o nó m representando a região R_m (o retângulo ou subconjunto do espaço preditor). O número de nós terminais é denotado por $|T|$. Considerando uma sequência de árvores indexadas por um parâmetro de ajuste não negativo α, a cada valor de α corresponde uma subárvore $T \subset T_0$, tal que

$$\sum_{m=1}^{|T|} \sum_{x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \tag{3.60}$$

é tão pequeno quanto possível. A variável $\hat{y}_{R_m}$ é a resposta prevista associada ao R_m, isto é, a média das observações de treinamento em R_m. O parâmetro de ajuste α controla o dilema entre o tamanho da árvore e a qualidade de ajuste para os dados. O

James et al. (2013) ajustam uma árvore de regressão no software `R` para o conjunto de dados `Boston`. Primeiramente, criaram um conjunto de dados de treinamento e então, ajustaram uma árvore de regressão para esses dados.

```
library(MASS)
library(tree)
set.seed(1)
train=sample(1:nrow(Boston),nrow(Boston)/2)
tree.boston=tree(medv~.,Boston,subset=train)
summary(tree.boston)
##
##Regression tree:
##tree(formula = medv ~ ., data = Boston, subset = train)
##Variables actually used in tree construction:
##[1] "lstat" "rm"    "dis"
##Number of terminal nodes:  8
##Residual mean deviance:  12.65 = 3099 / 245
##Distribution of residuals:
##     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
##-14.10000  -2.04200  -0.05357   0.00000   1.96000  12.60000
win.graph()
plot(tree.boston)
text(tree.boston, pretty=0)
```

A função `summary(tree.boston)` lista as variáveis que são usadas como nós internos na árvore, o número de nós terminais e a taxa de erro. Desta forma, podem ser verificadas na Figura 3.13 e na função `summary(tree.boston)` que foram utilizadas três variáveis para a construção da árvore: `lstat`, `rm` e `dis`. Além disso, a árvore possui 8 nós folha (nós terminais).

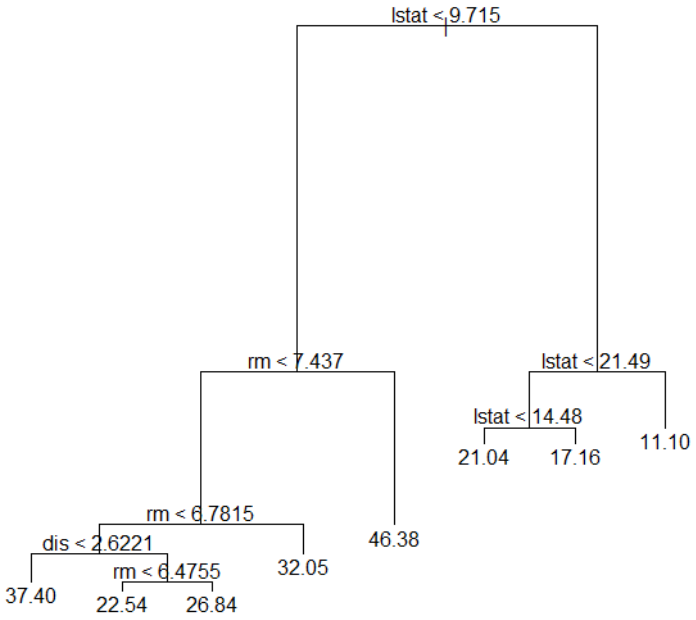


Figura 3.13: Árvore de regressão baseada em James et al. (2013).

De acordo com James et al. (2013), a árvore indica que os valores mais baixos de `lstat`, variável que mede a porcentagem de indivíduos com baixo status socioeconômico, correspondem a casas mais caras. A árvore prevê um preço médio de $\$46,4$ para casas maiores em subúrbios em que os residentes tem alto status socioeconômico (`rm` $>=$ `7.437` e `lstat` $<$ `9.715`).

Agora, foi utilizada a função `cv.tree()` para verificar a melhoria do desempenho da poda da árvore. Neste caso, a árvore mais complexa é selecionada por validação cruzada, para determinar o nível ótimo de complexidade da árvore.

```
cv.boston=cv.tree(tree.boston)
plot(cv.boston$size,cv.boston$dev,type='b')
```

Para podar a árvore pode ser utilizada a função `prune.tree()`. A Figura 3.14 apresenta a árvore formada pela função:

```
prune.boston=prune.tree(tree.boston,best=5)
plot(prune.boston)
text(prune.boston,pretty=0)
```

De acordo com os resultados da validação cruzada, James et al. (2013) escolheram utilizar a árvore não podada para fazer previsões sobre o conjunto de teste.

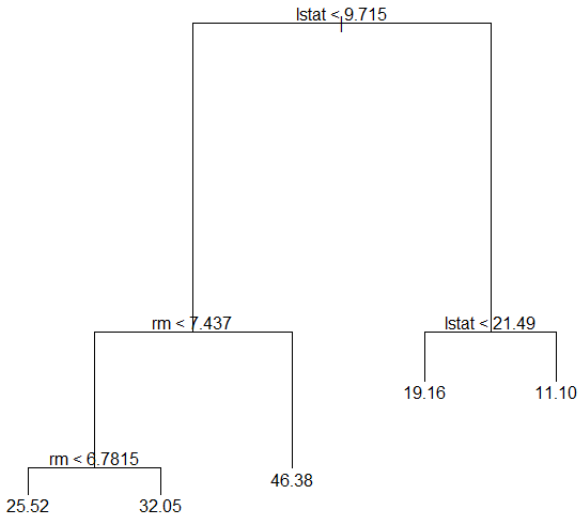


Figura 3.14: Árvore de regressão podada através da função `prune.tree()`. FONTE: James et al. (2013).

```
yhat=predict(tree.boston,newdata=Boston[-train,])
boston.test=Boston[-train,"medv"]
plot(yhat,boston.test)
abline(0,1)
mean((yhat-boston.test)^2)
##[1] 25.04559
```

Para o conjunto de teste, o MSE associado à árvore de regressão é 25.05. A raiz quadrada do MSE é em torno de 5.005, o que indica que esse modelo leva a predições de teste em torno de $5,005$ do verdadeiro valor mediano de casa para o subúrbio.

2.3.1.3 Árvores de Classificação

Conforme já foi mencionado, um nó pode ser denominado nó decisão ou nó folha. Um nó de decisão pode se dividir em dois nós (uma divisão binária). Essa divisão binária é determinada por uma condição Booleana que pode ser satisfeita ("sim") ou não satisfeita ("não") pelo valor observado dessa variável. Izenman (2008) cita um exemplo de um particionamento recursivo envolvendo duas variáveis de entrada X_1 e X_2, cuja árvore é representada na Figura 3.15. A Figura 3.16 apresenta a partição resultante em 5 regiões.

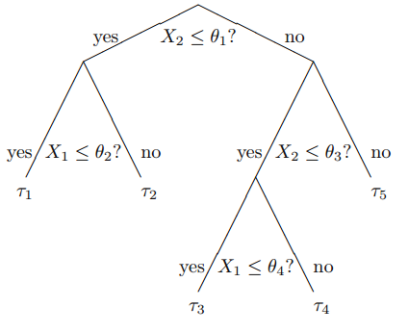


Figura 3.15: Árvore de decisão com cinco nós terminais, $\tau_1 - \tau_5$ e quatro divisões. Fonte: Izenman (2008).

As etapas possíveis desta árvore são as seguintes:

(1) $X_2 \leq \theta_1$? Caso a resposta seja "sim", siga o ramo esquerdo; caso contrário, siga o ramo direito.

(2) Se a resposta a (1) for "sim", então faz-se a seguinte pergunta: $X1 \leq \theta_2$? A resposta "sim" produz o nó folha τ_1 com a região correspondente $R_1 = X_1 \leq \theta_2, X_2 \leq \theta_1$; a resposta "não" produz o nó folha τ_2 com a região correspondente $R_2 = X_1 > \theta_2, X_2 \leq \theta_1$.

(3) Se a resposta a (1) for "não", faz-se a próxima pergunta: $X_2 \leq \theta_3$? Se a resposta é "sim", então pergunta-se: $X_1 \leq \theta_4$? Se a resposta for "sim" deve-se produzir o nó folha τ_3 com a região correspondente $R_3 = X_1 \leq \theta_4, \theta_1 < X_2 \leq \theta_3$; caso contrário, deve-se seguir o ramo direito para o nó folha τ_4 com a região correspondente $R_4 = X_1 > \theta_4, \theta_1 < X_2 \leq \theta_3$.

(4) Se a resposta a (3) for "não", chegou-se ao nó folha τ_5 com a região correspondente $R_5 = X_2 > \theta_3$.
Assume-se que $\theta_2 < \theta_4$ e $\theta_1 < \theta_3$.

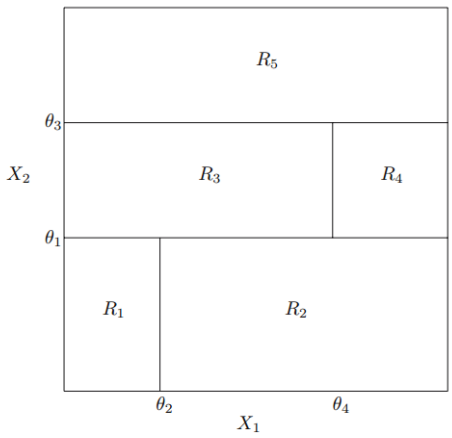


Figura 3.16: Exemplo de particionamento recursivo de cinco regiões em $\mathbb{R}^2$, $R_1 - R_5$, correspondendo aos cinco nós terminais. Fonte: Izenman (2008).

Na árvore de regressão, a resposta prevista para uma determinada observação é dada pela média das observações de treinamento que pertencem ao mesmo nó folha. Em contraste, na árvore de classificação, pode-se prever que cada observação pertence à classe de observações de treinamento mais frequente na região a que pertence.

O algoritmo de indução deve escolher qual o atributo preditivo que será utilizado em cada nó da árvore. Essa escolha pode ser baseada em diferentes critérios, tais como impureza,

distância ou dependência. A maior parte dos algoritmos tenta dividir os dados de um nó de forma a minimizar o grau de impureza dos nós filhos.

Na classificação, a soma dos quadrados não pode ser utilizada como critério para as divisões binárias. Uma alternativa a ser utilizada é a taxa de erro de classificação. O erro de classificação é uma das medidas de impureza comumente utilizadas em árvores de decisão binária.

Em um nó m representando uma região R_m com N_m observações, tem-se

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k)$$

como a proporção de observações da classe k no nó m. Classifica-se as observações no nó m para a classe $k(m) = \arg\max_k \hat{p}_{mk}$, a classe majoritária no nó. Desta forma, a taxa de erro de classificação é representado pela expressão:

$$E = \frac{1}{N_m} \sum\nolimits_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk(m)}, \tag{3.61}$$

em que $\hat{p}_{mk(m)}$ representa a proporção de observações de treinamento na m-ésima região que são da k-ésima classe. Hastie et al. (2001) afirmam que o erro de classificação não é suficientemente sensível para o cultivo de árvores e, na prática, são preferíveis duas outras medidas.

O Índice de Gini é definido por:

$$G = \sum\nolimits_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'} = \sum\nolimits_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}). \tag{3.62}$$

O índice de Gini mede o grau de impureza dos dados. Logo, pode ser usado para medir a pureza de um nó. Quando este índice é igual a zero, o nó é puro. Entretanto, quando se aproxima do valor um, o nó é impuro.

A entropia também é utilizada como medida de impureza e é dada pela expressão,

$$S = -\sum\nolimits_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}. \tag{3.63}$$

A entropia caracteriza a falta de homogeneidade dos dados de entrada em relação à sua classificação. Desta forma, a entropia é máxima (igual a 1) quando o conjunto de dados é heterogêneo.

Como exemplo de entropia, suponha o exemplo citado por Mitchell (1997) no qual foi criada uma árvore para auxiliar na decisão de jogar ou não tênis, com um total de 14 exemplos. Destes, pode-se dizer que 9 exemplos são positivos, uma vez que obtiveram a resposta para a classificação "sim" (ir jogar tênis) e, 5 são negativos devido à decisão "Não" (não ir jogar tênis). Pode-se então ser adotada a notação $[9+, 5-]$ para resumir tal amostra de dados. Então, a Entropia Cruzada pode ser calculada da seguinte forma:

$$S([9+, 5-]) = -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) = 0,940$$

Como existem números desiguais de exemplos positivos e negativos, a entropia está entre 0 e 1.

Na Figura 3.17 a função de entropia, a taxa de erro de classificação e o índice de Gini são representados graficamente para o caso de duas classes. Qualquer uma dessas três abordagens pode ser utilizada na poda da árvore. A taxa de erro de classificação é preferível se a meta for a precisão da predição da árvore final. A Entopia Cruzada e o Índice de Gini são diferenciáveis, portanto, são mais passíveis de otimização numérica. Além disso, devem ser usados ao crescer a árvore e são mais sensíveis às mudanças nas probabilidades do nó do que a taxa de erros de classificação.

A partir dos fatos apresentados até então, pode-se concluir que a construção de uma árvore de decisão tem como objetivos principais: diminuir a entropia, ou seja, a aleatoriedade da variável que define as classes, ser consistente com o conjunto de dados e possuir o menor número de nós possível.

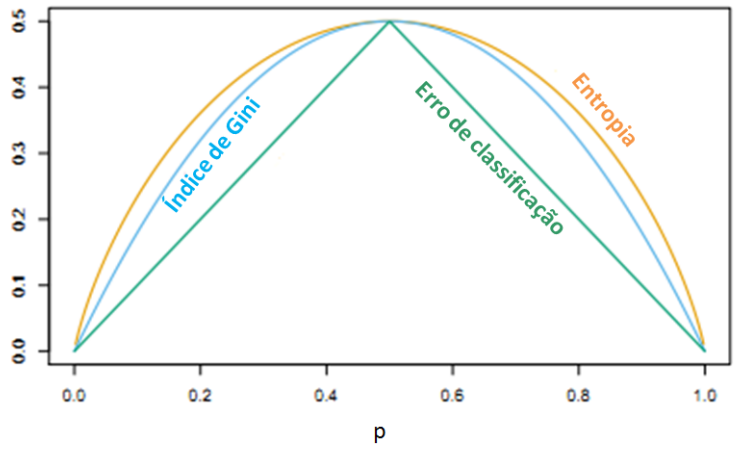


Figura 3.17: Comparação das medidas de impureza para a classificação de duas classes. A função de entropia é a curva laranja, o índice de Gini é a curva azul e a estimativa da taxa de erro de classificação é a curva verde. Fonte: Hastie et al. (2001).

3.3.2 Florestas Aleatórias

A técnica de **Florestas Aleatórias** (FA) trata-se de uma abordagem poderosa para a exploração de dados, análise de dados e modelagem preditiva. Esta técnica foi desenvolvida por Leo Breiman (criador do CART) na Universidade da Califórnia, Berkeley (Breiman, 2001).

De acordo com o site **Data Science Central** (`http://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm`), árvores de decisão individuais geralmente apresentam alta variabilidade ou grande viés. As FA's tentam mitigar tais problemas ao encontrar um equilíbrio natural entre estes dois extremos. Considerando que as FA's têm poucos parâmetros de sintonia (*tunning*) e podem ser usadas simplesmente com configurações padrão (*default*) de parâmetros, elas representam uma ferramenta simples de usar quando não se tem um modelo ou para produzir um modelo razoável rápido e eficientemente. Segundo o site os maiores benefícios das FA's são:

- Acurácia;
- Eficiência - mesmo para grandes bases de dados;
- Capacidade de lidar com milhares de variáveis de entrada;
- Proporciona estimativas de quais variáveis são importantes em problemas de classificação;
- Gera uma estimativa não viciada do erro de generalização à medida que a floresta é construída;
- Proporciona métodos efetivos para a estimação de dados faltantes;
- Mantém a acurácia mesmo na presença de dados faltantes;
- Proporciona métodos que funcionam bem para dados não balanceados;
- Gera florestas que podem ser salvas para análises futuras;
- Calcula as proximidades entre pares de casos que podem ser utilizadas para a formação de conglomerados, localização de outliers, ou construção de gráficos interessantes dos dados;
- As características acima podem ser estendidas para o caso de análises não supervisionadas, permitindo a formação não supervisionadas de conglomerados, gráficos e detecção de outliers;
- Oferece um método experimental para a detecção de interação entre as variáveis.

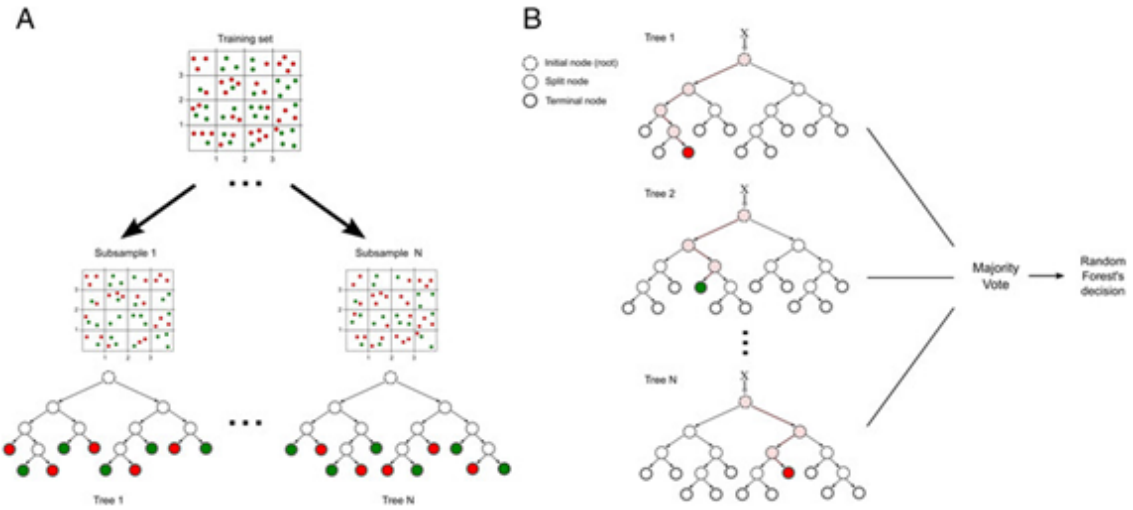


Figura 3.18: O *voto da maioria* é obtido a partir dos especialistas (árvores de classificação). Fonte: `https://dimensionless.in/introduction-to-random-forest/`.

Dentre as desvantagens das Florestas Aleatórias, incluem-se:

• Não são facilmente interpretáveis;

• Não são um algoritmo de última geração;

• Um grande número de árvores pode tornar o algoritmo lento e ineficaz para previsões em tempo real;

• Não são adequadas para variáveis contínuas: ao trabalhar com variáveis numéricas contínuas, a árvore de decisão perde informações quando categoriza variáveis em diferentes categorias.

Em suma, as FA's oferecem ferramentas para: a visualização de dados de dimensões elevadas (muitas colunas); análises de conglomerados; detecção de anomalias e outliers; análises de dados para pequenas amostras; identificação automática de preditores importantes; imputação de dados e geração de modelos preditivos acurados.

As FA's são baseadas no CART, Conjuntos de Aprendizagem (learning ensembles), Análise de especialistas (Committees of experts) e Bagging (Bootstrap Aggregation) (Dietterich, 2000). No Bagging trabalha-se com uma amostra de árvores. No entanto, utiliza-se o mesmo conjunto completo de preditores para determinar cada divisão dos dados. Isso resulta em grande correlação entre as árvores construídas, que são todas muito similares, resultando em pouca diversidade (James et al., 2013). Por outro lado, nas FA's introduz-se aleatoriedade ao selecionar-se subconjuntos aleatórios de preditores para cada divisão (Breiman, 2001).

De acordo com o site `https://dimensionless.in/introduction-to-random-forest/`, cada uma das árvores de decisão resulta em um classificador viesado (uma vez que considera apenas um subconjunto dos dados). Cada um captura diferentes tendências nos dados. Este conjunto de árvores é como uma equipe de especialistas, cada um com um pouco de conhecimento sobre o assunto geral, mas em sua área de especialização. Agora, em Problemas de Classificação, o *voto da maioria* é utilizado para classificar uma classe. Em analogia com os especialistas, é como fazer a mesma pergunta de múltipla escolha para cada especialista e utilizar a resposta mais frequente entre estes. No caso de problemas de Regressão, pode-se usar a média aritmética das predições obtidas em todas as árvores para descrever a predição global. Além disso, pode-se também atribuir mais peso a árvores mais decisivas (importantes) em relação a outras, ao utilizar-se dados de validação.

O esquema na Figura 3.18, proveniente do site `https://dimensionless.in/introduction-to-random-forest/`, exemplifica o problema de classificação pelos *votos da maioria*.

As florestas aleatórias (FA) representam uma coleção de árvores de decisão/classificação (CART) que seguem regras específicas na determinação do crescimento da árvore (tree growing),

na divisão (splitting), na combinação de árvores (tree combination), no auto-teste (self-testing) e no pós-processamento (post-processing). Neste processo, um conjunto contendo B amostras aleatórias de vetores dos preditores em estudo são aleatoriamente e independentemente selecionados. Para cada uma destas B amostras constrói-se uma árvore. As árvores selecionadas descrevem uma amostra I.I.D. (independentes e identicamente distribuídas) de árvores de uma dada floresta ou população (Breiman, 2001). As árvores construídas são combinadas de modo a obter-se uma predição conjunta. Isso remete ao conceito de "A união faz a força".

Crescimento - O crescimento (expansão) das árvores ocorre por particionamento binário (cada nó (pai) é dividido em não mais de dois filhos). Cada árvore é cultivada/expandida pelo menos parcialmente ao acaso.

A aleatoriedade é obtida ao expandir cada árvore baseada numa subamostra, escolhida ao acaso, a partir dos dados de treinamento. A aleatoriedade é também obtida durante o processo de divisão da árvore em cada nó. Tal divisão é determinada parcialmente ao acaso.

Divisão - Considere que há K variáveis preditoras no problema em questão. Seleciona-se, ao acaso, um pequeno subconjunto destas variáveis preditoras. Em geral utiliza-se $\sqrt{K}$. Por exemplo, se $K = 500$, seleciona-se cerca de $m_{try} = 23$ colunas da matriz de dados, em que a coluna denota variável. Posteriormente, divide-se cada nó com a "melhor" das 23 variáveis (não entre as 500).

Com as FA's obtém-se um grande número de modelos substancialmente diferentes e a aleatoriedade é introduzida de duas maneiras simultâneas:
(a) Por linha: os registros para treinamento são selecionados ao acaso, com reposição (da mesma forma como no resampling bootstrap do bagger);
(b) Por coluna: os preditores de candidatos em qualquer nó são escolhidos aleatoriamente e o "melhor divisor" selecionado do subconjunto aleatório.

Cada árvore é expandida até o seu tamanho máximo e podas não são realizadas. Experimentos têm mostrado, de forma convincente, que as podas prejudicam o desempenho dessas árvores. As árvores são deliberadamente sobre-ajustadas de modo a obter-se preditores que se assemelham ao *vizinho mais próximo*, uma técnica não-paramétrica bastante robusta. Possivelmente, árvores sobre-ajustadas se combinam para produzir conjuntos adequados.

Seja N_{tree} o número de árvores a construir. O algoritmo das Florestas Aleatórias segue os seguintes passos para cada uma das N_{tree} iterações:
1. Selecione uma nova amostra bootstrap (com reposição) a partir do conjunto de treinamento.
2. Expanda a árvore: adicione mais galhos (sem podar).
3. A cada nó interno, selecione ao acaso, m_{try} preditores e determine a melhor divisão usando somente estes m_{try} preditores.
4. Sem promover podas, registre (salve) a árvore, da forma como está, em um diretório próprio para estas árvores.
Obtenha uma predição global ou classificação modal (voto da maioria) de todas as árvores treinadas.

Hastie et al. (2001), pg. 588, apresentam um algoritmo detalhado para a técnica das Florestas Aleatórias. Tal algoritmo foi adaptado por Bill (2015) e é apresentado no Algoritmo 4.

Detalhes formais das Florestas Aleatórias podem ser encontrados no Apêndice A.

A seguir apresentaremos um exemplo de FA's utilizando comandos da biblioteca `randomForest` do R.

3.3.3 Exemplo no R sobre Florestas Aleatórias

A seguir, temos um exemplo de James et al. (2013) em que foi construído um algoritmo de florestas aleatórias por meio do pacote $R\{$`randomForest`$\}$ e a função de mesmo nome, nos

Algoritmo 4 - Florestas Aleatórias para Regressão ou Classificação

1. Para $b = 1$ até B:

(a) Selecione uma amostra bootstrap $\mathbf{Z}^*$ de tamanho N a partir dos dados de treinamento.

(b) Construa uma árvore T_b pertencente à Floresta Aleatória aos dados amostrados via bootstrap em (a). Isto é feito ao repetir, recursivamente (*loop*) os seguintes passos para cada nó terminal da árvore T_b, até que o nó de tamanho mínimo n_{min} seja alcançado.

Loop até que n_{min} o nó de tamanho mínimo n_{min} seja alcançado.

Selecione m variáveis ao acaso dentre as p variáveis disponíveis. Tipicamente, $m = \sqrt{p}$.

Conserve a "melhor variável" ou crie um ponto de divisão entre as m variáveis.

Divida o nó em dois nós-filhos.

Fim do loop.

2. Disponibilize as árvores $\{T_b\}, b = 1, \ldots, B$ de modo a obter uma predição de um novo ponto x:

Regressão: $\hat{f}_{fa}^{B}(x) - \frac{1}{B}\sum_{b=1}^{B} T_b(x)$.

Classificação: Seja $\hat{C}_b(x)$ a classificação obtida na árvore T_b.

Então seja $\hat{C}_{fa}^{B}(x) =$ voto da maioria entre $\hat{C}_b(x) b = 1, \ldots, B$.

permite utilizar a floresta aleatória no R. Os dados utilizados foram os mesmos dados usados em árvores de regressão: `Boston`.

```
install.packages("randomForest")
library(randomForest)
bag.boston=randomForest(medv~.,data=Boston,subset=train,mtry=13
    importance=TRUE) bag.boston
##
##Call:
## randomForest(formula = medv ~ ., data = Boston, mtry = 13,
## importance = TRUE,      subset = train)
##               Type of random forest: regression
##                     Number of trees: 500
##No. of variables tried at each split: 13
##
##          Mean of squared residuals: 11.1559
##                    % Var explained: 86.49
```

O argumento `ntree` pode mudar o número de árvores cultivadas pela `randomForest()`. Neste caso, indica que foram testadas 13 variáveis em cada divisão da árvore.

```
yhat.bag=predict(bag.boston,newdata=Boston[-train,])
plot(yhat.bag,boston.test)
abline(0,1)
mean((yhat.bag-boston.test)^2)
##[1] 13.2342
```

Para o conjunto de teste, o MSE associado à árvore de regressão *bagged* é 13.2342.

```
bag.boston=randomForest(medv~.,data=Boston,subset=train,
mtry=13,ntree=25)
yhat.bag=predict(bag.boston,newdata=Boston[-train,])
mean ((yhat.bag-boston.test)^2)
##[1] 13.44736
```

Para crescer a floresta aleatória será utilizado um menor valor para o argumento `mtry`. James et al. (2013) afirmam que a `randomForest()` utiliza $p/3$ variáveis ao construir uma floresta aleatória de árvores de regressão e $\sqrt{p}$ variáveis ao construir uma floresta aleatória de árvores de classificação. Neste caso, será utilizado `mtry` $= 6$. Observa-se que o conjunto de teste MSE é 11.48, o que indica que as florestas aleatórias renderam uma melhoria sobre o ensacamento neste caso.

```
set.seed(1)
rf.boston=randomForest(medv\~.,data=Boston,subset=train,
mtry=6,importance=TRUE)
yhat.rf=predict(rf.boston,newdata=Boston[-train,])
mean((yhat.rf-boston.test)^2)
##[1] 11.48
```

A função `importance()` avalia a importância de cada variável na floresta aleatória.

```
importance(rf.boston)
#          %IncMSE IncNodePurity
#crim    12.547772    1094.65382
#zn       1.375489      64.40060
#indus    9.304258    1086.09103
#chas     2.518766      76.36804
#nox     12.835614    1008.73703
#rm      31.646147    6705.02638
#age      9.970243     575.13702
#dis     12.774430    1351.01978
#rad      3.911852      93.78200
#tax      7.624043     453.19472
#ptratio 12.008194     919.06760
#black    7.376024     358.96935
#lstat   27.666896    6927.98475
```

Os autores ainda afirmam que são relatadas duas medidas de importância de variável. A primeira baseia-se na diminuição média da precisão nas previsões sobre as amostras fora do saco, quando uma determinada variável é excluída do modelo. A segunda é uma medida da diminuição da impureza total do nó que resulta de divisões sobre essa variável (calculada, em média, em todas as árvores). O gráfico de importância variável (Figura 3.19) também é uma ferramenta útil e pode ser plotado usando a função a função varImpPlot(). As 13 principais variáveis são selecionadas e plotadas com base na precisão do modelo e no valor de Gini.

```
varImpPlot(rf.boston)
```

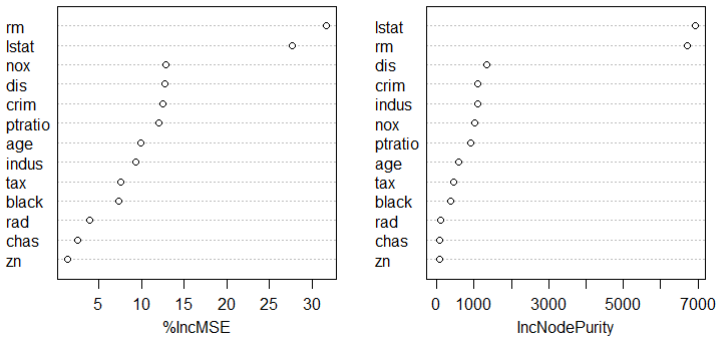


Figura 3.19: Gráfico de importância das 13 principais variáveis, baseado em James et al. (2008).

James et al. (2013) concluem, conforme os resultados, que dentre todas as árvores consideradas na floresta aleatória, o nível de riqueza da comunidade (`lstat`) e o tamanho da casa (`rm`) são as duas variáveis mais importantes.

Capítulo 4

Implementação Computacional

Devido ao esforço computacional deste trabalho, os algoritmos foram executados em máquinas com processador Intel Corel i7, 16 GB de memória. Todas as simulações e análises foram feitas utilizando o *software* R.

A seleção dos parâmetros de ajuste ótimos, através da validação cruzada (vide Capítulo 2), foi uma das partes computacionalmente mais intensivas desta dissertação, pois levou a inúmeras simulações, na tentativa de identificar o conjunto de parâmetros que retornaria o menor erro de teste.

Ao analisar os códigos em anexo, vemos que ao executar apenas uma validação cruzada (VC), obtém-se vários conjuntos de parâmetros que conduzem ao mesmo erro mínimo de VC. Desta forma, de modo a obter-se a unicidade dos parâmetros e a atingir-se o mínimo absoluto de VC foram realizadas algumas etapas, descritas a seguir:

Etapa 1: Definição dos conjuntos de treinamento e teste.
1.1- Fixa-se as frações dos dados a serem utilizadas como amostras para treinamento e teste. Nesta dissertação utilizou-se as frações de 2/3 dos dados para o treinamento e os 1/3 restantes para o teste.
1.2- Descreve-se os conjuntos de treinamento e teste.

Etapa 2: Realização do processo de validação cruzada (VC) para dada técnica de classificação.
2.1- Considera-se um conjunto de possíveis vetores de hiper-parâmetros. No caso de dois parâmetros, descreve-se uma matriz de possíveis pares de valores.
2.2- Define-se o número de *folds* a ser usado na análise e a composição dos mesmos. Nesta etapa os folds são definidos de acordo com o tipo de técnica de VC a ser utilizada (*Stratified k-fold, shuffle and split, leave-one-out*, etc). No entanto, os *folds* são fixos;
2.3- Calcula-se os erros de predição para cada vetor de hiper-parâmetros em análise.
2.4- Verifica-se se há um único vetor de hiper-parâmetros que minimiza o Erro de Treinamento (ET). Em caso afirmativo, este vetor representa o conjunto de *tunning* ótimo. Em caso contrário, procede-se à Etapa 3.

Etapa 3: No caso em que vários dos vetores de hiperparâmetros conduzem ao mesmo valor mínimo de ET, tenta-se encontrar a faixa de valores em que estão localizados os valores ótimos de *tunning*. Para tanto, descreve-se uma malha ou grade mais fina em torno dos valores dos hiper-parâmetros mais promissores (os que minimizaram o ET). Posteriormente, repete-se a Etapa 2 e, novamente, a Etapa 3 até que se encontre um vetor único de hiper-parâmetros que minimize o ET ou que não seja mais razoável ou producente continuar o processo de afinar a grade de valores dos hiper-parâmetros a serem estudados.

Etapa 4: Caso a Etapa 3 ainda não conduza a um único valor mínimo de ET, utiliza-se a última grade de vetores hiper-parâmetros obtida nessa etapa e replica-se, digamos R=500 vezes, todo o processo envolvido na Etapa 2. A diferença é que, agora, na Etapa 2.2 utilizam-se *folds* distintos do mesmo conjunto de treinamento.

Em geral, com as etapas acima consegue-se chegar ao vetor de *tunning* ótimo. De posse desse vetor ótimo de valores dos hiper-parâmetros para o procedimento de classificação em estudo, ajusta-se novamente o modelo e calcula-se o Erro de Predição (EP). Adicionalmente, constrói-se a Matriz de Confusão (*Confusion Matrix* - vide Seção 4.2), e a curva ROC (vide Seção 4.3), sendo ambas úteis para a avaliação do método de classificação empregado.

No intuito de calcular o Erro Médio de Teste (AVTE - *Average Test Error* - vide Seção 4.1), e produzir estatísticas mais robustas das taxas de má classificação e dos erros quadráticos médios, utilizou-se o processo de *Bagging* (vide Seção 4.4) considerando 1.000 replicações. Para o *Bagging* considerou-se o vetor ótimo de tunning dos hiper-parâmetros que foi encontrado nas etapas anteriores e, a cada replicação, utilizou-se um conjunto distinto de dados de treinamento e teste. Isso possibilitou a confecção de *Box-plots* para auxiliar nas análises comparativas das diversas técnicas de classificação utilizadas nesta dissertação.

De modo a tornar todo o processo reprodutível, fixou-se a mesma semente (de valor 107) no estudo. Para tanto, utilizou-se a função `set.seed` do R.

Neste Capítulo, serão apresentados o Erro Médio de Teste (Seção 4.1), a Matriz de Confusão (Seção 4.2), os gráficos ROC (Seção 4.3) e o *Bagging* (Seção 4.4). A Seção 4.5 relata a implementação dos três métodos de classificação RDA, SVM e RF. Os códigos em `R` de cada uma deles, estão disponíveis nos Apêndices C.1, C.2 e C.3.

4.1 Erro Médio de Teste

A classificação dos conjuntos de dados levou ao uso de várias implementações de validação cruzada na tentativa de identificar o conjunto de parâmetros que retornaria o erro mínimo absoluto. A avaliação final é baseada no erro médio de teste (*average test error*), referido, em aprendizado de máquinas, como *erro de generalização*, é estimado usando a medida do desempenho preditivo, AVTE($\hat{f}$), dado pela equação (4.1) (Bill, 2015):

$$\text{AVTE}(\hat{f}) = \frac{1}{R}\sum_{r=1}^{R}\left\{\frac{1}{m}\sum_{j=1}^{m} l(\mathbf{y}_j^{(r)}, \hat{f}^{(r)}(\mathbf{x}_j^{(r)}))\right\}, \tag{4.1}$$

em que l é definida de acordo com a expressão (2.4), $\hat{f}(\cdot)$ é um estimador para a função subjacente verdadeira, mas desconhecida, e $(\mathbf{x}_j^{(r)}, \mathbf{y}_j^{(r)})$ é a j-ésima observação a partir do conjunto de teste de tamanho m na r-ésima repetição aleatória da divisão dos dados. O objetivo final de qualquer técnica de classificação de aprendizagem supervisionada é minimizar o erro, o que implica em minimizar a equação (4.1) (Bill, 2015).

Os resultados para o AVTE são apresentados no Capítulo 5. Para a estimação do erro de teste foi utilizado o Algoritmo AVTE de Bill (2015) apresentado no Algoritmo 5.

4.2 Matriz de confusão

A matriz de confusão *(confusion matrix)* é uma matriz utilizada para descrever o número de previsões corretas e incorretas feitas por um classificador treinado. Segundo Sebastian (2015), ela estabelece o desempenho de um algoritmo de aprendizado.

Algoritmo 5 Replicação do Erro de Teste

R=1000
For $r = 1$ to R Loop
Usando a estratificação, divida aleatoriamente os dados em $\frac{2}{3}$ e $\frac{1}{3}$.
Construa o modelo com a técnica escolhida e obtenha os valores ótimos dos hiperparâmetros usando os $\frac{2}{3}$ de dados de treinamento.
Teste o modelo criado pelos dados de treinamento usando os $\frac{1}{3}$ de dados para o teste.
Armazene o r-ésimo erro na matriz $Merr_r$.
End Loop
(AVTE) $= \frac{1}{R}\sum_{r=1}^{R} Merr_r$.

A matriz de confusão, representada na Tabela 4.1, relaciona as contagens das previsões verdadeiras positivas, verdadeiras negativas, falsas positivas e falsas negativas de um classificador, conforme descritas as seguir: 1) VP - verdadeiro positivo, com TVP chamado de *sensibilidade*, referem-se às instâncias positivas que foram corretamente previstas pelo classificador; 2) FN - falso negativo - representam às instâncias positivas que foram incorretamente rotuladas como negativas; 3) FP - falso positivo - representam às instâncias negativas que foram incorretamente rotuladas como positivas; 4) VN - verdadeiro negativo, com TVN chamado de *especificidade*, referem-se às instâncias negativas que foram corretamente previstas pelo classificador.

É importante observar que VP e VN indicam o quanto o classificador está acertando nas predições, enquanto FP e FN indicam o quanto o classificador está errando nas predições.

Tabela 4.1: Representação de uma matriz de confusão e suas medidas de avaliação.

		Classe Prevista		
		Positivo	Negativo	
Classe Real	Positivo (P)	Verdadeiro Positivo (VP)	Falso Negativo (FN)	TVP $= \frac{VP}{P} = \frac{VP}{FN+VP}$
	Negativo (N)	Falso Positivo (FP)	Verdadeiro Negativo (VN)	TFP $= \frac{FP}{N} = \frac{FP}{FP+VN}$ TVN $= \frac{VN}{N}$
		VPP $= \frac{VP}{VP+FP}$	VPN $= \frac{VN}{FN+VN}$	ACC $= \frac{VP+VN}{FP+FN+VP+VN}$

Como exemplo prático, consideremos o conjunto de dados `colon` (Seção 5.1) composto pelas informações genéticas de pacientes com câncer no intestino grosso e de pacientes sem a doença. Observando a Tabela 4.2, a classe "positivo" é formada pelos indivíduos que apresentam a doença e a classe "negativo" é formada pelos indivíduos que não apresentam. Observa-se, ainda, que cada coluna apresenta os quantitativos previstos, enquanto cada linha relaciona o quantitativo da doença verdadeira.

No diagnóstico do câncer, objetiva-se não somente detectar pacientes doentes que foram incorretamente classificados sem a doença (FN), mas também detectar os pacientes não doentes que foram incorretamente classificados como doentes (FP), para minimizar diagnósticos equivocados. Desta forma, um classificador com um bom desempenho deve rotular corretamente os pacientes doentes (VP) e os não doentes (VN). A Tabela 4.2 mostra que os resultados para esses casos (FN, FP, VP e VN) foram: 0, 1, 15 e 4, respectivamente.

Para os dados de câncer de cólon, o modelo classificou corretamente 15 pacientes como doentes (VP) e 4 como não doentes (VN). Além disso, nenhum paciente doente foi classificado incorretamente (FN). No entanto, o modelo classificou incorretamente 1 paciente como doente, embora ele não apresentasse a doença (FP).

Tabela 4.2: Exemplo de tabela de confusão usando os dados de câncer de cólon (`colon`)na aplicação do método RDA.

		Classe Prevista		
		Com câncer	Sem câncer	
Classe Real	Com câncer	15	0	TVP $= \frac{15}{0+15} = 1$
	Sem câncer	1	4	TFP $= 0,2$ e $TVN = 0,8$
		VPP $= \frac{15}{15+1} = 0,9375$	VPN $= \frac{4}{0+4} = 1$	ACC $= \frac{15+4}{15+0+1+4} = 0,95$

A partir de agora serão descritas as medidas de avaliação que pode-se derivar de uma matriz de confusão, iniciando pela acurácia. Lopes (2015) descreve a acurácia (ACC) de um classificador para um determinado conjunto de teste como a porcentagem de instâncias corretamente previstas pelo classificador, sendo estas positivas ou negativas. Essa medida é calculada a partir da soma das previsões corretas divididas pelo número total de previsões, conforme pode ser observado na Tabela 4.1.

O erro de predição (ERR) pode ser entendido como a soma de todas as previsões falsas divididas pelo número total de predições. Ele pode ser calculado diretamente pela acurácia: $ERR = 1 - ACC$. Logo, como na Tabela 4.2 a $ACC = 0,95$, então $ERR = 1 - ACC = 0,05$.

O valor de predição positiva (VPP) é visto como o percentual de instâncias rotuladas como positivas e que realmente o são. Neste caso, temos uma $VPP = 0,9375$. A Taxa de Verdadeiros Positivos (TVP), fornece informações sobre a fração de instâncias positivas que foram identificadas corretamente do total de positivos (P). Na Tabela 4.2, tem-se que $TVP = 1$. Enquanto TVN fornece informações sobre a fração de instâncias negativas que foram corretamente identificadas do total de negativos (N). Neste exemplo, tem-se que $TVN = 0,8$.

A taxa TFP representa a proporção de instâncias negativas que foram incorretamente classificadas, como positivas, em relação ao total de negativos (N). No nosso exemplo, o resultado desta proporção foi igual a $0,2$. O ideal seria uma $TVP = 1$ e $TFP = 0$.

Nesta pesquisa, para cada um dos métodos de classificação em estudo: RDA, SVM e Florestas Aleatórias, encontrou-se a matriz de confusão referente a cada uma das simulações realizadas e, posteriormente, foram encontradas suas respectivas curvas ROC. Os gráficos podem ser encontrados no Capítulo 5.

4.3 Curva ROC

As curvas ROC (*Receiver Operator Characteristic*) foram introduzidas na aprendizado de máquinas como uma poderosa ferramenta para a avaliação de modelos de classificação, ilustrando o quão bem funciona o algoritmo classificador. Um modelo de classificação pode ser representado por um ponto no espaço ROC e, para se obter um ponto neste espaço usa-se um modelo de classificação e calcula-se a taxa de verdadeiros e falsos positivos.

O eixo x, ou variável independente, é representado pela Taxa de Falsos Positivos (TFP) e o eixo y, variável dependente, pela Taxa de Verdadeiros Positivos (TVP). Cada ponto no espaço ROC é um par de dados Verdadeiro Positivo (VP)/Falso Positivo (FP).

Na Figura 4.1, a linha diagonal que liga o ponto $(0, 0)$ e o ponto $(1, 1)$ denota o desempenho do classificador aleatório. As curvas ROC na área com o canto superior esquerdo (0, 1) indicam bons níveis de desempenho, enquanto as curvas ROC na área com o canto inferior direito (1,0) indicam níveis de desempenho insatisfatórios. Desta forma, um ponto de desempenho perfeito seria o ponto $(0, 1)$ indicando 0% falsos positivos e 100% de verdadeiros positivos. Sendo assim, quanto mais um ponto no espaço ROC estiver acima da linha diagonal, melhor será o valor preditivo do teste. No Capítulo 5 serão apresentados os gráficos ROC referentes ao desempenho de cada uma técnicas RDA, SVM e RF.

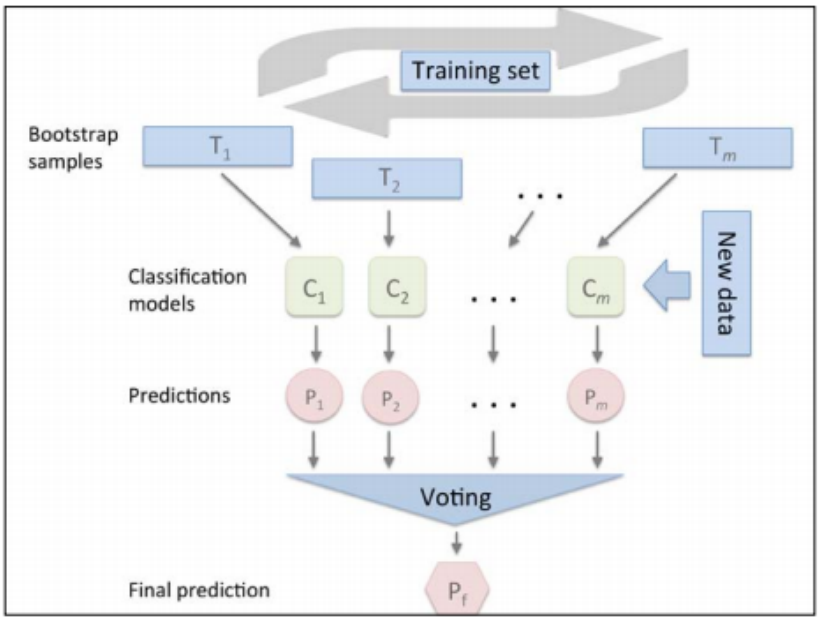


Figura 4.2: Representação do *Bagging*: construindo um conjunto de classificadores a partir de amostras *bootstrap*. Fonte: Sebastian (2015).

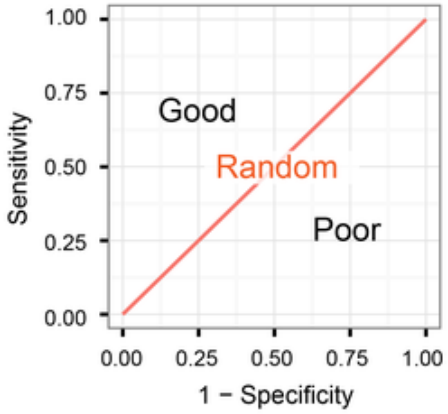


Figura 4.1: A curva ROC separa o espaço em duas áreas representando os bons e os baixos níveis de desempenho. Fonte: Takaya, S. and Marc R. (2015).

4.4 Bagging

O *Bagging*, representado na Figura 4.2, é uma maneira simples de aumentar o poder de um modelo estatístico preditivo, ao coletar várias amostras aleatórias (com reposição) do conjunto de treinamento inicial e usar cada uma dessas amostras para ajustar um classificador C_j e realizar as predições para o conjunto de teste. Em seguida, calcula-se a média dessas predições para obter o valor de predição final.

Sebastian (2015) diz que o *Bagging* pode melhorar a precisão de modelos instáveis e diminuir a variação, mas é ineficaz na redução do viés. O autor afirma que o *Bagging* também está relacionado ao classificador de florestas aleatórias. De fato, as florestas aleatórias são um caso especial de *Bagging*, onde usam-se subconjuntos de características aleatórias para ajustar as árvores de decisão individuais.

4.5 Técnicas de Classificação

4.5.1 Análise Discriminante Regularizada (RDA)

Para implementação da RDA, utilizou-se a função `rda` do software `R` (Guo et al., 2005). Este pacote depende de uma versão $R \geq 2.10$ e tem a capacidade de trabalhar com classificação em dados de alta dimensão. Os dados de câncer de cólon (`colon`) (Alon, et al. 1999) estão disponíveis na biblioteca `rda`.

A função `rda` requer que a entrada de um conjunto de dados seja uma matriz numérica formatada, de tal forma, que as colunas sejam as observações da amostra e a linhas sejam as características. Por exemplo, as colunas estariam representando as amostras dos indivíduos e as linhas, os genes. Desta forma, para utilizar a função `rda` nos dados de `colon`, foi necessário que a matriz de dados fosse previamente transposta.

O modelo RDA possui dois parâmetros: α e δ. O *parâmetro de regularização* α introduz uma pequena quantidade de viés para ajudar a estabilizar a matriz de covariância e resolver o problema da singularidade. O hiper-parâmetro α, pode ser qualquer valor entre 0 e $0,99$. O hiper-parâmetro δ é usado para controlar o encolhimento (shrinkage) das médias e pode ser qualquer valor entre 0 e 3. Para ambos os hiper-parâmetros, α e δ, pode-se estabelecer um vetor de valores ou um valor único (Bill, 2015).

Objetiva-se encontrar os valores de α e δ que minimizam o erro de estimação do modelo. Para isso, deve-se testar uma grade de pares de valores e verificar qual a combinação que maximiza a performance do modelo. Os parâmetros são determinados a partir do uso da validação cruzada, vide Seção 2.4.

As Tabelas 5.1 e 5.5 apresentam os hiper-parâmetros ótimos ajustados por meio da técnica de Análise Discriminante Regularizada (RDA).

4.5.2 Máquina de Vetor de Suporte (SVM)

Para utilização da técnica SVM foi instalado o pacote `e1071`. Esse pacote implementa uma interface para o código `libsvm` $C++$ para máquinas de vetores de suporte.

As SVMs podem ser utilizadas na classificação ou na regressão. Dependendo se y é um fator ou não, o pacote `e1071` oferece os tipos (`type`) de classificação *C-classification* ou *eps-regression*, respectivamente, dentre as opções: *C-classification, nu-classification, one-classification, eps-regression e nu-regression.* Neste trabalho foi utilizado o tipo de classificação *C-classification.*

No pacote `e1071` pode-se utilizar a função `tune`, para testar a grade de parâmetros fornecidos, e a função `svm`, para treinar o modelo SVM. A função `svm` aceita os seguintes kernels: linear, polynomial, radial basis ou sigmoid. Neste trabalho, foram ajustados modelos para cada tipo de kernel, visando encontrar-se aquele que apresentasse a melhor estimativa do erro de generalização. Dependendo do tipo de kernel utilizado, os parâmetros escolhidos foram: *degree* (default: 3), *gamma* (default: 1/(dimensão dos dados)), $coef0$ (default: 0) e *custo* (default: 1).

O parâmetro *degree* é necessário para o kernel do tipo polynomial e o parâmetro $coef0$ para os kernels do tipo polynomial e sigmoid. O parâmetro γ (*gamma*) é necessário para todos os tipos de kernel, exceto o linear. Segundo Lima (2014), para valores baixos de γ o limite de decisão é quase linear. Se o valor de γ aumenta, a flexibilidade do limite de decisão aumenta.

O parâmetro *custo* é a constante C do termo de regularização na formulação de Lagrange (3.45). Este parâmetro é necessário para todos os tipos de kernel. Segundo Ben-Hur e Weston (2008), aumentando-se o valor de C, aumenta-se o custo do erro de classificação, então o hiperplano aproxima-se de vários outros pontos. Se diminuirmos o valor de C o hiperplano provê uma margem maior.

As Tabelas 5.2 e 5.6 apresentam os resultados dos hiper-parâmetros ótimos ajustados por meio da SVM.

4.5.3 Florestas Aleatórias (FA)

Para a aplicação da técnica FA, instalou-se o pacote `randomForest`. Esse pacote implementa o algoritmo de floresta aleatória de Breiman (baseado no código Fortran original de Breiman e Cutler) e tem a capacidade de realizar aprendizado supervisionado e não supervisionado. Nesta dissertação, foi empregada apenas a aprendizagem supervisionada. Ao executar o aprendizado supervisionado com a técnica de florestas aleatórias, o vetor de rótulos de classe y deve ser do tipo fator.

Nesta pesquisa, *ntree* e *nodesize* foram os parâmetros de ajuste escolhidos para otimização. O parâmetro *ntree* controla o número de árvores a serem treinadas. Esse número não deve ser muito pequeno, para garantir que cada observação, ou linha de entrada, seja prevista algumas vezes. O parâmetro *nodesize* indica o tamanho mínimo dos nós terminais na árvore. O valor padrão do tamanho do nó para classificação é 1 e, para a regressão, é 5. Quanto maior for esse número, menores serão as árvores individuais na floresta (Bill, 2015).

O argumento `mtry` é o número de variáveis amostradas aleatoriamente como candidatos em cada divisão da árvore. Os valores padrão são diferentes para classificação ($\sqrt{p}$, onde p é o número de variáveis em x) e regressão ($\frac{p}{3}$).

As Tabelas 5.3 e 5.7 apresentam os resultados dos hiper-parâmetros ótimos ajustados utilizando RF.

Capítulo 5

Aplicação em dados reais

O Capítulo 3 mostrou três abordagens de aprendizado de máquinas adequadas à classificação de dados: RDA, SVM e FA. No Capítulo 4 descreveu-se como foi realizada a implementação computacional de cada uma das abordagens estudadas. Adicionalmente, suas respectivas rotinas em `R` estão disponíveis nos Apêndices C.1, C.2 e C.3..

Este estudo contou com a aplicação das técnicas de aprendizagem em dois bancos de dados. O primeiro deles relaciona o caso em que $n \lll p$, descrito na Seção 2.2. O outro conjunto de dados não apresenta tal característica. As Seções 5.1 e 5.2 fazem uma breve descrição sobre cada um dos bancos de dados utilizados e apresentam os resultados obtidos para cada um deles.

5.1 Dados I - Dados de Câncer de Cólon

O primeiro conjunto de dados a ser trabalhado (Dados I), relaciona dados de microarray de câncer de cólon (Alon et al., 1999), que é caracterizado pela alta dimensionalidade p dos vetores de variáveis explanatórias enquanto o tamanho da amostra n é muito pequeno, isto é, $n \lll p$.

Este conjunto de dados contém 62 observações sobre indivíduos classificados em dois grupos (**grupo 1**: pacientes com câncer de cólon, com 40 observações; **grupo 2**: pacientes sem a doença, com 22 observações) e medidos em 2000 variáveis (características genéticas colhidas a partir de tecido do intestino grosso).

Este conjunto de dados, bem conhecido na literatura moderna de classificação, é tratado como um conjunto de dados binário. Objetiva-se classificar os indivíduos em doentes ou não-doentes de acordo com as informações genéticas. A seguir é apresentado um resumo do processamento dos dados de cólon no estudo de Alon et al. (1999).

Segundo Alon et al. (1999), as matrizes de *oligonucleótidos* podem fornecer uma imagem ampla do estado da célula, monitorando o nível de expressão de milhares de genes ao mesmo tempo. Para analisar o conjunto de dados de câncer de cólon, o RNA foi extraído, hibridizado com matrizes de *oligonucleótidos* produzidos pela *Affymetrix Hum6000*, resultando em aproximadamente 65.000 características.

A flutuação do sinal de hibridização entre as diferentes características foi avaliada. Quando esse sinal de hibridização é muitas vezes mais forte que o sinal vizinho, é considerado um outlier e é descartado para que ele não interfira na leitura da intensidade média do gene.

Um algoritmo baseado em *deterministic-annealing* foi usado para organizar os dados numa árvore binária. Cada gene k foi organizado em um vetor V_k e cada vetor, ou gene, foi normalizado de modo que a intensidade média dos tecidos fosse 0 e o desvio padrão fosse 1. A normalização foi feita para enfatizar a variação relativa da intensidade e não a intensidade absoluta (Alon et al., 1999 Apud Bill, 2015).

O algoritmo desenvolvido com base no algoritmo de *deterministic-annealing* separa um conjunto de objetos (genes ou tecidos) em dois grupos. Em seguida, separa cada grupo em dois subgrupos e assim por diante, até que todos os objetos estejam dispostos em uma árvore binária. Como esse algoritmo gera uma árvore não ordenada, Alon et al. (1999) utilizaram um método para impor uma ordem nos ramos da árvore, para que seja obtida uma lista final ordenada. Então, usaram essa ordenação bidirecional de genes e tecidos para reorganizar as linhas e colunas do conjunto de dados, para que os genes e tecidos correlacionados sejam exibidos próximos uns aos outros.

Para ajudar a visualizar os dados foi utilizado um código de cores com intensidade gênica variando de vermelho (alta intensidade) a azul (baixa intensidade) (Figura 5.1). A intensidade de cada gene é normalizada de modo que a variação relativa na intensidade é enfatizada, ao invés da intensidade absoluta. O método de *clustering* bidirecional foi aplicado ao conjunto de dados de características genéticas produzindo uma matriz, que pareceu revelar padrões amplos e coerentes de genes (Figs. 5.1 B e 5.2). As áreas de alta ou baixa intensidade correspondem a grupos de dezenas a centenas de genes cuja expressão é coordenada em um grau substancial entre os grupos de amostras de tecido. Em contraste, o mesmo algoritmo aplicado a um conjunto de dados randomizados (Fig. 5.1 C e D) produziu uma matriz aparentemente com pouca estrutura. Essa diferença na padronização reflete a organização subjacente da expressão gênica no conjunto de dados reais (Alon, 1999).

O agrupamento dos genes no conjunto de dados revela grupos de genes cuja expressão está correlacionada entre os tipos de tecidos. A intensidade dos genes da proteína ribossômica é relativamente baixa (azul) nos tecidos normais do cólon e alta (vermelha) nos tecidos cancerígenos. Desta forma, o agrupamento separou o tecido cancerígeno do tecido normal (não cancerígeno).

Este trabalho utilizou os dados denominados `colon` disponibilizados no pacote `rda` do software `R`. Este dados são formados por uma matriz (62x2000). O repositório CRAN (2015) descreve que estes dados contêm dois objetos: `colon.x`, matriz de dados com cada linha representando uma amostra de câncer de cólon e `colon.y`, os labels para essas 62 amostras contendo o tipo de amostra de tecido (tumoral ou normal).

Nesta dissertação, não aplicou-se a padronização às covariáveis do conjunto de dados `colon`, uma vez que Bill (2015) concluiu que a padronização conduz a classificações menos precisas. Os hiper-parâmetros a serem mostrados neste capítulo serão os que proporcionaram o AVTE mínimo.

5.1.1 Parâmetros de Ajuste Ótimos e Erro Médio de Teste

As tabelas a seguir, apresentam os resultados das análises de validação cruzada realizadas no intuito de identificar o conjunto de parâmetros de ajuste ótimo ideal para cada técnica preditiva. O conjunto de parâmetros ideal é aquele que retorna o menor erro médio de teste (representado nas tabelas por AVTE) e descrito na Seção 4.1.

Análise Discriminante Regularizada (RDA)

A Tabela 5.1 apresenta os resultados obtidos para a RDA utilizando três métodos de validação cruzada (vide Seção 2.4) para estimação dos parâmetros. Nas colunas são apresentados os valores dos hiper-parâmetros ótimos, α e δ, e o valor do erro médio do teste (AVTE) para cada tipo de VC. Nas linhas, tem-se os tipos de validação cruzada (Shuffle and Split (SS), Stratified Cross Validation (SCV) e Leave-One-Out Cross Validation (LOOCV)) e os tamanhos de k-Folds a serem testados em cada tipo de VC. Nesta pesquisa, optou-se por valores de k iguais

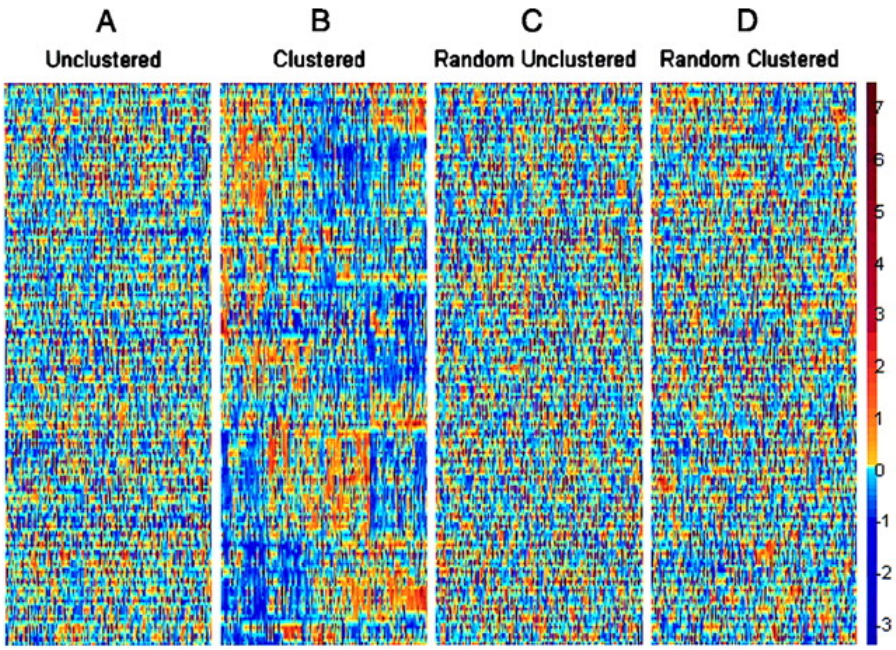


Figura 5.1: Conjunto de dados de câncer de cólon composto por 40 tecidos cancerígenos e 22 tecidos normais. Os 2000 genes escolhidos para o conjunto de dados de câncer de cólon foram os genes com a maior intensidade mínima entre todas as amostras. O eixo vertical corresponde aos genes e o eixo horizontal aos tecidos. Cada gene foi normalizado de modo que sua intensidade média entre os tecidos é 0 e seu SD é 1. O código de cores usado é indicado na escala adjacente. (A) Conjunto de dados não agrupados. (B) Dados agrupados. Os 62 tecidos estão dispostos no eixo vertical e os 2.000 genes estão dispostos no eixo horizontal. (C) Dados randomizados não agrupados, onde o conjunto de dados original foi randomizado (a localização de cada número na matriz foi alterada aleatoriamente). (D) Agrupamentos de dados aleatórios, sujeitos ao mesmo algoritmo de *clustering* como em B. Fonte: Alon et al. (1999).

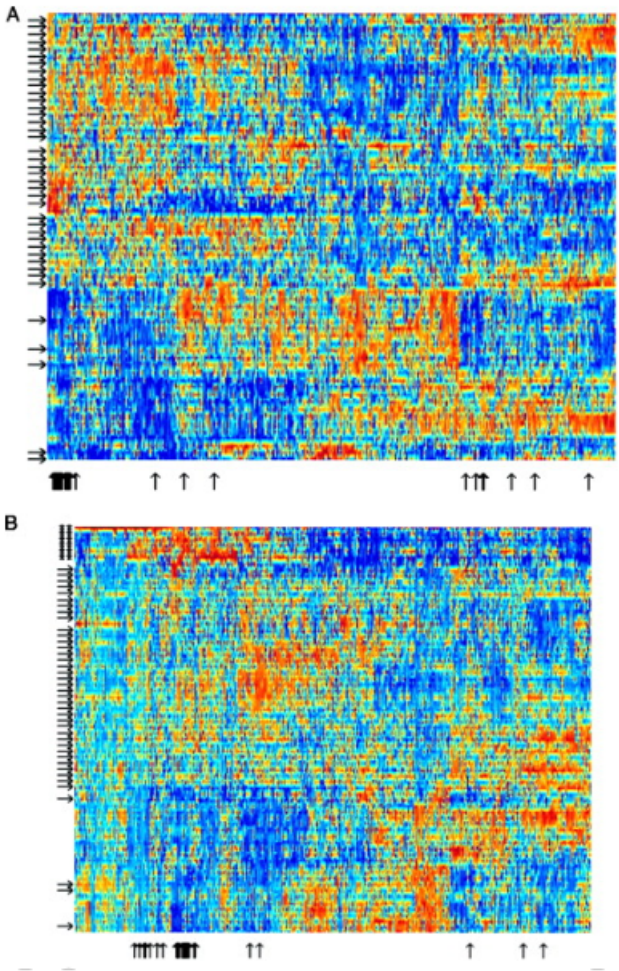


Figura 5.2: (A) Visão ampliada do conjunto de dados agrupados de 2.000 genes em 22 tecidos normais e 40 cancerígenos. Os genes escolhidos são os 2.000 genes com maior intensidade mínima nas amostras. Observa-se a separação dos tecidos normais e cancerígenos: os tecidos cancerígenos estão marcados com setas à esquerda e os tecidos normais não estão marcados. (B) O mesmo que A, mas com linhagens celulares de carcinoma de colon EB e EB1 adicionadas ao conjunto de dados. Fonte: Alon et al. (1999).

a 4 e 5, com base em um estudo realizado por Bill (2015, pg. 40), bem como k igual a 10, por ser um valor normalmente utilizado, segundo James et al. (2013).

Tabela 5.1: Método RDA e Dados I - Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação RDA, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10).

VC	K-Folds	α	δ	AVTE
SS	4	0,038	0,15	0,1155
	5	0,075	0,15	0,1108
	10	0,075	0,15	0,1147
SCV	4	0,038	0,15	0,1155
	5	0,075	0,15	0,1093
	10	0,075	0,15	0,1132
LOOCV	4	0,075	0,15	0,1132
	5	0,075	0,15	0,1132
	10	0,075	0,15	0,1132

Na Tabela 5.1 observa-se que δ é igual a $0,15$ em todos os resultados obtidos. Nas validações cruzadas SS e SCV, quando $k = 4$, tem-se que α é igual a $0,038$. Nos demais casos, o parâmetro de regularização α é igual a $0,075$.

Ao analisar o erro médio do teste (AVTE) apresentado na Tabela 5.1, verifica-se que o menor valor $(0,1093)$ para o AVTE foi obtido com o método SCV de validação cruzada e $k = 5$. Logo, o conjunto de hiper-parâmetros ideal é dado por $\alpha = 0,075$ e $\delta = 0,15$.

Máquina de Vetor de Suporte (SVM)

A Tabela 5.2 mostra os resultados da aplicação do método SVM comparando-se quatro tipos de kernels: linear, polynomial, sigmoid e radial.

Os parâmetros foram escolhidos dependendo do tipo de kernel. O parâmetro *degree*, necessário para o kernel do tipo polynomial, foi utilizado adotando-se $degree = 1$. Para o parâmetro *coef0*, necessário para os tipos de kernel polynomial e sigmoid, adotou-se $coef0 = 0$.

O γ foi utilizado para todos os tipos de kernels, exceto o linear. No parâmetro γ foi usado o valor padrão $\frac{1}{p}$, em que p é a dimensionalidade ou o número de características no conjunto de dados, logo, $\gamma = \frac{1}{2000}$.

O parâmetro *custo* foi utilizado em todos os tipos de kernels. Os valores de *custo* variaram entre os tipos de kernels, uma vez que foram testados diversos valores, $\gamma = (0.001, 0.01, 0.1, 0.5, 1, 5, 10, 100)$, a fim de encontrar o parâmetro ideal.

Dentre os resultados obtidos na Tabela 5.2, observa-se que a SVM utilizando o kernel sigmoid apresentou o menor AVTE (0,1246) nas técnicas de validação cruzada SS e SCV, com $k = 5$. Os parâmetros de ajuste ótimos foram: $\gamma = 0,0005$, $custo = 1$ e $coef = 0$.

Florestas Aleatórias

Os resultados para a FA são apresentados na Tabela 5.3. Os parâmetros escolhidos para otimização ($ntree = 280$ e $nodesize = 14$) também foram encontrados por Bill (2013, pág.86).

Tabela 5.2: Método SVM e Dados I - Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). A técnica de SVM foi aplicada testando-se quatro tipos de kernels: (a) linear, (b) polynomial, (c) sigmoid e (d) radial.

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	0,001	0,1266
	5	0,001	0,1257
	10	0,001	0,1286
SCV	4	0,001	0,1266
	5	0,001	0,1257
	10	0,001	0,1289
LOOCV	4	0,001	0,1271
	5	0,001	0,1271
	10	0,001	0,1271

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,0005	1	0,3409
	5	0,0005	1	0,3404
	10	0,0005	1	0,3455
SCV	4	0,0005	1	0,3409
	5	0,0005	1	0,3404
	10	0,0005	1	0,3460
LOOCV	4	0,0005	1	0,3438
	5	0,0005	1	0,3438
	10	0,0005	1	0,3438

(c) Kernel Sigmoid

VC	K-Folds	γ	Custo	AVTE
SS	4	0,0005	1	0,1273
	5	0,0005	1	0,1246
	10	0,0005	1	0,1267
SCV	4	0,0005	1	0,1273
	5	0,0005	1	0,1246
	10	0,0005	1	0,1265
LOOCV	4	0,0005	1	0,1270
	5	0,0005	1	0,1270
	10	0,0005	1	0,1270

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,0005	100	0,2025
	5	0,0005	100	0,1992
	10	0,0005	100	0,2071
SCV	4	0,0005	100	0,2025
	5	0,0005	100	0,1992
	10	0,0005	100	0,2052
LOOCV	4	0,0005	100	0,2022
	5	0,0005	100	0,2022
	10	0,0005	100	0,2022

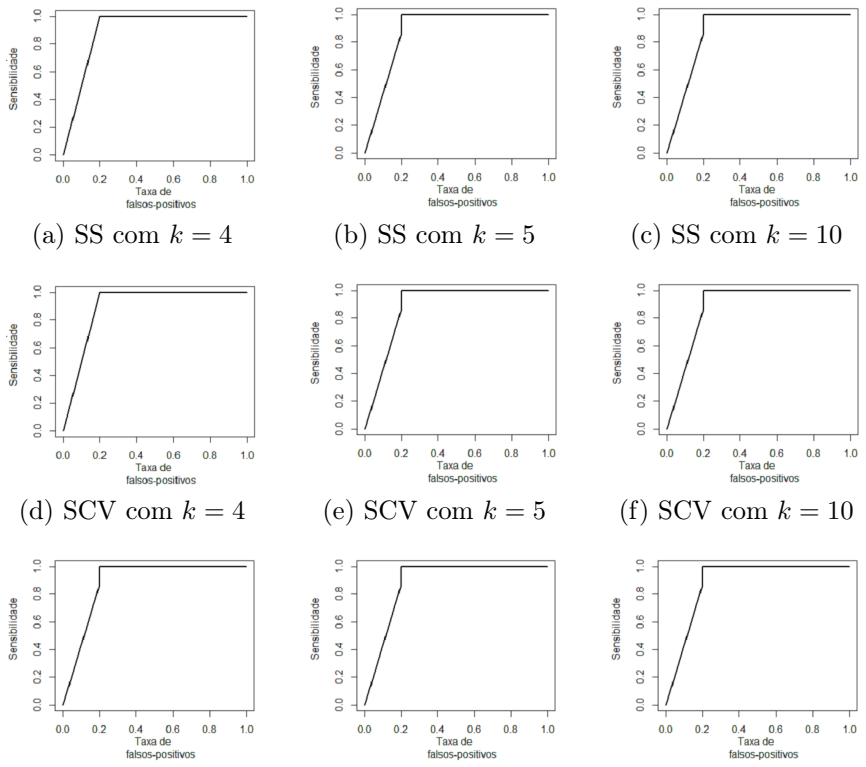
Tabela 5.3: Florestas Aleatórias e Dados I - Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação FA, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10).

VC	K-Folds	Ntree	Nodesize	AVTE
SS	4	280	14	0,1435
	5	280	14	0,1448
	10	280	14	0,1443
SCV	4	280	14	0,1447
	5	280	14	0,1457
	10	280	14	0,1415
LOOCV	4	280	14	0,1418
	5	280	14	0,1418
	10	280	14	0,1418

Com base nos resultados exibidos na Tabela 5.3, observa-se que os parâmetros ($Ntree = 280$ e $Nodesize = 14$) conduzem ao menor erro de teste (0,1415) quando utilizou-se o método de validação cruzada SCV com $k = 10$.

5.1.2 Curva ROC

Os gráficos ROC são úteis para selecionar modelos para classificação com base em seu desempenho em relação às taxas falsas positivas e verdadeiras positivas. Os pontos no espaço ROC de classificador perfeito ocorreriam no canto superior esquerdo do gráfico com uma TPV igual a 1 e uma TFP igual a 0 (Sebastian, 2015). Nesta Seção, serão apresentadas as curvas ROC referentes a cada um dos métodos apresentados na Seção 5.1.1.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

Figura 5.3: RDA - Dados I. Curvas ROC obtidas usando-se os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.

Análise Discriminante Regularizada (RDA)

Na Figura 5.3, apresenta-se as curvas ROC para a RDA e cada tipo de validação cruzada (SS, SCV e LOOCV). Cada tipo de VC foi testado com valores de k-folds iguais a 4, 5 e 10, respectivamente. Observa-se na Figura 5.3, que curvas ROC apresentadas nos gráficos mostram um bom desempenho dos classificadores.

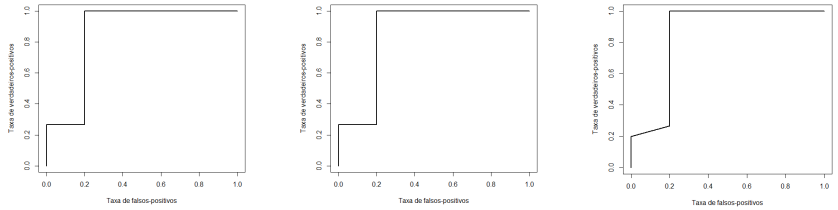
Máquina de Vetor de Suporte (SVM)

As Figuras 5.4 e 5.5 mostram os resultados da SVM testando-se os tipos de kernel linear e sigmoid. Observa-se uma semelhança gráfica entre as curvas ROC referentes a estes kernels. De fato, ao analisar-se os resultados da Tabela 5.2, verifica-se que os resultados para estes dois tipos de kernels são bem próximos e apresentam os melhores erros médios de teste em relação aos demais tipos de kernels (vide Apêndice B.1).

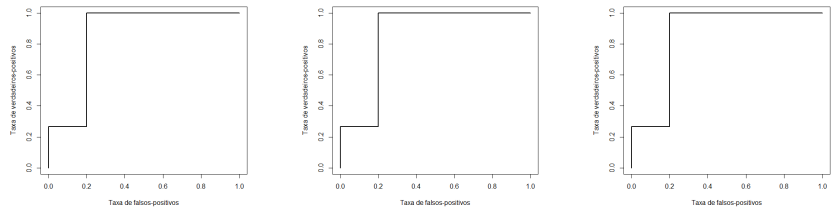
Comparando-se as curvas ROC obtidas na SVM (Figuras 5.4 e 5.5) com as curvas ROC da Figura 5.3, aparentemente o método RDA fornece um melhor desempenho classificador em comparação com o método SVM. As curvas ROC da RDA estão mais próximas do canto superior esquerdo, em comparação às curvas da SVM, e quanto mais próxima estiver do canto superior esquerdo, melhor será o valor preditivo do teste.

Florestas Aleatórias (FA)

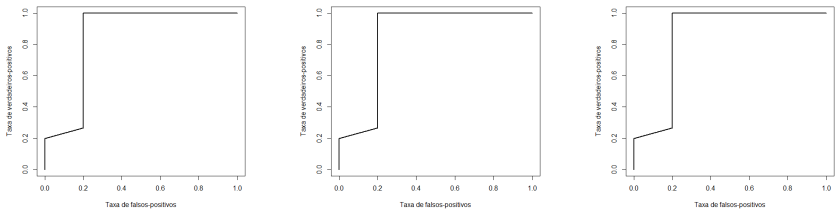
As curvas ROC ajustadas para FA usando os dados de cólon são ilustradas na Figura 5.6. Percebe-se que o desempenho classificador parece ser aceitável, uma vez que as curvas encontram-se acima da diagonal que liga o ponto $(0, 0)$ e o ponto $(1, 1)$. No entanto, em comparação com a Figura 5.3, aparentemente a RDA está mais próxima do canto superior esquerdo em comparação com a RF.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

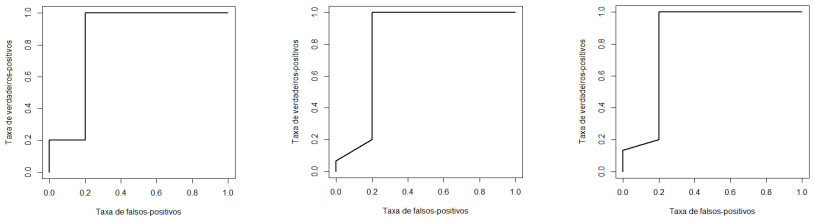


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

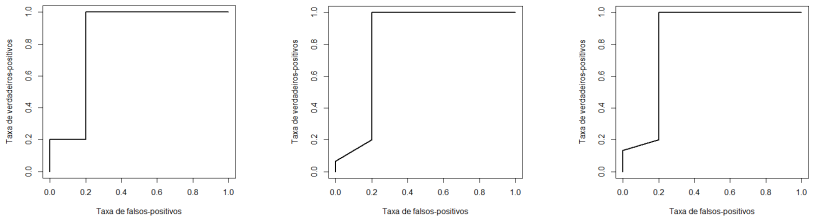


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

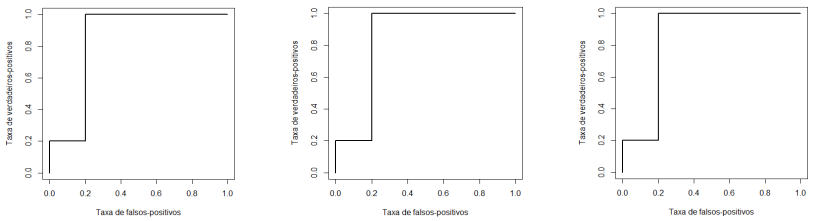
Figura 5.4: SVM (kernel Linear) - Dados I. Curvas ROC obtidas usando-se os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

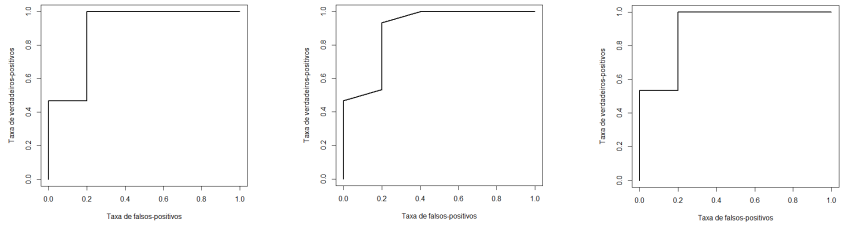


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

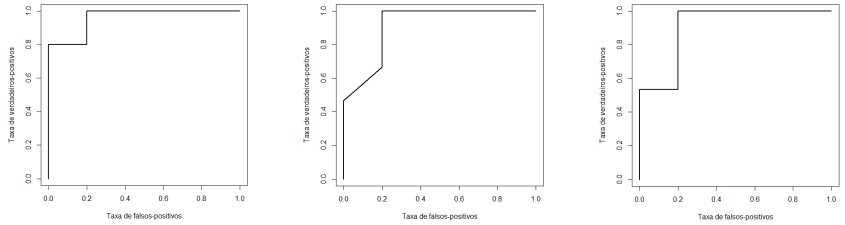


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

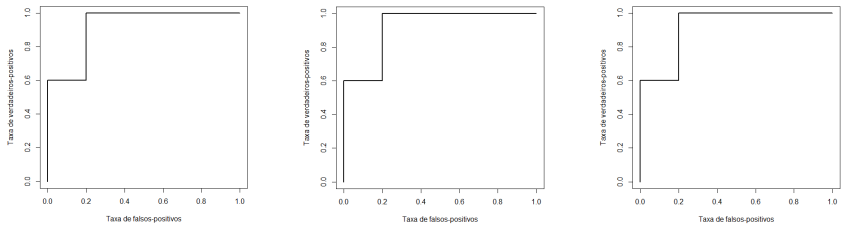
Figura 5.5: SVM (kernel Sigmoid) - Dados I. Curvas ROC obtidas usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

Figura 5.6: FA - Dados I. Curvas ROC obtidas usando-se os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.

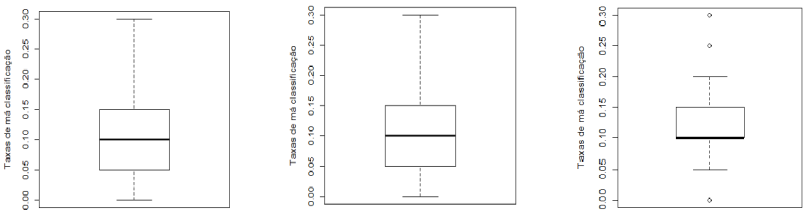
5.1.3 Replicações

Breiman (1996) relata que um procedimento de regularização é instável quando uma pequena alteração nos dados $\mathcal{L}$ pode ocasionar grandes mudanças na sequência regularizadora $(\hat{\mu}(\cdot, \mathcal{L}))$. O autor esclarece que certos procedimentos instáveis dificultam a busca do melhor modelo. Adicionalmente, a magnitude da perda preditiva pode ser uma fração substancial do erro de predição. O autor ainda afirma que os melhores métodos combinam estabilidade com uma melhor gama de ajustes.

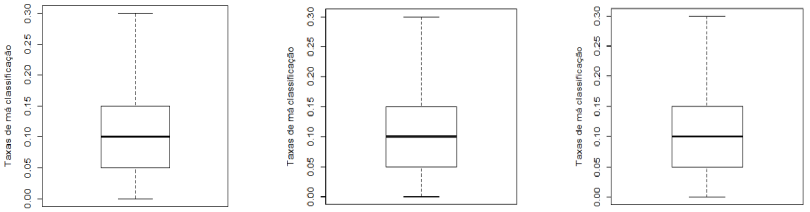
A replicação de execução ajuda na estabilização de procedimentos instáveis, uma vez que os preditores médios são uma sequência mais estável, com menor perda preditiva, e estimativas menos tendenciosas (Breiman, 1996 apud Bill, 2015). Isso pode ser visto claramente em um estudo empírico feito por Bill (2015, pág. 88).

Bill (2015) realizou testes empíricos em dois conjuntos de dados usando duas técnicas (RDA e SVM) para verificar qual seria o número ideal de replicações. O autor mostrou que os erros médios de teste com replicações mais numerosas são mais confiáveis para prever o erro de generalização.

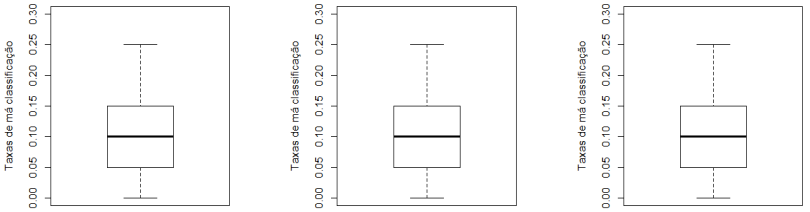
Baseando-se no estudo realizado por Bill (2015), utilizou-se, como padrão, 1000 réplicas. Desta forma, os conjuntos de dados foram divididos 1000 vezes em conjunto de treinamento ($\frac{2}{3}$ dos dados) e teste ($\frac{1}{3}$ dos dados) para a obtenção dos erros de teste médios. Ao final, foram gerados box-plots para as taxas de má classificação e os erros quadráticos médios, para medir o desempenho do classificador. Nesta Seção apresentam-se todos os box-plots obtidos em cada uma das técnicas de classificação.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.7: RDA - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

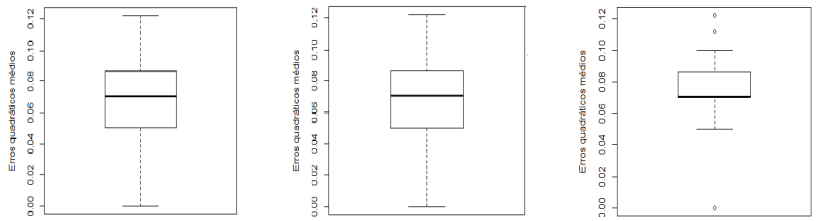
Análise Discriminante Regularizada (RDA)

Considerando a RDA, as Figuras 5.7 e 5.8 ilustram as taxas de má classificação e os erros quadráticos médios obtidos após as replicações (R=1000). Observa-se que as taxas de má classificação, para os três tipos de validação cruzada (SS, SCV e LOOCV) e K-Folds (4, 5 3 10), foram, aproximadamente, 0,10 (10%) e os erros quadráticos médios, em torno de 0,070.

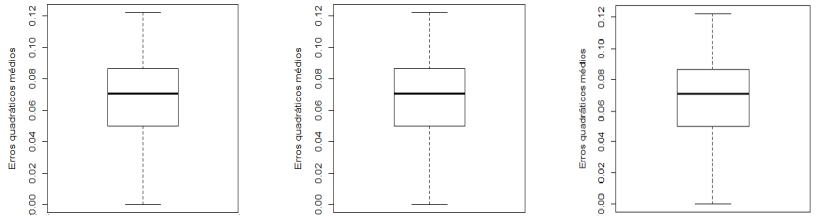
Na Tabela 5.1 obteve-se o menor erro de teste na validação cruzada SCV, com k=5. Inspecionando-se as Figuras 5.7 e 5.8, pode-se dizer que não há objeções quanto ao resultado apresentado na Tabela 5.1, isto é, a técnica RDA aparentemente conduz, realmente, ao melhor desempenho classificatório na validação cruzada SCV com $k = 5$.

Máquina de Vetor de Suporte (SVM)

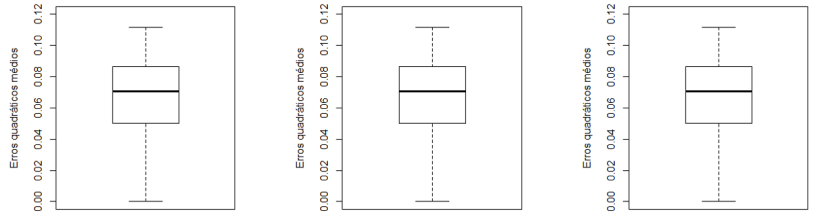
Uma vez que na Tabela 5.2 verificamos que o kernel sigmoid apresentou o menor AVTE (0,1246) nas técnicas SS e SCV com $k = 5$, este fato também pode ser verificado nas Figuras 5.9 e 5.10, observa-se taxas de má classificação, em torno de 0,10 (10%) e erros quadráticos médios em torno de 0.07. Em comparação com os demais resultados obtidos na SVM, em que os kernels linear, polynomial e radial foram usados (vide Apêndice B.2), estes foram os menores valores encontrados para as taxas de má classificação e os EQMs. Logo, os resultados apontam que a SVM com o kernel sigmoid obteve o melhor desempenho classificador em comparação com os demais tipos de kernels. Além disso, nota-se que, aparentemente, a SVM apresenta menor variância.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

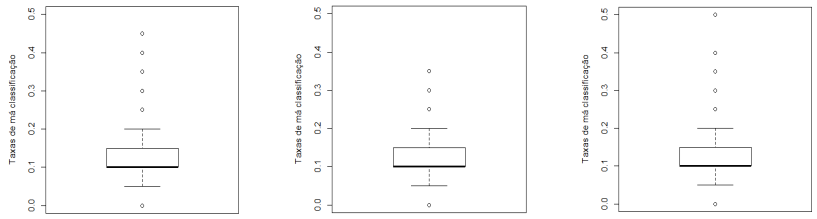


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

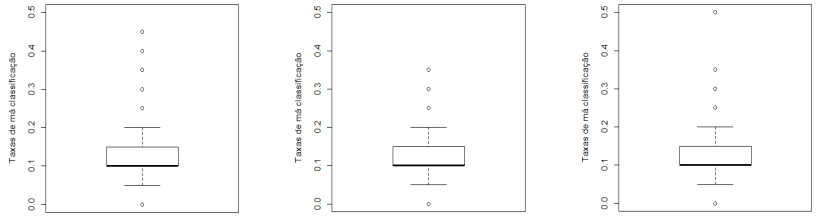


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

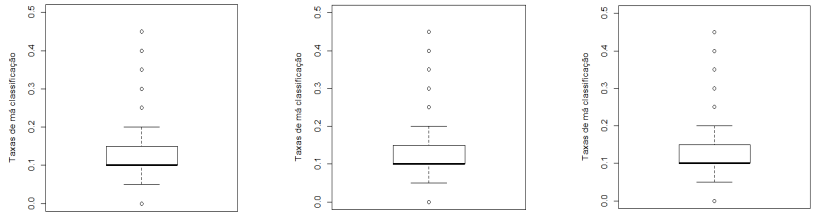
Figura 5.8: RDA - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

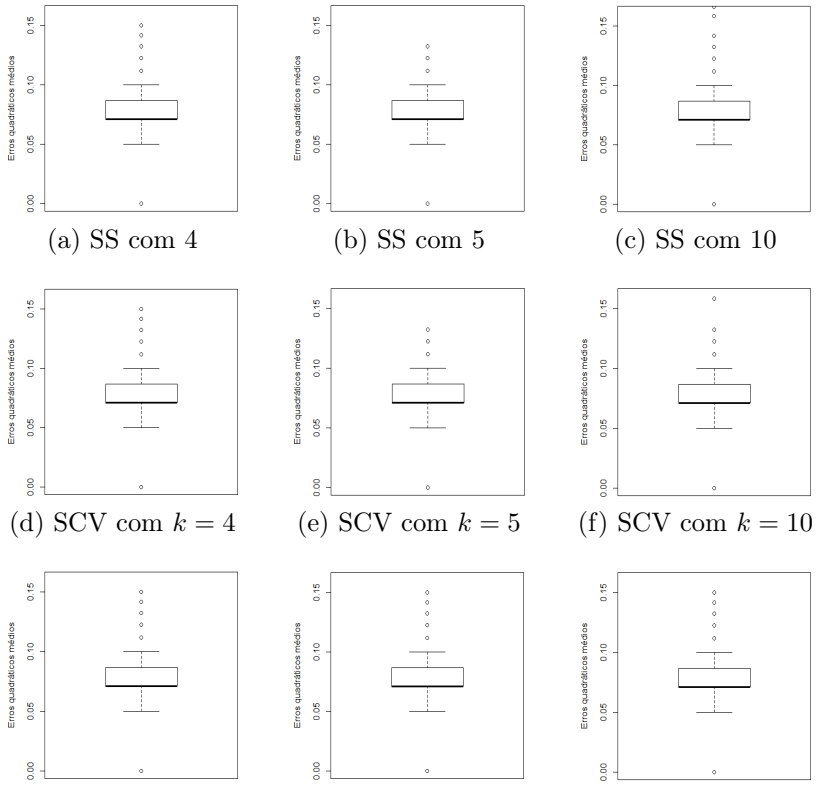


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.9: SVM (kernel: Sigmoid) - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com 4 (b) SS com 5 (c) SS com 10

(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

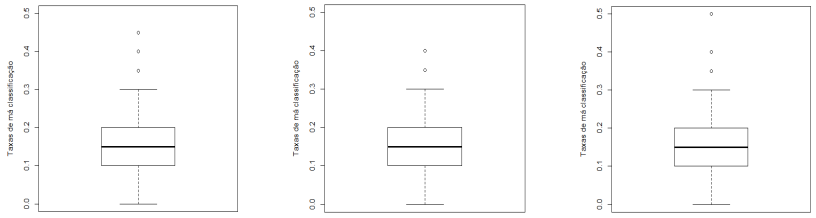
(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.10: SVM (kernel: Sigmoid) - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

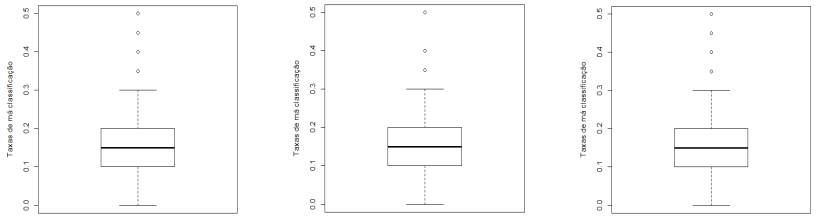
Florestas Aleatórias (FA)

As Figuras 5.11 e 5.12 referem-se às taxas de má classificação e aos EQMs obtidos utilizando a técnica de Florestas Aleatórias. As figuras mostram que as taxas de má classificação e os EQMs foram iguais a 0,15 (15%) e 0,086.

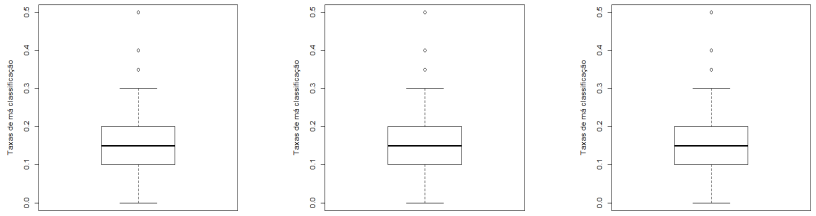
Os resultados da Tabela 5.3 mostram que o menor AVTE encontrado foi na técnica SCV com $k = 10$. Nas curvas ROCs da Figura 5.6, observa-se que a validação cruzada SCV, com $k = 4$, foi a que apresentou o melhor poder de classificação. As Figuras 5.11 e 5.12 não oferecem informações que auxiliem conclusões adicionais acerca dos tipos de validação cruzada. No entanto, escolheremos a técnica SCV com $k = 10$, cujo classificador obteve o melhor desempenho, devido ao menor valor encontrado para o AVTE.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

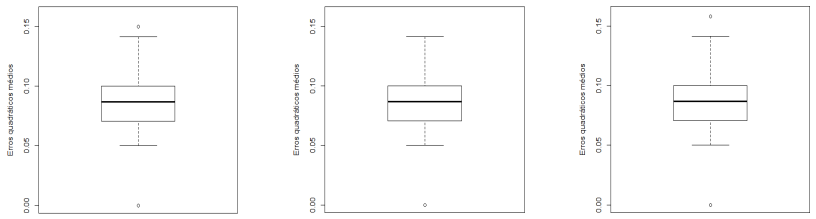


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

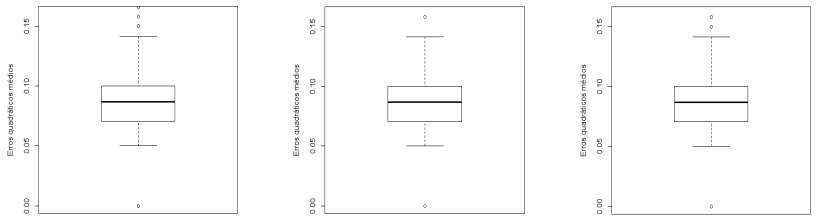


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

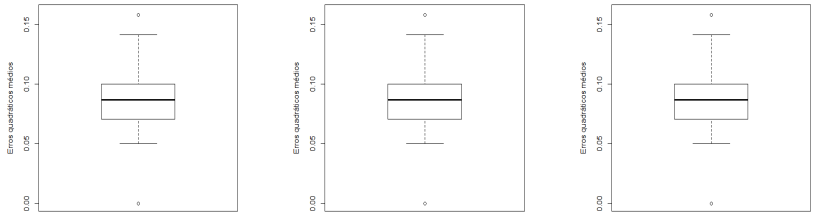
Figura 5.11: FA - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.12: FA - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

Conclusão

A Tabela 5.4 apresenta um resumo das taxas médias de má classificação e dos EQM usando-se os Dados I. Nas linhas tem-se os tipos de validação cruzada (VC), que se destacaram por fornecer o menor AVTE, as taxas médias de má classificação e o EQMs. Nas colunas tem-se os três métodos de classificação: RDA, SVM (com os quatro tipos de kernels) e FA.

Tabela 5.4: Valores das taxas médias de má classificação (%) e dos EQMs - Dados I

	RDA	SVM				FA
		Linear	Polynomial	Sigmoid	Radial	
CV k-Fold	SCV $k = 5$	SCV e SS $k = 5$	SCV e SS $k = 5$	SCV e SS $k = 5$	SS $k = 5$	SCV $k = 10$
Taxa média de má classificação (%)	10	12	34	12	19	14
EQM	0,070	0,075	0,129	0,074	0,097	0,081

Observa-se na Tabela 5.4, que a RDA foi o método que melhor classificou os dados, pois apresentou os menores valores para o EQM e para a taxa média de má classificação. Na Seção 5.1.2, ao analisar as curvas ROC, a RDA também destacou-se por fornecer o melhor desempenho classificador em relação às demais técnicas.

5.2 Dados II - Dados de Voz

O banco de dados voice gender data (Dados II), disponível em: https://www.kaggle.com/primaryobjects/voicegender, consiste em 3.168 amostras de voz gravadas, coletadas de falantes masculinos e femininos. Um total de 21 propriedades acústicas de cada voz foram medidas.

O conjunto de dados foi criado para identificar uma voz como sendo homem ou mulher, com base nas propriedades acústicas da voz e da fala. As amostras de voz são pré-processadas por análise acústica em R usando os pacotes seewave e tuneR, com uma faixa de freqüência analisada de 0hz-280hz. Os dados apresentam as seguintes propriedades acústicas de cada voz:

- meanfreq: frequência média (em kHz),
- sd: desvio padrão da frequência,
- median: mediana da frequência (em kHz),
- Q25: primeiro quantil (em kHz),
- Q75: terceiro quantil (em kHz),
- IQR: intervalo interquantil (em kHz),
- skew: skewness,
- kurt: curtose,
- sp.ent: entropia espectral,
- sfm: nivelamento espectral,
- mode: frequência de modo,
- centroid: centróide da frequência,
- meanfun: média da frequência fundamental medida através do sinal acústico,
- minfun: frequência fundamental mínima medida através do sinal acústico,
- maxfun: frequência fundamental máxima medida através do sinal acústico,
- meandom: média da frequência dominante medida através do sinal acústico,
- mindom: frequência dominante mínima medido através do sinal acústico,
- maxdom: frequência dominante máxima medida através do sinal acústico,
- dfrange: faixa de freqüência dominante medida através do sinal acústico,

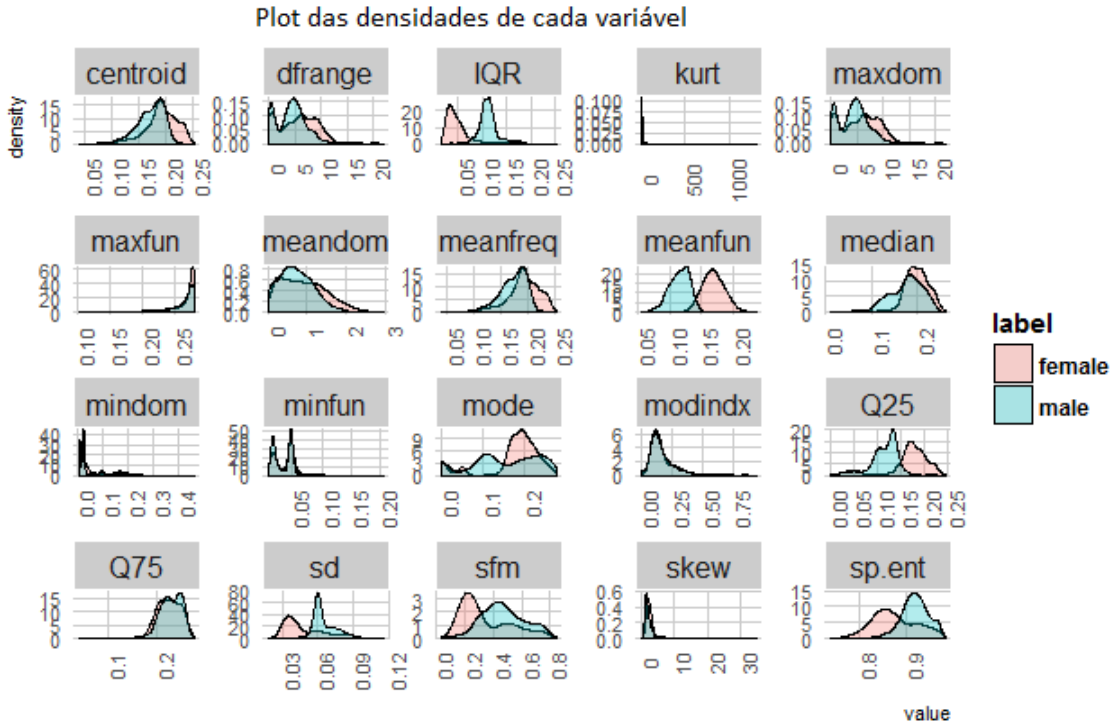


Figura 5.13: Plot da densidade de cada uma das 20 variáveis. Fonte: https://www.kaggle.com/adhok93/feature-importance-and-pca/.

• modindx: índice de modulação,
• label: masculino ou feminino.

O conjunto de dados de voz é composto de um total de 3.168 observações, sendo metade dos registros de pessoas do sexo masculino e metade do sexo feminino, e perfazendo um total de 20 propriedades acústicas das vozes (características), referentes a esses indivíduos.

A Figura 5.13 mostra os gráficos de densidade para cada uma das 20 variáveis separadas por gênero. Nota-se que há um pouco de sobreposição entre os valores dos dados classificados como masculino e feminino. Os gráficos que identificam as características masculinas e femininas distintamente são: IQR, meanfun, mode, Q25, sd, sfm e sp.ent.

Ao contrário do banco de dados anterior, este banco não faz parte do caso em que $n \lll p$. O objetivo é classificar os indivíduos, segundo o gênero, de acordo com as propriedades acústicas de suas vozes, utilizando as técnicas de classificação estudadas.

5.2.1 Parâmetros de ajuste ótimos e Erro Médio de Teste

Nesta subseção apresentaremos a aplicação dos dados de voz aos métodos de classificação RDA, SVM e RF. Os parâmetros de ajuste ótimos, de cada método de classificação foram encontrados após uma análise intensiva de validação cruzada, em que as técnicas SS, SCV e LOOCV foram testadas com tamanhos distintos de k-Folds. Os parâmetros ideais foram escolhidos com base no menor erro de teste. Inicialmente faremos análises sem proceder à padronização nas covariáveis (Seções 5.2.1 a 5.2.4) para, então, refazer as análises com padronização prévia das covariáveis.

Análise Discriminante Regularizada - sem padronização

A Tabela 5.5 apresenta os hiper-parâmetros ótimos ajustados utilizando a RDA nos dados de voz originais (sem padronização). Nota-se que os hiper-parâmetros ótimos α e δ foram iguais

a $0,99$ e 1, respectivamente, em todos os casos simulados. O menor valor para o AVTE foi encontrado com a técnica SCV, com $k = 4$ (AVTE $= 0,07041$).

Tabela 5.5: RDA - Dados II originais - Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação RDA, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10).

VC	K-Folds	α	δ	AVTE
SS	4	0,99	1,0	0,07124
	5	0,99	1,0	0,07136
	10	0,99	1,0	0,07145
SCV	4	0,99	1,0	0,07041
	5	0,99	1,0	0,07136
	10	0,99	1,0	0,07152
LOOCV	4	0,99	1,0	0,07172
	5	0,99	1,0	0,07172
	10	0,99	1,0	0,07172

Máquina de Vetor de Suporte - sem padronização

A Tabela 5.6 mostra os resultados da aplicação da técnica SVM aos dados de voz. Este conjunto de dados apresentou alguns problemas de convergência na SVM com kernel Linear uma vez que, para determinados valores de *custo*, o algoritmo não convergiu. A Tabela 5.6 exibe resultados SVM para valores de *custo* para os quais o algoritmo convergiu.

Nos dados de voz, assim como nos dados de câncer, o parâmetro *custo* foi utilizado em todos os tipos de kernels. O parâmetro γ foi utilizado para todos os tipos de kernels, exceto o linear. O parâmetro *degree*, necessário para o kernel do tipo polynomial, foi utilizado adotando-se $degree = 1$. Para o parâmetro *coef0*, necessário para os tipos de kernel polynomial e sigmoid, adotou-se $coef0 = 0$.

Nota-se que na Tabela 5.6 (a) (Kernel Linear), tem-se o único caso em que a técnica de validação cruzada LOOCV apresentou o menor erro médio de teste ($AVTE = 0,02570$). Verifica-se que o kernel radial com validação cruzada SS ($k = 4$) foi o que apresentou o menor erro médio de teste ($AVTE = 0,02180$), ou seja, no conjunto de dados de voz o kernel radial apresentou o melhor desempenho classificador dentre os demais tipos de kernel.

Florestas Aleatórias - sem padronização

Os resultados para a técnica FA estão dispostos na Tabela 5.7. Os parâmetros $ntree = 280$ e $nodesize = 2$ foram escolhidos para todas as técnicas de VC, dentre alguns valores testados, tais como $ntree = (100, 280, 500)$ e $nodesize = (2, 5, 7, 14, 20)$. O menor erro de teste ($TE = 0,02083$) foi encontrado na SCV com $k = 5$.

Observa-se na Tabela 5.7 que os valores encontrados para o AVTE são próximos aos melhores valores encontrados para a SVM na Tabela 5.6 (d) (Kernel Radial).

5.2.2 Curva ROC

A análise ROC foi introduzida na Aprendizagem de Máquinas como uma ferramenta poderosa para a avaliação de modelos de classificação. Este trabalho utilizou a análise ROC com o objetivo de prover uma avaliação mais rica, ao invés de apenas avaliar o modelo de classificação a partir de uma única medida. Os gráficos permitem uma melhor visualização do problema de avaliação.

Tabela 5.6: SVM - Dados II originais. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). A técnica de SVM foi aplicada testando-se quatro tipos de kernels: linear, polynomial, sigmoid e radial.

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	0,1	0,02598
	5	0,1	0,02614
	10	0,1	0,02599
SCV	4	0,1	0,02623
	5	0,1	0,02614
	10	0,1	0,02621
LOOCV	4	0,5	0,02570
	5	0,5	0,02570
	10	0,5	0,02570

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	1	0,04477
	5	0,05	1	0,04491
	10	0,05	1	0,04463
SCV	4	0,05	1	0,04497
	5	0,05	1	0,04491
	10	0,05	1	0,04463
LOOCV	4	0,05	1	0,04482
	5	0,05	1	0,04482
	10	0,05	1	0,04482

(c) Kernel Sigmoid

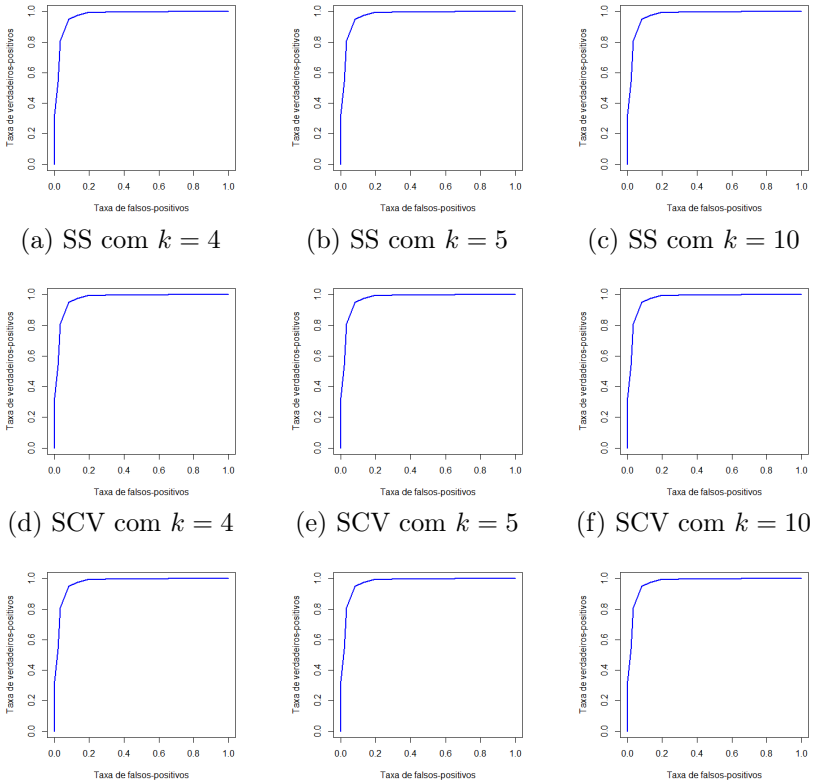
VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	0,01	0,10322
	5	0,05	0,01	0,10368
	10	0,05	0,01	0,10346
SCV	4	0,05	0,01	0,10340
	5	0,05	0,01	0,10368
	10	0,05	0,01	0,10346
LOOCV	4	0,05	0,01	0,10323
	5	0,05	0,01	0,10323
	10	0,05	0,01	0,10323

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	0,5	0,02180
	5	0,05	0,5	0,02191
	10	0,05	0,5	0,02198
SCV	4	0,05	0,5	0,02200
	5	0,05	0,5	0,02191
	10	0,05	0,5	0,02198
LOOCV	4	0,05	0,5	0,02182
	5	0,05	0,5	0,02182
	10	0,05	0,5	0,02182

Tabela 5.7: Florestas Aleatórias - Dados II originais. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação RF, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10).

VC	K-Folds	Ntree	Nodesize	AVTE
SS	4	280	2	0,02125
	5	280	2	0,02134
	10	280	2	0,02111
SCV	4	280	2	0,02108
	5	280	2	0,02083
	10	280	2	0,02119
LOOCV	4	280	2	0,02112
	5	280	2	0,02112
	10	280	2	0,02112



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.14: RDA - Dados II originais. Curvas ROC usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.

Nesta subseção, serão apresentadas as curvas ROC de cada um dos métodos de classificação utilizando os dados de voz.

Análise Discriminante Regularizada (RDA)

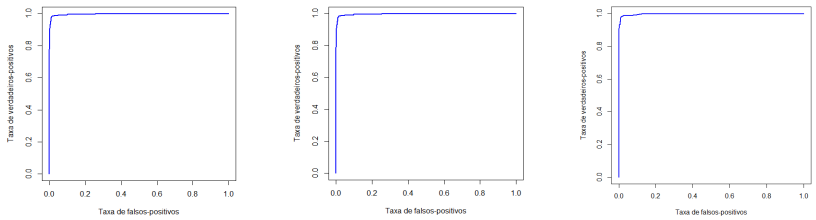
Para a RDA, a Figura 5.14 exibe as curvas ROC para cada tipo de validação cruzada e tamanhos de k-Folds. Como a curva aproxima-se do canto superior esquerdo do gráfico, pode-se dizer que o modelo teve um bom desempenho preditivo.

Máquina de Vetor de Suporte (SVM)

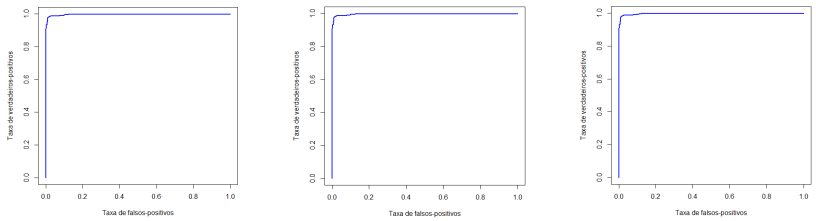
Para a SVM, uma vez que o kernel radial foi o que apresentou melhor desempenho classificador, apresenta-se a curva ROC apenas para este caso. Observa-se na Figura 5.15 que as curvas ROC estão próximas do ponto superior esquerdo $(0, 1)$ caracterizando uma ótima descrição do poder de classificação do SVM para o kernel radial. Os demais casos encontram-se no Apêndice B.3.

Florestas Aleatórias (FA)

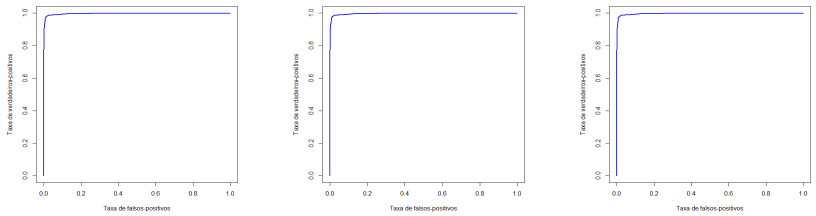
As curvas ROC obtidas através da técnica de Florestas Aleatórias estão dispostas na Figura 5.16. Assim como na SVM, os resultados gráficos ROC mostram um perfeito desempenho classificatório.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

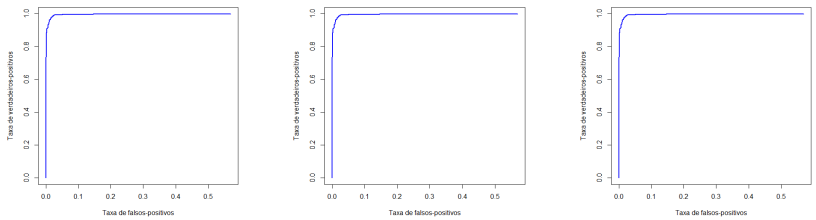


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

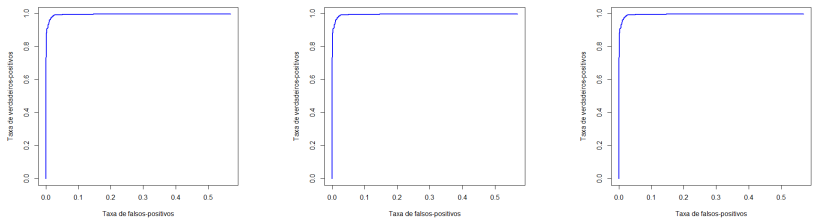


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

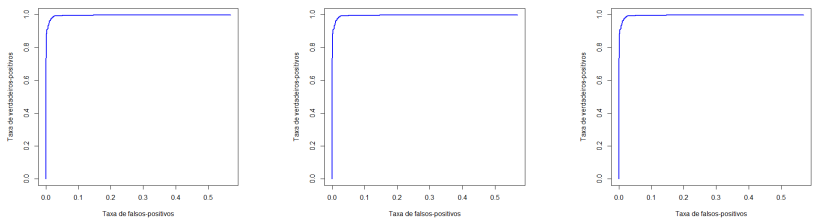
Figura 5.15: SVM (Kernel: Radial) - Dados II originais. Curvas ROC usando-se os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

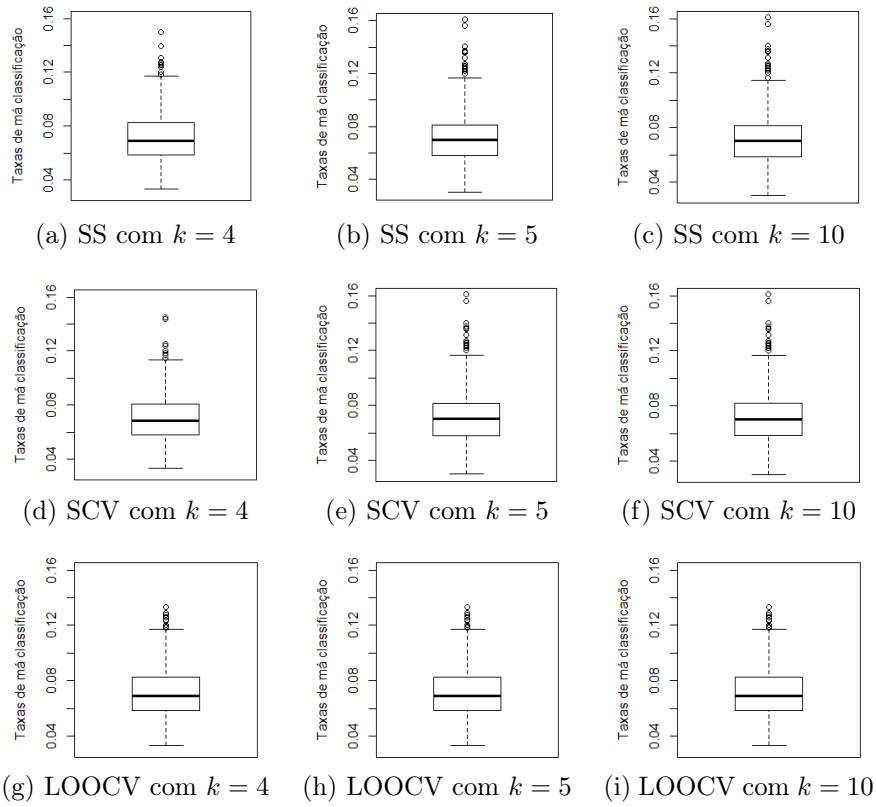


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura 5.16: Florestas Aleatórias - Dados II originais. Curvas ROC usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

Figura 5.17: RDA - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

5.2.3 Replicações

Este trabalho contou com o uso de replicações ($R = 1000$) para prever o erro de generalização. Esta subseção apresenta os box-plots gerados apresentando as taxas de má classificação e os EQMs de cada um dos métodos supervisionados de classificação.

Análise Discriminante Regularizada

Os mesmo processo de replicação ($R = 1000$), discutido na Seção 5.1.3, foi realizado utilizando o banco de voz. As Figuras 5.17 e 5.18 apresentam as taxas de má classificação em torno de 7% e os erros quadráticos médios em torno de 0,08, para o método RDA.

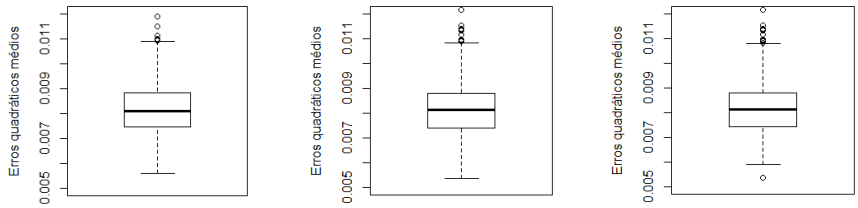
Máquina de Vetor de Suporte

As Figuras 5.19 e 5.20 descrevem os box-plots das taxas de má classificação e dos erros quadráticos médios para o método SVM (kernel Radial). No geral, as taxas de má classificação variam em torno de 0,0217 (2,17%) e 0,0045.

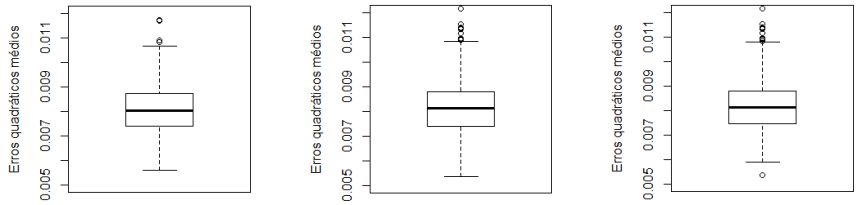
Na Tabela 5.6 tem-se o menor erro de teste para o SVM (Kernel Radial) na validação cruzada SS com $k = 4$. Este resultado também pode ser confirmado nas figuras. Os box-plots para os demais tipos de kernel, são encontrados no Apêndice B.4.

Florestas Aleatórias

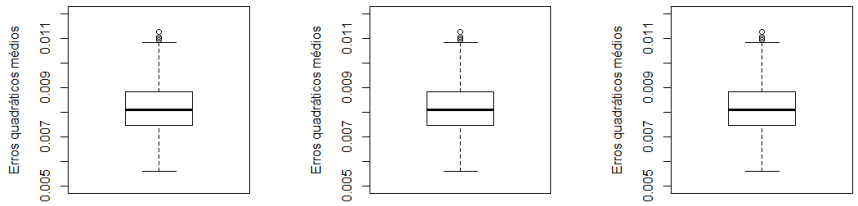
Os gráficos da técnica FA são apresentados nas Figuras 5.21 e 5.22. Percebe-se que as taxas de má classificação e os erros quadráticos médios foram iguais a aproximadamente 0.0208 (2,08%) e 0,0044, respectivamente. Observa-se que estes resultados são próximos os obtidos em SVM (kernel radial).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

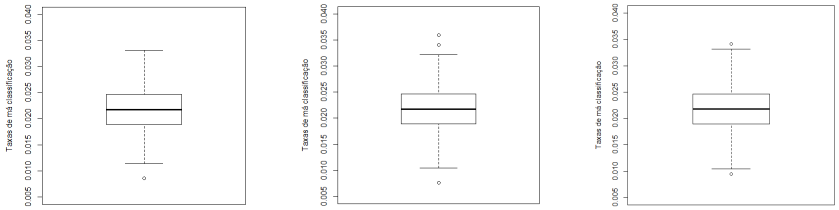


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

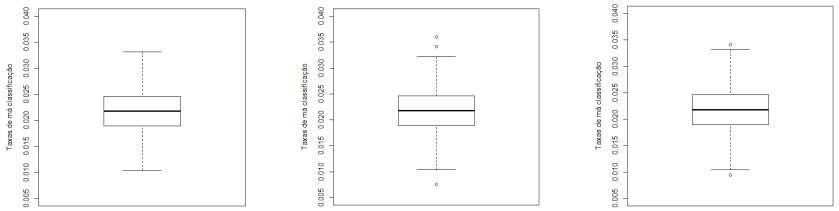


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

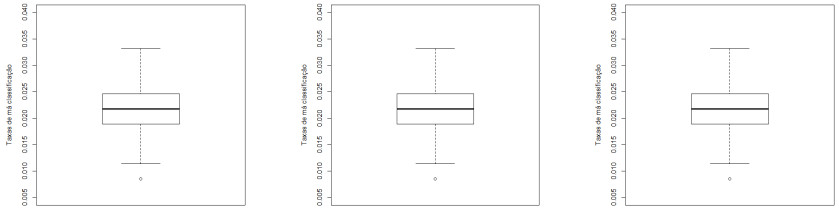
Figura 5.18: RDA - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

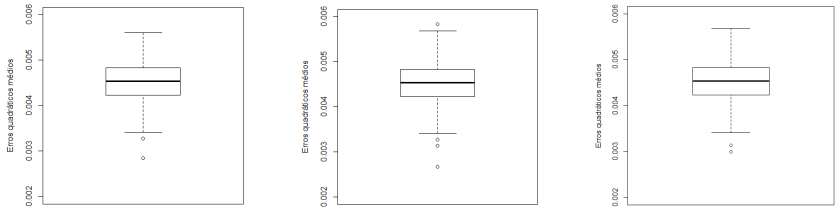


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

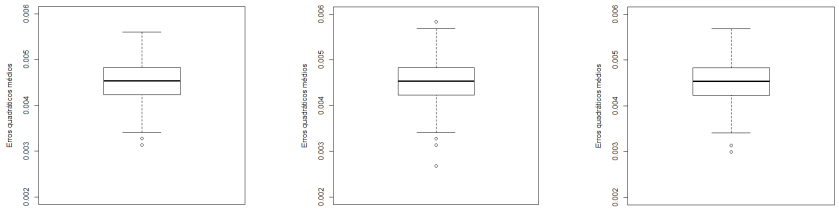


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

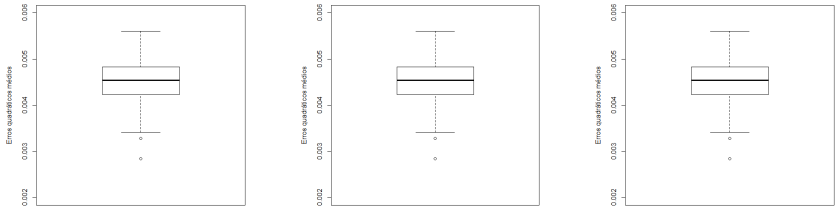
Figura 5.19: SVM (Kernel Radial) - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

Figura 5.20: SVM (Kernel Radial) - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

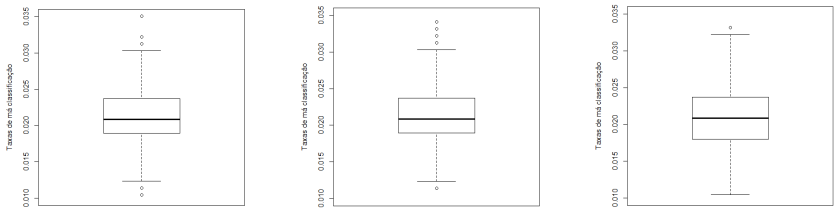
Nota-se por meio das figuras 5.21 e 5.22, que a técnica SCV com $k = 5$, aparentemente proporciona o melhor desempenho em relação às demais. Tal percepção visual pode ser confirmada observando-se os resultados da Tabela 5.7, que apontam o menor erro de teste na validação cruzada SCV com $k = 5$.

Como a FA e a SVM (kernel radial) apresentaram os menores valores em relação à RDA, pode-se dizer que com a SVM e a FA obteve-se o melhor desempenho classificatório dos dados.

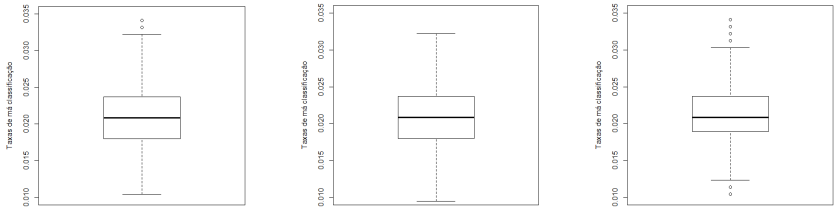
5.2.4 Modelo de Regressão Logística na Classificação

Conforme foi mencionado anteriormente (vide pg. 76), o algoritmo SVM (kernel Linear) não convergiu, devido a problemas de singularidade em matrizes utilizadas nos cálculos, para alguns valores de custo, tal como *custo* = 100. O uso da regressão logística deu-se na tentativa de amenizar tal problema de convergência, uma vez que tentou-se selecionar as variáveis mais importantes na análise, mas excluindo-se variáveis redundantes.

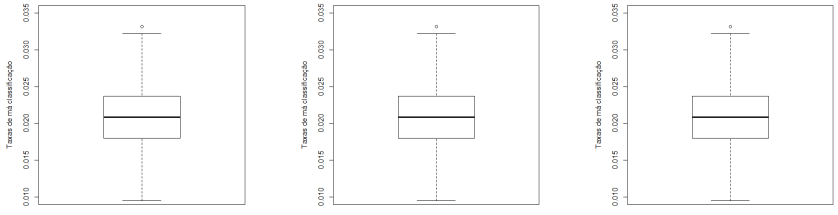
A Tabela abaixo faz uma comparação entre os resultados obtidos por meio da técnica SVM (kernel Linear) aplicada sem a regressão logística e, posteriormente, a técnica aplicada com a regressão logística usada para a seleção de covariáveis. Os resultados mostram que o pré-processamento realizado utilizando-se a regressão logística resolveu o problema da falta de convergência encontrado no kernel linear com *custo* = 100. As covariáveis inclusas na análise, por apresentarem significância (p-valor $\leq \alpha = 0,05$), foram: Q25, Q75, sp.ent, sfm, meanfun, minfun e nodindix. O código em `R` do método SVM, com a implementação da significância estatística via Regressão Logística, encontra-se no Apêndice C.2.1.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

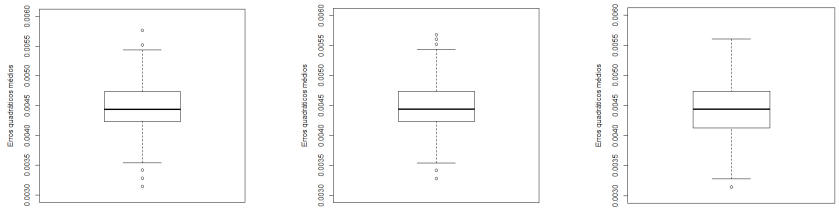


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

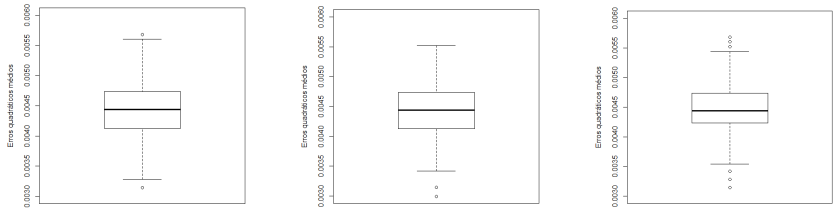


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

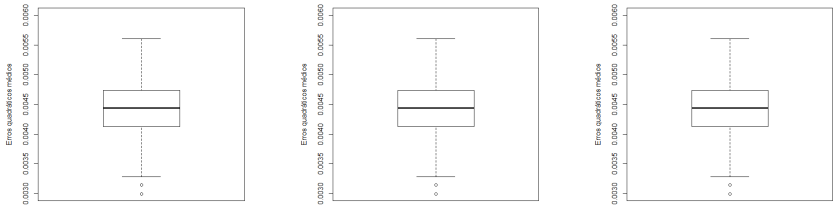
Figura 5.21: Florestas Aleatórias - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com k=10

Figura 5.22: Florestas Aleatórias - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

Tabela 5.8: SVM - Dados II originais. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM (kernel Linear e $custo = 100$), juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). As tabelas apresentam os resultados da seguinte forma: (a) SVM sem regressão logística, (b) SVM com regressão logística.

(a) SVM sem Regressão Logística

VC	K-Folds	Custo	AVTE
SS	4	100	NA
	5	100	NA
	10	100	NA
SCV	4	100	NA
	5	100	NA
	10	100	NA
LOOCV	4	100	NA
	5	100	NA
	10	100	NA

(b) SVM com Regressão Logística

VC	K-Folds	Custo	AVTE
SS	4	100	0,02520
	5	100	0,02527
	10	100	0,02525
SCV	4	100	0,02538
	5	100	0,02527
	10	100	0,02525
LOOCV	4	100	0,02527
	5	100	0,02527
	10	100	0,02527

A Tabela 5.9 apresenta os resultados do uso da regressão logística para a seleção de covariáveis a serem usadas na SVM utilizando os quatro tipos de kernels. Ao realizar uma comparação com os resultados da Tabela 5.6, observa-se que os resultados utilizando a regressão logística apresentaram uma diferença mais perceptível no kernel sigmoid, não diferindo muito nos demais resultados.

Tabela 5.9: SVM - Dados II originais - Regressão Logística. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). Aplicou-se a regressão logística testando-se quatro tipos de kernels: (a) linear, (b) polynomial, (c) sigmoid e (d) radial.

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	0,01	0,03135
	5	0,01	0,03137
	10	0,01	0,03117
SCV	4	0,01	0,03136
	5	0,01	0,03137
	10	0,01	0,03117
LOOCV	4	0,01	0,03137
	5	0,01	0,03137
	10	0,01	0,03137

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1428571	0,1	0,06704
	5	0,1428571	0,1	0,06742
	10	0,1428571	0,1	0,06717
SCV	4	0,1428571	0,1	0,06736
	5	0,1428571	0,1	0,06742
	10	0,1428571	0,1	0,06717
LOOCV	4	0,1428571	0,1	0,06708
	5	0,1428571	0,1	0,06708
	10	0,1428571	0,1	0,06708

(c) Kernel Sigmoid

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1428571	0,01	0,05904
	5	0,1428571	0,01	0,05958
	10	0,1428571	0,01	0,05910
SCV	4	0,1428571	0,01	0,05969
	5	0,1428571	0,01	0,05958
	10	0,1428571	0,01	0,05910
LOOCV	4	0,1428571	0,01	0,05897
	5	0,1428571	0,01	0,05897
	10	0,1428571	0,01	0,05897

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1428571	10	0,01941
	5	0,1428571	10	0,01922
	10	0,1428571	10	0,01946
SCV	4	0,1428571	10	0,01949
	5	0,1428571	10	0,01922
	10	0,1428571	10	0,01946
LOOCV	4	0,1428571	10	0,01943
	5	0,1428571	10	0,01943
	10	0,1428571	10	0,01943

5.2.5 Padronização dos dados

Uma das principais tarefas no pré-processamento de dados é normalizá-los, padronizá-los, ou realizar alguma transformação nos dados para colocar as variáveis na mesma escala de medição para uma comparação justa entre elas.

Tabela 5.10: SVM - Dados II padronizados - *custo* = 100. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM (kernel linear e *custo* = 100), juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). As tabelas apresentam os resultados dos dados padronizados: (a) SVM sem regressão logística, (b) SVM com regressão logística.

(a) SVM sem Regressão Logística

VC	K-Folds	Custo	AVTE
SS	4	100	NA
	5	100	NA
	10	100	NA
SCV	4	100	NA
	5	100	NA
	10	100	NA
LOOCV	4	100	NA
	5	100	NA
	10	100	NA

(b) SVM com Regressão Logística

VC	K-Folds	Custo	AVTE
SS	4	100	0,3855
	5	100	0,3845
	10	100	0,3850
SCV	4	100	0,3844
	5	100	0,3845
	10	100	0,3850
LOOCV	4	100	0,3854
	5	100	0,3854
	10	100	0,3854

Bill (2015) afirma que a padronização nos dados de cólon fez com que o desempenho de previsão piorasse. Isso pode ser justificado pelo fato de que os dados já haviam sido submetidos a várias transformações, impactando, negativamente, o desempenho preditivo.

Nesta pesquisa, testou-se a padronização nos dados de voz com o objetivo de verificar se esta implicaria em uma solução para o problema de convergência apresentado para alguns valores de *custo* na SVM (kernel linear).

A Tabela 5.10 mostra os resultados do algoritmo SVM (kernel linear e *custo* = 100), com os dados padronizados, antes e depois de aplicar a Regressão Logística.

Observa-se que somente o uso da padronização dos dados de voz não resolve o problema da falta de convergência. Além disso, mesmo aplicada juntamente com o uso da regressão logística (para a seleção de covariáveis), não proporciona melhora no desempenho preditivo, o que pode ser observado comparando-se os resultados das Tabelas 5.8 e 5.10.

O dados padronizados também foram aplicados para os demais tipos de kernel, conforme os resultados da Tabela 5.11.

Tabela 5.11: SVM - Dados II padronizados. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). A técnica de SVM foi aplicada nos dados padronizados testando-se quatro tipos de kernels: (a) linear, (b) polynomial, (c) sigmoid e (d) radial. .

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	10	0,11326
	5	10	0,11357
	10	10	0,11343
SCV	4	10	0,11319
	5	10	0,11357
	10	10	0,11343
LOOCV	4	10	0,11318
	5	10	0,11318
	10	10	0,11318

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	100	0,12615
	5	0,05	100	0,12652
	10	0,05	100	0,12598
SCV	4	0,05	100	0,12614
	5	0,05	100	0,12652
	10	0,05	100	0,12598
LOOCV	4	0,05	100	0,12622
	5	0,05	100	0,12622
	10	0,05	100	0,12622

(c) Kernel Sigmoid

VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	0,01	0,49487
	5	0,05	0,01	0,49504
	10	0,05	0,01	0,49483
SCV	4	0,05	0,01	0,49276
	5	0,05	0,01	0,49504
	10	0,05	0,01	0,49483
LOOCV	4	0,05	0,01	0,49430
	5	0,05	0,01	0,49430
	10	0,05	0,01	0,49430

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,05	100	0,08309
	5	0,05	100	0,08322
	10	0,05	100	0,08291
SCV	4	0,05	100	0,08305
	5	0,05	100	0,08322
	10	0,05	100	0,08291
LOOCV	4	0,05	100	0,08297
	5	0,05	100	0,08297
	10	0,05	100	0,08297

Na Tabela 5.11 observa-se que os custos que conduziram aos valores mínimos de AVTE são distintos daqueles observados na Tabela 5.6. No entanto, os valores de AVTE observados nesta última tabela são maiores do que os da Tabela 5.6, implicando que a padronização é desnecessária para o banco de voz.

Na Tabela 5.12 apresentam-se os resultados utilizando os dados padronizados, posteriormente, procedeu-se à seleção de covariáveis através do uso de Regressão Logística, para fins de comparação. Percebe-se que a padronização não trouxe melhorias com o uso da Regressão Logística. Bill (2015) justifica que realmente as técnicas de kernelização padronizam automaticamente os dados, não havendo necessidade de utilizar tal ferramenta.

Tabela 5.12: SVM - Dados II padronizados e Regressão Logística. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) obtidos usando-se a Regressão Logística nos dados II padronizados.

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	0,1	0,40609
	5	0,1	0,40475
	10	0,1	0,40638
SCV	4	0,1	0,40472
	5	0,1	0,40475
	10	0,1	0,40638
LOOCV	4	0,1	0,40588
	5	0,1	0,40588
	10	0,1	0,40588

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0.1428571	0,1	0,23010
	5	0.1428571	0,1	0,22925
	10	0.1428571	0,1	0,22939
SCV	4	0.1428571	0,1	0,22965
	5	0.1428571	0,1	0,22925
	10	0.1428571	0,1	0,22939
LOOCV	4	0.1428571	0,1	0,22990
	5	0.1428571	0,1	0,22990
	10	0.1428571	0,1	0,22990

(c) Kernel Sigmoid

VC	K-Folds	γ	Custo	AVTE
SS	4	0.1428571	0,01	0,40987
	5	0.1428571	0,01	0,40810
	10	0.1428571	0,01	0,40773
SCV	4	0.1428571	0,01	0,40747
	5	0.1428571	0,01	0,40810
	10	0.1428571	0,01	0,40773
LOOCV	4	0.1428571	0,01	0,40896
	5	0.1428571	0,01	0,40896
	10	0.1428571	0,01	0,40896

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0.1428571	100	0,04049
	5	0.1428571	100	0,04042
	10	0.1428571	100	0,04040
SCV	4	0.1428571	100	0,04061
	5	0.1428571	100	0,04042
	10	0.1428571	100	0,04040
LOOCV	4	0.1428571	100	0,04047
	5	0.1428571	100	0,04047
	10	0.1428571	100	0,04047

5.3 Avaliando a Classificação para algumas covariáveis selecionadas

Na Seção 4.2, a Figura 5.13 descreveu a densidade de cada uma das variáveis, separadas por gênero, pertencentes ao conjunto de dados de voz. Observa-se que, as variáveis que visualmente apresentaram um melhor poder de discriminação, devido às distribuições descritas para homens e mulheres serem bem separadas, foram: IQR, meanfun, sd, sp.ent, Q25, sfm e mode. As tabelas a seguir, apresentam os parâmetros de ajuste ótimos e os AVTEs referentes a aplicação das técnicas RDA, SVM e RF utilizando essa seleção de variáveis.

Objetiva-se comparar o poder de classificação dos métodos RDA, SVM e RF ao utilizar-se, nas análises, apenas essas variáveis, com os resultados obtidos usando os dados originais (Tabela 5.6), cujos resultados apresentam os menores valores de AVTE.

Análise Discriminante Regularizada

Na Tabela 5.13 apresentam-se os hiper-parâmetros ótimos ajustados utilizando a RDA. Nota-se que, assim como na Tabela 5.5, os hiper-parâmetros ótimos α e δ foram iguais a $0,99$ e 1, respectivamente, em todos os casos simulados.

Observa-se na Tabela 5.13, que o menor AVTE obtido foi em torno de $0,089$ (na Tabela 5.5, o menor AVTE foi $0,07041$). Dessa forma, analisando estes resultados, constata-se que para a RDA, o uso das covariáveis mais discriminantes, apenas, não conduz a resultados mais satisfatórios. No entanto, ao utilizar esse modelo bem mais simples, o desempenho não piora.

Tabela 5.13: RDA - Dados II originais e variáveis selecionadas. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação RDA, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). Nesta tabela observam-se os resultados utilizando apenas as variáveis: IQR, meanfun, sd, sp.ent, Q25, sfm e mode.

VC	K-Folds	α	δ	AVTE
SS	4	0,99	1,0	0,08943
	5	0,99	1,0	0,08943
	10	0,99	1,0	0,08943
SCV	4	0,99	1,0	0,08943
	5	0,99	1,0	0,08957
	10	0,99	1,0	0,09004
LOOCV	4	0,99	1,0	0,08900
	5	0,99	1,0	0,08900
	10	0,99	1,0	0,08900

Máquina de Vetor de Suporte

Para a SVM, na Tabela 5.6 verificou-se que o kernel radial proporcionou o menor erro médio de teste (AVTE $= 0,02180$). A Tabela 5.14 exibe os resultados utilizando-se apenas as variáveis que apresentaram um melhor poder discriminatório. Dentre os resultados obtidos, verifica-se, novamente, que o kernel radial conduz ao menor erro médio de teste (AVTE $= 0,01922$). Estes resultados também coincidem com os da Tabela 5.9.

Tabela 5.14: SVM - Dados II originais e variáveis selecionadas. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação SVM, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). Nesta tabela observam-se os resultados utilizando apenas as variáveis: IQR, meanfun, sd, sp.ent, Q25, sfm e mode.

(a) Kernel Linear

VC	K-Folds	Custo	AVTE
SS	4	0,1	0,02859
	5	0,1	0,02868
	10	0,1	0,02863
SCV	4	0,1	0,02878
	5	0,1	0,02868
	10	0,1	0,02863
LOOCV	4	0,5	0,02858
	5	0,5	0,02858
	10	0,5	0,02858

(b) Kernel Polynomial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1666667	0,5	0,05608
	5	0,1666667	0,5	0,05664
	10	0,1666667	0,5	0,05609
SCV	4	0,1666667	0,5	0,05622
	5	0,1666667	0,5	0,05664
	10	0,1666667	0,5	0,05609
LOOCV	4	0,1666667	0,5	0,05621
	5	0,1666667	0,5	0,05621
	10	0,1666667	0,5	0,05621

(c) Kernel Sigmoid

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1666667	0,01	0,08259
	5	0,1666667	0,01	0,08307
	10	0,1666667	0,01	0,08296
SCV	4	0,1666667	0,01	0,08279
	5	0,1666667	0,01	0,08307
	10	0,1666667	0,01	0,08296
LOOCV	4	0,1666667	0,01	0,08253
	5	0,1666667	0,01	0,08253
	10	0,1666667	0,01	0,08253

(d) Kernel Radial

VC	K-Folds	γ	Custo	AVTE
SS	4	0,1666667	100	0,01944
	5	0,1666667	100	0,01934
	10	0,1666667	100	0,01922
SCV	4	0,1666667	100	0,01958
	5	0,1666667	100	0,01934
	10	0,1666667	100	0,01922
LOOCV	4	0,1666667	100	0,01950
	5	0,1666667	100	0,01950
	10	0,1666667	100	0,01950

Florestas Aleatórias

Os resultados para a técnica FA dispostos na Tabela 5.7 mostraram que o menor erro de teste encontrado foi igual a $AVTE = 0,02083$. A Tabela 5.15 mostra que os resultados obtidos para os AVTEs são próximos aos encontrados na Tabela 5.7.

Tabela 5.15: FA - Dados II originais e variáveis selecionadas. Valores dos hiper-parâmetros ótimos e do erro médio de teste (AVTE) referentes ao ajuste do método de classificação FA, juntamente com as técnicas de validação cruzada (SS, SCV e LOOCV) para determinados valores de k (4, 5, e 10). Nesta tabela observam-se os resultados utilizando apenas as variáveis: IQR, meanfun, sd, sp.ent, Q25, sfm e mode.

VC	K-Folds	Ntree	Nodesize	AVTE
SS	4	280	2	0,02083
	5	280	2	0,02097
	10	280	2	0,02090
SCV	4	280	2	0,02074
	5	280	2	0,02098
	10	280	2	0,02104
LOOCV	4	280	2	0,02070
	5	280	2	0,02070
	10	280	2	0,02070

Conclusão

As análises SVM, com Kernel Linear e *custo* = 100 são apresentadas na Tabela 5.16. Os resultados são próximos dos obtidos na Tabela 5.8. Ou seja, o uso do confonto de variáveis selecionadas também resolve o problema de falta de convergência, mas os valores de AVTE são próximos aos obtidos com o uso da regressão logística.

Tabela 5.16: SVM - Dados originais - variáveis selecionadas e *custo* = 100. Valores dos AVTEs referentes ao ajuste do método SVM, com Kernel Linear e *custo* = 100, utilizando as variáveis: IQR, meanfun, sd, sp.ent, Q25, sfm e mode.

VC	K-Folds	AVTE
SS	4	0,028784
	5	0,028926
	10	0,028864
SCV	4	0,028964
	5	0,028926
	1	0,028864
LOOCV	4	0,028782
	5	0,028782
	10	0,028782

Nas análises da Seção 5.3 utilizou-se as variáveis Q25, sp.ent, sfm, meanfun, IQR, sd e mode, enquanto que as análises em que utilizou-se Regressão Logística, as variáveis selecionadas foram Q25, Q75, sp.ent, sfm, meanfun, minfun e nodindix. As variáveis em comum (Q25, sp.ent, sfm e meanfun) representam mais de 50% das variáveis utilizadas na Seção 5.3, sendo natural a concordância dos resultados.

Desta forma, observa-se a consistência dos resultados obtidos por meio da seleção de variáveis. Sendo assim, o subconjunto de variáveis selecionadas pareceu simplificar bem a interpretabilidade do modelo, com uma representação parcimoniosa. Na prática, com a utilização de um modelo simples, contendo apenas as variáveis mais importantes, e se, elas explicam tão bem quanto o modelo com mais variáveis, economiza-se tempo e fica mais fácil chegar às conclusões.

Comparando-se as técnicas de aprendizagem analisadas, conclui-se que não houve diferenças apreciáveis entre os resultado da SVM e da FA obtidos utilizando-se apenas as variáveis com maior poder de discriminação, dos resultados utilizando-se os dados de voz originais. Dessa forma, analisando a Tabela 5.17, que apresenta um resumo das taxas médias de má classificação e dos EQMs (vide Seção 5.2.3) para os dados de voz originais, comprova-se que a técnica FA apresentou o melhor desempenho classificatório, seguida da técnica SVM (Kernel Radial).

Tabela 5.17: Valores da Taxas Médias de má Classificação (%) e dos EQMs - Dados II originais

	RDA	SVM				FA
		Linear	Polynomial	Sigmoid	Radial	
CV k-Fold	SCV $k = 4$	LOOCV $k = 4$	SCV e SS $k = 10$	SS $k = 4$	SS $k = 4$	SCV $k = 5$
Taxa média de má classificação (%)	7	2,5	4	13	2,1	2
EQM	0,0081	0,0049	0,0064	0,0114	0,0045	0,0044

5.4 Comparações entre os resultados

Neste trabalho, três técnicas de classificação foram estudadas e aplicadas a dois bancos de dados. Na seção anterior, foram apresentados os resultados das análises para cada banco de dados: tabelas com os parâmetros de ajuste ótimos e os erros médios de generalização, curvas ROC, blox-plots das taxas de má classificação e EQMs.

Na Tabela 5.18, apresenta-se um resumo da análise final do desempenho dos métodos de classificação que baseia-se no erro médio de generalização (ou erro médio de teste) encontrados com as R=1000 replicações.

Tabela 5.18: Comparação das Técnicas de Classificação de acordo com o Erro Médio de Generalização (%).

Banco de Dados	n	p	RDA	SVM	FA
Câncer de Colon (Dados I)	62	2000	10,93	12,46	14,15
Dados de Voz (Dados II)	3168	21	7,04	2,18	2,04

A técnica de regularização RDA superou a SVM e a FA nos dados de câncer de cólon. Estes resultados estão de acordo com os obtidos por Bill (2015), em que os erros médios de generalização para as técnicas RDA, SVM e FA foram iguais a 11.05, 11.77 e 13.66, respectivamente. O autor associou o sucesso da RDA à capacidade dos algoritmos em diminuir a média através do seu parâmetro δ.

Nota-se que a RDA proporcionou o pior desempenho classificatório para o conjunto de dados de voz. O conjunto de dados II foi igualmente classificado pela FA e pela SVM, uma vez que a diferença entre os erros médios foi mínima.

As tabelas apresentadas nas seções anteriores deste capítulo mostram os menores erros médios de classificação obtidos para cada uma três técnicas de validação cruzada (com diferentes

k-folds). Esta forma de apresentação deu-se na tentativa avaliar qual técnica de VC se destacaria em relação às demais. Muitas das análises, em que obteve-se menor sucesso, foram colocadas nos apêndices para consulta.

Fazendo-se uma análise das inúmeras tabelas e box-plots apresentados, pode-se concluir que as técnicas de validação cruzada que mais se destacaram, por proporcionar os menores erros médios de teste, foram a SS e a SCV. Com a SCV obteve-se o menor erro médio de teste em cinco análises enquanto a SS, em três análises, mas em três casos, ambas retornaram o mesmo erro mínimo de teste. A LOOCV destacou-se apenas uma vez dentre os casos resumidos nas Tabelas 5.1, 5.2, 5.3, 5.5, 5.6 e 5.7. O tamanho de K-Fold ideal para esta pesquisa variou, quase na mesma proporção, entre $k = 4$ e $k = 5$.

5.4.1 Comparação em relação ao tempo de execução

A Tabela 5.19 descreve os tempos aproximados (em minutos) para execução de cada método. Os resultados apresentados na tabela referem-se aos métodos e às técnicas de validação cruzada que retornaram o menor erro médio de generalização.

Nos dados de cólon, tem-se os resultados para a RDA com SCV ($k = 5$), para a SVM (kernel sigmoid) com $k = 5$ (SS e SCV) e para a RF com SCV ($k = 10$). Nos dados de voz, tem-se a RDA com SCV ($k = 4$), SVM (kernel radial) com SS ($k = 4$) e FA com SCV ($k = 5$).

Tabela 5.19: Comparação dos tempos de execução, em minutos, de algumas análises utilizando-se a RDA, SVM e FA.

Banco de Dados	n	p	RDA	SVM	FA
Câncer de Colon (Dados I)	62	2000	40	50	90
Dados de Voz (Dados II)	3168	21	1440	60	35

Nota-se que no banco de Dados I, a técnica RDA apresentou o menor tempo de processamento, seguida das técnicas SVM E FA. No banco de Dados II, a RDA necessitou do maior tempo de processamento, enquanto o menor tempo foi obtido com a FA.

Capítulo 6

Considerações Finais

Neste trabalho foram descritos três métodos relativamente recentes de aprendizado supervisionado: Análise Discriminante Regularizada (RDA), Support Vector Machine (SVM) e Florestas Aleatórias (FA). A parte computacional do trabalho consistiu em implementar, utilizando o software R, estas técnicas de aprendizado máquinas para problemas de classificação de dados. Os códigos em `R`, que foram escritos para esta dissertação, estão disponíveis no anexo.

Avaliou-se a necessidade da utilização de padronização nos dados e conclui-se que a mesma não é necessária. Nesta pesquisa, também não aplicou-se métodos de balanceamento das classes, uma vez que estes métodos possuem problemas próprios que podem atrapalhar o aprendizado. Por exemplo, o problema causado pelo método *undersampling* acontece na eliminação de exemplos da classe majoritátia fazendo com que o classificador perca informações importantes pertencentes a esta classe. Por sua vez, o método *oversampling*, que replica exemplos da classe minoritária para obter uma distribuição mais balanceada, pode aumentar a probabilidade de ocorrer o *overfitting*.

Os métodos de aprendizado supervisionados foram aplicados a dois conjuntos de dados e foram realizados estudos de simulação para a comparação do desempenho destes métodos. Todos os resultados obtidos foram apresentados fazendo-se uma comparação não só entre os métodos estudados, como também entre as técnicas de validação cruzada, tomando-se tamanhos distintos de k-Folds.

Dentre os resultados obtidos na SVM utilizando os dados de voz, observa-se que a SVM com kernel Sigmoid apresentou os melhores resultados. Já a SVM com kernel linear não convergiu para alguns valores de *custo*, tal como $custo = 100$. No entanto, a seleção de covariáveis através do uso de regressão logística resolveu esse problema de convergência.

O uso de algumas covariáveis selecionadas por apresentarem distribuições bem distintas para homens e mulheres, também resolveu o problema da falta de convergência da SVM para o $custo = 100$ e, os resultados obtidos foram próximos àqueles conseguidos com o uso da regressão logística.

Além disso, observa-se que o uso de um modelo mais simples, com apenas algumas covariáveis selecionadas, não piora o desempenho, uma vez que os resultados foram próximos aos obtidos usando-se todo o conjunto de dados original.

Analisando os resultados nota-se que, nos dados de voz, a técnica FA apresentou o menor erro médio de teste, bem como o menor EQM. No entanto, a diferença foi mínima em comparação com a SVM (utilizando o kernel radial). Dessa forma, concluímos que o conjunto de voz foi bem classificado por ambas as técnicas.

Nas simulações dos dados, um fato importante a ser considerado, está relacionado ao tempo de execução da RDA nos dados de câncer de cólon, que foi claramente muito mais rápido em comparação ao tempo de execução da RDA nos dados de voz. Além disso, nos dados de cólon, esta técnica teve o melhor tempo de execução em relação às demais técnicas e, os

resultados deste trabalho mostraram que, para estes dados, a RDA obteve o melhor desempenho classificador.

No entanto, nesta pesquisa, não foi encontrado nenhum algoritmo ou método universalmente melhor para lidar com o problema de classificação, pois como pode ser visto, os métodos possuem comportamentos diferentes, dependendo da característica dos dados e do classificador.

Analisando as técnicas de validação cruzada, a técnica que mais se destacou foi a SCV. De acordo com Bill (2015), não é surpreendente que a SCV tenha sido mais bem sucedida na determinação dos parâmetros de ajuste ótimos do que a SS, pois a SCV realiza amostragem estratificada, o que é crucial para manter as probabilidades de classe e, assim, realizar uma predição de classificação mais precisa.

A técnica de validação cruzada, LOOCV foi imparcial em relação ao tamanho de k, uma vez que o algoritmo retornou sempre os mesmas estimativas para AVTE independente de k. Isso pode estar relacionado ao fato de que, na validação cruzada LOOCV, a diferença de tamanho entre o conjunto de treinamento usado em cada k-fold e o conjunto de dados completo é de apenas uma unidade.

O k-Fold ideal para esta pesquisa variou entre $k = 4$ e $k = 5$, apesar de o valor padrão para k ser, normalmente, $k = 10$. Observou-se que ao aumentar o valor de k (neste caso, para $k = 10$), aumentou-se também o tempo de execução do algoritmo. Desta forma, é possível escolher um valor menor para k e ainda obter uma estimativa precisa do desempenho médio do modelo enquanto reduz-se o custo operacional.

Para trabalhos futuros, novos métodos de classificação, baseados em análises *fuzzy* podem ser comparados aos já estudados nesta dissertação. Em especial, destaca-se o trabalho de Singh, Thakur e Sharua (2016).

Refências Bibliográficas

[1] Alon U. et al., (1999). Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays. *PNAS*, 96 (12), 6745-6750.

[2] Bellman, R. E, (1961). Adaptive Control Processes: A Guided Tour. *New Jersey: Princeton University Press.*

[3] Bill, Jo A., (2015). An Empirical Analysis of Predictive Machine Learning Algorithms on High-Dimensional Microarray Cancer Data. *Dissertação (Mestrado em Estatística aplicada) - Rochester Institute of Technology.*

[4] Bishop, C. M., (2006). Pattern Recognition and Machine Learning. *New York: Springer.*

[5] Bonesso, D., (2013). Estimação dos Parâmetros do Kernel em um Classificador SVM na Classificação de Imagens Hiperespectrais em uma Abordagem Multiclasse. *Dissertação (Mestrado) - Universidade Federal do Rio Grande do Sul, Centro Estadual de Pesquisas em Sensoriamento Remoto e Meteorologia, Porto Alegre, RS.*

[6] Breiman, L., Friedman, J., Olshen, R., Stone, C., Steinberg, D. and Colla, P., (1983). Classification and Regression Trees. *Belmont, CA: Wadsworth.*

[7] Breiman, L., (1996). Heuristics of instability and stabilization in model selection. *The Annals of Statistics, 24(6):2350?2383.*

[8] Breiman, L., (2001). Random forests. *Technical report, Berkeley University* `https://www.stat.berkeley.edu/ breiman/randomforest2001.pdf.`

[9] Breiman, L., Cutler, A., (2004). Random forest-manual. `http: // www. Stat. Berkeley. Edu / breiman / RandomForests / cc_manual. Htm.`

[10] Clarke, B., Fokoué, E., Zhang, H. H., (2009). Principles and Theory for Data Mining and Machine Learning. *New York: Springer Velarg.*

[11] Cortes, C., Vapnik, V., (1995). Support-vector network. *Machine Learning*, 20, 1-25.

[12] Data Science Central, (2013). Random Forests Algorithm. `http://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm`.

[13] Dietterich, T. G., (2000). An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40 (2), 139-157.

[14] Efron, B., (1979). Bootstrap methods: another look at the jackknife. *The annals of Statistics*, 1-26

[15] Fokoué, E., (2014). A Taxonomy of Big Data for Optimal Predictive Machine Learning and Data Mining. *Serdica Journal of Computing*, 8 (2), 111-136.

[16] Fokoué, E. and Bill, J., (2014). A Comparative Analysis of Predictive Learning Algorithms on High-Dimensional Microarray Cancer Data. *Serdica Journal of Computing*, 8 (2),137-168.

[17] Friedman, J. H., (1989). Regularized discriminat analysis. *Journal of the American Statistical Association*, 84 (405):165-175. `http://scholarworks.rit.edu/article/1750/.`

[18] Gonçalves, A. R., (2012). Máquina de vetores suporte. `http://www.dca.fee.unicamp.br/ andreric/arquivos/pdfs/svm.pdf.`

[19] Guo, B. Y., Hastie, T., Tibshirani, R., (2005) Regularized Discriminant Analysis and Its Application in Microarrays. *Bioestatistics*, 1 (1), 1-18

[20] Guarda, A., (2013). Aprendizado de máquina: árvore de decisão indutiva. `http://www.barbon.com.br/wp-content/uploads/2013/08/ArvoreDecisaoIndutiva.pdf.`

[21] Hastie, T., Tibshirani, R., Friedman, J., (2001). The Elements of Statistical Learning: Data Mining, Inference, and Predicition. *New York: Springer Velarg.*

[22] Hearst, M. A., Schölkopf, B., Dumais, S., Osuna, E., and Platt, J., (1998). Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13 (4), 18-28.

[23] Introduction to Random Forest, (2007). `https://dimensionless.in/introduction-to-random-forest/.`

[24] Izenman, A. J., (2008). Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning. *New York: Springer.*

[25] James, G., Witten, D., Hastie, T., Tibshirani, R., (2013). An Introduction to Statistical Learning: with Applications in R. *New York: Springer Velarg.*

[26] Koerich, A. L., (2012). Aprendizagem de Máquina. *Programa de Pós-Graduação em Informática. Pontifícia Universidade Católica do Paraná (PUCPR).*

[27] Körting, T. S., (2014). How SVM algorithm works. `https://www.youtube.com/watch?v=1NxnPkZM9bc&index=3&list=PLqS2sO7F2t3Wiw32XRhcDreBqfrKDbz_c.`

[28] Loh, W. Y., (2011). Clasification and regression trees. *John Wiley & Sons.*

[29] Lorena, A. C., Carvalho, A. C. P. L. F., (2003). Introdução às Máquinas de Vetores Suporte (Support Vector Machines): Relatórios Técnicos do ICMC. *São Carlos, SP.*

[30] Lorena, A. C., Carvalho, A. C. P. L. F., (2007). Uma Introdução às Support Vector Machines. *RITA*, Volume XIV, Número 2.

[31] Koerich, W. Y., (2005). Aprendizagem de Máquina: Aprendizagem de Árvores de Decisão. *Pontifícia Universidade Católica do Paraná -PUCPR.*

[32] Meloni, R. B. S., (2009). Classificação de imagens de sensoriamento remoto usando SVM. *Pontifícia Universidade Católica do Rio de Janeiro.*

[33] Mitchell T., (1997). Machine Learning. *WCB McGraw Hill.*

[34] Montaño R. A. N. R., (2016). Aplicação de Técnicas de Aprendizado de Máquina na Mensuração Florestalg. *Tese (Doutorado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.*

[35] Rodrigues, F. A. A., (2012). Modelo de Análise de Risco de Crédito utilizando Máquina de Vetor Suporte. *Rio de Janeiro: UFRJ/COPPE.*

[36] Santos, E. M., (2002). Teoria e Aplicação de Support Vector Machines à Aprendizagem e Reconhecimento de Objetos Baseado na Aparência. *Dissertação (Mestrado) - Universidade Federal da Paraíba, Curso de Pós-Graduação em Informática.*

[37] Sebastian, R. (2015). Python Mahine Learning. *Packt Publishing.*

[38] Shardlow, M., (2013). An Analysis of Feature Selection Techniques. `https://studentnet.cs.manchester.ac.uk/pgt/COMP61011/goodProjects/Shardlow-.pdf`.

[39] Singh, A. , Thakur, N. and Sharua, A (2016). A Review of Supervised Machine Learning Algorithms in Sustainable Global Development (INDIAcom).

[40] Sommer, C. and Gerlich, D. W., (2013). Machine learning in cell biology - teaching computers to recognize phenotypes. *Journal of Cell Science*, 126, 5529-5539.

[41] Tarca, A. L., Carey, V. J., Chen, X. W., Romero, R. and Draghici, S., (2007). Machine learning and its applications to biology. *PLOS Comput.* Biol. 3, e116.

[42] Takaya, S. and Marc R. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS One. 10(3):e011843.*

[43] Tibshirani, R., Hastie, T., Narasimhan, B. and Chu, G., (2003). Class Prediction by Nearest Shrunken Centroids, with Applications to DNA Microarrays. *Statistical Science* 18 (1), 104-107.

[44] Thomas, D., (2017). Educational research techniques: Research techniques and education. `https://educationalresearchtechniques.com/2017/03/22/linear-vs-quadratic-discriminant-analysis-in-r/`.

[45] Vapnik, V. N., (1995). The nature of Statistical learning theory. *Springer-Verlag, New York.*

[46] Vapnik, V. N., (1998). Statistical Learning Theory. *Wiley, New York.*

[47] Zuben, F. J. V., Attux, R. R. F., (2010). Árvores de Decisão. *DCA/FEEC/Unicamp.*

Apêndice A

Florestas Aleatórias

De acordo com Clarke et al. (2009), floresta aleatória consiste em árvores formadas a partir das amostras de bootstrap. Contudo, nenhuma poda das árvores é feita. A árvore acaba de crescer até que cada nó folha contenha apenas membros de uma única classe. Normalmente, são geradas cerca de 100 árvores, cada uma de uma amostra de bootstrap independente. Para uma nova observação ser classificada, usa-se cada uma das árvores na floresta. Caso uma variedade de árvores combinar com uma certa classificação, então essa é a categoria prevista da observação.

Ainda segundo Clarke et al. (2009), existem dois "truques" inteligentes que tornam as florestas aleatórias eficazes. O primeiro é uma estimativa de erro "fora do saco"(out-of-bag) e o segundo é uma seleção aleatória da característica (para maiores detalhes, ver Clarke et al. (2009), páginas 255 a 257).

Para obter melhorias significativas na precisão da classificação deve-se crescer um conjunto de árvores e deixá-los votar na classe mais popular. Para crescer esses conjuntos, muitas vezes são gerados vetores aleatórios que regem o crescimento de cada árvore no conjunto. Um exemplo é o ensacamento (Breiman, 1996), em que para crescer cada árvore, uma seleção aleatória (sem substituição) é feita a partir dos exemplos no conjunto de treinamento (Breiman, 2001).

Breiman (2001) explica que nesse procedimento, um vetor aleatório Θ_k é gerado para a k-ésima árvore, independente dos vetores aleatórios passados $\Theta_1, \ldots, \Theta_{k-1}$, mas com a mesma distribuição; uma árvore é cultivada usando o conjunto de treinamento e Θ_k resulta em um classificador $h(\mathbf{x}, \Theta_k)$, sendo $\mathbf{x}$ um vetor de entrada. A natureza e a dimensionalidade de Θ dependem da sua utilização na construção de árvores. Depois de gerar um grande número de árvores, a classe mais popular é votada. Esse conjunto de árvores é chamado de floresta aleatória.

Definição A.0.1 (Breiman, 2001). *Uma floresta aleatória é um classificador que consiste em uma coleção de classificadores estruturados em árvores* $\{h(\boldsymbol{x}, \Theta_k), k = 1, \ldots\}$ *em que* Θ_k *são vetores aleatórios independentes e identicamente distribuídos e cada árvore lança uma unidade de votação para a classe mais popular na entrada* $\boldsymbol{x}$.

Clarke et al. (2009) mostram que o número médio de classificações corretas é a proporção dos classificadores que identificam a classe correta:

$$AV(Y) = \frac{1}{J} \sum_{j=1}^{J} I_{\{h_j(\mathbf{X})=y\}}, \tag{A.1}$$

em que I_A é a função indicadora para um conjunto A. O número médio de classificações erradas

do tipo k, para $k \neq Y$ é a proporção de $h_j s$ classificando erradamente Y como k,

$$AV(k) = \frac{1}{J} \sum_{j=1}^{J} I_{\{h_j(\mathbf{X})=k\}}. \tag{A.2}$$

Um conjunto de classificadores $h_1(\mathbf{X}), \ldots, h_k(\mathbf{X})$ e um conjunto de treinamento formado aleatoriamente a partir da distribuição do vetor aleatório $Y, \mathbf{X}$ definem a função margem como

$$MG(\mathbf{X}, Y) = AV_k I(h_k(\mathbf{X}) = Y) - \max_{j \neq Y} AV_k I(h_k(\mathbf{X}) = j), \tag{A.3}$$

em que $I(\cdot)$ é a função indicadora. Ou simplesmente,

$$MG(\mathbf{X}, Y) = AV(Y) - \max_{j \neq Y} AV(j). \tag{A.4}$$

em que MG representa quantos classificadores obtêm a classe certa ao invés de obter a classe errada. Clarke et al. (2013) afirmam que um bom classificador fornece uma MG grande. Se MG fosse negativa e $K = 2$, seria obtido um melhor classificador se fossem trocadas as classes previstas.

A margem mede o quanto o número médio de votos em $\mathbf{X}, Y$ para a classe correta excede o voto médio para qualquer outra classe. Quanto maior a margem, maior a confiança na classificação.

Em média, o comportamento da MG é descrito pela probabilidade do erro de generalização

$$PE = \mathbb{P}_{\mathbf{X},Y}(MG(\mathbf{X}, Y) < 0). \tag{A.5}$$

em que os subscritos $\mathbf{X}, Y$ indicam que a probabilidade está sobre o espaço $\mathbf{X}, Y$. O PE é a probabilidade de que a classificação correta, pelos classificadores combinados, seja dada com menos frequência do que a mais provável das classificações erradas. Obviamente, é desejável que o PE seja pequeno.

Suponha que os classificadores J são estruturados em árvore com $h_j(\mathbf{x}) = h(\mathbf{x}, \theta_j)$ e a floresta aleatória gerada a partir deles é formada por voto majoritário. Para um grande número de árvores, Breiman (2001) afirma que o tamanho da floresta converge para um valor limitante. O que nos leva ao seguinte Teorema de Breiman:

Teorema A.0.1 (Breiman, 2001). *À medida que o número de árvores aumenta, quase certamente todas as sequências* $\Theta_1, \ldots$ *PE convergem para* $P_{\boldsymbol{X},Y}(P_\Theta(h(\boldsymbol{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\boldsymbol{X}, \Theta) = j) < 0)$

Esse resultado explica por que as florestas aleatórias não se sobrepõem à medida que mais árvores são adicionadas, mas produzem um valor limitante do erro de generalização (Breiman, 2001).

Força e correlação

Segundo Breiman (2001), o erro de generalização de uma floresta de classificadores de árvores depende de dois parâmetros que estão relacionados à força das árvores individuais na floresta e à correlação entre elas. A interação entre dois parâmetros fornece a base para compreensão do funcionamento das florestas aleatórias.

Breiman (2001) define a função de margem para uma floresta aleatória como

$$MR(\mathbf{X}, Y) = (P_\Theta(h(\mathbf{X}, \Theta) = Y) - \max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j), \tag{A.6}$$

e a força do conjunto de classificadores $\{h(x, \Theta)\}$ é

$$s = E_{\mathbf{X},Y} MR(\mathbf{X}, Y). \tag{A.7}$$

Assumindo $s \geq 0$, a desigualdade de Chebychev fornece

$$PE^* \leq \frac{\text{var(MR)}}{s^2}. \tag{A.8}$$

Breiman (2001) mostra que uma expressão para a variância de MR é derivada de:

$$\hat{j}(\mathbf{X}, Y) \arg\max_{j \neq Y} P_\Theta(h(\mathbf{X}, \Theta) = j), \tag{A.9}$$

assim,

$$MR(\mathbf{X}, Y) = P_\Theta(h(\mathbf{X}, \Theta) = Y) - P_\Theta(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y)) \tag{A.10}$$

$$= E_\Theta[I(h(\mathbf{X}, \Theta) = Y) - I(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y))]. \tag{A.11}$$

Desta forma, o autor ainda afirma que a função da margem bruta é dada por:

$$RMG(\Theta, \mathbf{X}, Y) = I(h(\mathbf{X}, \Theta) = Y) - I(h(\mathbf{X}, \Theta) = \hat{j}(\mathbf{X}, Y)). \tag{A.12}$$

em que $MR(\mathbf{X}, Y)$ é a expectativa de $RMG(\Theta, \mathbf{X}, Y)$ em relação à Θ

Suponha que foram geradas cópias independentes de Θ: Θ e Θ', a expectativa de seus fatores de produto é dada por

$$MR(\mathbf{X}, Y)^2 = E_{\Theta,\Theta'} RMG(\Theta, \mathbf{X}, Y) RMG(\Theta', \mathbf{X}, Y), \tag{A.13}$$

em que X e Y permanecem aleatórios. A variância sobre X e Y é

$$Var(MR) = E_{\Theta,\Theta'}(Cov_{\mathbf{X},Y} RMG(\Theta, \mathbf{X}, Y) RMG(\Theta', \mathbf{X}, Y)) \tag{A.14}$$

$$= E_{\Theta,\Theta'}(\rho(\Theta, \Theta')) SD(\Theta) sd(\Theta'), \tag{A.15}$$

em que $\rho(\Theta, \Theta')$ é a correlação entre $RMG(\Theta, \mathbf{X}, Y)$ e $RMG(\Theta', \mathbf{X}, Y)$ e SD é o desvio padrão de $RMG(\Theta, \mathbf{X}, Y)$ mantendo Θ fixo. Então,

$$Var(MR) = \overline{\rho}(E_\Theta SD(\Theta))^2 \tag{A.16}$$

$$\overline{\rho} E_\Theta Var(\Theta), \tag{A.17}$$

onde $\overline{\rho}$ é o valor médio da correlação; isto é,

$$\overline{\rho} = \frac{E_{\Theta,\Theta'}(\rho(\Theta, \Theta') SD(\Theta) SD(\Theta'))}{E_{\Theta,\Theta'} SD(\Theta) SD(\Theta')}. \tag{A.18}$$

O erro de generalização é dado por (Breiman, 2001)

$$PE^* \leq \frac{\overline{\rho}(1 - s^2)}{s^2}.$$

Os dois ingredientes envolvidos no erro de generalização para florestas aleatórias são a força dos classificadores individuais na floresta e a correlação entre eles em termos de funções da margem bruta. A relação $c/s2$ é a correlação dividida pelo quadrado da força. Ao entender o funcionamento das florestas aleatórias, essa proporção será um guia útil - quanto menor for, melhor (Breiman, 2001).

Breiman (2001) definiu a relação $c/s2$ para uma floresta aleatória como

$$c/s2 = \frac{\overline{\rho}}{s^2.}$$

Para o caso de duas classes a função de margem é apresentada pela expressão

$$MR(\mathbf{X}, Y) = 2P_\Theta(h(\mathbf{X}, \Theta) = Y) - 1.$$

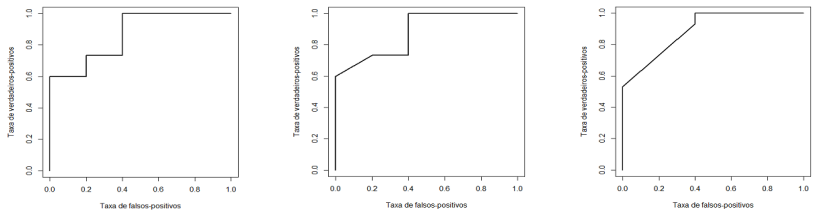
A função de margem bruta é $2I(h(\mathbf{X}, \Theta) = Y) - 1$ e a correlação $\overline{\rho}$ é realizada entre $I(h(\mathbf{X}, \Theta) = Y)$ e $I(h(\mathbf{X}, \Theta') = Y)$.

Apêndice B

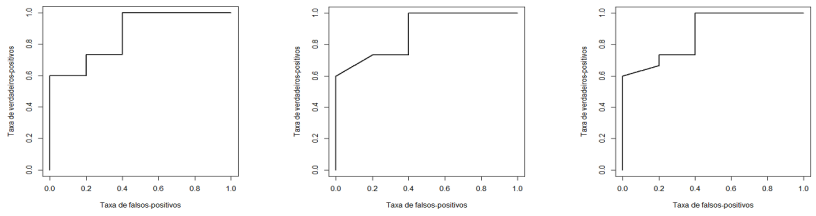
Gráficos

B.1 Curvas ROC - Dados I

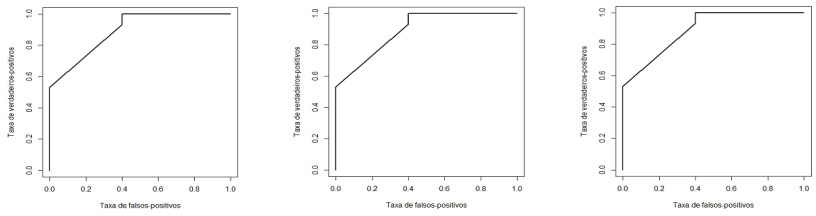
A seguir, são apresentadas as curvas ROC para a SVM (kernels Polynomial e radial).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

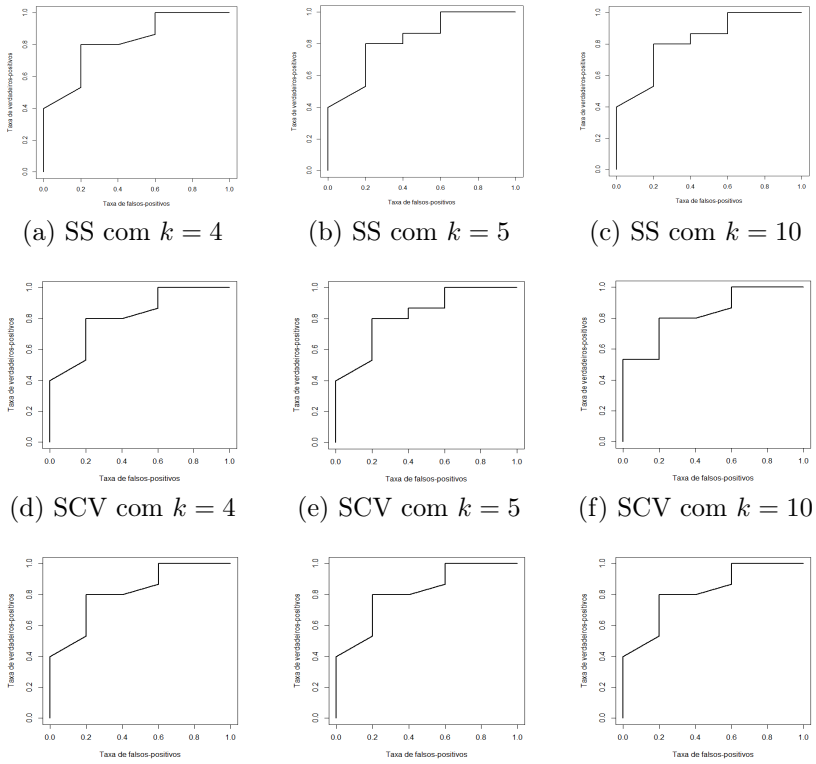


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura B.1: SVM (kernel: Polynomial) - Dados I. Curvas ROC obtidas usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

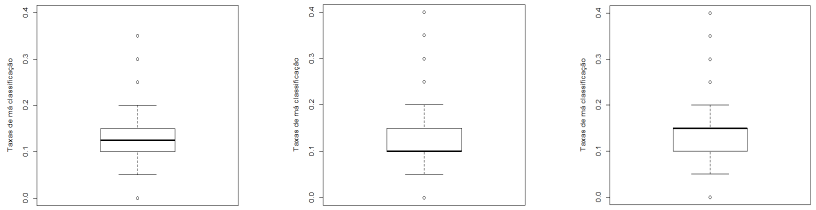
(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

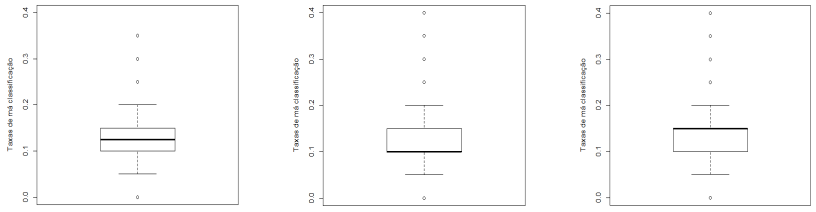
Figura B.2: SVM (kernel: radial) - Dados I. Curvas ROC obtidas usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.

B.2 Box-Plots - Dados I

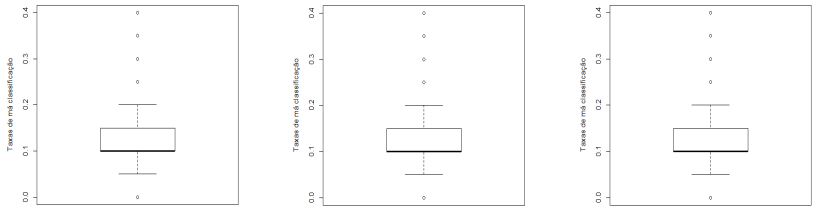
Os box-plots das Figuras B.3 e B.4 mostram as taxas de má classificação e os erros quadráticos médios para a SVM (kernel Linear). As taxas de má classificação e os erros quadráticos médios da SVM (kernel Polynomial), são apresentados nas Figuras B.5 e B.6. Por sua vez, as as Figuras B.7 e B.8 mostram os mesmos resultados para a SVM (kernel Radial).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

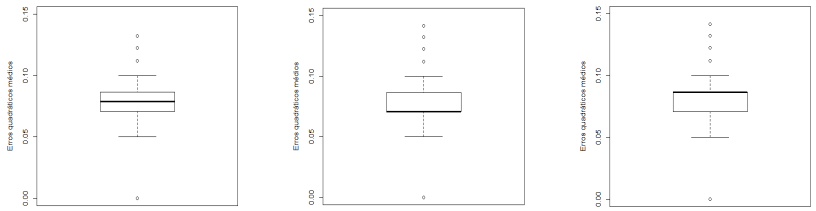


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

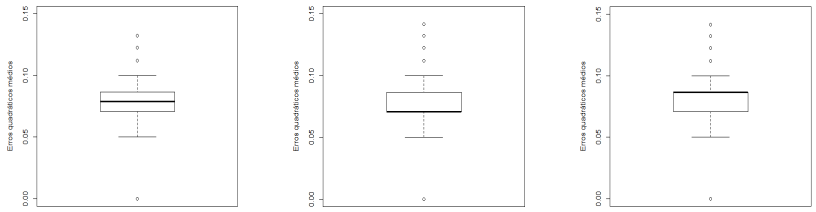


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

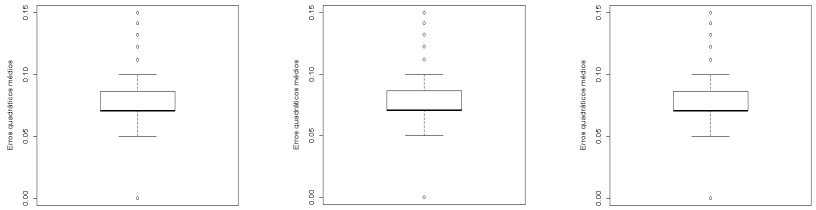
Figura B.3: SVM (kernel linear) - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

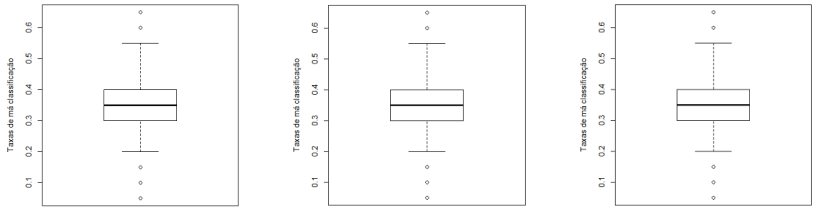


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

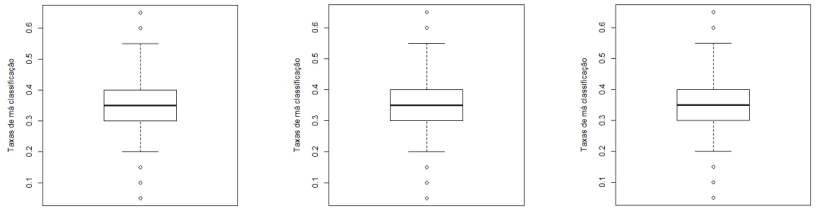


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

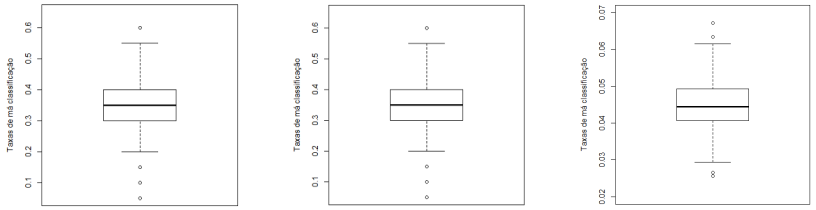
Figura B.4: SVM (kernel linear) - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

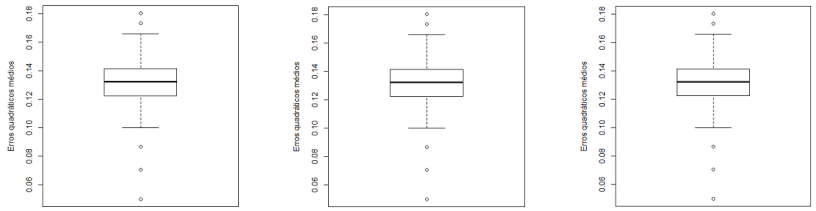


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

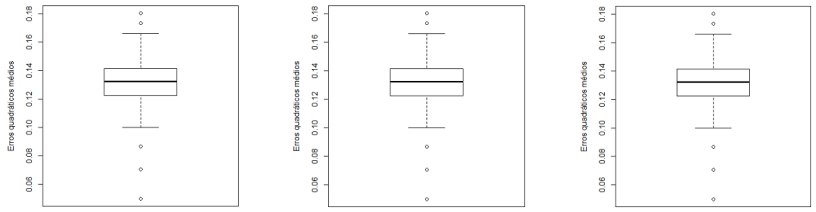


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

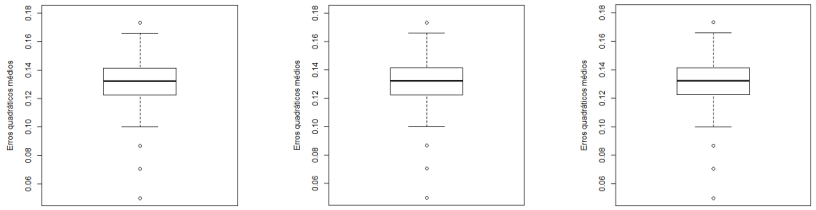
Figura B.5: SVM (kernel: polynomial) - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

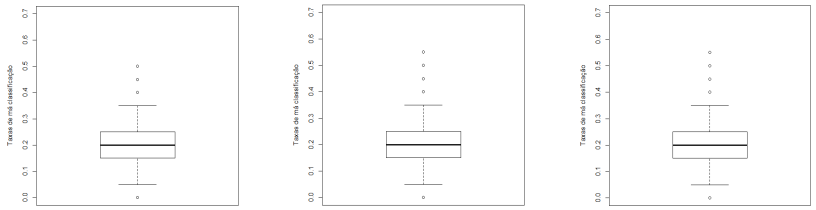


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

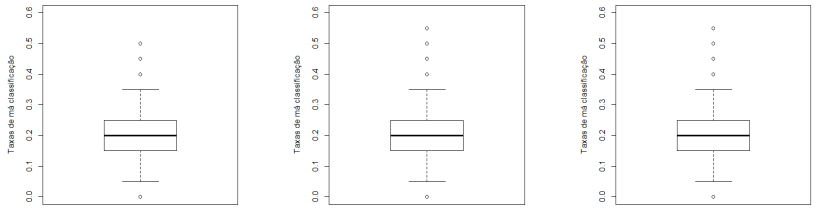


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

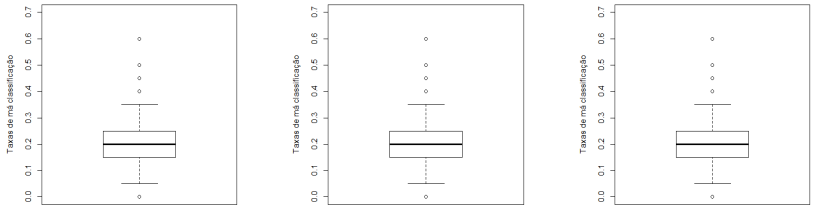
Figura B.6: SVM (kernel: polynomial) - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

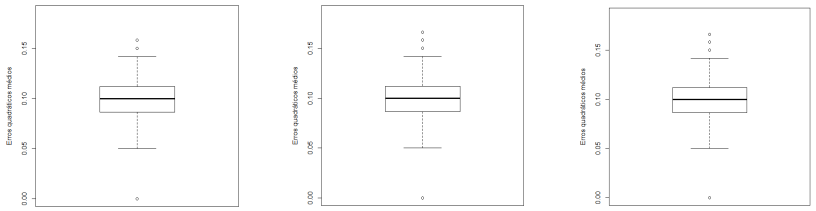


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

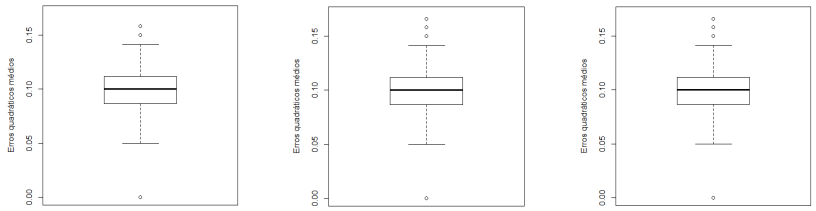


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

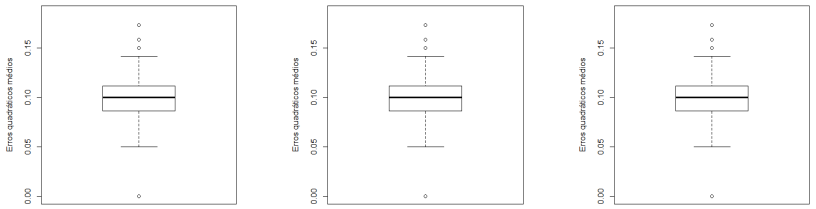
Figura B.7: SVM (kernel: Radial) - Dados I. Box-plots das taxas de má classificação obtidas usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com 4 (b) SS com 5 (c) SS com 10



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

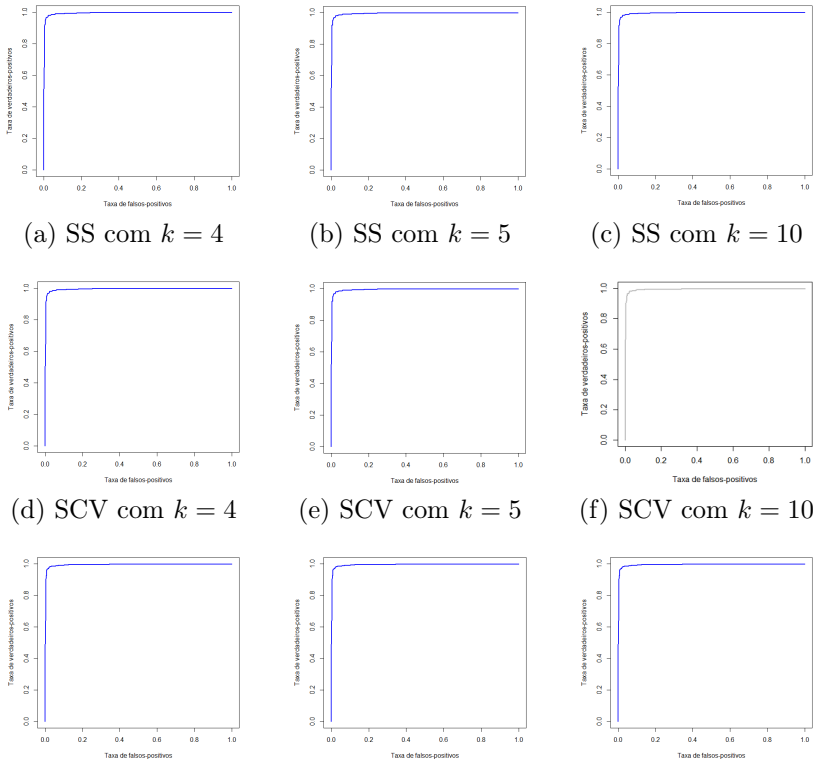


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

Figura B.8: SVM (kernel: Radial) - Dados I. Box-plots dos EQMs obtidos usando-se a validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).

B.3 Curvas ROC - Dados II

A seguir, são apresentadas as curvas ROC da SVM (kernels Linear, Polynomial e Sigmoid) realizadas usando-se Dados II.

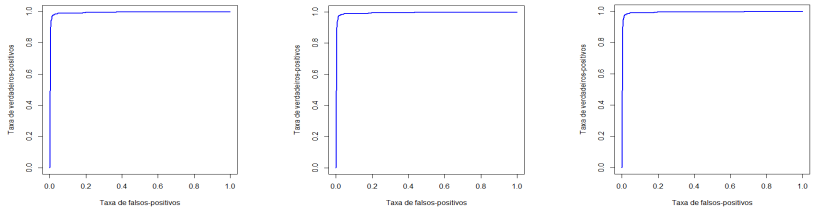


(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

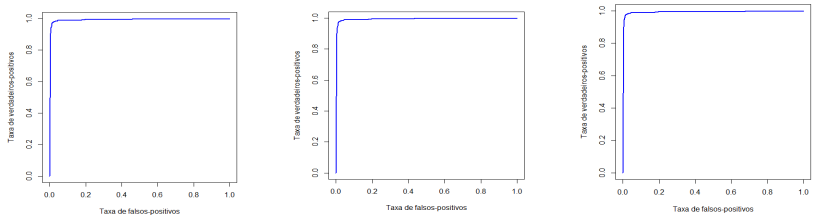
(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

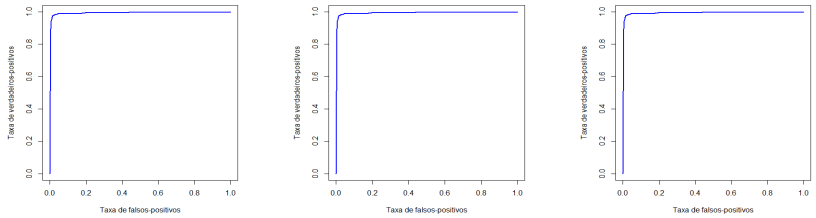
Figura B.9: SVM (Kernel: Linear) - Dados II originais. Curvas ROC usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

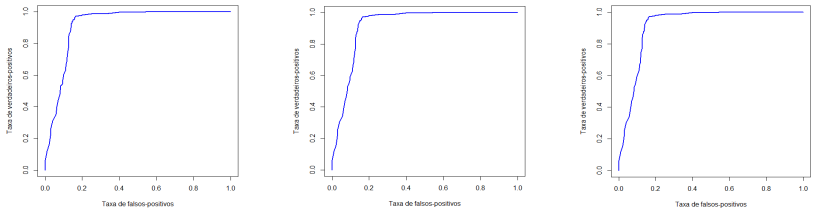


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

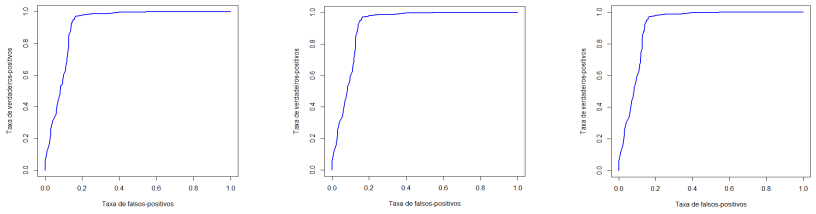


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

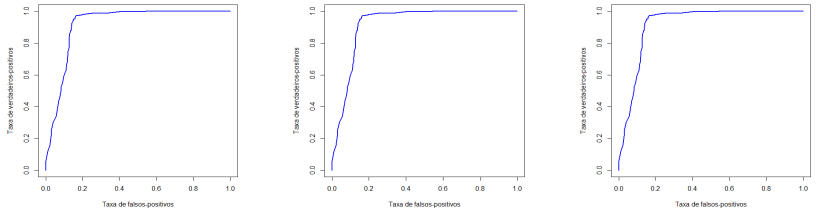
Figura B.10: SVM (Kernel: Polynomial) - Dados II originais. Curvas ROC usando os métodos de validação cruzada SS, SCV e LOOCV.



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

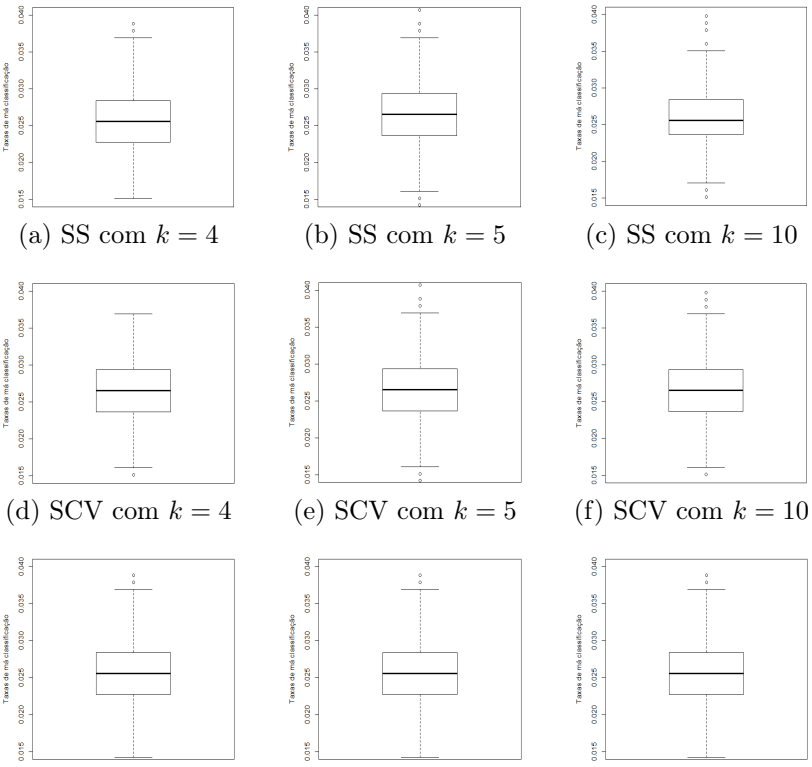


(g) LOOCV com k=4 (h) LOOCV com k=5 (i) LOOCV com k=10

Figura B.11: SVM (Kernel: Sigmoid) - Dados II originais. Curvas ROC usando os métodos de validação cruzada SS, SCV e LOOCV, para valores de k iguais a 4, 5 e 10, respectivamente.

B.4 Box-Plots Dados II

As figuras a seguir, apresentam que as taxas de má classificação e os erros quadráticos médios para a SVM (kernels Linear, Polynomial e sigmoid).

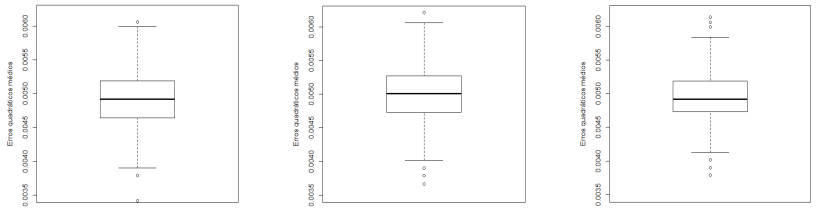


(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

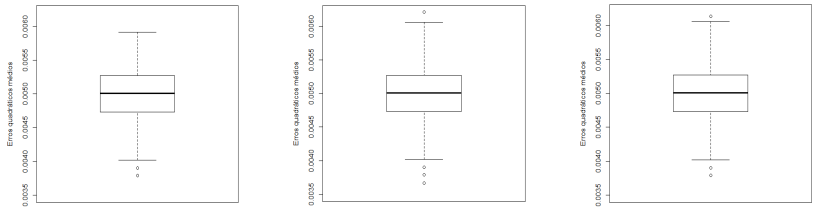
(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

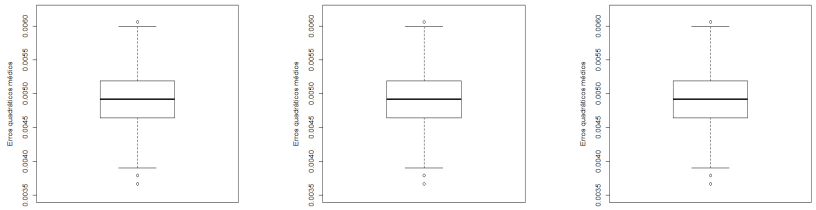
Figura B.12: SVM (Kernel Linear) - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

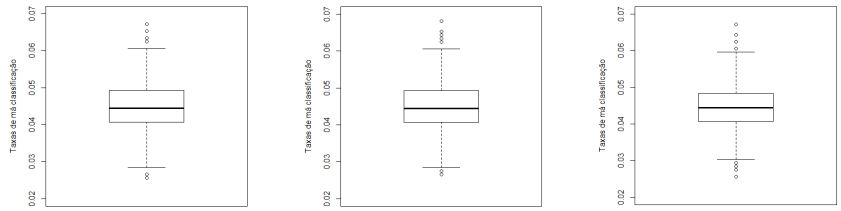


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

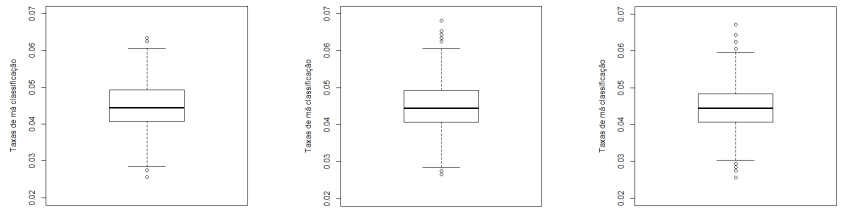


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

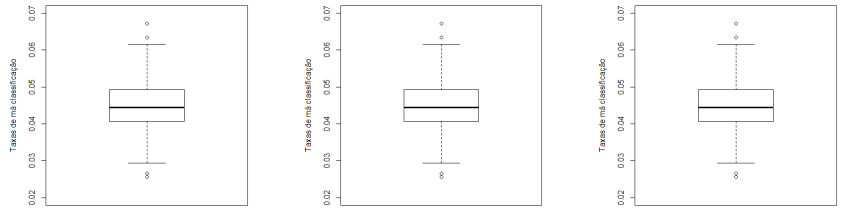
Figura B.13: SVM (Kernel Linear) - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV) e diferentes valores de k (4, 5 e 10).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

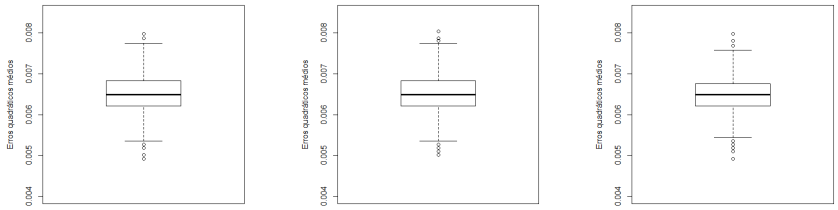


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

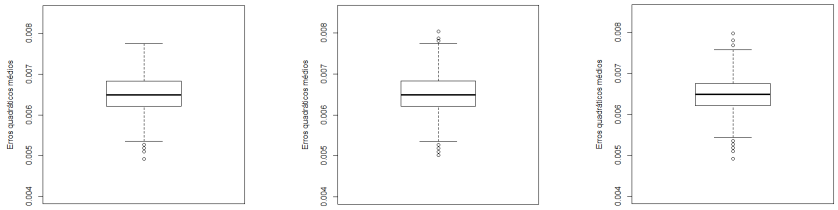


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

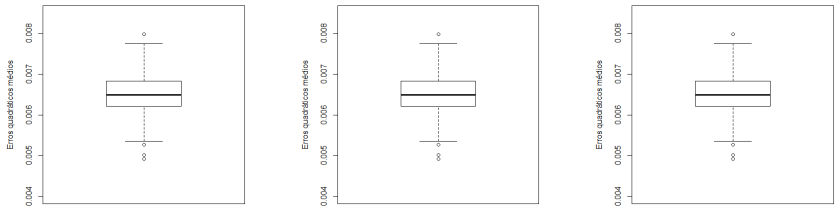
Figura B.14: SVM (Kernel Polynomial) - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

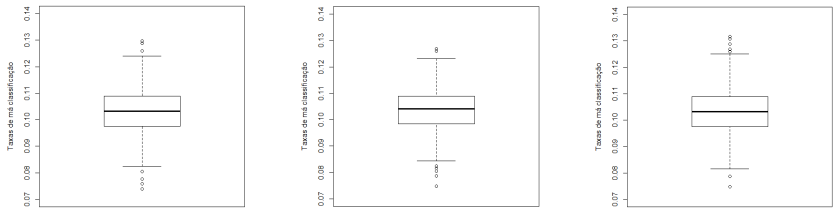


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

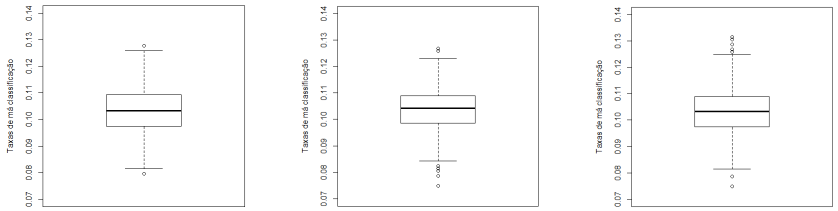


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

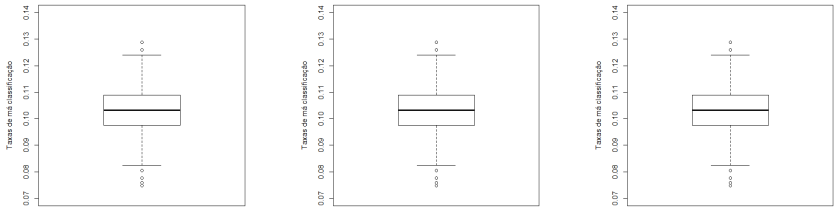
Figura B.15: SVM (Kernel Polynomial) - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$

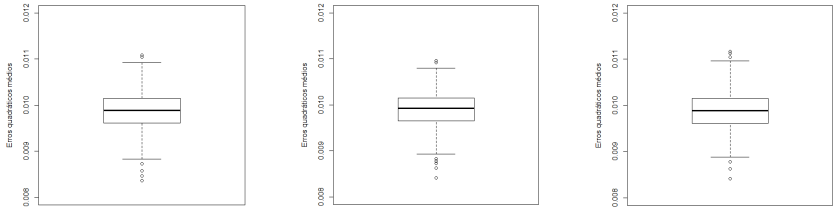


(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$

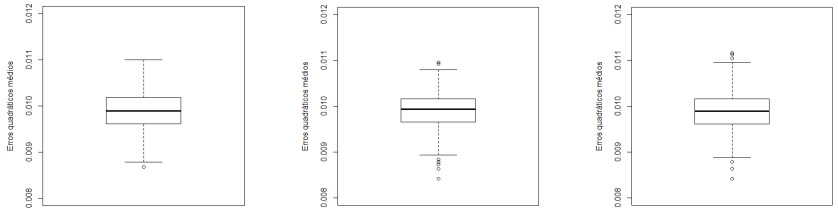


(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

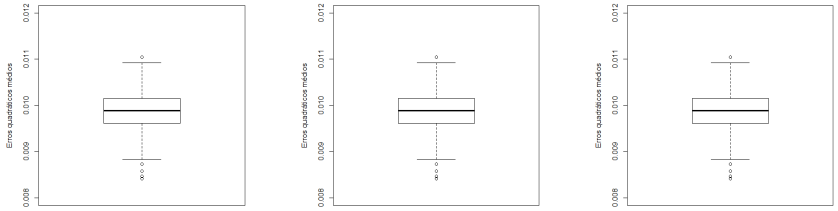
Figura B.16: SVM (Kernel Sigmoid) - Dados II originais. Box-plots das taxas de má classificação obtidas usando-se validação cruzada (SS, SCV e LOOCV).



(a) SS com $k = 4$ (b) SS com $k = 5$ (c) SS com $k = 10$



(d) SCV com $k = 4$ (e) SCV com $k = 5$ (f) SCV com $k = 10$



(g) LOOCV com $k = 4$ (h) LOOCV com $k = 5$ (i) LOOCV com $k = 10$

Figura B.17: SVM (Kernel Sigmoid) - Dados II originais. EQMs obtidos usando-se validação cruzada (SS, SCV e LOOCV).

Apêndice C

Programas desenvolvidos em R

No Apêndice C.1.1 apresenta-se o programa desenvolvido para a RDA utilizando-se os dados de câncer de cólon. Estes comando foram adaptadas para os dados de voz. Desta forma, foi desenvolvido um só programa que possa ser utilizado para os dois bancos de dados, conforme Apêndice C.1.2. Os Apêndices C.2 e C.3 mostram os programas desenvolvidos para as técnicas SVM e RF. Estes dois programas foram desenvolvidos e utilizados, tanto para os dados de cólon, como para os dados de voz.

C.1 Análise Discriminante Regularizada (RDA)

C.1.1 RDA - Dados I

```
############################ FUNÇÕES ##########################################################

#########################################################################################################
##### CHOOSE.MAT
#
choose.mat <- function(alpha1, delta1, ymat, ynew, ii){
l1 <- length(alpha1)
l2 <- length(delta1)
if(l1==1 & l2==1) pe <- ynew[ii] != ymat
if((l1==1 & l2>1) || (l1>1 & l2==1)) pe <- ynew[,ii] != ymat
if( l1>1 & l2>1 & length(dim(ynew)) > 2) pe <- ynew[,,ii] != ymat
if( l1>1 & l2>1 & length(dim(ynew)) == 2) pe <- ynew != ymat
return(pe)
}
#

#########################################################################################################
##### RDA.BASICS
#

rda.basics <- function(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta){
alpha1 <- alpha
delta1 <- delta
MEtest.ss <- 0
for(k in 1:nrFolds) {
fold <- which(folds == k)             # Trabalhe com o k-ésimo
fold
#
x.train <- training.x[,-fold]           #x.train é o conjunto dados de treinamento (matriz)
y.train <- training.y[-fold]            #y.train é vetor que  informa a classe do conjunto de treinamento
#
x.test<- training.x[,fold]       #x.test é o conjunto dados de teste (matriz)
if(length(fold) == 1)
x.test = matrix(x.test, ncol = 1)
y.test <- training.y[fold]              #y.test é vetor que informa a classe do conjunto de teste
rda1 = rda(x.train,y.train,alpha = alpha1, delta = delta1)
ynew <- predict(rda1, x=x.train, y=y.train,xnew=x.test, alpha=alpha1, delta=delta1)

# CALCULO DO ERRO DE PREDIÇÃO:
```

```
pe <- PE1 <- matrix(0,length(alpha1),length(delta1))
for(ii in 1:length(fold)){
ymat <- matrix(y.test[ii],length(alpha1),length(delta1))
# choose.mat ANALISA A DIMENSÃO DE ynew:
pe <- choose.mat(alpha1,delta1,ymat,ynew,ii)
pe[pe=="FALSE"] <- 0
pe[pe=="TRUE"] <- 1
PE1 <- PE1 + pe
}
#
MEtest.ss <- MEtest.ss + PE1/length(fold)
}
##
MEtest.ss <- MEtest.ss/nrFolds
return(MEtest.ss)
}
#

#############################################################################################################
##### RDA1.BASICS
#
rda1.basics <- function(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta){

alpha1 <- alpha
delta1 <- delta
MEtest.ss <- 0

for(k in 1:nrFolds) {
fold <- which(folds == k)             # Trabalhe com o k-ésimo
fold
#
x.train<-training.x[,-fold]           #x.train é o conjunto dados de treinamento (matriz)
y.train<-training.y[-fold]            #y.train é vetor que informa a classe do conjunto de treinamento
#
x.test<-training.x[,fold]             #x.test é o conjunto dados de teste (matriz)
if(length(fold) == 1)
x.test = matrix(x.test, ncol = 1)
y.test<-training.y[fold]              #y.test é vetor que informa a classe do conjunto de teste
rda1 = rda(x.train,y.train, x.test, y.test, alpha = alpha1, delta = delta1)
ynew <- predict(rda1, x=x.train, y=y.train, xnew=x.test, alpha=alpha1, delta=delta1)

# LEVANDO EM CONTA OS TAMANHOS DISTINTOS DOS FOLDS:
MEtest.ss = MEtest.ss + rda1$testerror/length(fold)
}
##
MEtest.ss<-MEtest.ss/nrFolds
return(MEtest.ss)
}
#

#############################################################################################################
##### CONFUSION MATRIX AND ROC CURVE
#
roc.curve=function(s,print=FALSE){
Ps = (S>s)*1
FP = sum((Ps==1)*(Y==0))/sum(Y==0)
TP = sum((Ps==1)*(Y==1))/sum(Y==1)
if(print==TRUE){
print(table(Observed=Y,Predicted=Ps))
}
vect = c(FP,TP)
names(vect) = c("FPR","TPR")
return(vect)
}
#



#############################################################################################################
##### BAGGING
#
bagging.rda <- function(x,y,tr.index,s.size,iterations){
testing.y  <- y[-tr.index]
#
predictions<-foreach(m=1:iterations,.combine=cbind) %do% {
print(m)
```

```
bagg1<-sample(tr.index,s.size,replace=TRUE)
training1.x <-x[, bagg1]
training1.y <-y[bagg1]
rda1 = rda(training1.x, training1.y, alpha = alpha1, delta = delta1)
predict(rda1, x=training1.x, y=training1.y,xnew=testing.x, alpha=alpha1, delta=delta1)
}
#
predictions <- rowMeans(predictions)
error <- sqrt((sum((testing.y-predictions)^2))/length(testing.y))
list(predictions,error)
}
#

#############################################################################################################

#################### ANÁLISES ###############################################################################
#
require(rda)
require(caret)
require(foreach)
data(colon)
x <- colon.x <- t(colon.x)
y <- colon.y
nn<- ncol(colon.x)
nrFolds<-5
train.nn<-ceiling(62*(2/3))
alpha=seq(0, 0.99, len=10)
delta=seq(0, 3, len=10)
seed1<-107
set.seed(seed1)
tr.index <- sample(1:nn, train.nn)
training.x <-x[, tr.index]
training.y <-y[tr.index]
testing.x  <-x[, -tr.index]
testing.y  <-y[-tr.index]
#

#############################################################################################################
##### UTILIZANDO O MÉTODO K-folds DE VALIDAÇÃO CRUZADA DO TIPO ESTRATIFICADO (SCV) PARA O RDA:
##### STRATIFIED CROSS-VALIDATION
#
folds <- createFolds(factor(training.y), k = nrFolds, list = FALSE)

#### VERIFICANDO A PROPORÇÃO DE INDIVÍDUOS COM A DOENÇA EM CADA FOLD:
analisa<-cbind(training.y,folds)
props<-rep(0,nrFolds)
for(i in 1:nrFolds){
aux<-analisa[analisa[,2]==i,]
aux<-table(aux[,1])/nrow(aux)
props[i]<-aux[2]
}
##
props

#### PROPORÇÃO DE INDIV. COM A DOENÇA NO training data:
b1<-table(training.y)/length(training.y)
b1
##

#### PORTANTO, CADA FOLD CONTEM, APROXIMADAMENTE, A MESMA PROPORÇÃO DO training set (training.y).

#### rda.basics - ERROS DE PREDIÇÃO CALCULADOS COM BASE EM CÓDIGO DESENVOLVIDO NESTA DISSERTAÇÃO.
st.RDA<-rda.basics(training.x, training.y, testing.x, testing.y, nrFolds,folds,alpha,delta)
st.RDA

#### rda1.basics - ERROS DE PREDIÇÃO CALCULADOS COM BASE NA FUNÇÃO testerror do rda.
st.RDA1<-rda1.basics(training.x, training.y, testing.x, testing.y, nrFolds,folds,alpha,delta)
st.RDA1
#
# CONCLUSÃO: rda.basics e rda1.basics conduziram aos mesmos valores.

#############################################################################################################
##### UTILIZANDO O MÉTODO K-folds DE VALIDAÇÃO CRUZADA DO TIPO Shuffe and Split (SS) PARA O RDA:
#
folds <- createFolds(training.y, k = nrFolds, list = FALSE)
st.RDA<-rda.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta)
```

```
st.RDA
st.RDA1<-rda1.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta)
st.RDA1
#

#############################################################################################################
##### UTILIZANDO O MÉTODO DE VALIDAÇÃO CRUZADA DO TIPO LEAVE-ONE-OUT PARA O RDA:
#
nrFolds <- length(training.y)
folds <- createFolds(factor(training.y), k = nrFolds, list = TRUE)
st.RDA <- rda.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta)
st.RDA
st.RDA1<-rda1.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta)
st.RDA1
#

#############################################################################################################
##### TENTANDO LOCALIZAR A FAIXA ONDE ESTÃO alpha e delta ÓTIMOS:

############## NOVOS PARES (alpha,delta) ######################
# delta com escala menos ampla:
alpha=c(seq(0, 0.90, len=10),0.99)
delta=seq(0, 1, len=10)
st.RDA<-rda1.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, alpha, delta)
st.RDA

##### ENCONTRANDO AS LINHAS E COLUNAS CUJO ERRO DE TREINAMENTO É MÍNIMO:
wh<-which(st.RDA == min(st.RDA), arr.ind = TRUE)
wh
#

############## NOVOS PARES (alpha,delta) ##############################
# delta com escala menos ampla:
alpha=seq(0, 0.5, len=10)
delta=seq(0, 0.5, len=10)
st.RDA<-rda1.basics(training.x, training.y, testing.x, testing.y, nrFolds,folds,alpha,delta)
st.RDA
wh<-which(st.RDA == min(st.RDA), arr.ind = TRUE)
wh
#

############## NOVOS PARES (alpha,delta) ##############################
# delta com escala menos ampla:
alpha=seq(0, 0.15, len=5)
delta=seq(0, 0.30, len=5)

st.RDA<-rda1.basics(training.x, training.y, testing.x, testing.y,
nrFolds,folds,alpha,delta)
st.RDA

wh<-which(st.RDA == min(st.RDA), arr.ind = TRUE)
wh
#

##### AGORA PRECISA REPETIR O PROCESSO.
##### TOMA-SE FOLDS DISTINTOS DO MESMO training set:
#
mat.error1<-matrix(0,length(alpha),length(delta))
kk<-500
for(ii in 1:kk){
print(ii)
folds <- createFolds(factor(training.y), k = nrFolds, list =
FALSE)
st.RDA<-rda1.basics(training.x, training.y, testing.x, testing.y,
nrFolds,folds,alpha,delta)
mat.error1<-mat.error1 + st.RDA
#   print(st.RDA)
}
mat.error1<-mat.error1/kk
mat.error1
#

##### ENCONTRANDO AS LINHAS E COLUNAS CUJO ERRO DE TREINAMENTO É MÍNIMO:
wh<-which(mat.error1 == min(mat.error1), arr.ind = TRUE)
wh
#
```

```
##### TRABALHANDO, AGORA, COM OS VALORES DE TUNNING ÓTIMOS:
alpha1<-0.075
delta1<-0.15

##### Erro de predição para os valores ótimos de alpha e delta:
#
fit<- rda(training.x, training.y,alpha = alpha1, delta = delta1)
ynew<-predict(fit, x=training.x, y=training.y,xnew=testing.x,
alpha=alpha1, delta=delta1)
pred.error<-sum(ynew != testing.y)
pred.error
#

##### CONFUSION MATRIX - VIDE PGS. 190 E 191 DE SEBASTIAN RASCHKA:
#
S<-predict(fit, x=training.x, y=training.y,xnew=testing.x,
alpha=alpha1, delta=delta1,type="posterior")
S<-S[,2]
#S
#
Y<-testing.y-1  #Note que no caso de Colon data, os labels são 1 e 2.
threshold = 0.5
roc.curve(threshold,print=TRUE)
#

##### ROC CURVE

ROC.curve=Vectorize(roc.curve)
M.ROC=ROC.curve(seq(0,1,by=.01))
plot(M.ROC[1,], M.ROC[2,], col="grey", lwd=2,type="l", xlab="Taxa de
falsos-positivos", ylab="Taxa de verdadeiros-positivos -
Sensibilidade")

# FONTE:
# https://www.r-bloggers.com/illustrated-guide-to-roc-and-auc/
# http://maths-people.anu.edu.au/~johnm/courses/dm/math3346/2008/pdf/r-exercisesVI.pdf
# https://github.com/ChicagoBoothML/MachineLearning_Fall2015/blob/master/Programming%20Scripts/Lecture03/
knn-bagging.R


##### AGORA PRECISA REPETIR O PROCESSO:

# s.size: #tamanho da amostra de indices relativos à training sample
s.size<-length(tr.index)
# iterations: número de iterações no bagging process.
iterations<-1000
aa<-bagging.rda(x,y,tr.index,s.size,iterations)
aa
cbind(aa[[1]],testing.y)
#

##### REPETIR O PROCESSO DE PARTIÇÃO EM TRAINING/TEST SETS 1000 VEZES

alpha1 <- 0.075
delta1 <- 0.15
iterations <- 100
nn <- ncol(colon.x)
train.nn <- ceiling(62*(2/3))
stats <- matrix(0,iterations,2)
for(m in 1:iterations){
print(m)
tr.index <- sample(1:nn, train.nn)
training.x <- x[, tr.index]
training.y <- y[tr.index]
testing.x <- matrix(x[, -tr.index], ncol = nn - train.nn)
testing.y <- y[-tr.index]
rda1<- rda(training.x, training.y, alpha = alpha1, delta = delta1)
predic<-predict(rda1, x=training.x, y=training.y, xnew=testing.x, alpha=alpha1, delta=delta1)
#
# Misclassification rate: mcr
mcr<-sum(testing.y != predic)/length(testing.y)
# Mean squared error: mse
mse<-sqrt(sum((testing.y-predic)^2))/length(testing.y)
stats[m,]<-c(mcr,mse)
}
```

```
##
### AVTE: Average Test Error:
AVTE<-mean(stats[,1])
AVTE

### estatisticas das taxas de má classificação:
summary(stats[,1])

### estatisticas das erros quadráticos médios:
summary(stats[,2])

boxplot(stats[,1], ylab="Taxas de má classificação")
#
boxplot(stats[,2], ylab="Erros quadráticos médios")
#
```

C.1.2 RDA - Dados II

```
#########################################################################################################
##### rda.basics
#
rda.basics = function(training.x, training.y, testing.x, testing.y, nrFolds, folds, hpars){
MEtest.ss = data.frame(matrix(0, ncol = length(hpars$delta), nrow = length(hpars$alpha)))
names(MEtest.ss) = round(hpars$delta, 3); row.names(MEtest.ss) = round(hpars$alpha, 3)
for(k in 1:nrFolds){
fold = which(folds == k)
x.train = training.x[, -fold]
y.train = training.y[-fold]
x.test = matrix(training.x[, fold], ncol = length(fold))
y.test = training.y[fold]
MEtest.ss = MEtest.ss +
rda(x.train, y.train, x.test, y.test, alpha = hpars$alpha, delta = hpars$delta)$testerror / length(y.test)
}
MEtest.ss = MEtest.ss / nrFolds
return(MEtest.ss)
}
#

#########################################################################################################
##### Função para gerar um Cross-Validation
#
rda.rep = function(training.x, training.y, testing.x, testing.y, nrFolds, hpars, reps, balanced = F){
err = data.frame(matrix(0, ncol = length(hpars$delta), nrow = length(hpars$alpha)))
names(err) = round(hpars$delta, 3); row.names(err) = round(hpars$alpha, 3)
for(rep in 1:reps){
if(reps > 1)
print(paste0("Iteração: ", rep, " / ", reps))

folds = createFolds(if(balanced){factor(training.y)}else{training.y}, k = nrFolds, list = F)
st.RDA = rda.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, hpars)
err = err + st.RDA
}
err = err / reps
return(err)
}
#


#########################################################################################################
##### Função para gerar uma matriz de confusão
#
roc.curve=function(s, print = FALSE){
Ps = (S > s) * 1
FP = sum((Ps == 1) * (Y == 0)) / sum(Y == 0)
TP = sum((Ps == 1) * (Y == 1)) / sum(Y == 1)

if(print == TRUE){
print(table(Observed = Y, Predicted = Ps))
}
vect = c(FP, TP)
names(vect) = c("FPR", "TPR")
return(vect)
}
#
```

```
#############################################################################################################
##### Função para se gerar um bagging do modelo
#
bagging.rda = function(x, y, tr.index, s.size, iterations, hpars){
testing.x = x[, -tr.index]
testing.y  = y[-tr.index]
predictions = foreach(m = 1:iterations, .combine = cbind) %do% {
print(paste0("Iteração: ", m))
bagg1 = sample(tr.index, s.size, replace = TRUE)
training1.x = x[, bagg1]
training1.y = y[bagg1]

RDA = rda(training1.x, training1.y, alpha = hpars$alpha, delta = hpars$delta)
pred = predict(RDA, x = training1.x, y = training1.y, xnew = testing.x, alpha = hpars$alpha,
delta = hpars$delta)
}
predictions = rowMeans(predictions)
error = sqrt((sum((testing.y - predictions)^2)) / length(testing.y))
list(Predictions = predictions, Error = error)
}
#

#############################################################################################################
##### Análises
#
require(rda)             #Dados/Modelo
require(caret)           #createFolds
require(foreach)         #foreach

###-------------------------------------------------------------------------------------------------------
voice = data.matrix(read.csv2("C:...\\voice.csv",sep= "," , dec=".", header=T))
attr(voice, "dimnames")[[2]] = paste0("X", 1:ncol(voice))
x = t(voice[, 1:(ncol(voice) - 1)])
y = voice[, ncol(voice)]
###-------------------------------------------------------------------------------------------------------
# data(colon)
# x = t(colon.x)
# y = colon.y
###-------------------------------------------------------------------------------------------------------

nrFolds = 5
train.nn = ceiling(ncol(x) * (2 / 3)) #66% para Training
cv.reps = 500                          #Número de repetições no Cross-Validation
nbag = 1000                            #Tamanho do Bagging no treinamento

hpars = list(
alpha = c(seq(0, 0.90, len = 10), 0.99),
delta = seq(0, 1, len = 10)
)

seed1 = 107
set.seed(seed1)
tr.index = sample(1:ncol(x), train.nn)
training.x = x[, tr.index]
training.y = y[tr.index]
testing.x = x[, -tr.index]
testing.y = y[-tr.index]


#############################################################################################################
##### K-Folds - Validação Cruzada Estratificada (SCV)
#
SCV.RDA = rda.rep(training.x, training.y, testing.x, testing.y, nrFolds, hpars, cv.reps, balanced = T)
SCV.RDA
#

#############################################################################################################
##### K-Folds - Validação Cruzada Shuffe and Split (SS)
#
SS.RDA = rda.rep(training.x, training.y, testing.x, testing.y, nrFolds, hpars, cv.reps)
SS.RDA
#

#############################################################################################################
##### K-Folds - Validação Cruzada Leave-One-Out (LOOCV)
```

```
#
LOOCV.RDA = rda.rep(training.x, training.y, testing.x, testing.y, length(training.y), hpars, reps = 1)
LOOCV.RDA
#

#########################################################################################################
##### Valores ótimos para os hiper-parâmetros, utilizados para as análises a seguir
#
result = data.frame(CVtype = c("SCV", "SS", "LOOCV"),
alpha = hpars$alpha[c(which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 1],
which(SS.RDA == min(SS.RDA), arr.ind = TRUE)[1, 1],
which(LOOCV.RDA == min(LOOCV.RDA), arr.ind = TRUE)[1, 1])],
delta = hpars$delta[c(which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 2],
which(SS.RDA == min(SS.RDA), arr.ind = TRUE)[1, 2],
which(LOOCV.RDA == min(LOOCV.RDA), arr.ind = TRUE)[1, 2])],
error = c(SCV.RDA[which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 1],
which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 2]],
SS.RDA[which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 1],
which(SCV.RDA == min(SCV.RDA), arr.ind = TRUE)[1, 2]],
LOOCV.RDA[which(LOOCV.RDA == min(LOOCV.RDA), arr.ind = TRUE)[1, 1],
which(LOOCV.RDA == min(LOOCV.RDA), arr.ind = TRUE)[1, 2]]))
result

hpars = list(
alpha = 0.99,
delta = 1
)
#

#########################################################################################################
##### Matriz de confusão
#
RDA = rda(training.x, training.y, alpha = hpars$alpha, delta = hpars$delta)
pred = predict(RDA, x = training.x, y = training.y, xnew = testing.x, alpha = hpars$alpha, delta = hpars$delta)
print(paste0("Test Error: ", 1 - mean(pred == testing.y)))

check = as.numeric(attr(predict(RDA, x = training.x, y = training.y, xnew = testing.x, alpha = hpars$alpha,
delta = hpars$delta, type = "posterior"), "dimnames")[[2]])
S = as.vector(predict(RDA, x = training.x, y = training.y, xnew = testing.x, alpha = hpars$alpha,
delta = hpars$delta, type = "posterior")[, seq(1, length(check))[check == max(check)]])
Y = testing.y - 1
threshold = 0.5
roc.curve(threshold, print = T)
#

#########################################################################################################
##### ROC Curve
#
ROC.curve = Vectorize(roc.curve)
M.ROC = ROC.curve(seq(0, 1, by = .01))
plot(M.ROC[1,], M.ROC[2,], col = "blue", lwd = 2, type = "l", xlab = "Taxa de falsos-positivos",
ylab = "Taxa de verdadeiros-positivos")
#

#########################################################################################################
##### Bagging
#
st.RDA = bagging.rda(x, y, tr.index, length(tr.index), nbag, hpars)
#

#########################################################################################################
##### Model Performance
#
iterations = 1000
stats = matrix(NA, iterations, 2)
for(it in 1:iterations){
tr.index = sample(1:ncol(x), train.nn)

RDA = rda(x[, tr.index], y[tr.index], alpha = hpars$alpha, delta = hpars$delta)
pred = predict(RDA, x = x[, tr.index], y = y[tr.index], xnew = x[, -tr.index],
alpha = hpars$alpha, delta = hpars$delta)

mcr = 1 - mean(pred == y[-tr.index])
mse = sqrt(sum((y[-tr.index] - pred)^2)) / length(pred)
stats[it, ] = c(mcr, mse)
}
```

```
### AVTE: Average Test Error
AVTE = mean(stats[, 1])
AVTE

### Estatisticas das taxas de má classificação
summary(stats[, 1])

### Estatisticas dos erros quadráticos médios
summary(stats[, 2])

### Boxplot
boxplot(stats[, 1], ylab = "Taxas de má classificação")
boxplot(stats[, 2], ylab = "Erros quadráticos médios")
```

C.2 Máquina de Vetor de Suporte (SVM)

```
###############################################################################################################
##### svm.basics
#
svm.basics = function(training.x, training.y, testing.x, testing.y, nrFolds, folds, kernel, hpars){
MEtest.ss = 0
for(k in 1:nrFolds) {
fold = which(folds == k)

x.train = training.x[-fold, ]
y.train = training.y[-fold]

x.test = matrix(training.x[fold, ], nrow = length(fold))
y.test = training.y[fold]

tune.out = tune(svm, y.train~., data = data.frame(cbind(x.train, y.train)),
validation.x = x.test, validation.y = y.test,
kernel = kernel, ranges = hpars, tunecontrol = tune.control(sampling = "fix", fix = 1))

MEtest.ss = MEtest.ss + tune.out$performances$error
}
MEtest.ss = data.frame(cbind(tune.out$performances[, 1:length(hpars)], MEtest.ss / nrFolds))
names(MEtest.ss) = c(names(hpars), "error")
return(MEtest.ss)
}
#

###############################################################################################################
##### Função para gerar um Cross-Validation
#
svm.rep = function(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, reps, balanced = F){
err = 0
for(rep in 1:reps){
if(reps > 1)
print(paste0("Iteração: ", rep, " / ", reps))

folds = createFolds(if(balanced){factor(training.y)}else{training.y}, k = nrFolds, list = F)
st.SVM = svm.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, kernel, hpars)
err = err + st.SVM$error
}
err = data.frame(cbind(st.SVM[, 1:length(hpars)], err / reps))
names(err) = c(names(hpars), "error")
return(err[order(err$error),])
}
#


###############################################################################################################
##### Função para gerar uma matriz de confusão
#
roc.curve = function(s, print = FALSE){
Ps = (S > s) * 1
FP = sum((Ps == 1) * (Y == 0)) / sum(Y == 0)
TP = sum((Ps == 1) * (Y == 1)) / sum(Y == 1)

if(print == TRUE){
print(table(Observed = Y, Predicted = Ps))
}
```

```
vect = c(FP, TP)
names(vect) = c("FPR", "TPR")
return(vect)
}
#

#########################################################################################################
##### Função para se gerar um bagging do modelo
#
bagging.svm = function(x, y, tr.index, s.size, iterations, kernel, cost, gamma = 1){
testing.x = x[-tr.index, ]
testing.y = y[-tr.index]
predictions = foreach(m = 1:iterations, .combine = cbind) %do% {
print(m)
bagg1 = sample(tr.index, s.size, replace=TRUE)
training1.x = x[bagg1, ]
training1.y = y[bagg1]

SVM = svm(x = training1.x, training1.y, type = "C-classification",
kernel = kernel, cost = cost, gamma = gamma)
pred = predict(SVM, testing.x)
}
predictions = rowMeans(predictions)
error = sqrt((sum((testing.y - predictions)^2)) / length(testing.y))
list(predictions, error)
}
#

#########################################################################################################
##### Análises
#
require(rda)      #Dados
require(e1071)    #Modelo
require(caret)    #createFolds
require(foreach)  #foreach


###-------------------------------------------------------------------------------------------------------
voice = data.matrix(read.csv2("C:\\...\\voice.csv",sep= "," , dec=".", header=T))
attr(voice, "dimnames")[[2]] = paste0("X", 1:ncol(voice))
x = voice[, 1:(ncol(voice) - 1)]
y = voice[, ncol(voice)]
###-------------------------------------------------------------------------------------------------------
#data(colon)
#x = colon.x
#y = colon.y
###-------------------------------------------------------------------------------------------------------


nrFolds = 5
train.nn = ceiling(nrow(x) * (2 / 3))
cv.reps = 500        #Número de repetições no Cross-Validation
nbag = 1000          #Tamanho do Bagging no treinamento

kernel = "linear"     #linear / polynomial / radial basis / sigmoid
hpars = list(
cost = c(.1, .5)         #linear / polynomial / radial basis / sigmoid
#gamma = 1/ncol(x)       #       / polynomial / radial basis / sigmoid
#degree = 1              #       / polynomial /              /
#coef0 = 0               #       / polynomial /              / sigmoid
)

seed1 = 107
set.seed(seed1)
tr.index = sample(1:nrow(x), train.nn)
training.x = x[tr.index ,]
training.y = y[tr.index]
testing.x = x[-tr.index ,]
testing.y = y[-tr.index]


#########################################################################################################
##### K-Folds - Validação Cruzada Estratificada (SCV)
#
SCV.SVM = svm.rep(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, cv.reps, balanced = T)
SCV.SVM
```

```
#

#########################################################################################################
##### K-Folds - Validação Cruzada Shuffe and Split (SS)
#
SS.SVM = svm.rep(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, cv.reps)
SS.SVM
#

#########################################################################################################
##### K-Folds - Validação Cruzada Leave-One-Out (LOOCV)
#
LOOCV.SVM = svm.rep(training.x, training.y, testing.x, testing.y, length(training.y), kernel, hpars, reps = 1)
LOOCV.SVM
#

#########################################################################################################
##### Valores ótimos para os hiper-parâmetros, utilizados para as análises a seguir
#
cost = .1
#gamma
#

#########################################################################################################
##### Matriz de confusão
#
SVM = svm(x = training.x, training.y, type = "C-classification", kernel = kernel, cost = cost, probability = T)
pred = predict(SVM, testing.x)
print(paste0("Test Error: ", 1 - mean(pred == testing.y)))

check = as.numeric(attr(attr(predict(SVM, testing.x, probability = T), "probabilities"), "dimnames")[[2]])
S = as.vector(attr(predict(SVM, testing.x, probability = T), "probabilities")
[, seq(1, length(check))[check == max(check)]])
Y = testing.y - 1
threshold = 0.5
roc.curve(threshold, print = T)
#

#########################################################################################################
##### ROC Curve
#
ROC.curve = Vectorize(roc.curve)
M.ROC = ROC.curve(seq(0, 1, by = .01))
plot(M.ROC[1,], M.ROC[2,], col = "blue", lwd = 2, type = "l",
xlab = "Taxa de falsos-positivos", ylab = "Taxa de verdadeiros-positivos")
#

#########################################################################################################
##### Bagging
#
st.SVM = bagging.svm(x, y, tr.index, length(tr.index), nbag, kernel, cost = cost)
#

#########################################################################################################
##### Model Performance
#
iterations = 1000
stats = matrix(NA, iterations, 2)
for(it in 1:iterations){
tr.index = sample(1:nrow(x), train.nn)
SVM = svm(x = x[tr.index ,], y = y[tr.index], type = "C-classification", kernel = kernel, cost = cost)
pred = as.numeric(as.character(predict(SVM, x[-tr.index ,])))
mcr = 1 - mean(pred == y[-tr.index])
mse = sqrt(sum((y[-tr.index] - pred)^2)) / length(pred)
stats[it, ] = c(mcr, mse)
}

### AVTE: Average Test Error
AVTE = mean(stats[, 1])
AVTE

### Estatisticas das taxas de má classificação
summary(stats[, 1])

### Estatisticas dos erros quadráticos médios
summary(stats[, 2])
```

```
### Boxplot
boxplot(stats[, 1], ylab = "Taxas de má classificação")
boxplot(stats[, 2], ylab = "Erros quadráticos médios")
```

C.2.1 SVM - com implementação da significância estatística via Regressão Logística

```
#############################################################################################################
#Função para ajustar um modelo de Reg. Logística nos dados e retornar as variáveis significantes via p-valor
#
sigVars = function(x, y, scale = FALSE, sig = 0.05){
temp = data.frame(cbind(x, y))
if(scale)
temp = data.frame(cbind(scale(x), y))

fit = glm(paste0(names(temp)[ncol(temp)],"~",paste(names(temp)[1:(ncol(temp)-1)], collapse = "+")),
data = temp, family = "binomial")
p = summary(fit)$coefficients[-1, 4]
return(names(p)[p <= sig])
}

#############################################################################################################
##### Função base para treinar um SVM e retornar sua performace calculada nos dados de validação
#
svm.basics = function(training.x, training.y, testing.x, testing.y, nrFolds, folds, kernel, hpars){
MEtest.ss = 0
for(k in 1:nrFolds) {
fold = which(folds == k)

x.train = training.x[-fold, ]
y.train = training.y[-fold]

x.test = matrix(training.x[fold, ], nrow = length(fold))
y.test = training.y[fold]

tune.out = tune(svm, y.train~., data = data.frame(cbind(x.train, y.train)),
validation.x = x.test, validation.y = y.test,
kernel = kernel, ranges = hpars,
tunecontrol = tune.control(sampling = "fix", fix = 1))

MEtest.ss = MEtest.ss + tune.out$performances$error
}
MEtest.ss = data.frame(cbind(tune.out$performances[, 1:length(hpars)], MEtest.ss / nrFolds))
names(MEtest.ss) = c(names(hpars), "error")
return(MEtest.ss)
}
#

#############################################################################################################
##### Função para gerar um Cross-Validation
#
svm.rep = function(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, reps, balanced = F){
err = 0
for(rep in 1:reps){
if(reps > 1)
print(paste0("Iteração: ", rep, " / ", reps))

folds = createFolds(if(balanced){factor(training.y)}else{training.y}, k = nrFolds, list = F)
st.SVM = svm.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, kernel, hpars)
err = err + st.SVM$error
}
err = data.frame(cbind(st.SVM[, 1:length(hpars)], err / reps))
names(err) = c(names(hpars), "error")
return(err[order(err$error),])
}
#


#############################################################################################################
##### Função para gerar uma matriz de confusão
#
roc.curve = function(s, print = FALSE){
```

```
Ps = (S > s) * 1
FP = sum((Ps == 1) * (Y == 0)) / sum(Y == 0)
TP = sum((Ps == 1) * (Y == 1)) / sum(Y == 1)

if(print == TRUE){
print(table(Observed = Y, Predicted = Ps))
}
vect = c(FP, TP)
names(vect) = c("FPR", "TPR")
return(vect)
}
#

#########################################################################################################
##### Função para se gerar um bagging do modelo
#
bagging.svm = function(x, y, tr.index, s.size, iterations, kernel, cost, gamma = 1){
testing.x = x[-tr.index, ]
testing.y = y[-tr.index]
predictions = foreach(m = 1:iterations, .combine = cbind) %do% {
print(m)
bagg1 = sample(tr.index, s.size, replace=TRUE)
training1.x = x[bagg1, ]
training1.y = y[bagg1]

SVM = svm(x = training1.x, training1.y, type = "C-classification",
kernel = kernel, cost = cost, gamma = gamma)
pred = predict(SVM, testing.x)
}
predictions = rowMeans(predictions)
error = sqrt((sum((testing.y - predictions)^2)) / length(testing.y))
list(predictions, error)
}
#

#########################################################################################################
##### Análises
#
require(rda)      #Dados
require(e1071)    #Modelo
require(caret)    #createFolds
require(foreach)  #foreach


###-------------------------------------------------------------------------------------------------------
voice = data.matrix(read.csv2("C:...\\voice.csv", sep= "," , dec=".", header=T))
attr(voice, "dimnames")[[2]] = paste0("X", 1:ncol(voice))
x = voice[, 1:(ncol(voice) - 1)]
y = voice[, ncol(voice)]
###-------------------------------------------------------------------------------------------------------
#data(colon)
#x = colon.x
#y = colon.y
###-------------------------------------------------------------------------------------------------------

vsig = sigVars(x, y - 1) #Passar y em (0,1)
x = x[, vsig]

nrFolds = 5
train.nn = ceiling(nrow(x) * (2 / 3))
cv.reps = 500        #Número de repetições no Cross-Validation
nbag = 1000          #Tamanho do Bagging no treinamento

kernel = "linear"    #linear / polynomial / radial basis / sigmoid
hpars = list(
cost = c(.1, .5)  #linear / polynomial / radial basis / sigmoid
#gamma = 1/c(2000, 1000, 500, 10, .5)       #       / polynomial / radial basis / sigmoid
#degree = 1                                 #       / polynomial /              /
#coef0 = 0                                  #       / polynomial /              / sigmoid
)

seed1 = 107
set.seed(seed1)
tr.index = sample(1:nrow(x), train.nn)
training.x = x[tr.index ,]
training.y = y[tr.index]
```

```
testing.x = x[-tr.index ,]
testing.y = y[-tr.index]


#############################################################################################################
##### K-Folds - Validação Cruzada Estratificada (SCV)
#
SCV.SVM = svm.rep(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, cv.reps, balanced = T)
SCV.SVM
#

#############################################################################################################
##### K-Folds - Validação Cruzada Shuffe and Split (SS)
#
SS.SVM = svm.rep(training.x, training.y, testing.x, testing.y, nrFolds, kernel, hpars, cv.reps)
SS.SVM
#

#############################################################################################################
##### K-Folds - Validação Cruzada Leave-One-Out (LOOCV)
#
LOOCV.SVM = svm.rep(training.x, training.y, testing.x, testing.y, length(training.y), kernel, hpars, reps = 1)
LOOCV.SVM
#

#############################################################################################################
##### Valores ótimos para os hiper-parâmetros, utilizados para as análises a seguir
#
cost = .1
#gamma
#

#############################################################################################################
##### Matriz de confusão
#
SVM = svm(x = training.x, training.y, type = "C-classification", kernel = kernel, cost = cost, probability = T)
pred = predict(SVM, testing.x)
print(paste0("Test Error: ", 1 - mean(pred == testing.y)))

check = as.numeric(attr(attr(predict(SVM, testing.x, probability = T), "probabilities"), "dimnames")[[2]])
S = as.vector(attr(predict(SVM, testing.x, probability = T), "probabilities")
[, seq(1, length(check))[check == max(check)]])
Y = testing.y - 1
threshold = 0.5
roc.curve(threshold, print = T)
#

#############################################################################################################
##### ROC Curve
#
ROC.curve = Vectorize(roc.curve)
M.ROC = ROC.curve(seq(0, 1, by = .01))
plot(M.ROC[1,], M.ROC[2,], col = "red", lwd = 2, type = "l",
xlab = "Taxa de falsos-positivos", ylab = "Taxa de verdadeiros-positivos")
#

#############################################################################################################
##### Bagging
#
st.SVM = bagging.svm(x, y, tr.index, length(tr.index), nbag, kernel, cost = cost)
#

#############################################################################################################
##### Model Performance
#
iterations = 1000
stats = matrix(NA, iterations, 2)
for(it in 1:iterations){
tr.index = sample(1:nrow(x), train.nn)
SVM = svm(x = x[tr.index ,], y = y[tr.index], type = "C-classification", kernel = kernel, cost = cost)
pred = as.numeric(as.character(predict(SVM, x[-tr.index ,])))
mcr = 1 - mean(pred == y[-tr.index])
mse = sqrt(sum((y[-tr.index] - pred)^2)) / length(pred)
stats[it, ] = c(mcr, mse)
}
```

```
### AVTE: Average Test Error
AVTE = mean(stats[, 1])
AVTE

### Estatisticas das taxas de má classificação
summary(stats[, 1])

### Estatisticas dos erros quadráticos médios
summary(stats[, 2])

### Boxplot
boxplot(stats[, 1], ylab = "Taxas de má classificação")
boxplot(stats[, 2], ylab = "Erros quadráticos médios")
```

C.3 Florestas Aleatórias (FA)

```
#########################################################################################################
##### rf.basics
#
rf.basics = function(training.x, training.y, testing.x, testing.y, nrFolds, folds, hpars){
MEtest.ss = data.frame(expand.grid(hpars))
MEtest.ss$error = 0
for(k in 1:nrFolds){
fold = which(folds == k)

x.train = training.x[-fold, ]
y.train = training.y[-fold]

x.test = matrix(training.x[fold, ], nrow = length(fold))
y.test = training.y[fold]

for(i in 1:nrow(MEtest.ss)){
aux = data.frame(x.train, y.train); aux$y.train = as.factor(aux$y.train)
rf = randomForest(y.train~. , data = aux, ntree = MEtest.ss$ntree[i], mtry = MEtest.ss$mtry[i],
nodesize = MEtest.ss$nodesize[i])
err = mean(predict(rf, data.frame(x.test)) != y.test)
MEtest.ss$error[i] = MEtest.ss$error[i] + err
}
}
MEtest.ss$error = MEtest.ss$error / nrFolds
return(MEtest.ss)
}
#

#########################################################################################################
##### Função para gerar um Cross-Validation
#
rf.rep = function(training.x, training.y, testing.x, testing.y, nrFolds, hpars, reps, balanced = F){
err = 0
for(rep in 1:reps){
if(reps > 1)
print(paste0("Iteração: ", rep, " / ", reps))

folds = createFolds(if(balanced){factor(training.y)}else{training.y}, k = nrFolds, list = F)
st.RF = rf.basics(training.x, training.y, testing.x, testing.y, nrFolds, folds, hpars)
err = err + st.RF$error
}
err = data.frame(cbind(st.RF[, 1:length(hpars)], err / reps))
names(err) = c(names(hpars), "error")
return(err[order(err$error),])
}
#


#########################################################################################################
##### Função para gerar uma matriz de confusão
#
roc.curve=function(s, print = FALSE){
Ps = (S > s) * 1
FP = sum((Ps == 1) * (Y == 0)) / sum(Y == 0)
TP = sum((Ps == 1) * (Y == 1)) / sum(Y == 1)

if(print == TRUE){
print(table(Observed = Y, Predicted = Ps))
```

```
}
vect = c(FP, TP)
names(vect) = c("FPR", "TPR")
return(vect)
}
#

#########################################################################################################
##### Função para se gerar um bagging do modelo
#
bagging.rf = function(x, y, tr.index, s.size, iterations, hpars){
testing.x = x[-tr.index, ]
testing.y  = y[-tr.index]
predictions = foreach(m = 1:iterations, .combine = cbind) %do% {
print(paste0("Iteração: ", m))
bagg1 = sample(tr.index, s.size, replace=TRUE)
training1.x = x[bagg1, ]
training1.y = y[bagg1]

aux = data.frame(cbind(training1.x, as.factor(training1.y)))
aux[ncol(aux)] = as.factor(aux[[ncol(aux)]])
RF = randomForest(aux[-ncol(aux)], as.factor(aux[[ncol(aux)]]), data = aux, ntree = hpars$ntree,
mtry = hpars$mtry, nodesize = hpars$nodesize)
pred = predict(RF, data.frame(testing.x))
}
predictions = rowMeans(predictions)
error = sqrt((sum((testing.y - predictions)^2)) / length(testing.y))
list(Predictions = predictions, Error = error)
}
#

#########################################################################################################
##### Análises
#
require(rda)
require(randomForest)
require(caret)           #createFolds
require(foreach)


###-------------------------------------------------------------------------------------------------------
voice = data.matrix(read.csv2("C:...\\voice.csv", sep= "," , dec=".", header=T))
attr(voice, "dimnames")[[2]] = paste0("X", 1:ncol(voice))
x = voice[, 1:(ncol(voice) - 1)]
y = voice[, ncol(voice)]
###-------------------------------------------------------------------------------------------------------
#data(colon)
#x = colon.x
#y = colon.y
###-------------------------------------------------------------------------------------------------------

nrFolds = 5
train.nn = ceiling(nrow(x) * (2 / 3))
cv.reps = 500                           #Número de repetições no Cross-Validation
nbag = 1000                             #Tamanho do Bagging no treinamento

hpars = list(
mtry = floor(sqrt(ncol(x))),        #Default randomForest
ntree = 280,
nodesize = 14
)

seed1 = 107
set.seed(seed1)
tr.index = sample(1:nrow(x), train.nn)
training.x = x[tr.index ,]
training.y = y[tr.index]
testing.x = x[-tr.index ,]
testing.y = y[-tr.index]


#########################################################################################################
##### K-Folds - Validação Cruzada Estratificada (SCV)
#
SCV.RF = rf.rep(training.x, training.y, testing.x, testing.y, nrFolds, hpars, cv.reps, balanced = T)
SCV.RF
```

```
#

#########################################################################################################
##### K-Folds - Validação Cruzada Shuffe and Split (SS)
#
SS.RF = rf.rep(training.x, training.y, testing.x, testing.y, nrFolds, hpars, cv.reps)
SS.RF
#

#########################################################################################################
##### K-Folds - Validação Cruzada Leave-One-Out (LOOCV)
#
LOOCV.RF = rf.rep(training.x, training.y, testing.x, testing.y, length(training.y), hpars, reps = 1)
LOOCV.RF
#

#########################################################################################################
##### Valores ótimos para os hiper-parâmetros, utilizados para as análises a seguir
#
hpars = list(
mtry = floor(sqrt(ncol(x))),
ntree = 280,
nodesize = 14
)
#

#########################################################################################################
##### Matriz de confusão
#
aux = data.frame(cbind(training.x, as.factor(training.y)))
RF = randomForest(aux[-ncol(aux)], as.factor(aux[[ncol(aux)]]), data = aux, ntree = hpars$ntree,
mtry = hpars$mtry, nodesize = hpars$nodesize)
testing.x = data.frame(testing.x)
pred = predict(RF, testing.x)
print(paste0("Test Error: ", 1 - mean(pred == testing.y)))

check = as.numeric(attr(predict(RF, testing.x, type = "prob"), "dimnames")[[2]])
S = as.vector(predict(RF, testing.x, type = "prob")[, seq(1, length(check))[check == max(check)]])
Y = testing.y - 1
threshold = 0.5
roc.curve(threshold, print = T)
#

#########################################################################################################
##### ROC Curve
#
ROC.curve = Vectorize(roc.curve)
M.ROC = ROC.curve(seq(0, 1, by = .01))
plot(M.ROC[1,], M.ROC[2,], col = "black", lwd = 2, type = "l", xlab = "Taxa de falsos-positivos",
ylab = "Taxa de verdadeiros-positivos")
#

#########################################################################################################
##### Bagging
#
st.RF = bagging.rf(x, y, tr.index, length(tr.index), nbag, hpars)
#

#########################################################################################################
##### Model Performance
#
iterations = 1000
stats = matrix(NA, iterations, 2)
for(it in 1:iterations){
tr.index = sample(1:nrow(x), train.nn)
aux = data.frame(cbind(x[tr.index ,], as.factor(y[tr.index])))
aux[ncol(aux)] = as.factor(aux[[ncol(aux)]])
RF = randomForest(aux[-ncol(aux)], as.factor(aux[[ncol(aux)]]), ntree = hpars$ntree, mtry = hpars$mtry,
nodesize = hpars$nodesize)
pred = as.numeric(as.character(predict(RF, data.frame(x[-tr.index ,]))))
mcr = 1 - mean(pred == y[-tr.index])
mse = sqrt(sum((y[-tr.index] - pred)^2)) / length(pred)
stats[it, ] = c(mcr, mse)
}

### AVTE: Average Test Error
```

```
AVTE = mean(stats[, 1])
AVTE

### Estatisticas das taxas de má classificação
summary(stats[, 1])

### Estatisticas dos erros quadráticos médios
summary(stats[, 2])

### Boxplot
boxplot(stats[, 1], ylab = "Taxas de má classificação")
boxplot(stats[, 2], ylab = "Erros quadráticos médios")
```