Universidade do Estado do Rio de Janeiro

Centro de Tecnologia e Ciências

Instituto de Química

Bárbara Martins de Andrade

Comparação do desempenho de classificadores multiclasses em dados químicos: abordando o problema de sobreajuste com o teste de permutação

Rio de Janeiro

2020

Bárbara Martins de Andrade

Comparação do desempenho de classificadores multiclasses em dados químicos: abordando o problema de sobreajuste com o teste de permutação

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Química, da Universidade do Estado do Rio de Janeiro. Área de concentração: Bioprocessos e Tecnologia Química.

Orientadores: Prof. Dr. Aderval Severino Luna

Prof. Dr. Jefferson Santos de Gois

Prof. Dr. Vinicius Layter Xavier

Rio de Janeiro

2020

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/Q

A553 Andrade, Bárbara Martins de.
Comparação do desempenho de classificadores multiclasses em dados químicos: abordando o problema de sobreajuste com o teste de permutação. – 2020.
131 f.

Orientador: Aderval Severino Luna
Orientador: Jefferson Santos de Gois
Orientador: Vinicius Layter Xavier

Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro. Instituto de Química.

1. Quimiometria – Teses. 2. Vinho – Teses. I. Luna, Aderval Severino. II. Gois, Jefferson Santos de. III. Xavier, Vinicius Layter. IV. Universidade do Estado do Rio de Janeiro. Instituto de Química. V. Título.

CDU 54.08::663.2

Bárbara Martins de Andrade
_____
Assinatura

26/11/2020
_____
Data

Bárbara Martins de Andrade

Comparação do desempenho de classificadores multiclasses em dados químicos: abordando o problema de sobreajuste com o teste de permutação

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Engenharia Química, da Universidade do Estado do Rio de Janeiro. Área de concentração: Bioprocessos e Tecnologia Química.

Aprovado 18 de fevereiro de 2020.

Banca Examinadora:

Prof. Dr. Aderval Severino Luna (Orientador)
Programa de Pós-Graduação em Engenharia Química – UERJ

Prof. Dr. Jefferson Santos de Gois (Orientador)
Programa de Pós-Graduação em Engenharia Química – UERJ

Prof. Dr. Vinicius Layter Xavier (Orientador)
Instituto de Matemática e Estatística – UERJ

Prof. Dr. Alexandre Rodrigues Torres
Faculdade de Tecnologia – UERJ

Prof. Dr. André Luiz Hemerly Costa
Programa de Pós-Graduação em Engenharia Química – UERJ

Dr. Leonardo Silva de Souza
COPPE – UFRJ

Rio de Janeiro

2020

DEDICATÓRIA

A Deus, pela força e coragem e ao meu Pai, pelo qual cultivo amor entre o Céu e a Terra.

AGRADECIMENTOS

A conclusão do mestrado não poderia chegar a bom porto sem o precioso apoio de várias pessoas como meus orientadores, Prof. Luna, Prof. Jefferson e Prof. Vinicius, por toda ajuda e demonstração de força de vontade, pela excelente orientação, apontando os melhores caminhos, dando estímulos para o desenvolvimento deste trabalho e pela motivação constante.

Aos professores, pelos ensinamentos na minha vida acadêmica, aliados ao meu desenvolvimento pessoal, durante a época da graduação e agora do mestrado. Não conheço missão maior e mais nobre.

A minha família e amigos pela paciência e incentivo com os meus estudos, em especial meus tios Jucimar e Juliana.

Ao meu querido esposo Pedro pelo apoio incondicional em todos os momentos, principalmente nos de incerteza, muito comuns para quem tenta trilhar novos caminhos.

Aos meus colegas de mestrado, pelo companheirismo, receptividade e pelo inegável apoio quando necessário.

A UERJ, porque sem ela não poderia ter realizado este sonho de conquista.

A todos aqueles, que embora não citados nominalmente, contribuíram direta e indiretamente para a execução deste trabalho.

Não há acaso, sina, destino, que possa limitar, impedir ou controlar a firme resolução de uma alma determinada.

Ella Wheller Wilcox

RESUMO

ANDRADE, B. M. Comparação do desempenho de classificadores multiclasses em dados químicos: abordando o problema de sobreajuste com o teste de permutação. 2020. 131 f. Dissertação (Mestrado em Engenharia Química) – Instituto de Química, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

O objetivo deste trabalho foi aplicar técnicas de reconhecimento de padrões em três conjuntos de dados disponíveis na literatura. O conjunto de dados denominado *Glass Identification Dataset* foi avaliado usando-se os modelos de classificação: Análise Discriminante Linear, Análise Discriminante Regularizada, Análise Discriminante de Mistura, Análise Discriminante de Mínimos Quadrados Parciais, Redes Neurais Artificiais, Máquina de Vetor de Suporte com função de núcleo radial, K-vizinhos mais próximos, Naive Bayes, Vetor de Quantização de Aprendizagem de Máquina, Árvore de Classificação e Regressão, *Bagging*, Floresta Aleatória, C5.0 e *Generalized Boosted Machine* usando-se como parâmetros de mérito a exatidão e o índice Kappa. O classificador Vetor de Quantização de Aprendizagem de Máquina (OLVQ1) foi o único modelo que forneceu 100% de exatidão e o índice Kappa igual a 1,000. Tendo em vista o resultado obtido, foi proposta uma nova abordagem para o teste de permutação para verificar a ocorrência de sobreajuste do modelo, a qual indicou a inexistência do sobreajuste. O conjunto de dados denominado *Wine Quality Dataset* foi dividido em Vinho Tinto e Vinho Branco. Para os dois conjuntos de vinho foram testados os modelos: Regressão por Mínimos Quadrados Parciais, Máquina de Vetor de Suporte com função de núcleo radial, Máquina de Vetor de Suporte com função linear, K-vizinhos mais próximos, Árvore de Classificação e Regressão, *Bagging*, Floresta Aleatória e *Generalized Boosted Machine* usando-se como parâmetros de mérito o erro absoluto médio (MAE), a raiz do erro quadrático médio (RMSE) e o coeficiente de determinação (R^2). Para todos os modelos testados, os resultados foram insatisfatórios. Em função dos resultados dos modelos de regressão preditivos, foram empregados os mesmos modelos de classificação usados no conjunto de dados Glass Identification Dataset. Mais uma vez, o classificador Vetor de Quantização de Aprendizagem de Máquina (OLVQ1) foi o modelo com melhor desempenho indicando igual a exatidão de 99,54% e 99,33% e o índice Kappa igual a 0,9932 e 0,9896 para os Vinhos Branco e Tinto, respectivamente. Os resultados obtidos superaram a classificação dos trabalhos publicados anteriormente na literatura. Portanto, os experimentos computacionais mostram que o Vetor de Quantização de Aprendizagem de Máquina foi o único método capaz de classificar corretamente todos os três conjuntos de dados, atingindo a exatidão de aproximadamente 100% e o índice Kappa perfeito. Os resultados mostraram que o problema do excesso de ajuste estava ausente, o que foi confirmado pelo teste de Wilcoxon pareado.

Palavras-chave: Reconhecimento de padrões. Regressão. Sobreajuste. Teste de permutação.

ABSTRACT

ANDRADE, B. M. Comparison of the performance of multiclass classifiers in chemical data: addressing the problem of overfitting with the permutation test. 2020. 131 f. Dissertação (Mestrado em Engenharia Química) – Instituto de Química, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2020.

This work aimed to apply pattern recognition techniques to three sets of data available in the literature. The dataset called Glass Identification Dataset was evaluated using the classification models: Linear Discriminating Analysis, Regularized Discriminating Analysis, Mixture Discriminating Analysis, Discriminating Analysis of Partial Least Squares, Artificial Neural Networks, Support Vector Machine with the function of the radial core, K-nearest neighbors, Naive Bayes, Learning Vector Quantization, Classification, and Regression Tree, Bagging, Random Forest, C5.0 and Generalized Boosted Machine using accuracy and Kappa index as figures of merit. The Learning Vector Quantization (OLVQ1) classifier was the only model that provided 100% accuracy and the Kappa index equal to 1,000. Because of the result obtained, a new approach for the permutation test was proposed to verify the occurrence of overfitting of the model, which indicated the absence of overfitting. The dataset called Wine Quality Dataset was divided into Red Wine and White Wine. Regression models for both sets of wine were tested: Partial Least Squares, Support Vector Machine with radial base function, Support Vector Machine with a linear function, K-nearest neighbors, Classification and Regression Tree, Bagging, Random Forest and Generalized Boosted Machine using the mean absolute error (MAE), the root of the mean square error (RMSE) and the coefficient of determination (R^2) as figures of merit. For all models tested, the results were unsatisfactory. Because of the results of the predictive regression models, the same classification models used in the Glass Identification Dataset data set were used. The obtained results outperformed the classification concerning previously works published in the literature. Once again, the Learning Vector Quantization (OLVQ1) classifier was the model with the best performance indicating the accuracy of 99.54% and 99.33% and the Kappa index equal to 0.9932 and 0.9896 for White and Red Wines, respectively. Therefore, the computational experiments show that the Learning Vector Quantification was the one method able to correctly classify all three data sets, reaching the accuracy of approximately 100% and the perfect Kappa index. The results showed that the overfitting problem was absent, which was confirmed by the pairwise Wilcoxon signed-rank test.

Keywords: Pattern Recognition. Regression. Overfitting. Permutation test.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	Rede neural artificial (do inglês, *Artificial Neural Network*)
CART	Árvores de classificação e regressão (do inglês, *Classification and Regression Trees)*
EDX	Espectroscopia de raios X por energia dispersiva
FDR	Intervalo de dados do recurso, do inglês *feature data range*
FN	Falsos negativos
FP	Falsos positivos
GBM	Máquina generalizada boosted (do inglês, *Generalized Boosted Models*)
GLVQ	Vetor de Quantização de Aprendizagem de Máquina Generalizado (do inglês, *Generalized Learning Vector Quantization)*
GNU	Licença pública geral
HNB	Do inglês, *Hidden Naïve Bayes*
k-NN	k- Vizinhos mais próximos (do inglês, *K-Nearest Neighbor*)
LDA	Análise discriminante linear (do inglês, *Linear Discriminant Analysis)*
LVQ	Vetor de Quantização de Aprendizagem de Máquina (do inglês, *Learning Vector Quantization)*
MAE	Erro absoluto médio
MDA	Análise discriminante de mistura (do inglês, *Mixture Discriminant Analysis)*
NB	Naive Bayes
OLVQ1	Vetor de Quantização de Aprendizagem de Máquina Otimizado (do inglês, *Optimized Learning Vector Quantization)*
PC	Componente principal
PCA	Análise de componentes principais
pH	Potencial hidrogeniônico
PLS-DA	Análise Discriminante por Mínimos Quadrados Parciais (do inglês, *Partial Least Squares Discriminant Analysis)*
R^2	Coeficiente de determinação

RDA	Análise discriminante regularizada (do inglês, *Regularised Discriminant Analysis)*
RF	Floresta aleatória (do inglês, *Random Forest*)
RI	Índice de refração
RMSE	Raiz do erro médio quadrático
ROBPCA	Análise de componentes principais robusta
SEM	Microscopia de varredura eletrônica
SVM	Máquina de vetores de suporte (do inglês, *Support Vector Machines*)
TN	Verdadeiros negativos
TP	Verdadeiros positivos
UCI	Universidade da Califórnia, Irvine
wt	Peso (do inglês, *weight)*

SUMÁRIO

INTRODUÇÃO

Em diversos setores da química é necessário o estudo de propriedades de um determinado produto, as quais podem estar diretamente relacionadas a inúmeras variáveis. A utilização de ferramentas estatísticas multivariadas é uma valiosa alternativa no estudo de casos para auxiliar na interpretação das relações existentes.

Neste contexto, um dos desafios para os engenheiros químicos é desenvolver sistemas para auxiliarem em análises químicas visando facilitar a interpretação e tomada de decisão. Métodos quimiométricos são particularmente úteis, os quais empregam métodos matemáticos e estatísticos para definir ou selecionar as condições ótimas de medidas e experimentos e, permitir a obtenção do máximo de informações a partir da análise de dados químicos multivariados (SENA et al., 2000).

Os padrões são os meios de interpretação que levam a certas decisões, que são aprimoradas ao longo do tempo (CASTRO; PRADO, 2001). Os métodos de reconhecimento de padrões, particularmente importantes na quimiometria, são eficientes na análise, obtenção e interpretação de dados. Estes métodos são alvo constante de estudos da inteligência artificial e sua dificuldade consiste na classificação ou categorização de dados, sendo que a avaliação dos classificadores pode ser realizada pelos interpretação dos parâmetros de mérito.

A quimiometria também auxilia em estudos de quantificação aplicando modelos de regressão. A previsão tem sido um componente essencial da ciência de dados moderna, seja na disciplina de análise estatística ou no aprendizado de máquina. A tecnologia moderna facilitou a geração de dados, logo, há o desenvolvimento de novos métodos e algoritmos para extrair informações e criar modelos preditivos robustos (RIMAL; ALMØY; SÆBØ, 2019). Modelos clássicos, como regressão linear simples, se tornam insuficientes para sistemas complexos, frente à quantidade de dados que podem ser obtidos a partir de análises químicas. Assim, modelos de regressão multivariada para a obtenção de dados quantitativos apresentam diversas vantagens já que podem apresentar bons resultados em sistemas complexos, quando aplicados no conjunto de dados em questão (BRO, 2003).

Com o avanço e disponibilidade de diversos recursos computacionais, os métodos de análise e classificação foram facilitados. Em muitas aplicações, é possível a

combinação de vários classificadores e avaliação de qual melhor descreve o conjunto de dados.

- Estudos de caso

Para o desenvolvimento do trabalho foram considerados dois estudos de casos, descritos a seguir.

Identificação de vidro (Glass Identification Dataset): O vidro encontrado em uma cena de crime pode ser uma evidência generalizada, já que é um material onipresente e pode ter inúmeras aplicações, comumente encontrado em vários ambientes. Tais fragmentos podem ser aderidos à roupa de um criminoso e associados à origem do mesmo. São evidências formidáveis, uma vez que são quimicamente estáveis e fornecem resultados significativos em laboratórios forenses sem sofrerem alterações temporais (JENSEN; SHEN, 2009). Fragmentos de vidro para fins forenses são geralmente comparados em proposições concorrentes ou são classificados em categorias de aplicação, ou seja, pelo tipo de vidro. Tipicamente, as amostras deste material são minúsculas, sendo mais viáveis para avaliar características físico-químicas, como índice de refração (RI) ou composição elementar (NAPIER; NEOCLEOUS; NOBILE, 2015). Uma análise elementar quantitativa precisa pode fornecer informações sobre a fonte do material e levar à corroboração de um álibi. Assim, profissionais nessa área devem procurar minimizar o número de falso-positivos, que, se considerados equivocadamente podem culpar um inocente (BUSCAGLIA, 1994). Devangad et al. (2016) considera que em estudos forenses, os dados são vastos e complexos e a interpretação daqueles gerados por métodos analíticos nem sempre é capaz de gerar resultados exatos. Com isso, a quimiometria oferece várias vantagens de ferramentas avançadas baseadas em métodos estatísticos e matemáticos, para solucionar problemas desse tipo (KUMAR; SHARMA, 2018).

Qualidade de vinhos (Wine Quality Dataset): O vinho é uma bebida cada vez mais apreciada por uma ampla gama de consumidores. A certificação de vinhos e avaliação de qualidade são elementos-chave dentro deste contexto, que impede a adulteração ilegal e garante a qualidade do mercado vitivinícola. A avaliação da

qualidade é muitas vezes parte do processo de certificação e pode ser usada para melhorar a produção de vinho, identificando os fatores mais influentes e para estratificar vinhos como marcas premium, parâmetro ideal para definir o valor agregado (CORTEZ et al., 2009). Essa certificação de vinhos é geralmente avaliada por análises físico-químicas e testes sensoriais. Exames laboratoriais físico-químicos rotineiramente usados incluem determinação de densidade, teor de álcool e pH, enquanto testes sensoriais dependem principalmente de especialistas humanos. Assim, a classificação do vinho é uma tarefa difícil. Além disso, as relações entre a análise físico-química e sensorial são complexas e ainda não totalmente compreendidas.

- Organização do trabalho

Esta dissertação está dividida em 7 capítulos. O primeiro capítulo descreve a introdução, a motivação do estudo e descreve os casos que serão estudados ao longo do trabalho. O segundo capítulo ressalta os principais objetivos do estudo. O terceiro capítulo descreve a fundamentação teórica de reconhecimento de padrões e regressão. São descritas de forma breve, as técnicas específicas do aprendizado de máquina, o pré-processamento e o tratamento de dados aplicados nos conjuntos de dados, a análise exploratória e o conceito de superajuste. No capítulo 4 são apresentados os métodos do trabalho, incluindo o *software* R e os pacotes utilizados, bem como os conjuntos de dados e suas etapas de aplicação. Os resultados obtidos pela aplicação dos modelos quimiométricos são apresentados e discutidos para cada conjunto de dados no capítulo 5, avaliando principalmente a performance dos mesmos. Por fim, o capítulo 6 aborda as conclusões finais e o capítulo 7 apresenta as perspectivas para trabalhos futuros.

1. OBJETIVOS

1.1. Gerais

Este trabalho tem como objetivo geral aplicação de diferentes métodos de Reconhecimento de Padrões em três diferentes conjuntos de dados químicos para avaliação do modelo que apresenta o melhor desempenho.

1.2. Específicos

- Aplicar e comparar diferentes modelos quimiométricos lineares, não lineares e baseados em árvores e regras de classificação em conjunto de dados previamente conhecidos;

- Aplicar e comparar diferentes modelos quimiométricos para regressão buscando avaliar sua aplicabilidade frente à tarefa de classificação;

- Avaliação dos principais parâmetros de mérito gerados pelos modelos de regressão e classificação;

- Aplicar uma nova abordagem de teste de permutação para verificar a ocorrência de sobreajuste do modelo de classificação;

2. REVISÃO DA LITERATURA

2.1. Quimiometria

Segundo Flumignan (2010), a quimiometria é o segmento da química que utiliza ferramentas estatísticas e matemáticas para planejamento e otimização de experimentos, além de extração de informações químicas relevantes de dados químicos multivariados. Com o desenvolvimento de novos instrumentais aplicados aos laboratórios de pesquisas e a velocidade de processos, a quantidade de dados cresceu substancialmente, surgindo a necessidade do desenvolvimento de uma ferramenta estatística sofisticada aplicada à química.

O uso de computadores para analisar dados químicos cresceu significativamente, e esse aumento foi causado por recentes avanços em *hardware* e *software*, além do interfaceamento de instrumentos com os computadores, principalmente na área de química analítica, produzindo grande quantidade de dados (FERREIRA et al., 1999). Assim, diante da necessidade, surgiu a quimiometria.

Até a segunda metade dos anos 80, a presença da quimiometria na literatura era escassa. Sua aplicação sofreu aumento significativo a partir dos anos 90 (SENA et al., 2000), levando pesquisas das mais diferentes áreas à possibilidade de extração de informações de dados.

No contexto atual, a quimiometria tem sido inserida em programas de graduação e pós-graduação em diversos segmentos acadêmicos e apresentado grande relevância na pesquisa científica e consequente diferencial para a formação profissional (SCHNEIDER et al., 2018). A quimiometria certamente encontra-se em uma posição mais favorável para contribuir com a interpretação de dados, quando se considera as mudanças tremendas que ocorrem constantemente na química e nas várias disciplinas relacionadas.

Diversos estudos são voltados para sistemas multivariados, em que é possível medir muitas variáveis de determinada amostra, necessitando de técnicas estatísticas multivariadas, álgebra linear e análise numérica. (ADAMS, 1995).

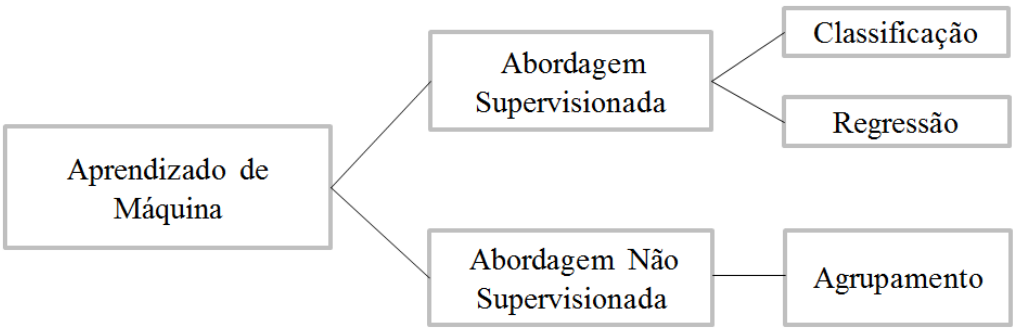
2.2. **Aprendizado de Máquina**

A mineração de dados abrange o processo *knowledge discovery databases* (KDD), que trata da descoberta de conhecimento em banco de dados (WITTEN; FRANK; HALL, 2011). Esse processo busca extrair padrões e informações previamente desconhecidos em conjuntos de dados, que podem ser úteis. Para atingir os objetivos do KDD, os algoritmos realizam um aprendizado de máquina (AM).

Os algoritmos são usados para prever modelos preditivos através da observação de um conjunto de objetos rotulados em AM supervisionado e são chamados de conjunto de treinamento (LUXBURG; SCHÖLKOPF, 2011). Tais rótulos correspondem a classes ou valores obtidos por alguma função desconhecida. Assim, os algoritmos de classificação buscam determinar um classificador capaz de generalizar as informações do conjunto de treinamento visando classificar objetos cujo rótulo seja desconhecido. Quando os valores dos rótulos são discretos, tem-se um problema de classificação, visando identificar futuros casos em cada uma das classes pré-estabelecidas. Quando tais valores são contínuos, tem-se um problema de regressão e o objetivo é prever o valor dessa variável com base nas características dos ensaios.

De acordo com Brereton (2015), o aprendizado de máquina pode ser dividido em dois tipos: abordagens supervisionadas e não supervisionadas. Essas duas vertentes resultam em fundamentos para estudo e desenvolvimento dos classificadores e agrupadores, como representado na Figura 1.

Figura 1- Tipos de algoritmos no aprendizado de máquina de acordo com Brereton (2015)



Fonte: O autor, 2020.

No aprendizado supervisionado, toda entrada é precedida de uma saída desejada que o método deve ser capaz de estimar a classe ou o valor da saída. Assim, dado um conjunto de treinamento T_r composto por pares (x_i, y_i) onde $x_i \epsilon X$ são as entradas para a i-ésima amostra do conjunto de treinamento e $y_i \epsilon Y$, a saída dessa amostra, o aprendizado supervisionado visa determinar uma função $y = f(x, \Theta)$, buscando minimizar o erro. Também é selecionado um conjunto de validação, em que são conhecidos os pares (x_i, y_i). O modelo é ajustado no conjunto T_r e o objetivo geral é minimizar os erros no conjunto de validação V_a, para que o modelo possa ter maior capacidade de generalização possível. As aplicações mais comuns desse tipo de algoritmo são a predição e a classificação (MOHRI, 2012).

Já no aprendizado não supervisionado, a entrada contém somente variáveis das amostras $(x_i \epsilon X)$, assim a saída é desconhecida inicialmente. Técnicas de aprendizado não supervisionado têm como objetivo prever pontos não vistos em dados de treinamento não rotulados, como *Clustering* e a redução da dimensionalidade (MOHRI; ROSTAMIZADEH; TALWALKAR, 2012).

Como é estabelecido nos modelos, esse trabalho trata de dados multivariados e é dito como supervisionado, aplicando um conjunto de treinamento para classificar amostras já conhecidas.

2.3. **Reconhecimento de Padrões**

O reconhecimento de padrões (RP) é o campo da ciência que busca classificar objetos em categorias ou classes pré-estabelecidas, de acordo com suas características. Não há definição universal para RP (BRERETON, 2015) mas é amplamente entendida como a representação simples de um determinado conjunto de dados a partir de suas propriedades com maior relevância (GOMES, 2007). Sucintamente, Bishop (2006) define essa ferramenta como busca automática que visa descobrir regularidades em dados por meio de algoritmos computacionais, sendo que essas regularidades são usadas para realizar ações como classificar dados em diferentes categorias ou agrupá-los.

O reconhecimento de padrões cresceu a partir da engenharia (BISHOP, 2006) e de acordo com Jain, Duin & Mao (2000), é crescente o interesse na área de

reconhecimento de padrões, visto que as aplicações não são apenas desafiadoras, mas também computacionalmente mais simples. A Tabela 1 demonstra exemplos de aplicação e suas respectivas classes preditoras.

Tabela 1- Aplicações de Reconhecimento de Padrões em diferentes domínios de problemas

Aplicação	**Padrão de Entrada**	**Classe Preditora**
Análise de Sequência	DNA/ Sequência de proteína	Tipos de genes conhecidos/padrões
Busca por Padrões significantes	Pontos espacial multidimensional	Compactar e separar grupos
Busca na Internet	Documento texto/imagem	Categorias semânticas (negócios, entre outros)
Máquina de leitura de deficientes visuais	Imagem de documento	Caracteres alfa numéricos, palavras
Inspeção de placas de circuito impresso	Intensidade ou alcance de imagem	Natureza do produto (defeituosa ou não)
Identificação pessoal	Face, impressão digital	Usuários autorizados para controle de acesso
Inquérito por telefone sem assistência de operador	Voz em forma de onda	Palavras faladas

Fonte: Adaptado de JAIN, 2000.

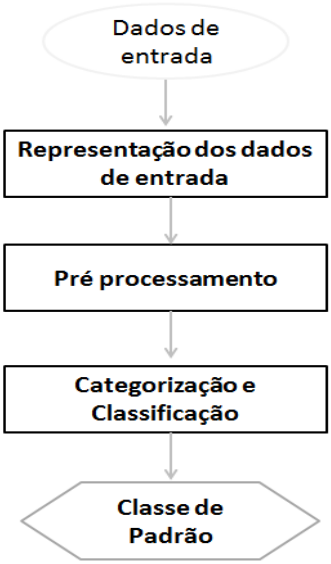
O reconhecimento de padrões é aplicado em diversas áreas de pesquisas, abrangendo desde exemplos comuns, como detecção de padrões de amostras simples até casos mais complexos, como aqueles de aprendizagem de máquina. Assim, ela se torna importante em situações cotidianas do homem que necessitam de padrões como formação da linguagem, reconhecimento de imagens e modo de falar. Apresenta-se como uma tarefa complexa, que envolve a caracterização de relações existentes no meio para adaptação da mesma (MORAIS, 2010).

O reconhecimento de padrões se originou no campo do processamento de imagens e sinais, onde as técnicas foram desenvolvidas para categorizar as amostras com base nas regularidades dos dados observados. As primeiras aplicações de técnicas de reconhecimento de padrões na química foram estudos envolvendo espectrometria de massa de baixa resolução, que apareceram na literatura em 1969. Numerosos livros e artigos de revisão foram publicados sobre este assunto (LAVINE, 2006).

Utilizando ferramentas computacionais, essa área da quimiometria permite investigações em resultados obtidos através de análises químicas buscando semelhança entre as amostras quanto a sua composição química. Assim, o reconhecimento de padrões se torna um importante instrumento de estudo na química analítica, trazendo informações mais apuradas quando comparado a outros métodos univariados (CORREIA; FERREIRA, 2007).

Basicamente, sistemas de reconhecimento de padrões abrangem as fases: aquisição de dados e pré-processamento, representação dos dados e tomada de decisões, como descrito na Figura 2.

Figura 2- Fases do reconhecimento de padrões para determinação de classe de um conjunto de dados



Fonte: Adaptado de CASTRO; PRADO, 2001, f. 133.

O desafio consiste em escolher quais técnicas serão mais eficientes para concluir o projeto. A primeira etapa está relacionada à representação dos dados de entrada que podem ser medidos de acordo com o domínio do problema. Tal mensuração deve representar padrões do objeto, tornando possível ser posteriormente classificado. A segunda fase extrai propriedades intrínsecas e atributos do objeto. A determinação dessas características é de fundamental relevância, influenciando na eficiência do classificador, tendo como principais objetivos, a redução da dimensionalidade sem que haja perda de informações principais, reduzindo o esforço computacional e estimar as

características mais significativas para classificação. Por último, a terceira fase consiste na determinação dos procedimentos que identificaram e classificaram os objetos em uma classe.

2.4. **Regressão**

Métodos de regressão surgiram há mais de dois séculos com Legendre e Gauss, com objetivo de prever órbitas ao redor do Sol utilizando o método dos mínimos quadrados. Hoje em dia, o problema de estimação de uma função de regressão possui papel central em estatística (IZBICKI; SANTOS, 2018).

O objetivo dessa tarefa é estabelecer a relação entre essas características a partir de um modelo matemático, auxiliando no entendimento de como uma(s) variável(is) pode influenciar no comportamento da variável de saída.

A análise de regressão possibilita encontrar uma relação razoável entre as variáveis de entrada e saída. O uso desta abordagem necessita de coleta de dados e do uso de métodos estatísticos.

Modelos de regressão são construídos com os objetivos (VINING; MONTGOMERY; PECK, 2001):

- **Predição** – Procedimento para determinar a variável preditora a partir das variáveis de entrada;
- **Seleção de variáveis** – Eliminação das variáveis cuja contribuição não seja importante para determinar a variável preditora;
- **Estimação de parâmetros** - Dado um modelo e um conjunto de dados referente às variáveis resposta e preditoras, estimar parâmetros ou ajustar um modelo aos dados significa obter valores ou estimativas para os parâmetros, por algum processo, tendo por base os dados observados;
- **Inferência** - O ajuste de um modelo de regressão em geral tem por objetivos básicos, além de estimar os parâmetros, realizar inferências sobre eles, tais como, testes de hipóteses e intervalos de confiança.

Na literatura, existem diversos métodos cujo objetivo é fazer previsão, como por exemplo os métodos de regressão não paramétricos kernel e vizinhos mais próximos (ALTMAN, 1992), os de redes neurais (VINING; MONTGOMERY; PECK, 2001) dentre outros. Apesar de a predição ser a aplicação mais comum para os modelos de regressão, quando se utiliza modelos de regressão o objetivo é, geralmente, utilizá-los de forma a se fazer a descrição (modelagem) dos dados em estudo e/ou algum teste de hipóteses em relação aos seus parâmetros.

Nesse contexto, o presente trabalho busca para determinado conjunto de dados a utilização de modelos de regressão pré existentes, visando gerar previsões para os dados em questão e a geração de previsão para os modelos assumidos através de classificadores.

2.5. **Métodos quimiométricos**

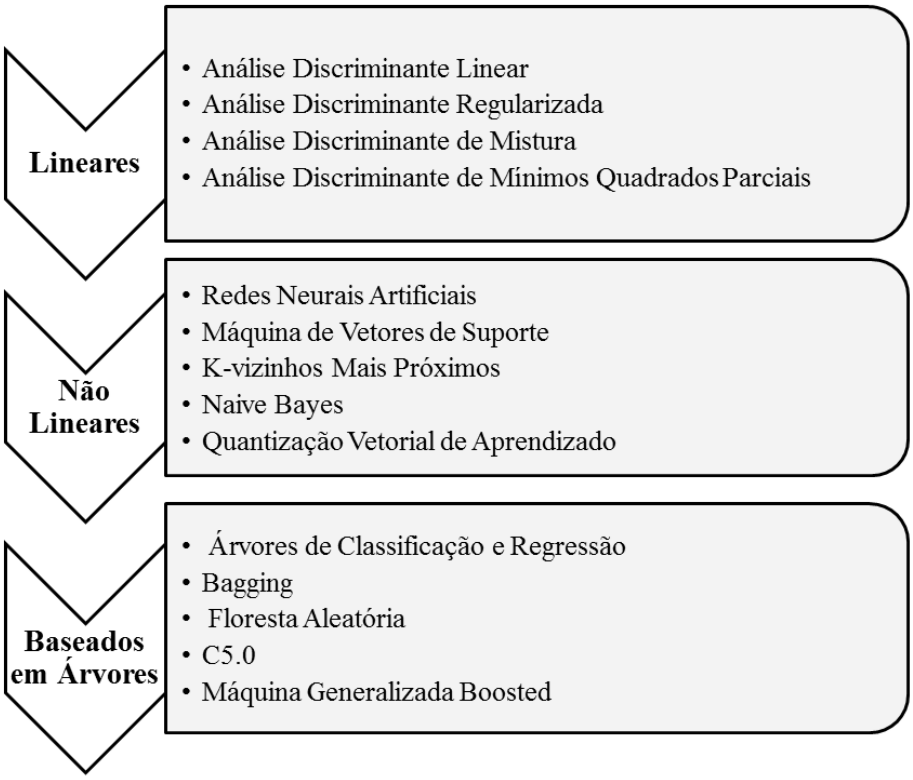
Conceitualmente, o aprendizado envolve a abstração e generalização de dados em uma representação estruturada. Em termos práticos, um algoritmo de aprendizagem de máquina usa dados contendo exemplos e características do conceito a ser aprendido, e resume esses dados na forma de um modelo, que é então usado para fins preditivos ou descritivos. Essas finalidades podem ser agrupadas em tarefas, incluindo classificação, previsão numérica, detecção de padrões e agrupamentos. Entre as muitas opções, os algoritmos de aprendizado de máquina são escolhidos com base nos dados de entrada e nas tarefas de aprendizagem (LANTZ, 2015).

Modelos classificadores podem ser definidos como um mapeamento que, a partir de um conjunto de dados de entrada, responde com um valor numérico ou associa uma amostra a uma classe (SALIM, 2015). A dificuldade de obtenção de um classificador de aprendizagem de máquina consiste em um processo de busca visando encontrar um modelo que apresente melhor capacidade de descrever o domínio que ocorre o aprendizado, uma vez que o mesmo deve ser o mais generalizado possível para que não haja comprometimento da capacidade de predição. Assim, esses classificadores podem sofrer superajuste (*over-fitting*) (MITCHELL, 1997), ocorrência que será discutida nos próximos tópicos. Então, os estudos desenvolvidos nesse sentido visam estimar modelos que sejam capazes de classificar novas amostras para as quais os rótulos são ainda

desconhecidos, seja atribuindo um rótulo ou retornando uma probabilidade da amostra pertencer a cada classe do problema.

Sistematizando, os modelos quimiométricos de classificação podem ser divididos em três tipos diferentes, de acordo com a Figura 3: o primeiro tipo compreende os modelos de classificação lineares, o segundo tipo engloba os modelos não lineares, e o terceiro tipo está relacionada com as árvores e regrasde decisão.

Figura 3 - Modelos quimiométricos aplicados nas tarefas de classificação ao longo do trabalho



Fonte: O autor, 2020.

Como descrito, foram selecionados quatorze modelos de classificação diferentes, a saber: Análise Discriminante Linear, Análise Discriminante Regularizada, Análise Discriminante de Mistura, Análise Discriminante de Mínimos Quadrados Parciais, Redes Neurais Artificiais, Máquina de Vetores de Suporte com função de base radial, *K*-vizinhos mais próximos, *Naive Bayes*, Vetor de Quantização de Aprendizagem de Máquina, Árvores de Classificação e Regressão, *Bagging*, Floresta Aleatória, C5.0 e Máquina Generalizada *Boosted*. Nesta seção será apresentada um

sumário das técnicas de classificação que foram empregadas nesta dissertação com a intenção de fornecer as principais características das mesmas.

Os algoritmos de aprendizagem de regressão avaliam os parâmetros que modelam o conjunto de dados de forma significativa. Assim, visando avaliar os resultados obtidos pela classificação e compará-los, foram empregados os modelos que poderiam tanto ser usados na classificação quanto na regressão, que são: Análise Discriminante por Mínimos Quadrados Parciais (PLS-DA), Árvores de Classificação e Regressão (CART), *Bagging*, Floresta Aleatória (RF), k-Vizinhos mais próximos (k-NN), Máquina de vetores de suporte (SVM) com *kernel* radial e linear, e Máquina Generalizada *Boosted* (GBM), capazes de estimar uma relação entre as variáveis e posteriormente e utilizá-los para classificar as respostas geradas.

2.5.1. Modelos Lineares

2.5.1.1. Análise Discriminante Linear

A análise discriminante linear (LDA) tem sido usada com frequência em tarefas de classificação, como no trabalho desenvolvido por Lee et al. (2016), que usa o modelo para verificar os espectros de infravermelho (IR) de diferentes tipos de papel branco que tendem a parecer indiferentes em casos de falsificação, concluindo que a técnica empregada é viável para identificação forense deste material.

O modelo projeta a matriz de entrada em um espaço de dimensão reduzida, sendo baseado na distância entre as médias das diferentes classes e a variação dentro de cada classe. Assim, o espaço dimensional gerado aumenta a variação entre classes e minimiza a variação interna das mesmas (THARWAT et al., 2017).

A LDA faz previsões de margem linear, estimando a probabilidade de um novo conjunto de entrada pertencer a determinada classe. A classe que obtém a maior probabilidade é a classe de saída e então é feita uma previsão. O modelo usa o Teorema de Bayes para estimar as probabilidades condicionais. Resumidamente, o Teorema de Bayes pode ser usado para estimar a probabilidade da classe de saída considerando a

entrada, usando a probabilidade de cada classe e a probabilidade dos dados pertencentes a cada classe.

Embora apresente alguns problemas, essa técnica é utilizada com grande frequência na literatura (LEE et al., 2016), sendo destinada a problemas de classificação em que a variável de saída é categórica.

2.5.1.2. Análise Discriminante Regularizada

Com o intuito de demonstrar outro método de discriminação que minimizasse os problemas de técnicas linear e quadrática, Friedman (1989) propôs o método conhecido como Análise Discriminante Regularizada - RDA. A partir disso, Wu et al. (1996), compararam os três classificadores: análise discriminante linear (LDA), quadrática (QDA) e regularizada (RDA) para dados químicos de alta dimensionalidade obtidos por Espectroscopia no Infravermelho Próximo (NIR). Os autores concluíram que, em diversos casos, RDA reduz-se a LDA ou a QDA dependendo de alguns parâmetros, podendo inclusive apresentar-se com melhor desempenho. Nos casos em que pequenos ganhos na qualidade da classificação foram importantes, a aplicação do RDA foi considerada mais útil.

O método RDA classifica a amostra teste, na classe que retorna maior valor numérico, utilizando a matriz covariância regularizada. Essa técnica faz uso de dois parâmetros lambda (λ) e gama (γ), que variam entre 0 e 1. Através da combinação entre λ e γ, obtém-se valores intermediários entre os métodos QDA e LDA, combinando diferentes graus da matriz covariância comum das classes, quanto da matriz covariância particular de cada classe, resultando em uma estimativa mais confiável.

Considerando as conclusões de Friedman (1989), o método de regularização aumenta o poder da análise discriminante em configurações em que o tamanho das amostras é relativamente pequeno e o número de parâmetros a serem estimados é grande, propiciando ganhos substanciais na exatidão de classificação.

2.5.1.3. Análise Discriminante de Mistura

A análise discriminante de mistura (MDA) é um método para classificar as observações em classes não normais conhecidas, estimando a densidade de misturas para cada classe (ÇALIS et al., 2011). Ele é tratado como uma extensão do LDA que melhora a exatidão da classificação, modelando os relacionamentos não-lineares multivariados entre variáveis de cada classe ou determinando subclasses latentes.

Em princípio, essa abordagem é útil para duas finalidades: modelar relações não-normais ou não-lineares multivariadas entre variáveis dentro de cada grupo, permitindo uma classificação mais precisa; ou para determinar se subclasses latentes / subjacentes estão presentes em cada grupo.

Nas misturas Gaussianas, cada observação ou vetor de dados tem uma probabilidade de classificação em cada classe, sendo selecionados aqueles que maximizam a exatidão da classificação (RAUSCH; KELLEY, 2009).

2.5.1.4. Análise Discriminante por Mínimos Quadrados Parciais

A Análise Discriminante por Mínimos Quadrados Parciais (PLS-DA) também é uma excelente opção para reduzir a dimensão de dados para classificação (LEE et al., 2016), além da possibilidade de ser utilizado em casos de regressão. Ao longo dos estudos, diversas variantes do modelo PLS foram desenvolvidas e podem ser aplicadas em casos de regressão, em abordagens discriminatórias e problemas de classificação a partir de um modelo pré-determinado (MEHMOOD; AHMED, 2016).

Resumidamente, um modelo de regressão linear é estimado projetando variáveis preditas e variáveis observadas em um novo espaço (modelo de fator bilinear); isto é, modela as estruturas de covariância em dois espaços. Assim, o melhor modelo é escolhido como o que explica a direção máxima da variação multidimensional. É considerado um dos modelos mais aplicados em quimiometria e áreas afins (WOLD; SJÖSTRÖM; ERIKSSON, 2001).

2.5.2. Modelos Não Lineares

2.5.2.1. Redes Neurais Artificiais

A rede neural artificial (ANN) é um sistema matemático de manipulação similar ao princípio de funcionamento das redes neurais biológicas, com processadores conectados, os neurônios (CHITSAZAN; NADIRI; TSAI, 2015). De acordo com Raghu,Poongodi e Yegnanarayana (1995), ANN apresenta as seguintes vantagens:

- Os relacionamentos funcionais entre os padrões de entrada e saída não necessitam ser conhecidos para serem capturados por uma rede neural;
- Não é necessário hipóteses sobre as distribuições estatísticas dos padrões de entrada;
- Mesmo quando algum elemento de conexão do processamento se mostre degradado, há tolerância do modelo à falha.

No treinamento supervisionado, são fornecidas as entradas e as saídas. A rede processa as entradas e compara as saídas resultantes com as saídas desejadas. Os erros são propagados de volta pelo sistema, fazendo com que o sistema ajuste os pesos que controlam a rede. Esse processo ocorre repetidamente à medida que os pesos são continuamente ajustados. Durante o treinamento de uma rede, o mesmo conjunto de dados é processado muitas vezes à medida que os pesos de conexão são refinados. O modelo tem sido amplamente aplicado em vários campos.

O estudo de El-khatib, Abu-nasser e Abu-naser (2019) auxilia o investigador a identificar o tipo de vidro encontrado na área do crime através do modelo de rede neural e alcança exatidão de 96,7%. Debska e Guzowska-Świder (2011) utiliza também o modelo para avaliação da qualidade dos alimentos com exatidão de 97,2% em um conjunto de dados de vinho.

2.5.2.2. Máquina de Vetores de Suporte

Máquina de vetores de suporte (SVM) é um método estatístico usado para resolver problemas de estimativa de função multidimensional, seja na classificação ou regressão. Em seus primórdios, as funções de decisão foram expandidas para as funções

não-lineares parametrizadas pelos vetores de suporte (SVs). Posteriormente foi empregado em funções com valores reais, abrindo novas possibilidades de problemas de aproximação e estimação de função tipo *Splines*, com uma ou duas dimensões, são construídas com um número arbitrário de pontos de grade, aproximando-se as funções lineares das não-lineares. Esse método permite resolver problemas que estimam funções, tanto em estatística quanto em matemática aplicada (VAPNIK; GOLOWICH; SMOLA, 1997).

Schölkopf e Smola (2002) afirmam que nos últimos anos, o escopo desse método é crescente, tanto em termos de algoritmos quanto a um entendimento teórico mais desenvolvido. A função *kernel* no SVM, é baseada na construção de uma função que mapeia os dados que estão em uma dimensão qualquer para outra maior, tornando os mesmos linearmente separáveis. Intuitivamente, uma boa separação é alcançada pela função que tem a maior distância do ponto de dados de treinamento mais próxima de qualquer classe (a chamada margem funcional), uma vez que, em geral, quanto maior a margem, menor o erro de generalização do classificador.

Cortez et al. (2009) utilizou esse modelo para estudo de caso de certificação de vinhos e alcançou uma exatidão em torno de 89% para classificação do vinho tinto e 86,8% para o vinho branco. Ao mesmo tempo, Zadora (2007) avaliou a eficiência dos Classificadores Naive Bayes (NB) e Máquinas de Vetor de Suporte (SVM) quando aplicados a problemas de classificação de objetos de vidro com base na concentração elementar maior e menor por SEM-EDX e alcançou uma exatidão de 98,6% com o SVM.

2.5.2.3. K-Vizinhos mais próximos

O modelo de vizinho mais próximo compara a distância entre as instâncias e localiza a classe que tem a semelhança mais próxima entre as variáveis. Várias métricas de distância podem ser usadas, incluindo a Euclideana, que apresenta bom desempenho quando as variáveis contínuas são normalizadas, de modo que têm a mesma influência na medida de distância entre instâncias (ALDAYEL, 2012).

Durante o processamento, verifica-se a quais classes pertencem os *k* padrões mais próximos e a classificação é feita, associando-se o padrão de teste à classe que for

predominante (MITCHELL, 1997). Nessa fase, *k* é uma constante definida pelo usuário e um vetor não rotulado (uma consulta ou ponto de teste) é classificado atribuindo o rótulo que é mais frequente entre as *k* amostras de treinamento mais próximas a esse ponto de consulta. A melhor escolha de *k* depende dos dados; geralmente, valores maiores de *k*, reduzem o efeito do ruído na classificação, mas tornam os limites entre as classes menos distintos. Aldayel (2012) investigou a aplicação do KNN na identificação de vidro para permitir comparações com outras técnicas de mineração de dados, alcançando exatidão de 80,6%. Assim como Nababan, Sitompul and Tulus (2018), que utilizou para esse mesmo conjunto de dados, esse modelo com variações de *k*, alcançando uma exatidão máxima de 87,5%.

2.5.2.4. Naive Bayes

Naive Bayes (NB) é baseado no teorema de Bayes associado aos pressupostos ingênuos na independência de recursos, mostrando-se conveniente em casos reais, sendo um modelo probabilístico (VURAL; GÖK, 2017). O modelo atribui uma probabilidade aos vetores que possuem características específicas. Desta forma, o problema de estimar uma função de densidade de probabilidade p dimensional é reduzido a fim de estimar p funções unidimensionais.

Uma das vantagens desse método é de não ter a necessidade de estimar distribuições multivariadas, o que pode ser inviável nos casos de alta dimensão. No entanto, pode apresentar inconvenientes quando há uma grande quantidade de dados (GUPTA, 2015). Este modelo requer uma pequena quantidade de dados de treinamento para supor os parâmetros essenciais para classificação.

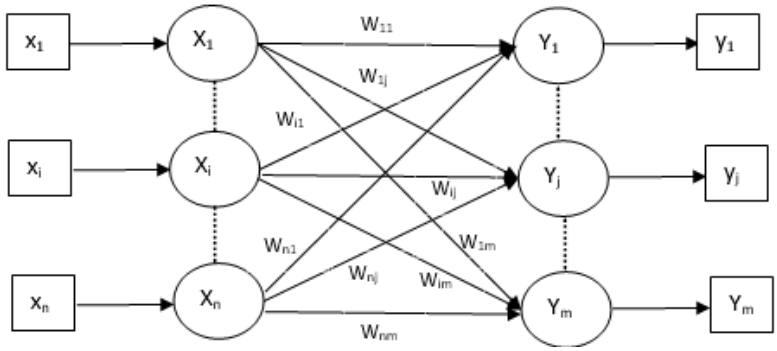
2.5.2.5. Vetor de Quantização de Aprendizagem de Máquina

O modelo de Vetor de Quantização de Aprendizagem de Máquina (LVQ) é um caso particular de uma rede neural artificial baseada no aprendizado de *Hebbian*. É

definido como um processo de classificação dos padrões em que cada unidade de saída representa uma classe. A Figura 4 mostra a arquitetura do LVQ. Observando a mesma, há um número "n" de unidades de entrada e um número "m" de unidades de saída. As camadas estão totalmente interconectadas com pesos sobre elas.

É um precursor de mapas de auto-organização (SOM), mas sem uma estrutura topológica. O sistema é determinado por protótipos $W = (w(i), \ldots, w(n))$, sendo o vencedor o mais próximo do *input* de acordo com uma medida de distância Euclideana estimada e é classificado desta forma se ele classificar corretamente um certo ponto (amostra). Esses protótipos são fáceis de aplicar e podem ser usados na presença de várias classes (LVQ, 2019). O LVQ pode causar um desempenho ruim quando os dados são de distribuição não esférica e, especialmente, se contém ruídos ou *outliers* (AL-DAOUD, 2009).

Figura 4 - Arquitetura do modelo LVQ demonstrando unidades de entrada e de saída interconectados por protótipos



Fonte: Adaptado de OMATU; YANO; IKEDA, 2015, f.199.

Três outras variantes, nomeadamente LVQ2, LVQ2.1 e LVQ3, foram desenvolvidas por Kohonen. A complexidade de todas essas três variantes, devido ao conceito de que o vencedor e a unidade finalista aprenderão, é mais do que na LVQ. Esses algoritmos de treinamento produzem precisões quase semelhantes, embora diferentes técnicas estejam subjacentes uma à outra.

O LVQ1 (KOHONEN, 1990b) seleciona o único conjunto de melhores vetores de livros de códigos correspondentes que é selecionado e movido para mais perto ou mais longe de cada vetor de dados, por iteração, se a decisão de classificação estiver correta ou errada, respectivamente. Já o LVQ1 otimizado é o mesmo que o LVQ1,

exceto que cada vetor do livro de códigos tem sua própria taxa de aprendizado (KOHONEN, 1992).

No LVQ2.1, dois conjuntos dos melhores vetores correspondentes do livro de códigos são selecionados e atualizados apenas se um pertencer à classe desejada e outro não, e a taxa de distância estiver dentro de uma janela definida. O valor da janela é definido como o ponto médio dos dois vetores do livro de códigos (KOHONEN, 1990a).

Enquanto o LVQ3 é mesmo que LVQ2.1, exceto se os dois conjuntos de vetores do livro de códigos forem da classe correta. Eles são atualizados, mas ajustados usando um valor de épsilon. O valor de épsilon é usado para ajustar a taxa global de aprendizado (KOHONEN, 1990b). O LVQ3 otimizado é igual ao LVQ3, exceto que cada vetor do livro de códigos tem sua própria taxa de aprendizado da mesma maneira que OLVQ1.

HU and LIU (2013) utilizaram o LVQ nos conjuntos de identificação de vidro e qualidade de vinhos, resultando respectivamente em valores de exatidão de 72,36% e 92,28%, respectivamente.

2.5.3. Modelos Baseados Árvores e Regras de Decisão

Árvores de decisão determinam modelo para representar decisões e possíveis consequências à medida que se caminha pelos nós da árvore (da raiz a uma das folhas). Em aprendizado de máquina, existem algoritmos que possibilitam a construção destas árvores de decisão a partir de um conjunto de dados fornecido para treinamento, criando assim um mapeamento entre as características de uma amostra e a respectiva classe desta amostra (SALIM, 2015). Alguns modelos que usam esse método, foram escolhidos e estão descritos de forma resumida abaixo.

2.5.3.1. Árvores de Classificação e Regressão

Árvores de Classificação e Regressão (CART) consistem em técnicas não-paramétricas que induzem árvores de classificação e árvores de regressão, dependendo se a variável é nominal ou contínua (regressão). É uma técnica que gera uma árvore binária a partir de "particionamento recursivo binário", agrupando os nós em respostas sim ou não, como valores de previsão. Cada uma dessas respostas é prevista de acordo com uma variável, minimizando a variabilidade desse conjunto criado.

Ela foi considerado uma técnica de interpretação direta, que a torna acessível, mas não aplicável em muitos casos (YOUSSEF et al., 2016). O valor que é gerado pelo nó "terminal" corresponde à média dos valores de resposta desse nó. Os resultados gerados pelo modelo não são influenciados pelas transformações monótonas e por diferentes escalas entre as variáveis, além de não serem sensíveis aos *outliers*. Ao contrário das abordagens adotadas por outros modelos, que utilizam a pré-poda, o CART expande a árvore de forma exaustiva, realizando a pós-poda, que reduz o fator custo-complexidade.

2.5.3.2. Bagging

Bootstraping e agregação (ou *Bagging*, abreviadamente) é um método simples, composto de duas fases:

- *Bootstraping:* Amostras são geradas por amostragem aleatória simples com reposição e em cada amostra é treinado um classificador.
- Agregação: As previsões dos classificadores são agregadas para fazer uma previsão combinada final.

Bagging pode ser aplicado para reduzir a variação de algoritmos com alta variância. Eles são sensíveis aos dados de treinamento, e os resultados podem ser alterados facilmente. Como as próprias árvores de decisão, o *bagging* pode ser usado para problemas de classificação e regressão (BROWNLEE, 2016).

O empacotamento leva a "melhorias para procedimentos instáveis", que incluem, por exemplo, redes neurais artificiais , árvores de classificação e regressão e seleção de subconjuntos na regressão linear (BREIMAN, 1994), através da melhoria da estabilidade e a exatidão dos algoritmos de aprendizado de máquina. Também, reduz a variação e ajuda a evitar ajustes excessivos.

2.5.3.3. Floresta Aleatória

A Floresta Aleatória (RF) baseia-se no estabelecimento de um grande conjunto de árvores aleatórias, cada uma construída de acordo com uma amostra de *bootstrap*, a partir do conjunto de treinamento, podendo executar as tarefas de regressão ou de classificação. Na classificação, a melhor divisão de cada nó na formação das árvores é feita de modo a ter a maior redução no índice de Impureza de Gini (*Gini Impurity*), sendo assim, em cada divisão busca-se formar nós que contêm uma alta proporção de amostras de uma única classe, idealmente nós puros, com somente uma classe. Na regressão a melhor divisão de cada nó é dada pelo ponto que reduz a variância. Os principais parâmetros que devem ser ajustados são " *mtry* " e " *ntree* " (BREIMAN, 2001). Estes parâmetros representam o número de variáveis amostradas aleatoriamente como candidatas em cada divisão e número de árvores utilizadas, respectivamente.

O principal benefício deste modelo é a capacidade de modelagem de relações não-lineares entre preditores e variável de resposta para lidar com ruído e em situações que contêm pequeno conjunto de dados associado a um grande número de variáveis (JEUNE et al., 2018).

2.5.3.4. C5.0

C5.0 é um modelo que divide a amostra sobre a variável que oferecerá o máximo ganho de informações. Essa divisão é sucessiva e o processo são repetidos até que já não seja possível fazer a separação. A divisão do nível inferior, que não contribui significativamente com o modelo, é podada. O nó desse algoritmo só pode prever um objetivo específico e gera dois tipos de modelos: uma árvore de decisão ou um conjunto de regras. A árvore de decisão fornece em cada nó um conjunto de dados de treinamento. O conjunto de regras é derivado de árvores de decisão e gera um conjunto de regras para fazer previsões de registros individuais, combinando votos ponderados recebidos em cada regra. Os modelos C 5.0 são considerados fáceis de entender e com melhoria eficiente para obter maior exatidão na classificação (“Nodo C5.0”, 2019).

2.5.3.5. Máquina Generalizada Boosted

O GBM é baseado em um modelo aditivo progressivo gerado pela implementação do gradiente descendente no espaço. Nesse modelo, o algoritmo *AdaBoost* treina uma árvore de decisão e determina que cada nó tenha pesos equivalentes. Depois de passar pela primeira árvore, o peso das observações mais complexas de serem classificadas é elevado e o peso das observações menos complexas é reduzido. A segunda árvore é então avaliada com os dados ponderados e assim por diante até um número específico de iterações. Em seguida, é avaliado e classificado qual informação não é classificada efetivamente , sendo definida como o modelo final a partir da soma ponderada das predições feitas nos modelos de cada árvore (RIDGEWAY, 2007).

2.6. **Pré-processamento e transformação de dados**

É necessário organizar e avaliar os dados, observando sua natureza e a possibilidade de um tratamento prévio ou transformação dos dados e até a determinação do tipo de método de análise multivariada mais adequado.

Como descrito por Flumignan (2010), os dados possuem ***n*** medidas de diferentes propriedades, as variáveis, executadas sobre ***m*** amostras, formando uma matriz de dados ***X***, constituída por ***m x n*** elementos, como descrito na Figura 5. Na matriz, cada variável é classificada como um vetor coluna e cada amostra como vetor linha, sendo que cada um desses vetores é determinado como ponto no espaço de dimensão m, onde o eixo das coordenadas é descrito pelas variáveis.

Figura 5 - Matriz de dados X constituída de m amostras com n medidas

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Fonte: Adaptado de Flumignan, 2010, f. 29.

Uma organização adequada e avaliação de dados são de extrema importância para que a análise multivariada seja executada corretamente, levando-se em consideração a forma da criação do conjunto de dados, as medidas utilizadas, a confiabilidade do mesmo e, com maior importância, deve ser avaliado a representatividade do conjunto de amostras que se pretende avaliar.

Assim, o pré-processamento e transformação dos dados são ferramentas de grande valia, visto que muitas vezes os dados são expressos em grandezas diferentes, o que pode levar a um entendimento errôneo na análise (FLUMIGNAN, 2010). Assim, essas técnicas consistem em adequar as amostras do conjunto, maximizando ou minimizando o efeito de determinada variável no todo (PARREIRA, 2003). No presente trabalho, foram aplicados o autoescalamento como técnica de pré-processamento, as transformações Box-Cox e sinal espacial.

2.6.1. Autoescalamento

O autoescalamento alia os métodos de centralização na média e o escalamento pela variância, aplicando-as de uma só vez. Assim, cada variável apresenta média igual a zero e desvio padrão igual a 1. Então, as variáveis passam a apresentar mesma importância. Para a centralização na média, calcula-se a média $\overline{x_j}$ dos dados x_{ij} (Equação 1), em que $i = 1, \dots, m$ e $j = 1, \dots, n$.

$$\overline{x_j} = \sum_{i=1}^{m} \frac{x_{ij}}{m} \tag{1}$$

Subtrai-se então os dados originais pela média calculada (Equação 2), deslocando a origem dos eixos e colocando de forma mais conveniente à visualização.

$$x_{ij(cm)} = x_{ij} - \overline{x_j} \quad (2)$$

Para escalar os dados pela variância ($x_{ij\,(ev)}$), cada variável *j* é dividida pelo seu desvio padrão (S_j) (Equação 3), obtendo o valor escalado pela variância de *i* amostras. O desvio padrão é calculado a partir da Equação 4.

$$x_{ij\,(ev)} = \frac{x_{ij}}{S_j} \quad (3)$$

$$S_j^2 = \frac{1}{m-1}\sum_{i=1}^{m}(x_{ij-}\ \overline{x_j})^2 \quad (4)$$

Finalmente, o valor transformado da variável *j* para a amostra *i*, ($x_{ij\,(as)}$), é dado pela Equação 5.

$$x_{ij\,(as)} = \frac{x_{ij} - \overline{x_j}}{S_j} \quad (5)$$

2.6.2. Transformação Box-Cox

A transformação Box-Cox é uma maneira de transformar variáveis dependentes que não seguem uma distribuição normal em uma distribuição normal. A distribuição normal é uma suposição importante para muitas técnicas estatísticas. Os estatísticos Box e Cox (1964) desenvolveram um procedimento para identificar um expoente apropriado (Lambda = λ) e tornar os dados normais. O parâmetro λ indica a potência para a qual todos os dados devem ser elevados. Para fazer isso, é necessário fazer uma busca em grade, que varia de -2 a 2, até que o melhor valor λ seja encontrado. A transformação de y é dada de acordo com a Equação 6, sendo somente ser possível, aplicá-la em valores positivos:

$$y(\lambda) = \begin{cases} \frac{y^{\lambda}-1}{\lambda}, & \text{if } \lambda \neq 0; \\ \log y, & \text{if } \lambda = 0. \end{cases} \tag{6}$$

O objetivo da transformação Box-Cox é converter a variável y *de* forma que a nova variável resultante $\tilde{y}$ tenha uma distribuição tão próxima quanto possível da distribuição normal, ou seja, $\tilde{y} \sim \mathrm{N}(\mu, \sigma^2)$ com média μ e desvio padrão σ, representado pela Equação 7.

$$f(\tilde{y}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{\tilde{y}-\mu}{\sigma}\right)^2\right] \tag{7}$$

2.6.3. Transformação do sinal espacial

A transformação do sinal espacial (*spatial sign transformation*) é uma maneira simples de conferir robustez moderada a estimadores multivariados. Uma projeção do sinal espacial dos dados transformará os dados na superfície de uma esfera multidimensional. Os resultados podem ser usados para destacar a existência de *outliers* que podem ser modificados ou removidos dos dados (SERNEELS; DE NOLF; VAN ESPEN, 2006). Matematicamente, cada variável é dividida por sua norma (Equação 8) (KUHN; JOHNSON, 2013):

$$x_{ij}^{*} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}}^{2}} \tag{8}$$

Assim, após o tratamento matemático do conjunto de dados, adequando o mesmo e até removendo possíveis fontes de variação, procede-se para as análises de reconhecimento de padrões.

2.7. **Análise exploratória**

De forma geral, um conjunto de dados envolvendo análises químicas envolve um certo número de amostras, descritas por determinado número de variáveis. Essas variáveis são por exemplo, concentração de elementos importantes, altura de pico em perfis cromatográficos, características físico-químicas, comprimento de onda em perfis espectroscópicos (CRUZ, 2003). A análise exploratória visa obter informações desses dados, sendo possível que a partir das relações entre variáveis e objetos encontradas, sejam feitas hipóteses e modelos possam ser propostos.

O desempenho de um classificador é dependente do inter-relacionamento entre o conjunto de exemplos, a quantidade de variáveis e sua complexidade. Para uma melhor relação do custo e exatidão, é vantajoso reduzir a dimensionalidade dos dados. Porém, o número reduzido de variáveis pode levar a uma fraca discriminação e consequentemente a uma exatidão inferior no reconhecimento de padrões. A necessidade de redução de dimensionalidade surge por exemplo, da codificação de um grande número de variáveis redundantes, mas também pode implicar na perda de informações fundamentais para discriminação de padrões. Assim, o objetivo dessa técnica é reservar o máximo das informações relevantes do conjunto de dados. A principal decisão da redução de dimensionalidade é a escolha da função de critério.

A análise de componentes principais (PCA) (PEARSON, 1901), sem dúvida, é o método de redução de dimensão mais amplamente aplicado, desempenhando um papel significativo na análise de dados em uma ampla gama de áreas, incluindo aprendizado de máquina, estatística, finanças, bioestatística e muitas outras. O PCA padrão realiza a decomposição da matriz de covariância da amostra, seleciona os vetores próprios correspondentes aos maiores valores próprios e, em seguida, constrói um subespaço de baixa dimensão com base nos vetores próprios selecionados. É notado que o PCA padrão, dependendo das aplicações, pode sofrer algumas deficiências, como (YANG; XU, 2015):

- O PCA é notoriamente frágil para os discrepantes e seu desempenho pode se degradar significativamente na presença de poucas amostras corrompidas, devido ao critério de erro quadrático usado;
- O PCA não pode utilizar informações adicionais dos componentes principais;
- Sua saída pode não ser interpretável visto que não é indicado para soluções esparsas.

Muitos esforços foram feitos para mitigar essas fraquezas do PCA. Nos últimos anos, numerosos algoritmos PCA robustos foram propostos para abordar principalmente o primeiro problema descrito.

Considerando os estudos em questão, para um resultado eficiente, optou-se por não reduzir os dados às suas principais componentes e aplicar tal ferramenta para detectar e avaliar apenas os valores discrepantes. Para isso foi estimado o algoritmo PCA robusto. ROBPCA é uma técnica valiosa na análise de dados. Na prática, o nível de robustez do contra discrepantes parece adequado em muitas situações e, portanto, o ROBPCA fornece uma maneira útil de examinar a estrutura dos dados e testar ao mesmo tempo discrepantes (CARONI, 2000)

O problema dos *outliers* é semelhante ao problema dos dados ausentes, no sentido de que valores extremos fornecem informações inexistentes ou incorretas (STACKLIES; REDESTIG, 2019). Então, tais pontos devem ser avaliados e tratados.

2.8. **Detecção do sobreajuste**

A qualidade do ajuste de uma função se refere a quão bem se aproxima a uma função de destino considerando que os algoritmos de aprendizado de máquina supervisionados procuram aproximar a função de mapeamento subjacente desconhecida para as variáveis de saída, dadas as variáveis de entrada. A habilidade de generalização de classificadores se refere a seu desempenho ao classificar padrões de teste que não foram utilizados durante o treinamento.

Deficiências na capacidade de generalização de um classificador podem ser atribuídas ao sobreajuste (*overfitting*), que se refere a um modelo que modela os dados de treinamento muito bem, sendo incapaz de detectar resultados diferentes em novos conjuntos de teste. Para Hawkins (2004), o sobreajuste é o uso de modelos ou procedimentos que violam a parcimônia, ou seja, que incluem mais termos do que o necessário ou usam abordagens mais complicadas do que o necessário. Assim, em vez de aprender a distribuição geral dos dados, o modelo aprende a saída esperada para cada ponto de dados.

A super adaptação é mais provável nos modelos não paramétricos e não lineares que têm mais flexibilidade ao aprender uma função de destino. Como tal, muitos

algoritmos não paramétricos de aprendizado de máquina também incluem parâmetros ou técnicas para limitar e restringir a quantidade de detalhes que o modelo aprende (BROWNLEE, 2016).

Um modelo adequadamente treinado não deve ser capaz de atingir uma exatidão maior que a fração de pontos de dados rotulados corretamente. Caso contrário, o modelo será superajustado.

Existem duas técnicas importantes que podem ser utilizadas ao avaliar algoritmos de aprendizado de máquina para limitar o super ajuste:

1. Técnica de reamostragem para estimar a exatidão do modelo.
2. Reter um conjunto de dados de validação.

A técnica de reamostragem mais popular é a validação cruzada *k-fold*. Ele permite que haja o teste do modelo o número de vezes estimado em diferentes subconjuntos de dados de treinamento e crie uma estimativa do desempenho de um modelo de aprendizado de máquina em dados não vistos.

Depois de selecionar e ajustar os algoritmos de aprendizado de máquina no conjunto de dados de treinamento, é possível avaliar os modelos aprendidos no conjunto de dados de validação.

Tais técnicas foram aplicadas no presente trabalho para evitar o problema de superajuste, mais ainda sim é necessário confirmar a existência desse problema. Para isso foram criados novos conjuntos de dados para ensaios e avaliação do efeito do superajuste.

Testes de permutação (ou testes de aleatorização ou testes exatos) são aplicados para identificar um modelo de super ajuste nas mesmas condições, mas em dados aleatórios. Quando as condições de modelagem são excessivamente ajustadas, elas geralmente fornecem um ajuste para dados aleatórios, o que é melhor do que seria esperado.

Os testes de permutação envolvem reordenar repetidamente e aleatoriamente o y-bloco, reconstruindo o modelo com as configurações de modelagem atuais após cada reordenamento. Após permutação do y-bloco, são registradas a exatidão e o índice Kappa para cada previsão usando o método de validação cruzada.

A verificação do super ajuste consiste em comparar os parâmetros de mérito alcançados em cada possibilidade desse teste com os parâmetros do conjunto original.

Se caso alguma dessas possibilidades apresente resultados iguais aos do original, é constatado a presença do problema.

2.9. **Parâmetros de mérito**

Os parâmetros de mérito são utilizadas para avaliar o desempenho dos modelos de classificação. A matriz de confusão de cada técnica demonstra a partir das linhas da matriz, as instâncias em uma classe prevista, enquanto as colunas correspondem às instâncias em uma classe real. A partir disso, são calculados os parâmetros na classificação. Existem muitas medidas de discriminação, sendo a exatidão de classificação a mais comum, devido à sua simplicidade e facilidade de compreensão (KIL; SHIN, 1996), além do índice Kappa. Já para avaliar a regressão, são estimados a média da diferença absoluta entre os valores previstos e o valor observado (MAE), a raiz do erro médio quadrático (RMSE) e coeficiente de determinação (R^2).

2.9.1. Exatidão

A exatidão de um classificador é a capacidade de diferenciar as classes entre si. Matematicamente relaciona a proporção dos verdadeiros positivos e verdadeiros negativos em todos os casos (BARATLOO et al., 2015), sendo calculada a partir dos valores preditivos, ou seja, verdadeiros positivos (TP), falsos positivos (FP), falsos negativos (FN) e verdadeiros negativos (TN), como na Equação 9, em que a soma dos valores verdadeiros positivos e negativos são divididos pela soma de todos valores preditivos.

$$\textbf{Exatidão} = \frac{\textbf{TP} + \textbf{TN}}{\textbf{TP} + \textbf{FP} + \textbf{FN} + \textbf{TN}} \tag{9}$$

2.9.2. Índice Kappa

O kappa é uma forma de coeficiente de correlação padronizado, muito usado para testar a confiabilidade entre classes e é definido como a quantidade de precisão nos dados devido à congruência entre os coletores de dados. Se estende a faixa de -1 a +1, em que 1 é a concordância perfeita entre os dados previstos e reais. Embora nos testes, valores negativos apareçam, na prática isso é improvável. Há sugestões de interpretação dos dados em que valores ≤ 0 são indicadores sem concordância, 0,01 – 0,20 são ligeiros, 0,21 – 0,40 são regulares, 0,41 – 0,60 são moderados, 0,61 – 0,80 são substanciais e 0,81 – 1,00 são considerados quase perfeito (MCHUGH, 2012). Allouche, Tsoar e Kadmon (2006) sugerem que Kappa de Kohonen pode ser deduzida de acordo com a Equação 10.

$$\kappa = \frac{\Pr(a) - \Pr(e)}{1 - \Pr(e)} \quad (10)$$

Em que Pr (a) representa a concordância real observada e Pr (e) representa acordo casual, definidos matematicamente como

$$\Pr(a) = \frac{TP + TN}{TP + FP + FN + TN} \quad (11)$$

$$\Pr(e) = \frac{(TP + FP)(TP + FN) + (FN + TN)(TN + FP)}{(TP + FP + FN + TN)^2} \quad (12)$$

2.9.3. MAE

Erro absoluto médio (MAE) é a média da diferença absoluta entre os valores previstos e o valor observado. O MAE é a mais intuitiva das métricas, já que é observado a diferença absoluta entre os dados e as previsões do modelo. Um pequeno MAE sugere que o modelo é ótimo na previsão, enquanto um grande MAE sugere que seu modelo pode ter problemas em determinadas áreas. Apresentando valor 0, o MAE indica que o modelo é um preditor perfeito das saídas.

2.9.4. RMSE

A raiz do erro médio quadrático (RMSE) mede a magnitude média dos erros em um conjunto de previsões sem considerar a sua direção, determinando a precisão para variáveis contínuas. Uma vez que os erros são elevados ao quadrado antes da média ser calculada, o RMSE atribui um peso relativamente elevado a um grande erro. Isto significa que o RMSE é útil quando grandes erros são particularmente indesejáveis (VINING; MONTGOMERY; PECK, 2001). O erro estimado de previsão para um modelo de regressão ajustado é mensurado pela raiz do erro quadrático médio na Equação 13.

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2} \tag{13}$$

Onde $\hat{y}_i$ é o valor estimado de i-ésima observação.

2.9.5. R^2

O coeficiente de determinação (R^2) é responsável por avaliar o grau de ajuste do modelo. Relaciona a soma quadrática da regressão e a soma quadrática total.

3. MATERIAIS E MÉTODOS

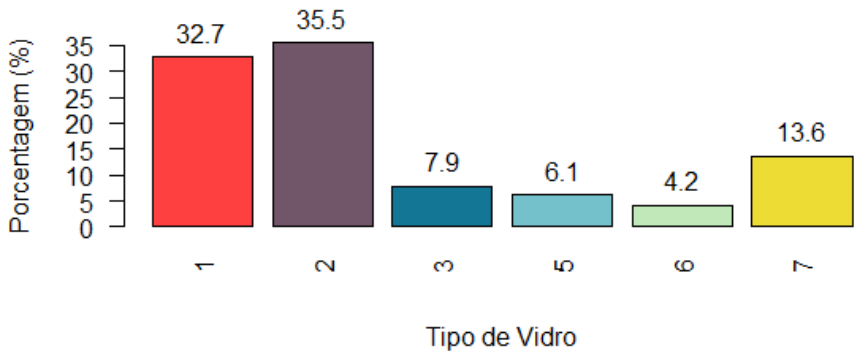
Para o presente trabalho, foram explorados três diferentes conjuntos de dados químicos de multi-classificação, todos disponíveis no Repositório de Aprendizado de Máquina da Universidade da Califórnia, Irvine (UCI) (BLAKE; MERZ, 2018) descritos como *Glass Identification Dataset e Wine Quality Dataset*, que por sua vez, possui dois conjuntos de dados, um para vinho tinto e outro para vinho branco.

3.1. Conjunto de dados: Identificação de Vidros

O conjunto de dados avaliado compreende em 214 casos e 10 variáveis, sem nenhum dado faltante. A variável 1 representa o índice de refração (RI) e as variáveis de 2-9 consistem na percentagem em massa do óxido correspondente de sódio, magnésio, alumínio, silício, potássio, cálcio, bário e ferro. Como última variável, está a classe preditora Tipo. O material é agrupado em sete tipos, distribuídos de acordo com a Figura 6, sendo o tipo 2 mais frequente, já o tipo 4 não é característico de nenhuma amostra do conjunto pré-determinado. Esses tipos são descritos como:

- Tipo 1: Vidro usado para construir janela processada com ar;
- Tipo 2: Vidro usado para construir janela processada sem ar;
- Tipo 3: Vidro usado para construir veículo processado com ar;
- Tipo 4: Vidro usado para construir veículo processado sem ar;
- Tipo 5: Vidro usado em contêineres;
- Tipo 6: Vidro usado na fabricação de louça;
- Tipo 7: Vidro usado na fabricação de faróis.

Figura 6 - Distribuição de amostras de vidro de acordo com os tipos 1 a 7 do Conjunto de Identificação de Vidro, exceto para o tipo 4, não presente no conjunto.

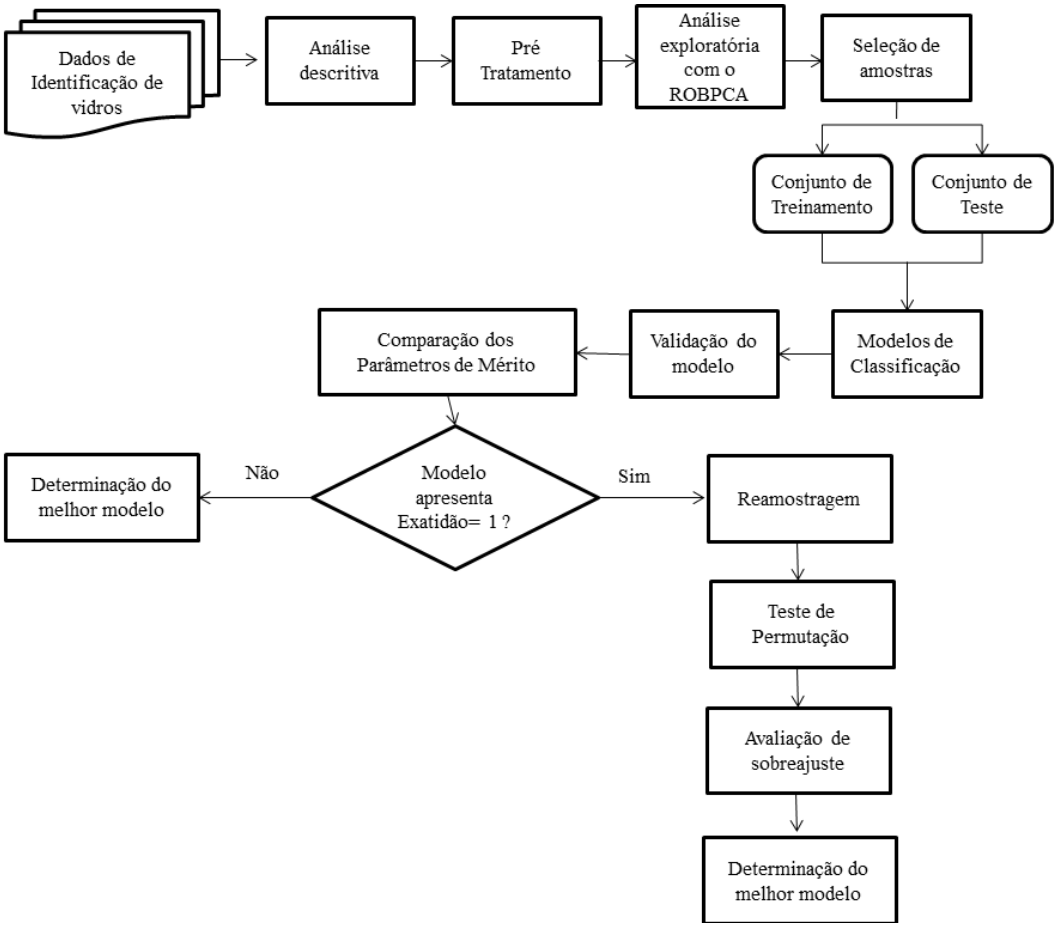


Fonte: O autor, 2020.

A abordagem do teste de permutação para detecção de super ajuste foi aplicado somente no conjunto de identificação de vidro, visto que foi o conjunto que alcançou valores máximos. Como nesse caso, há 6 tipos de vidro que são efetivamente encontrados, deveriam ser levantados 720 novos ensaios para avaliação dos parâmetros de exatidão e índice Kappa. Porém, estabeleceu o uso de 10% desse número de ensaios, ou seja, 72, considerando a extensão do valor e que com essa porcentagem, já seria possível alcançar o objetivo. Com os dados obtidos, foram construídos o histograma e a curva de densidade de probabilidade da precisão e do índice Kappa.

O fluxograma da Figura 7 mostra as etapas do estudo. A mesma inclui desde a importação do conjunto de dados no *software* R; a análise descritiva que inclui a avaliação de histogramas, gráfico de caixas e gráficos de correlação; o pré-tratamento de dados a partir do autoescalamento, da transformação Box-Cox e da transformação do sinal espacial; a análise exploratória através do PCA robusto; a seleção de amostras, dividindo-as em conjunto de treinamento e teste; aplicação dos modelos de classificação citados anteriormente; comparação dos parâmetros de mérito, exatidão e índice Kappa; Caso o melhor parâmetro de mérito atinja o valor máximo, seguir ao processo de reamostragem, fazer o teste de permutação com o modelo e comparar os parâmetros com os do conjunto original, verificando a possibilidade de sobreajuste.

Figura 7 - Fluxograma das etapas de estudos do conjunto de identificação de vidros



Fonte: O autor, 2020.

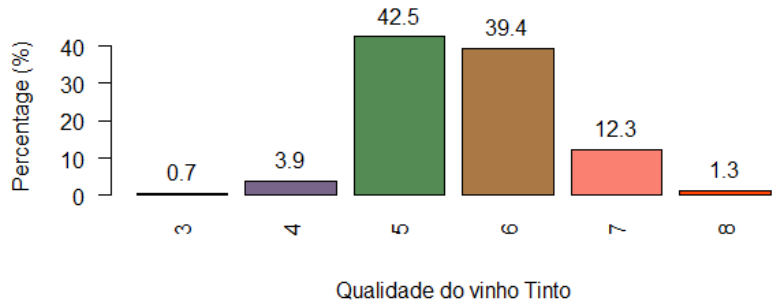
3.2. **Conjunto de dados: Qualidade de Vinhos**

De acordo com Ribeiro et al. (2009), este trabalho adotou parte dos dados recolhidos durante a fase de produção de vinho durante quatro anos na Quinta do Vinho no Minho (Norte de Portugal). É descrito as características físico-químicas das variantes do vinho "Vinho Verde" tinto e branco (CORTEZ et al., 2009), a fim de detectar a adulteração do vinho. Neste projeto, dois grandes conjuntos de dados são separados contendo 1.599 instâncias de vinho tinto e 4.989 instâncias de vinho branco caracterizados por 12 variáveis físico-químicas: acidez fixa, acidez volátil, ácido cítrico, açúcar residual, cloretos, dióxido de enxofre livre, dióxido de enxofre total, densidade, pH, teor de sulfato, teor alcoólico e classificação de qualidade. A classificação de

qualidade foi baseada em um teste sensorial realizado por pelo menos três sommeliers e escalado em 11 classes de qualidade de 0 - muito ruim a 10 – muito excelente (GUPTA, 2018).

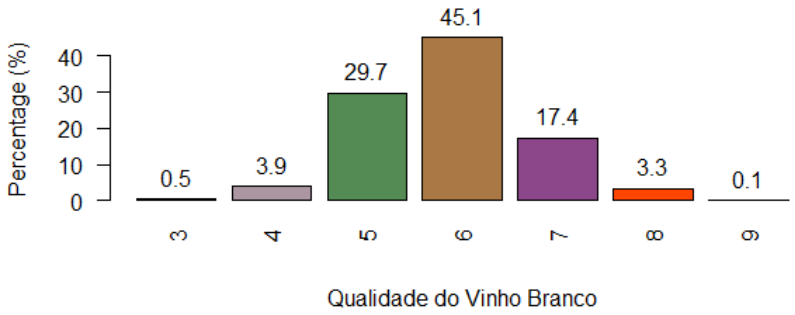
As frequências das qualidades de cada tipo de vinho estão descritas nas Figuras 8 e 9. O vinho tinto apresenta qualidade 5 mais frequente, seguida pela qualidade 6, sendo que as frequências mais inferiores, 1 e 2, nem as mais superiores, 9 e 10, não aparecem. Já o vinho branco apresenta mais amostras de qualidade 6, e ainda há presença de amostras com qualidade 9.

Figura 8 - Distribuição de amostras de vinho tinto de acordo com a qualidade estimada por sensoriamento do conjunto de qualidade de vinhos tinto



Fonte: O autor, 2020.

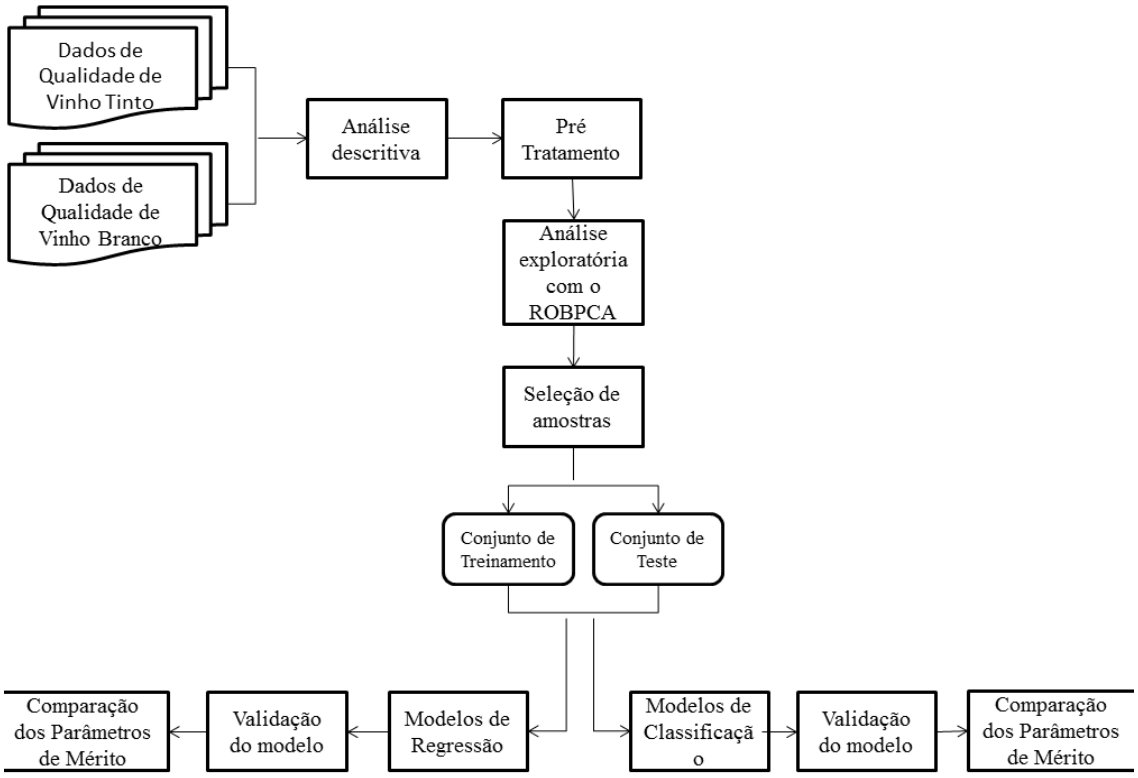
Figura 9 - Distribuição de amostras de vinho branco de acordo com a qualidade estimada por sensoriamento do conjunto de qualidade de vinhos branco



As etapas de estudo desses dois conjuntos são apresentadas no fluxograma da Figura 10. Trata-se da importação dos conjuntos de dados no *software* R; a análise

descritiva que inclui a avaliação de histogramas, Gráfico de caixas e gráficos de correlação; o pré-tratamento de dados utilizando o autoescalamento, as transformações Box-Cox e o sinal espacial; a análise exploratória através do PCA robusto; a seleção de amostras, dividindo-as em conjunto de treinamento e teste. No primeiro momento, foram estabelecidos os modelos de regressão para avaliar o melhor modelo e fazer a predição das amostras, avaliando a exatidão da classificação desse modelo. Posteriormente, com os conjuntos de treinamento e teste, foram aplicados os 15 modelos de classificação e validados, logo em seguida, foi comparado os parâmetros de mérito: exatidão e índice Kappa.

Figura 10 - Fluxograma das etapas de estudos dos conjuntos de vinho tinto e branco



Fonte: O autor, 2020.

3.3. **Software R**

Este estudo abrange diversas técnicas para sumarização e visualização dos conjuntos de dados analisados. Todas essas técnicas foram desenvolvidas no *software* R, uma linguagem e ambiente para a programação de estatísticas e construção de gráficos (R Core Team, 2014). Trata-se de um ambiente de programação e *software* livre para computação estatística e gráficos, que compila e roda em uma ampla variedade de plataformas.

A linguagem de programação R para computação estatística é um ambiente originalmente desenvolvido por Ihaka e Gentleman (1996), dando continuidade a linguagem S desenvolvida nos Laboratórios AT & T e S-Plus (CHAMBERS, 1998); S é uma linguagem projetada especificamente para visualização e exploração de dados, modelagem estatística e programação com dados, e sua evolução deu origem ao ambiente R com código livre.

A linguagem R é comumente usada para ensino e pesquisas nas áreas de estatística e previsão, mas ainda vem ganhando espaço aos poucos na comunidade acadêmica e de pesquisa dedicada à inteligência computacional devido às suas características, particularmente nos métodos de classificação e reconhecimento de padrões (VELÁSQUEZ; BRANCH, 2012).

O ambiente R é um *software* livre sob a licença GNU dada pela *Free Software Foundation*, obtido diretamente do *site* http://www.rproject.org/. Usuários podem interagir com o sistema através de uma linha de comando e vários pacotes fornecem ao ambiente uma interface gráfica baseada em menus e caixas de diálogo, como uma linguagem de programação fortemente relacionada com os paradigmas de programação funcional e orientada a objeto. O R apresenta alguns recursos como:

- Mecanismo para lidar com grandes quantidades de informações;
- Extensa coleção de ferramentas estatísticas para análise de dados;
- Sistema para criar e manipular gráficos de maior complexidade;
- Milhares de pacotes para estender a funcionalidade do ambiente, sendo adicionados novos constantemente;
- Linguagem de programação simples, eficaz e versátil;
- Sistemas para detecção e tratamento de exceções.

O R fornece suporte para aprendizado de máquina na forma de pacotes criados por uma comunidade. Essas ferramentas poderosas são gratuitas para *download*, mas

precisam ser instaladas antes de poderem ser usadas. Os pacotes aplicados neste trabalho foram *mlbench* (LEISCH; DIMITRIADOU, 2015), que contêm o conjunto de identificação de vidro salvo; *corrgram* (FRIENDLY, 2002) para criação do correlogramo; *caret* (KUHN, 2008), que contêm funções de pré-processamento, seleção de amostras e modelos quimiométricos; *rrcov* (TODOROV, 2018) contendo a função que realiza o PCA robusto; *class* (VENABLES; RIPLEY, 2002) para a função do LVQ.

4. RESULTADOS E DISCUSSÃO

4.1. Conjunto de Identificação de vidros

4.1.1. Estatística descritiva

Em busca de informações iniciais, é avaliado o sumário descritivo (Tabela 2) do conjunto de Identificação de vidros, onde é verificado que todas as variáveis são numéricas, representando o índice de refração e as porcentagens em massa do óxido correspondente, sem apresentar dados faltantes. O desenvolvimento da linguagem desse conjunto está contido no Anexo A. Observando a composição química das amostras avaliadas, o elemento em maior concentração é o silício e em alguns casos há ausência (ou estavam abaixo do limite de detecção) de elementos como magnésio, potássio, bário e ferro. O desenvolvimento do *script* no *software* desse conjunto está no Anexo A.

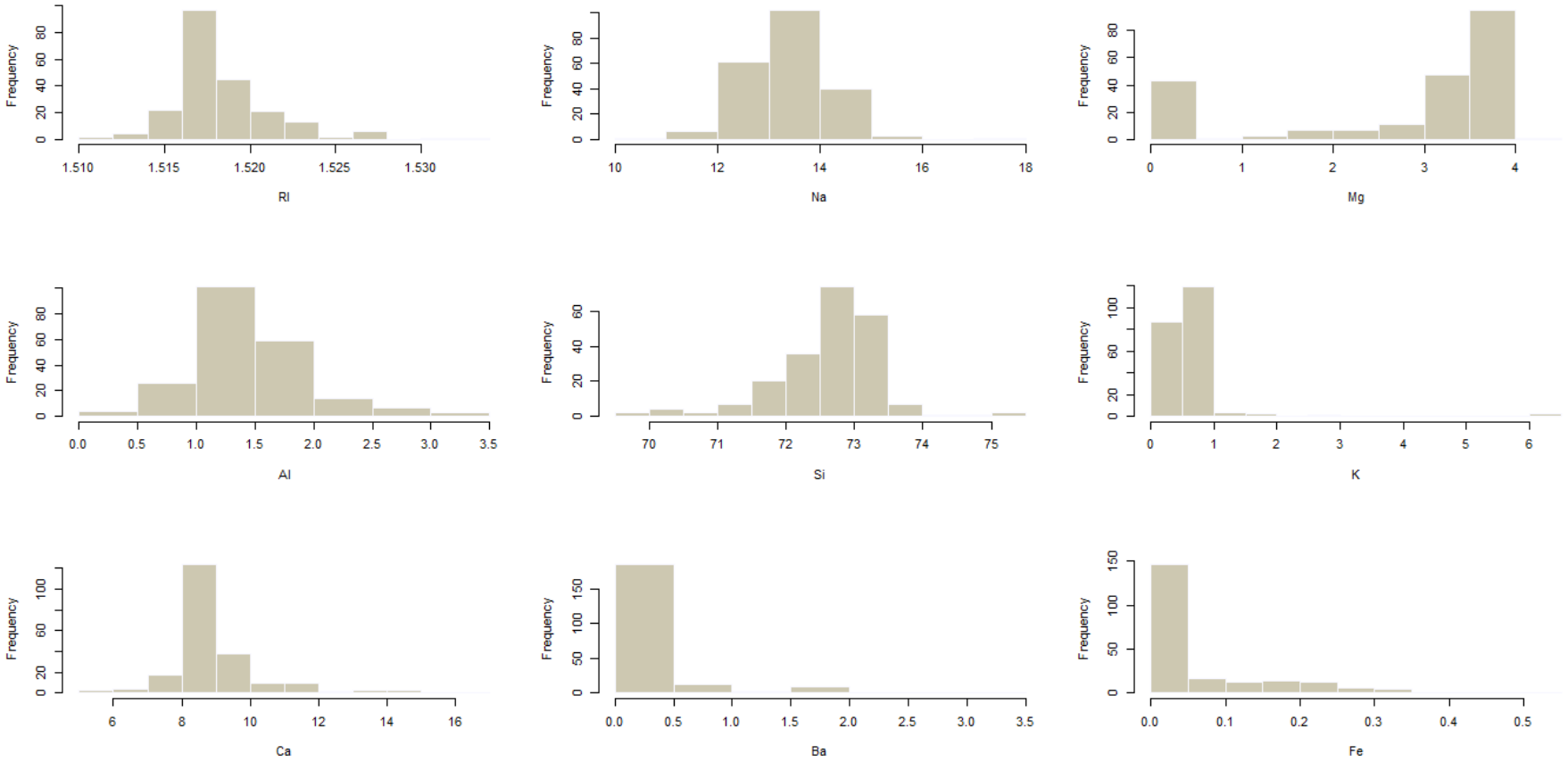
Tabela 2 - Sumário descritivo das variáveis do conjunto de Identificação de Vidros

Variável	**Mínimo**	**1° Quartil**	**Mediana**	**Média**	**3° Quartil**	**Máximo**
RI	1,51	1,52	1,52	1,52	1,52	1,53
Na (wt %)	10,73	12,91	13,30	13,41	13,82	17,38
Mg (wt %)	0,00	2,11	3,48	2,68	3,60	4,49
Al (wt %)	0,29	1,19	1,36	1,44	1,63	3,50
Si (wt %)	69,81	72,28	72,79	72,65	73,09	75,41
K (wt %)	0,00	0,12	0,55	0,50	0,61	6,21
Ca (wt %)	5,43	8,24	8,60	8,96	9,17	16,19
Ba (wt %)	0,00	0,00	0,00	0,18	0,00	3,15
Fe (wt %)	0,00	0,00	0,00	0,06	0,10	0,51

Fonte: O autor, 2020.

Na Figura 11 constam os histogramas univariados de todas as variáveis, destacando que as variáveis Si, Al e Na apresentam distribuição normal aproximada. Já as outras variáveis apresentam distribuições assimétricas.

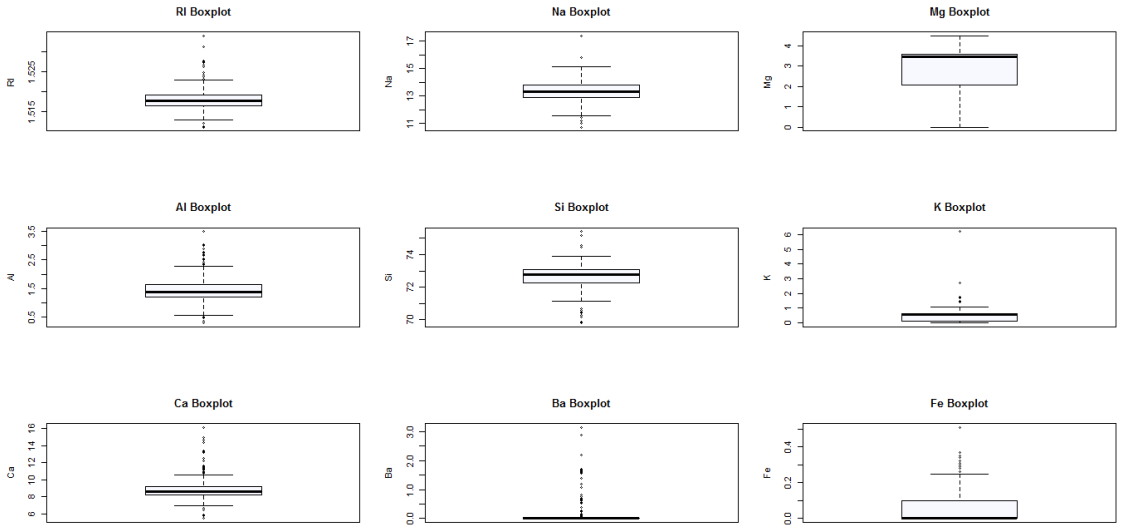
Figura 11 - Histogramas univariados do conjunto de Identificação de vidros demonstrando assimetria em todas variáveis



Fonte: O autor, 2020.

Nos gráficos de caixa univariados expostos na Figura 12, é possível notar a presença de valores atípicos, também chamados de *outliers*, em todas as variáveis. Assim, é notável a necessidade da etapa de pré-processamento de dados para as etapas posteriores.

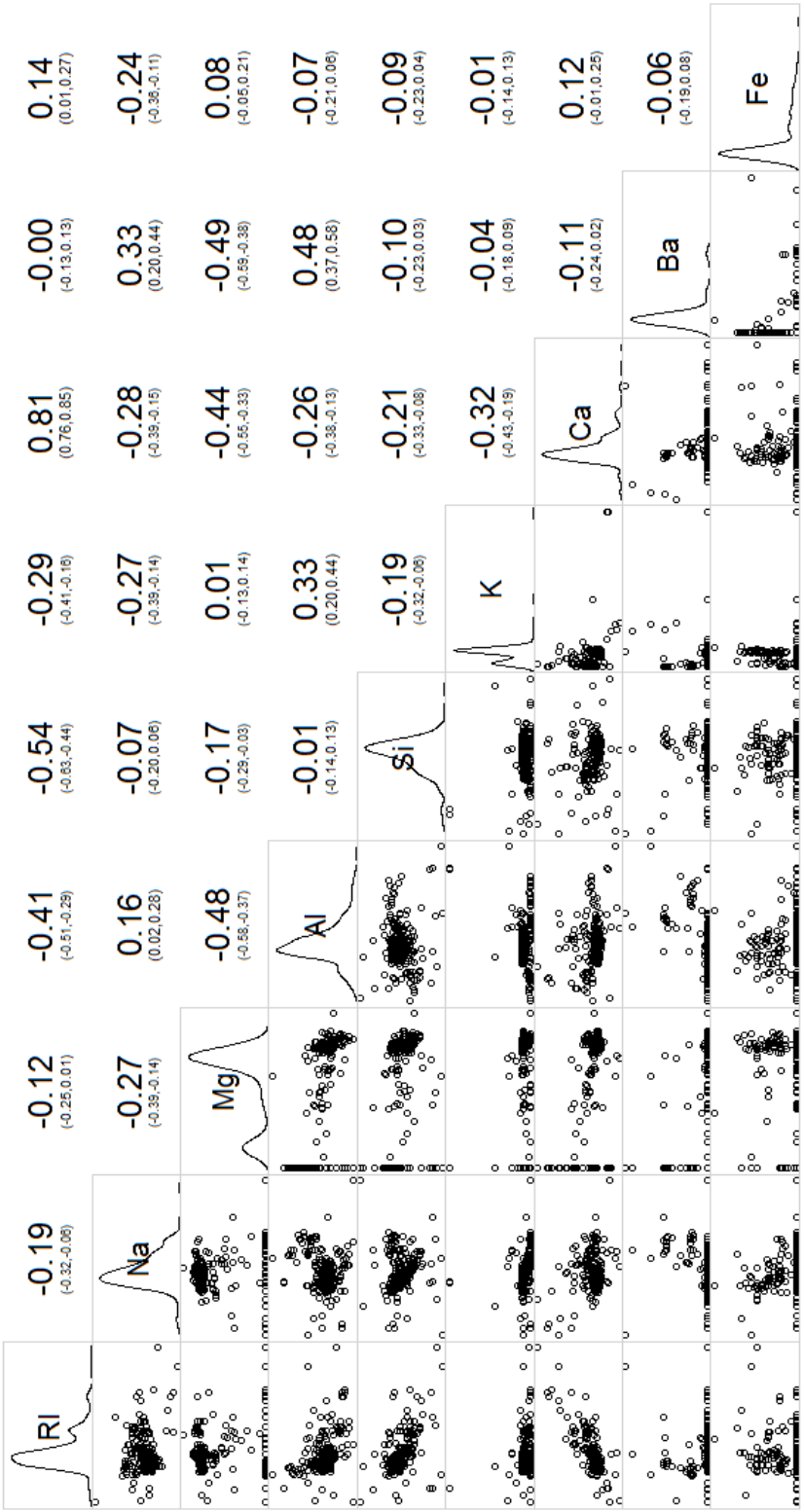
Figura 12 - Gráfico de caixas univariados do conjunto de Identificação de vidros em que os dados não seguem distribuição normal, com presença de *outliers.*



Fonte: O autor, 2020.

Ao avaliar a relação entre as variáveis (Figura 13) geradas a partir do pacote *corrgram* (FRIENDLY, 2002), é sugerido que não há relação do índice de refração com a porcentagem de óxido de bário, mas está altamente correlacionado com a presença de óxido de cálcio. Além disso, altos níveis de Si tendem a reduzir RI. As variáveis que apresentam sinal de coeficiente negativo são inversamente proporcionais, ou seja, a presença progressiva de óxido de alumínio tende a baixos níveis de óxido de magnésio, por exemplo. Ao contrário disso, as relações positivas são proporcionais, como o óxido de Bário que aumenta com presença gradativa de óxido de alumínio.

Figura 13 - Correlograma das variáveis do conjunto de Identificação de Vidro indicando baixa correlação entres as mesmas, confirmando que os dados não seguem distribuição normal



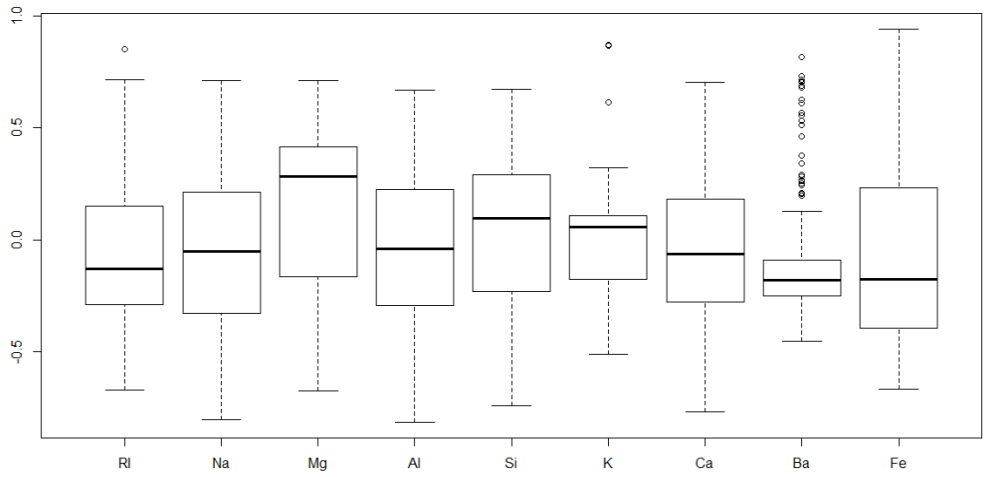
Fonte: O autor, 2020.

O pré-processamento dos dados garante que todas as variáveis tenham a mesma influência nos resultados da previsão, apresentando-se adimensional, com significados

reais, magnitudes e unidades semelhantes. Aplicando os métodos de processamento discutidos no capítulo 2 aliados às funções contidas no pacote *caret*, nota-se que as transformações causadas por esses métodos são eficientes, observando que os dados estão contidos no mesmo intervalo de distribuição (Figura 14).

Os métodos "*center*" e "*scale*" transformam o dimensionamento de cada preditor das amostras usando as médias e os desvios padrão do conjunto de dados, tornando os dados padronizados. Ao aplicar o "Box-Cox", as variáveis passam a seguir distribuição simétrica e a transformação do sinal espacial permite que os valores preditivos de todas as amostras sejam limitados a uma esfera multidimensional, eliminando os efeitos dos valores atípicos.

Figura 14 - Gráfico de caixas das variáveis do conjunto de Identificação de vidros após aplicação dos métodos de autoescalado, transformação Box Cox e transformação de sinais espaciais.



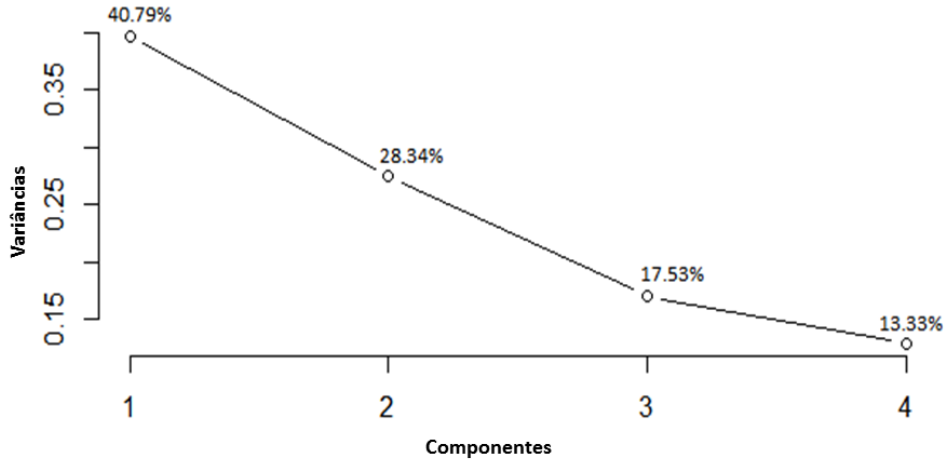
Fonte: O autor, 2020.

4.1.2. Análise exploratória de dados

O uso do algoritmo de PCA robusto elimina a influência de *outliers* na classificação, reduzindo os dados limitados as principais componentes. A Figura 15 mostra as variações explicadas para cada um dos quatro componentes principais determinados pela função *PcaHubert* contida no pacote *rrcov* (TODOROV;

FILZMOSER, 2009) e estima-se que 87% da variação seja explicada por três componentes principais.

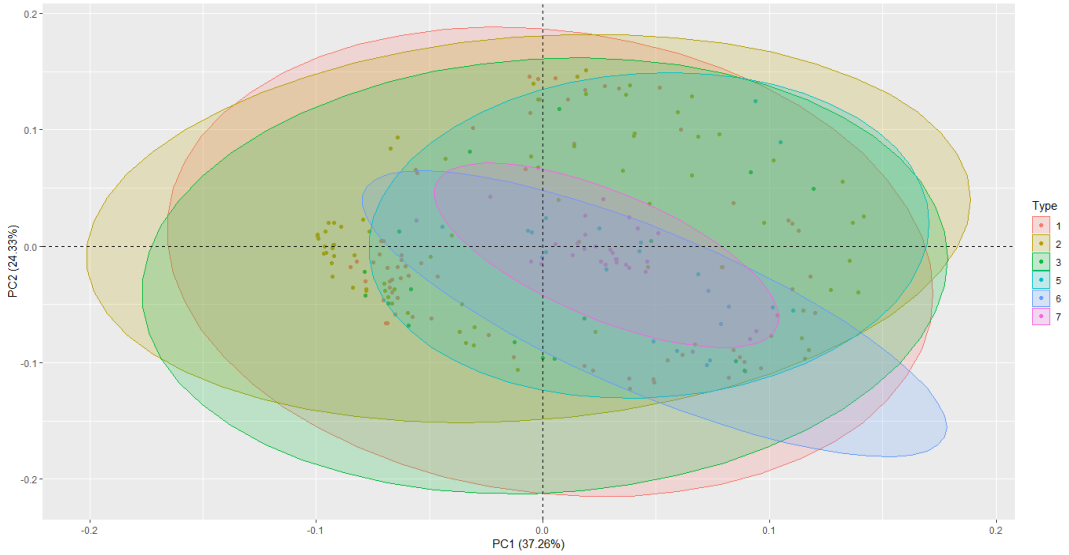
Figura 15 - Variação explicada ao longo dos principais componentes estimados pelo PCA Robusto no conjunto de dados de Identificação de Vidro



Fonte: O autor, 2020.

A contribuição de cada variável para as respectivas componentes principais é de complexa identificação, visto que os tipos de vidro não estão bem agrupados, como visto na Figura 16.

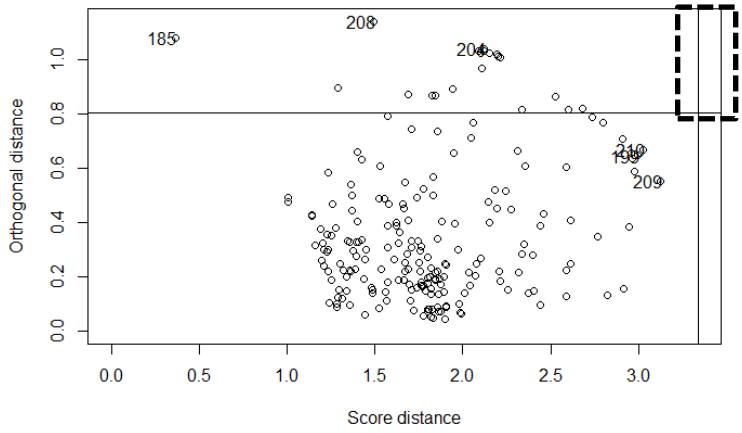
Figura 16 - Gráfico dos *scores* PC1 x PC2 das variáveis de amostras de Vidro sem agrupamentos definidos



Fonte: O autor, 2020.

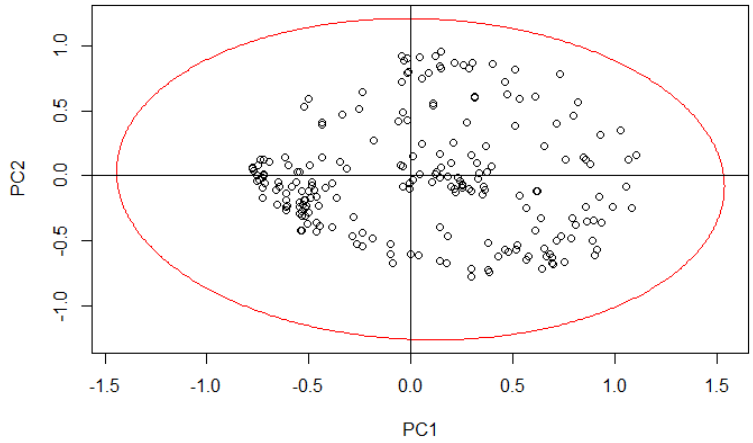
Na representação da distância ortogonal versus a distância dos *scores* do conjunto de dados (Figura 17), a linha vertical é a linha de corte dos valores discrepantes da distância dos *scores*, a linha horizontal é o ponto de corte dos valores discrepantes para a distância ortogonal. Portanto, os pontos situados à direita da linha vertical e acima da linha horizontal são identificados como *outliers*, delimitados pela linha tracejada. Nesse caso, os valores discrepantes não são encontrados, assim como apresentado na Figura 18 em que todos os pontos são agrupados dentro da elipse do intervalo de confiança de 95%.

Figura 17 - O mapa de *outliers* dos tipos de vidro, baseados em dois componentes principais obtidos com o ROBPCA sem indicação de *outliers*



Fonte: O autor, 2020.

Figura 18 - *Scores* de PC1 versus PC2 para amostras de vidro, delimitados pela elipse de 95% de confiança.



Fonte: O autor, 2020.

A ausência de *outliers* é uma situação esperada ao aplicar as transformações realizadas. Esse procedimento reduz o impacto da assimetria de dados e é de alta relevância, considerando que dados atípicos afetam significativamente o desempenho dos modelos aplicados na classificação.

4.1.3. Seleção de amostras

Antes de iniciar efetivamente a aplicação dos dados nos modelos, deve-se criar conjuntos de treinamento e teste. A função *createDataPartition* encontrada no pacote *caret* (KUHN, 2008) permite uma divisão equilibrada do conjunto de dados. Essa função prova ser adequada devido à amostragem aleatória dentro de cada classe, preservando a distribuição geral da classe dos dados. Assim, o algoritmo é treinado com um grande volume de dados de treino, depois validado o resultado deste algoritmo com os dados de teste e só então é possível aplicar o algoritmo em produção com confiança em que é possível prever dados reais. A frequência dos conjuntos subdivididos está descrita na Tabela 3.

Tabela 3 - Frequência dos conjuntos de treinamento e teste para o conjunto de vidro

Conjunto de dados	**Frequência**	
	Treino	**Teste**
Identificação de Vidros	162	52

Fonte: O autor, 2020.

4.1.4. Modelos preditivos de Classificação

O conjunto de treinamento estimado foi usado em todos os algoritmos e a validação cruzada de 10 vezes foi usada para cada modelo, visando comparar o

desempenho de cada um deles nos conjuntos de testes. Os parâmetros de mérito resultantes de cada modelo estão expostos na Tabela 4.

No conjunto de identificação de vidros, o modelo que apresentou melhores resultados foi o OLVQ1, apresentando exatidão máxima e índice Kappa perfeito. Os modelos lineares apresentaram exatidão média de 64,6%. Enquanto que os modelos de classificação baseados em árvores e regras têm exatidão média de 71,7%.

Tabela 4 - Desempenho dos índices de exatidão e Kappa para os modelos submetidos no conjunto de dados de Identificação de Vidros

Tipos de Modelos	**Modelos**	**Exatidão (%)**	**Índice Kappa**
Classificadores Lineares	PLS-DA	59,62	0,4295
	MDA	61,54	0,4707
	LDA	67,31	0,5474
	RDA	69,23	0,5646
Baseados em árvores	CART	67,31	0,5237
	C5.0	67,31	0,5483
	Bagging	71,15	0,6067
	GBM	73,08	0,6190
	RF	79,62	0,6738
Classificadores não lineares	ANN	57,69	0,4085
	SVM	57,69	0,3892
	K-NN	59,62	0,4301
	NB	61,54	0,4647
	OLVQ1	100,00	1,0000

Fonte: O autor, 2020.

Classificadores lineares são ferramentas simples que podem ser usadas para classificar amostras de um conjunto de dados, mas em todos os conjuntos existem amostras que não são linearmente separáveis, portanto, esses modelos apresentam resultados ineficientes.

Entretanto, a classificação baseada em árvores de decisão gera regras de interpretação natural, mas se no conjunto de dados de treinamento houver classes predominantemente maiores em várias instâncias, isso poderá afetar o aprendizado da máquina, como no caso desses conjuntos de dados, além de grandes possibilidades de modelos com sobreajuste. Tais situações podem ter influenciado os resultados

apresentados. Já os classificadores não lineares geram hipóteses mais sofisticadas e robustas, com separação de classes com mais eficiência.

Como tratado na introdução, os trabalhos contidos na literatura apresentam resultados mais satisfatórios em alguns desses modelos, como o estudo de El-khatib, Abu-Nasser e Abu-Naser (2019) que utilizou o modelo de rede neural e alcançou exatidão de 96,7%, enquanto que Zadora (2007) alcançou através do modelo de Máquinas de Vetor de Suporte (SVM), exatidão de 98,6%. Aldayel (2012) investigou a aplicação do k-NN na identificação de vidro que permitiu comparações com outras técnicas de mineração de dados, alcançando exatidão de 80,6%. De forma geral, resultados encontrados utilizando métodos diferentes contidos na literatura estão descritos na Tabela 5.

Tabela 5 – Exatidão de diferentes modelos quimiométricos descritos na literatura aplicados no conjunto de Identificação de Vidro

Método	**Exatidão %**	**Referência**
ANN	96,7	(EL-KHATIB; ABU-NASSER; ABU-NASER, 2019)
SVM	98,6	(ZADORA, 2007)
k-NN	80,6	(ALDAYEL, 2012)
Boosting KK	75,5	(ATHITSOS; SCLAROFF, 2004)
Naïve KK	73,2	(ATHITSOS; SCLAROFF, 2004)
Adaptive metric NN	75,2	(DOMENICONI; GUNOPULOS, 2001)
Discriminante Adaptivo NN	72,9	(DOMENICONI; GUNOPULOS, 2001)
C4.5	68,2	(DOMENICONI; GUNOPULOS, 2001)
Edição Wilson	67.4	(ZEIDAT; WANG; EICK, 2005)
Multi-edição	60.1	(ZEIDAT; WANG; EICK, 2005)
Edição de citações	70.0	(ZEIDAT; WANG; EICK, 2005)
Clustering Supervisionado	71.5	(ZEIDAT; WANG; EICK, 2005)
k-NN + HNB*	80.4	(ALDAYEL, 2012)
k-NN	61.9	(NABABAN; SITOMPUL; TULUS, 2018)
k-NN + Gain Ratio	63.0	(NABABAN; SITOMPUL; TULUS, 2018)
GLVQ - FDR	82.7	(HU; LIU, 2013)
J48	68.7	(GUPTA, 2015)
Mesa de decisão	68.2	(GUPTA, 2015)
Logistico	63.1	(GUPTA, 2015)
Perceptron multicamada	65.9	(GUPTA, 2015)
NB	47.7	(GUPTA, 2015)
RF	80.4	(GUPTA, 2015)
VFI	54.7	(GUPTA, 2015)
Zero R	35.5	(GUPTA, 2015)

Programação genética	66.8	(GUPTA, 2015)
k-NN	81.4	(DASGUPTA et al., 2019)
RF	79.1	(DASGUPTA et al., 2019)
SVM	81.4	(DASGUPTA et al., 2019)
LVQ	61,9	(AL-DAOUD, 2009)
LVQ	82,7	(HU; LIU, 2013)
OLVQ1	100.0	Presente trabalho

Fonte: O autor, 2020.

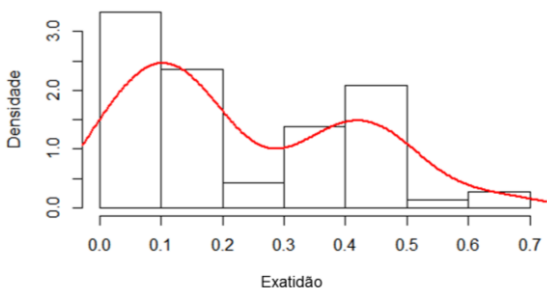
Em oposição aos resultados anteriores, o Vetor de Quantização de Aprendizagem de Máquina (OLVQ1) foi o modelo que apresentou o melhor desempenho em relação a todos os demais modelos empregados. Na literatura, há alguns trabalhos que aplicam o LVQ nesse mesmo conjunto de dados, como Al-Daoud (2009) que obteve uma exatidão de 61,9%, enquanto que Hu e Liu (2013), utilizando variantes do LVQ, conseguiram uma exatidão máxima de 82,7%. Assim, o presente trabalho consegue superar a exatidão encontrada por outros autores, mostrando-se como boa alternativa para classificação de dados desse tipo.

Para a aplicação desse algoritmo, foi utilizada a versão OLVQ1, otimização do LVQ1 original, que apresenta treinamento básico para a rede neural competitiva. Ao encontrar a posição ideal no espaço para seus protótipos, o modelo classificou os dados de acordo com a distância desses protótipos. A representação do modelo é um conjunto fixo de vetores de livro de códigos que se parecem com instâncias de treinamento, mas os valores de cada variável são adaptados com base no procedimento de aprendizado, permitindo alta exatidão. O LVQ é uma alternativa que usa uma abordagem probabilística, na qual os protótipos não são distribuídos nas centróides das classes, sendo colocados os vetores de livro de códigos dentro dos *clusters* individuais de cada classe, garantindo maior flexibilidade de modelagem em caso de sobreposição de dados, sem fazer suposições de distribuição de classe. Outro fator determinante nos resultados desse modelo é a normalização dos dados na etapa de transformação, uma vez que os valores reais nos vetores de entrada devem estar nessa condição.

4.1.5. Sobreajuste

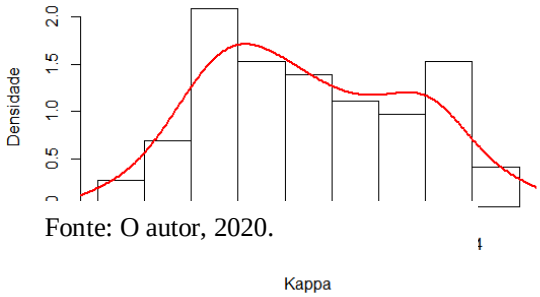
Ao aplicar o modelo LVQ nos 72 novos conjuntos de dados, criados pela aleatorização da classe principal (Tipo) na reamostragem, a exatidão máxima obtida foi de 65,38% e índice kappa alcançou valor máximo de 0,4994. A distribuição dos valores de exatidão se encontra na Figura 19 e do índice kappa na Figura 20. Assim, ambos os parâmetros não são compatíveis com os resultados do conjunto original; ou seja, eles não atingem 100% de precisão e o índice Kappa igual a 1. Isso pode ser confirmado pelo teste de Wilcoxon, que rejeita a hipótese nula de que os valores assumidos são iguais a 1 nos dois parâmetros.

Figura 19 - Distribuição dos valores de exatidão dos 72 ensaios do Teste de permutação



Fonte: O autor, 2020.

Figura 20 - Distribuição dos valores do Índice Kappa das 72 ensaios do Teste de permutação



Fonte: O autor, 2020.

Assim, considerando os resultados obtidos pelo teste de permutação, o sobreajuste do método de classificação não foi observado nesta análise.

Esse desfecho está diretamente ligado à utilização da validação cruzada de 10 vezes, que ao separar uma pequena amostra do conjunto em questão, otimiza o modelo e testa os parâmetros determinados durante a otimização nessa parte separada.

4.2. **Conjunto de Vinhos Branco**

4.2.1. Estatística descritiva

Avaliando todo o conjunto de dados inicial do conjunto de vinhos branco, é possível observar a presença de 937 amostras com as mesmas características, ou seja, amostras duplicadas. Essas amostras foram excluídas, a fim de reduzir a dimensionalidade e consequentemente aumentar a agilidade das etapas posteriores. Com isso, esse conjunto passou a ter 3961 amostras.

Todas as variáveis são numéricas e representam dados baseados em testes físico-químicos. O nível médio de açúcar residual é de 5,91 g/dm^3, mas há uma amostra de vinho muito doce com 65,8 g/dm^3. O dióxido de enxofre livre médio é de 34,89 mg/dm^3, sendo o valor máximo de 289 mg/dm^3, o que é bastante alto, pois 75% dos casos se apresenta até 45 g/dm^3. O pH do vinho está entre 2,72 e 3,82, com média de 3,19, não possuindo casos de vinhos básicos neste conjunto de dados, ou seja, sem altos níveis de pH conforme era esperado. Quanto ao teor alcoólico, a taxa se limita em 8% a 14,20 % (Tabela 6). O desenvolvimento do script desse conjunto está contido no Anexo B.

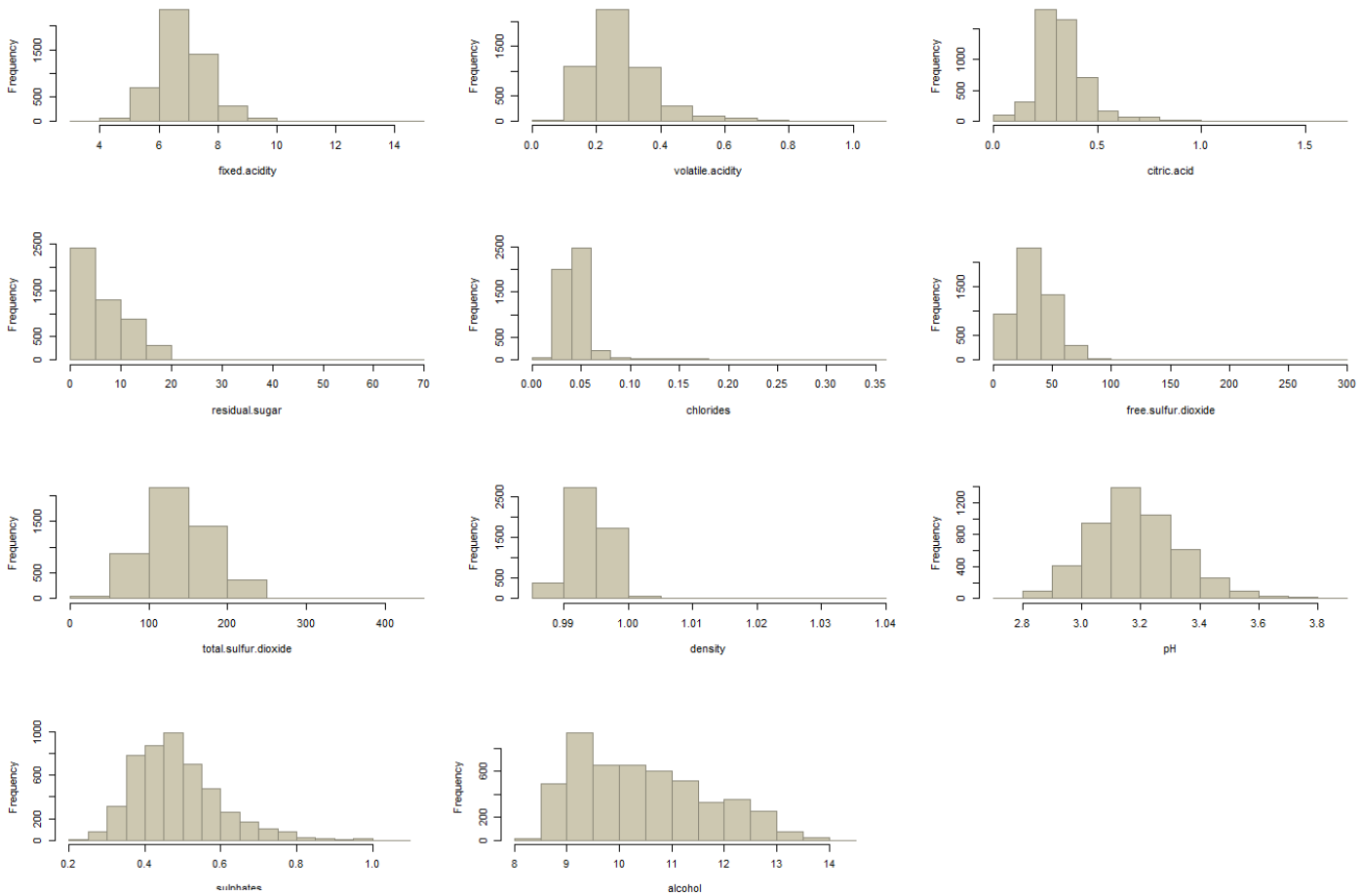
Tabela 6 - Sumário descritivo do conjunto de Qualidade do vinho Branco

Variável	**Mínimo**	**1° Quartil**	**Mediana**	**Média**	**3° Quartil**	**Máximo**
Acidez Fixa (g(ácido tártarico)/dm^3)	3,80	6,30	6,80	6,84	7,30	14,20
Acidez Volátil (g(ácido acético)/dm^3)	0,08	0,21	0,26	0,28	0,33	1,10
Ácido Cítrico (g/dm^3)	0,00	0,27	0,32	0,33	0,39	1,66
Açúcar Residual (g/dm^3)	0,60	1,60	4,70	5,91	8,90	65,80
Cloretos (g(cloreto de sódio)/dm^3)	0,01	0,03	0,04	0,04	0,05	0,34
SO_2 livre (mg/dm^3)	2,00	23,00	33,00	34,89	45,00	289,00
SO_2 Total (mg/dm^3)	9,00	106,0	133,00	137,20	166,00	440,00
Densidade (g/cm^3)	0,98	0,99	0,99	0,99	0,99	1,03
pH	2,72	3,09	3,18	3,19	3,29	3,82
Sulfatos (g(sulfato de potássio)/dm^3)	0,22	0,42	0,48	0,49	0,55	1,08
Teor alcoólico (vol.%)	8,00	9,50	10,40	10,59	11,40	14,20

Fonte: O autor, 2020.

A distribuição das variáveis observadas no histograma (Figura 21) sugere que somente o pH apresenta distribuição normal aproximada. Enquanto as outras variáveis apresentam assimetria para a direita sendo característica de distribuição assimétrica.

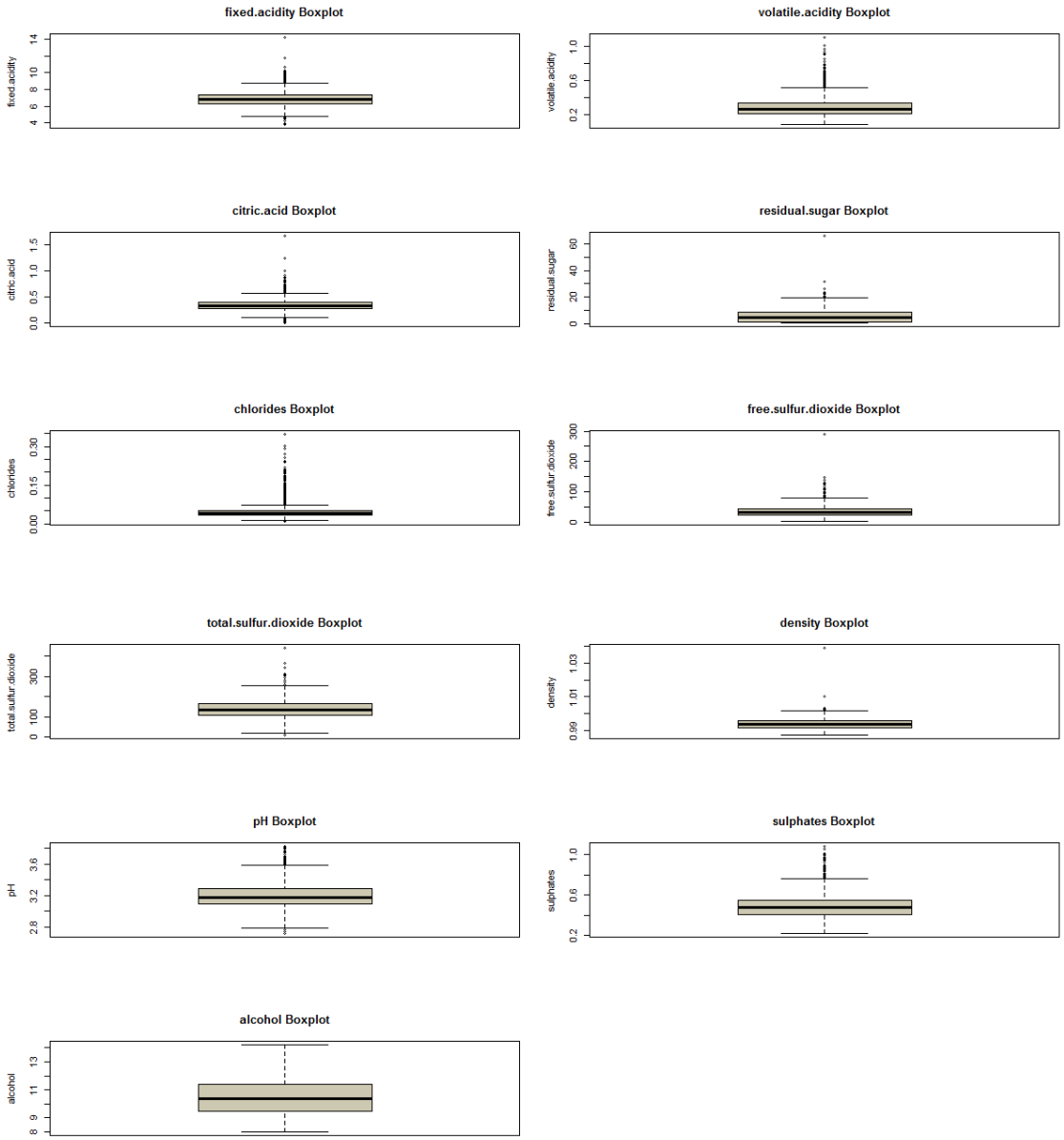
Figura 21 - Histogramas univariados do conjunto de Qualidade de vinhos Branco com ausência de distribuição normal em todos as variáveis.



Fonte: O autor, 2020.

A distribuição assimétrica é confirmada no Gráfico de caixas (Figura 22), além da presença de *outliers* em todas variáveis, exceto na variável álcool, que apresenta valores contidos apenas entre os valores mínimo e máximo. O pH é normalmente distribuído, com poucos valores discrepantes. Com isso, surge também a necessidade da etapa de pré-processamento.

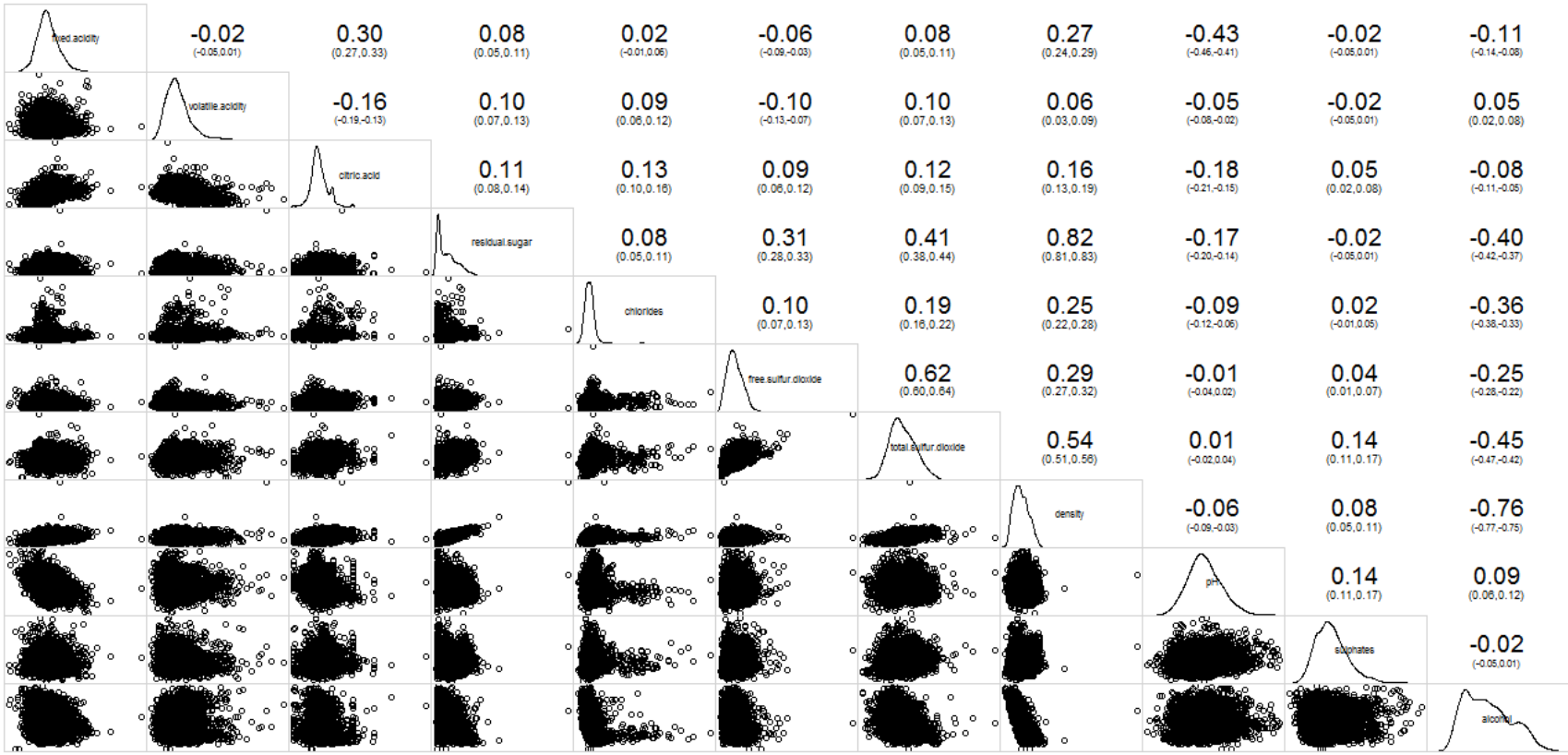
Figura 22 - Gráfico de caixas univariados do conjunto de Qualidade de vinhos Branco com a presença de *outliers* em todos as variáveis



Fonte: O autor, 2020.

A correlação entre todas as variáveis pode ser observada na Figura 23.

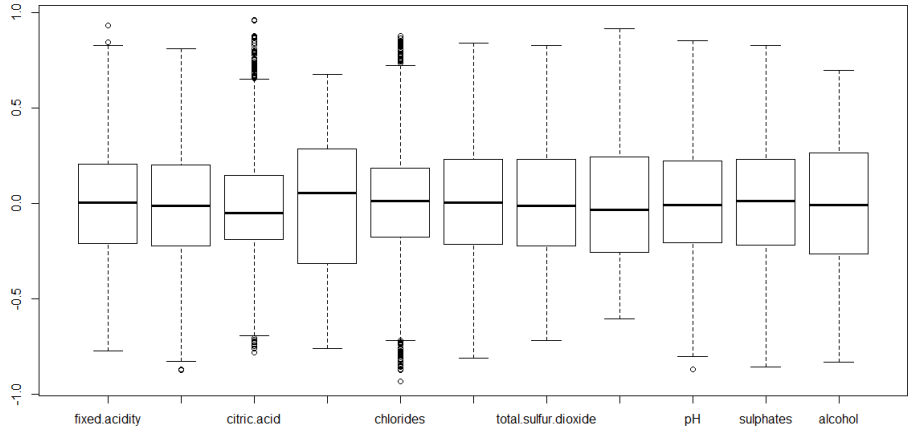
Figura 23 - Correlograma das variáveis do conjunto de Qualidade de Vinhos Branco apresentando baixa corelação entre as mesmas.



Fonte: O autor, 2020.

O pré-processamento garante a distribuição normal de todas variáveis (Figura 24), confirmadas pelo teste de normalidade Shapiro-Wilk.

Figura 24 - Gráfico de caixas das variáveis após aplicação dos métodos de autoescalado, transformação Box Cox e transformação de sinais espaciais do conjunto de Qualidade de vinhos Branco

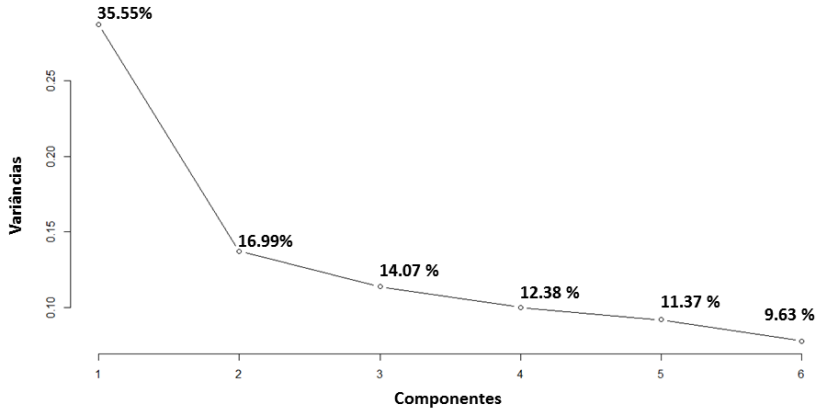


Fonte: O autor, 2020.

4.2.2. Análise exploratória de dados

Foram selecionados seis componentes pois eles explicam 99,99% da variância total, como apresentado na Figura 25, visto que houve redução considerável dos dados ao excluir as amostras duplicadas e o intuito da aplicação do PCA robusto no presente trabalho é apenas para indicação dos possíveis valores atípicos.

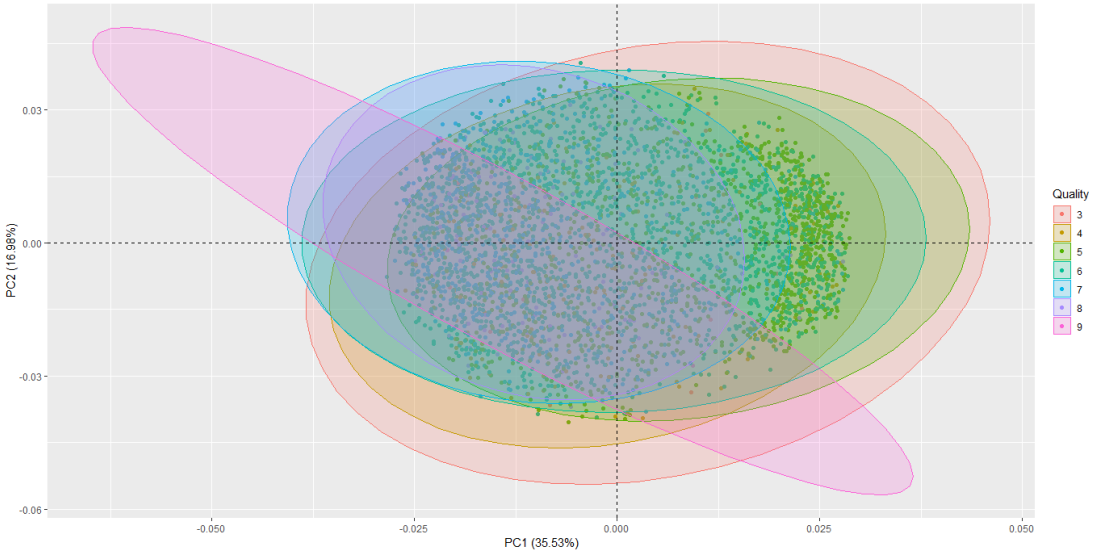
Figura 25 - Variação explicada ao longo dos principais componentes estimados pelo PCA Robusto no conjunto de dados de Qualidade de vinhos Branco



Fonte: O autor, 2020.

Nesse conjunto, também há grande sobreposição da classe preditora qualidade estimada do vinho branco como visualizado na Figura 26.

Figura 26 - Gráfico dos *scores* PC1 x PC2 das variáveis de amostras de vinho branco com sobreposição dos agrupamentos das qualidades

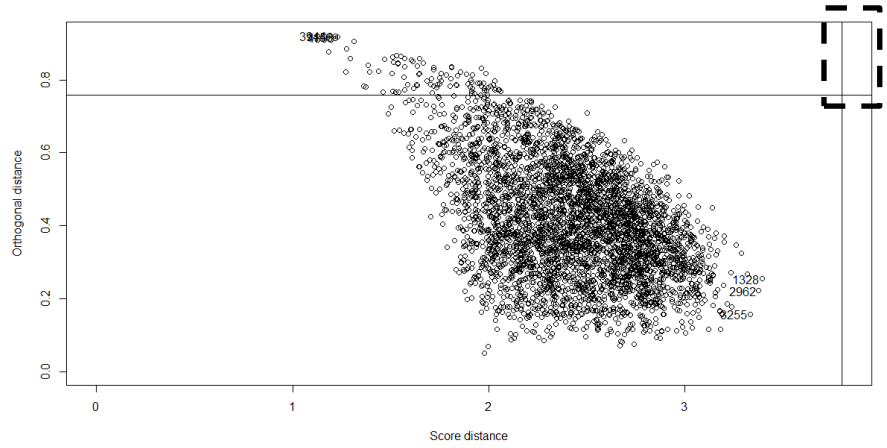


Fonte: O autor, 2020.

No mapa de *outliers* (Figura 27) e no gráfico de *scores* (Figura 28), verifica-se que todos os pontos extremos encontrados antes do pré processamento, foram agrupados dentro da elipse do intervalo de confiança de 95%. Assim, sem a presença de

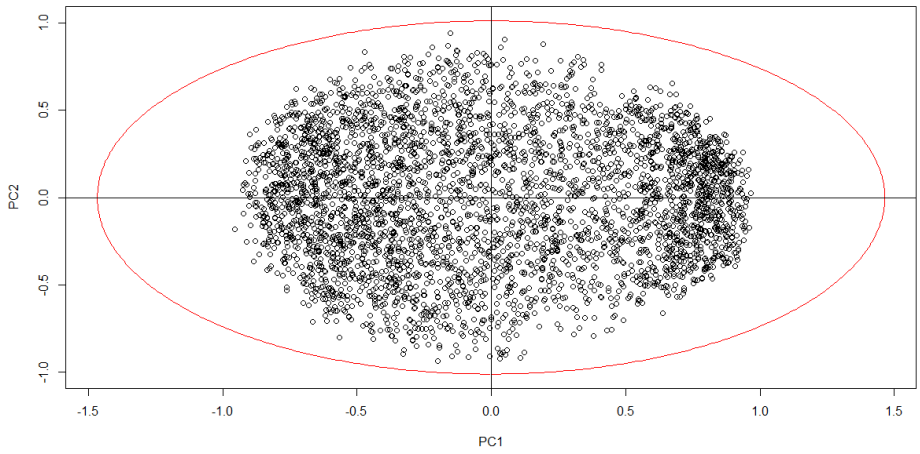
outliers é possível a aplicação confiável dos modelos de classificação.

Figura 27 - O mapa de *outliers* das qualidade de vinho branco, baseados em dois componentes principais obtidos com o ROBPCA



Fonte: O autor, 2020.

Figura 28 - *Scores* de PC1 versus PC2 para amostras de vinho branco, delimitados pela elipse de 95% de confiança.



Fonte: O autor, 2020.

4.2.3. Seleção de amostras

Para predição dos modelos, o conjunto de dados foi dividido em conjunto de treinamento e teste, estabelecendo uma frequência de 2/3 para treino e 1/3 para teste, como mostra a Tabela 7.

Tabela 7 - Dimensão dos conjuntos de treino e teste para os conjuntos de qualidade de vinhos branco

Conjunto de dados	**Frequência**	
	Treino	**Teste**
Qualidade de Vinhos Branco	2641	1320

Fonte: O autor, 2020.

4.2.4. Modelos preditivos de Regressão

Globalmente, a indústria do vinho vale quase 300 bilhões de dólares. Ser capaz de prever a qualidade do vinho seria uma adição muito valiosa para esta indústria. Assim, é necessário considerar um modelo que estime a qualidade do vinho para uma determinada amostra com base em um determinado conjunto de variáveis e que seja capaz de logo em seguida, classificar essa amostra corretamente.

A classificação de qualidade variável é considerada variável dependente e outras 11 variáveis são assumidas como preditores ou variáveis independentes neste trabalho (GUPTA, 2018). Hoje existem sistemas que trabalham com a qualidade do vinho, mas nenhum deles modela o problema dessa maneira para avaliar a qualidade do vinho com base em uma determinada amostra de vinho (UNIYAL; BARTHWAL, 2017).

Com o pacote *caret*, os modelos de regressão por Mínimos Quadrados Parciais (PLS), Árvores de Classificação e Regressão (CART), *Bagging*, Floresta Aleatória (RF), k-Vizinhos mais próximos (k-NN), Máquina de vetores de suporte (SVM) com *kernel* radial e linear, e Máquina Generalizada *Boosted* (GBM), apesentam os parâmetros de mérito descritos nas Tabelas 8, 9 e 10. O melhor modelo de regressão foi o modelo SVM com base radial, que levou a um erro absoluto médio de 0,5460. Enquanto, o modelo CART propõe uma função com MAE de 0,6098.

Tabela 8 - MAE referente aos modelos de regressão aplicados ao conjunto de Vinho Branco

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.5360742	0.5603706	0.5868167	0.5812122	0.6007003	0.6174390	0
svmLinear	0.5341584	0.5609138	0.5867392	0.5798526	0.5953846	0.6165793	0
svmRadial	0.5092071	0.5334236	0.5460670	0.5466979	0.5691759	0.5780827	0
knn	0.5259119	0.5540825	0.5654534	0.5673191	0.5890152	0.5999579	0
cart	0.5784592	0.5911848	0.6098312	0.6153848	0.6449290	0.6569899	0
bagging	0.5523987	0.5723798	0.5895363	0.5909278	0.6088571	0.6231802	0
rf	0.5059611	0.5393057	0.5480555	0.5476765	0.5684592	0.5736120	0
gbm	0.5333735	0.5573642	0.5743257	0.5703517	0.5846225	0.6064958	0

Fonte: O autor, 2020.

Tabela 9 - RMSE referente aos modelos de regressão aplicados ao conjunto de Vinho Branco

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.6927478	0.7193296	0.7501781	0.7460400	0.7759302	0.7898400	0
svmLinear	0.6983023	0.7238560	0.7523667	0.7485853	0.7758811	0.7915370	0
svmRadial	0.6634842	0.6917175	0.7173787	0.7156249	0.7480566	0.7597029	0
knn	0.6674241	0.7065705	0.7417888	0.7333724	0.7552156	0.7980033	0
cart	0.7247360	0.7407805	0.7723854	0.7779349	0.8211948	0.8329577	0
bagging	0.6987256	0.7197005	0.7440712	0.7460892	0.7753327	0.7982137	0
rf	0.6381097	0.6803767	0.7145760	0.7040772	0.7236554	0.7595637	0
gbm	0.6673787	0.7156719	0.7228110	0.7243095	0.7385992	0.7933318	0

Fonte: O autor, 2020.

Tabela 10 - R^2 referente aos modelos de regressão aplicados ao conjunto de Vinho Branco

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.2295955	0.2821424	0.2944154	0.2948941	0.3082773	0.3882986	0
svmLinear	0.2331055	0.2844088	0.2893837	0.2929449	0.3041906	0.3812375	0
svmRadial	0.2751659	0.3485030	0.3533745	0.3526342	0.3670604	0.4249652	0
knn	0.2238789	0.3183701	0.3380790	0.3234409	0.3465933	0.3618484	0
cart	0.1397208	0.2105961	0.2363849	0.2337579	0.2492902	0.3052945	0
bagging	0.1907309	0.2853374	0.2974259	0.2949303	0.3130747	0.3827028	0
rf	0.2647420	0.3460545	0.3897482	0.3764212	0.4055308	0.4599266	0
gbm	0.2248524	0.3158995	0.3459840	0.3372507	0.3686619	0.4080500	0

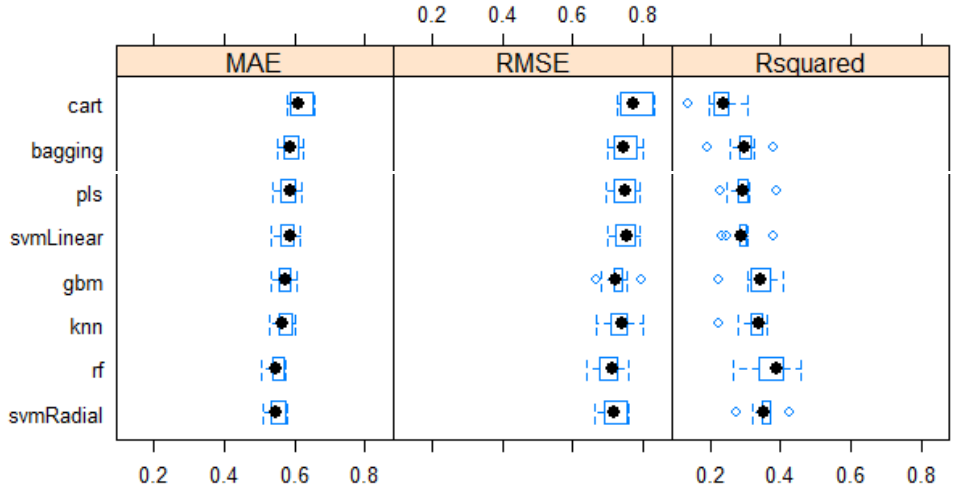
Fonte: O autor, 2020.

Na Tabela 10, os coeficientes de determinação estão muito abaixo do esperado. Isso confirma, que esses modelos não são adequados a esse caso e irão implicar em baixas precisões na tarefa de modelagem. As informações das tabelas acima estão graficamente dispostas na Figura 29.

Após aplicação dos modelos de regressão, os dados gerados foram classificados usando os mesmos algoritmos, alcançando os parâmetros descritos na Tabela 11. O SVM com base radial, que apresentou melhores parâmetros na regressão, é o modelo

que alcança maior exatidão (56,7%). De qualquer forma, esse resultado não é satisfatório.

Figura 29 - Gráfico de pontos das métricas dos modelos de regressão para o conjunto de Vinhos Branco



Fonte: O autor, 2020.

Cortez et al. (2009) utilizou esse modelo para estudo de caso de certificação de vinhos e alcança uma exatidão de 86,8% para o vinho branco, sendo superior ao encontrado pelo presente trabalho.

Tabela 11 - Exatidão e índice Kappa da classificação do conjunto de vinhos branco

Modelos	Exatidão (%)	Índice Kappa
CART	51,89	0,1964
BAGGING	52,27	0,2437
KNN	55,08	0,3022
PLS	52,50	0,2267
SVM LINEAR	54,17	0,2550
GBM	54,09	0,2720
RF	56,29	0,2933
SVM RADIAL	56,67	0,3054

Fonte: O autor, 2020.

Com os modelos de regressão verifica-se resultados que não apresentam boas predições. Isso é explicado pela complexidade dos dados, considerando que a qualidade

estimada é dada por uma pesquisa que considera testes sensoriais e apresenta baixa correlação com as características físico-químicas correspondentes. Assim, para esse caso, a regressão multivariada utilizando essa metodologia não é adequada, pois apresenta resultados insatisfatórios.

4.2.5. Modelos Preditivos de classificação

O conjunto de vinhos brancos é classificado a partir dos 14 modelos descritos anteriormente. Na Tabela 12, os classificadores lineares apresentam média apenas de 52,99%, valor bem próximo da média da exatidão dos modelos de árvores de classificação e baseados em regras, com 52,65%.

Já nos classificadores lineares, assim como no conjunto de identificação de vidro, o modelo LVQ também apresentam resultados satisfatórios, com exatidão de 99,54% e Índice Kappa de 0,9932. Esse resultado supera ao encontrado no trabalho de Omatu, Yano e Ikeda (2015), que classifica essas mesmas amostras com exatidão de 97,5%. Er e Atasoy (2016), após aplicar PCA e utilizando validação cruzada nesse mesmo conjunto, alcançou exatidão de 39,8%, 64,5%, e 70,7% para os modelos SVM, k-NN e RF, respectivamente.

Tabela 12 - Desempenho da exatidão e do índice Kappa para os modelos submetidos no conjunto de dados de Qualidade do Vinho Branco

Tipos de Modelos	**Modelos**	**Exatidão (%)**	**Índice Kappa**
Classificadores Lineares	PLS-DA	53,23%	0,2301
	MDA	50,95%	0,2477
	LDA	53,45%	0,2651
	RDA	53,15%	0,2750
Árvores classificatórias e baseadas em Regras	CART	51,03%	0,2229
	C5.0	49,58%	0,2131
	Bagging	53,15%	0,2616
	GBM	53,38%	0,2579
	RF	56,11%	0,2951
Classificadores não lineares	ANN	53,76%	0,2718

SVM	52,54%	0,2593
K-NN	53,53%	0,2657
NB	47,38%	0,2357
OLVQ1	99,54%	0,9932

Fonte: O autor, 2020.

4.3. **Conjunto de vinho tinto**

4.3.1. Estatística descritiva

No caso do vinho tinto, há presença de 240 amostras duplicadas. Com a exclusão das mesmas, esse conjunto passa a possuir 1359 instâncias. De acordo com a Tabela 13, o valor do pH parece exibir uma distribuição normal, com amostras principais exibindo valores entre 2,74 e 4,01. O teor alcoólico parece variar de 8,40 a 14,90, com picos maiores em torno de 9, com uma contagem menor entre 13 e 14. O desenvolvimento do script desse conjunto está contido no Anexo C.

Tabela 13 - Sumário descritivo do conjunto de Qualidade do vinho Tinto

Variável	**Mínimo**	**1° Quartil**	**Mediana**	**Média**	**3° Quartil**	**Máximo**
Acidez Fixa (g(ácido tártarico)/dm^3)	4,60	7,10	7,90	8,311	9,20	15,90
Acidez Volátil (g(ácido acético)/dm^3)	0,12	0,39	0,52	0,52	0,64	1,58
Ácido Cítrico (g/dm^3)	0,00	0,09	0,26	0,27	0,43	1,00
Açúcar Residual (g/dm^3)	0,90	1,90	2,20	2,52	2,60	15,50
Cloretos (g(cloreto de sódio)/dm^3)	0,01	0,07	0,07	0,08	0,09	0,61
SO_2 livre (mg/dm^3)	1,00	7,00	14,00	15,89	21,00	72,00
SO_2 Total (mg/dm^3)	6,00	22,00	38,00	46,83	63,00	289,00
Densidade (g/cm^3)	0,99	0,99	0,99	0,99	0,99	1,00
pH	2,74	3,21	3,31	3,31	3,40	4,01
Sulfatos (g(sulfato de potássio)/dm^3)	0,33	0,55	0,62	0,65	0,73	2,00
Teor alcoólico (vol.%)	8,40	9,50	10,20	10,43	11,10	14,90

Fonte: O autor, 2020.

Os histogramas das variáveis do vinho tinto (Figura 30) são semelhantes com os do vinho branco, exceto que, além do pH apresentar distribuição normal aproximada, a densidade também apresenta essa condição.

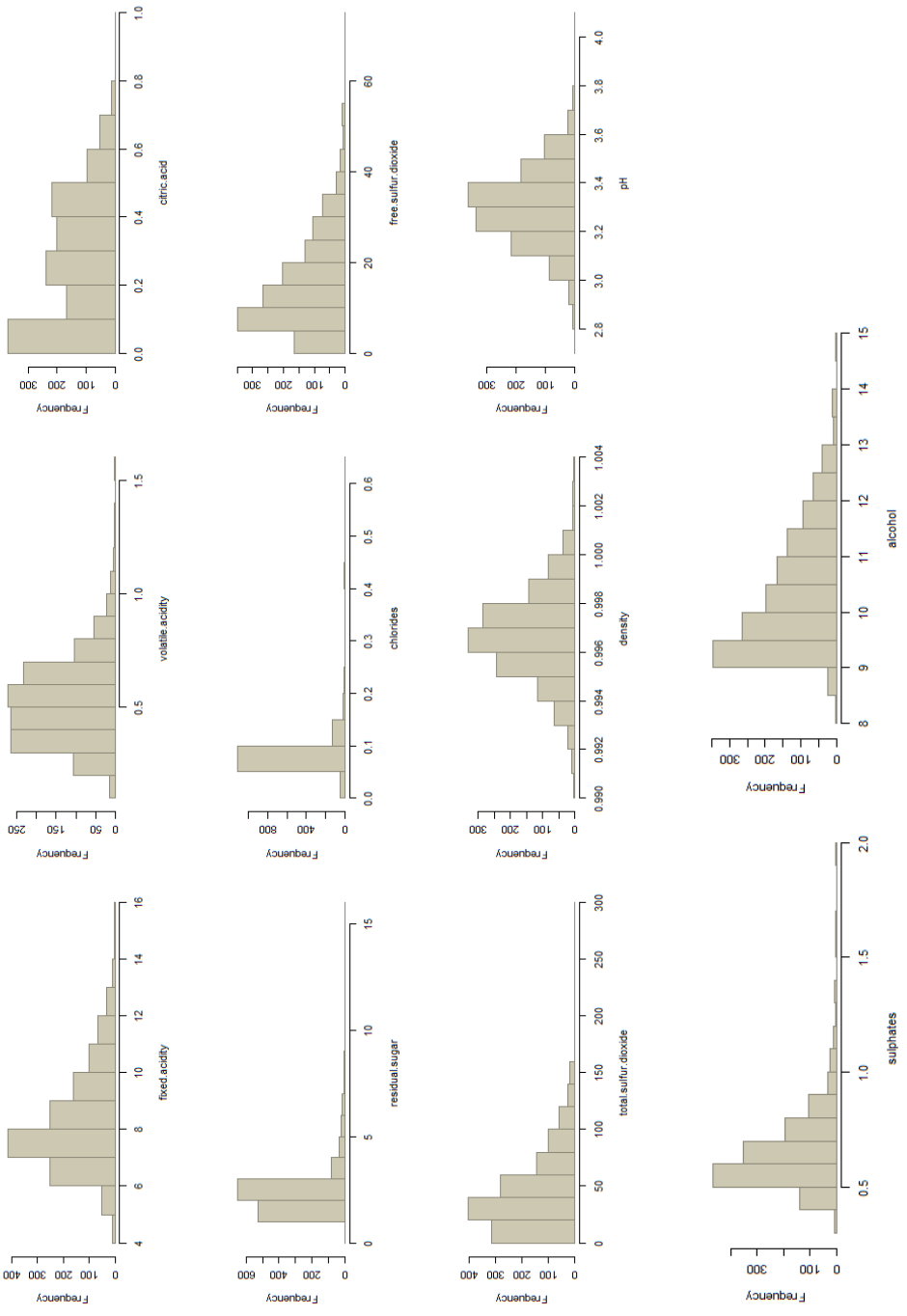
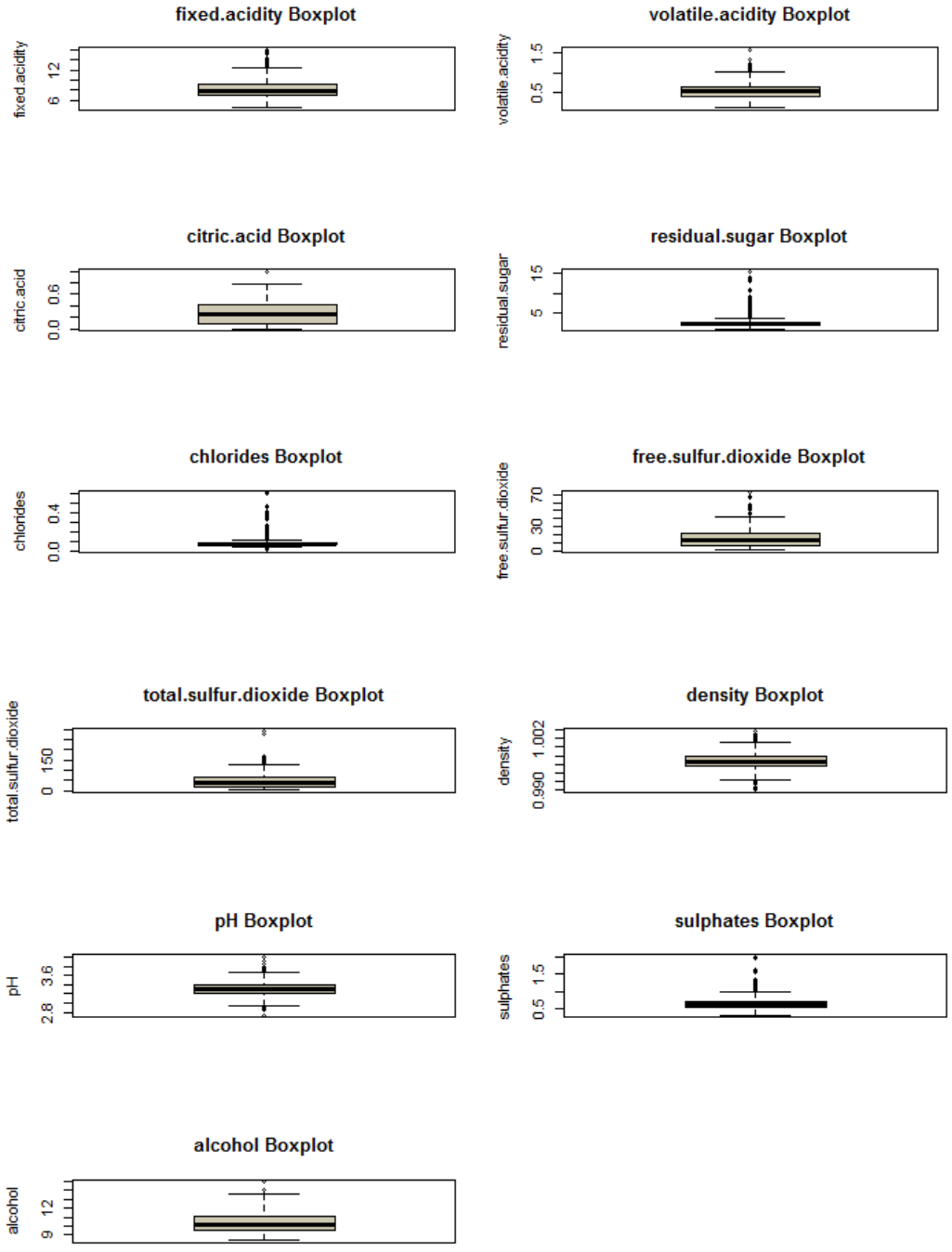


Figura 30 - Histogramas univariados do conjunto de Qualidade de vinhos Tinto com ausência de distribuição normal em todas variáveis

Fonte: O autor, 2020.

Já no gráfico de caixas (Figura 31), as variáveis são assimétricas e há possíveis *outliers*, sugerindo a aplicação dos métodos de pré processamento.

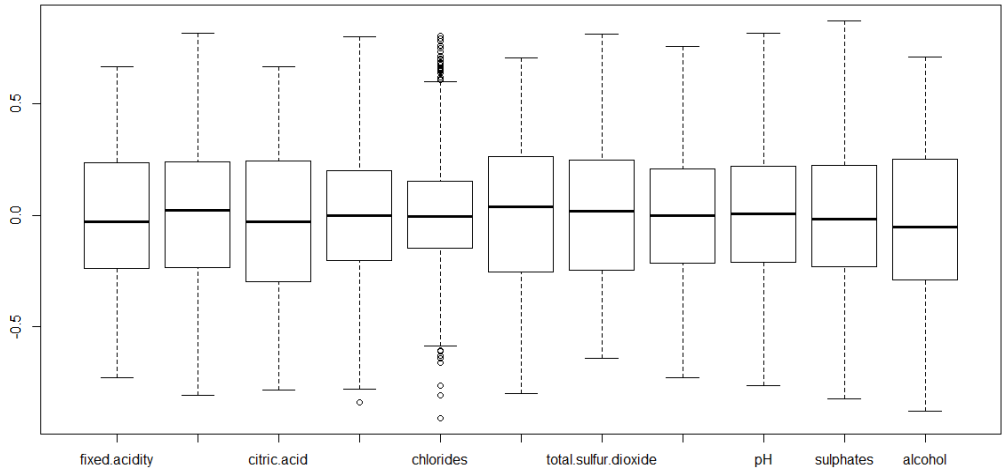
Figura 31 - Gráfico de caixas univariados do conjunto de Qualidade de vinhos Tinto com presença de *outliers* em todas as variáveis



Fonte: O autor, 2020.

Após o uso dos métodos de tratamento de dados, a distribuição segue uma distribuição aproximadamente normal para todas variáveis, apresentando média igual a zero e desvio padrão igual a 1, como observado na Figura 32.

Figura 32 - Gráfico de caixas das variáveis após os métodos de autoescalado, transformação Box-Cox e transformação de sinais espaciais do conjunto de qualidade de vinhos tinto

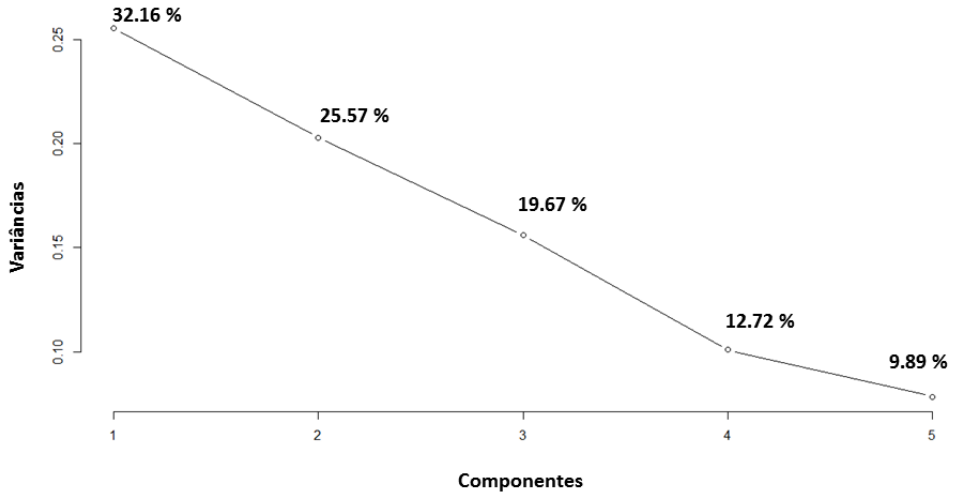


Fonte: O autor, 2020.

4.3.2. Análise exploratória de dados

O PCA robusto aplicado nesse conjunto de dados estima cinco componentes principais (Figura 33) e todos foram utilizados.

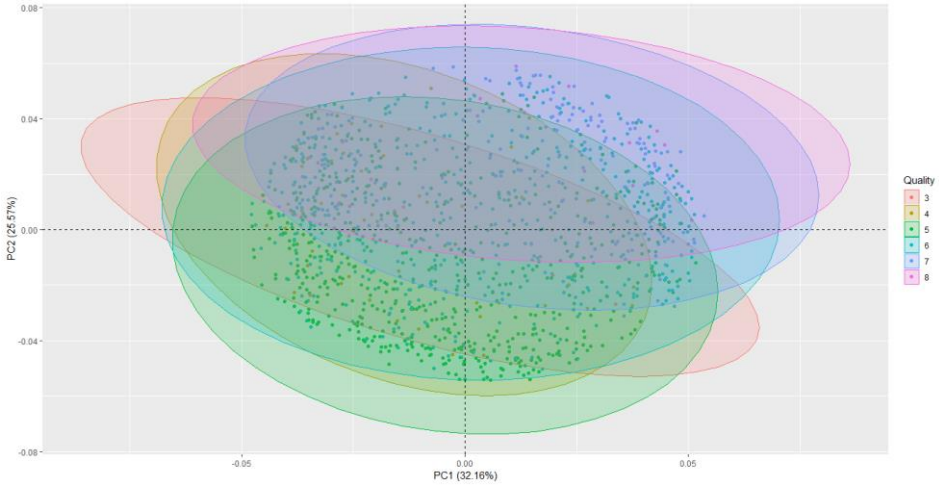
Figura 33 - Variação explicada ao longo dos principais componentes estimados pelo PCA Robusto no conjunto de dados de qualidade de vinhos tinto



Fonte: O autor, 2020.

Essa técnica também permite verificar regiões de sobreposição das qualidades do vinho e que não há presença de pontos fora das elipses de agrupamento, como demonstrado na Figura 34, o que sugere a ausência de *outliers*.

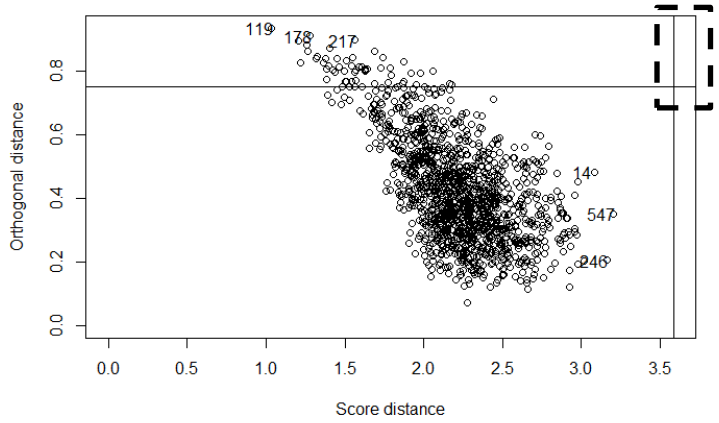
Figura 34 - Gráfico dos *scores* PC1 x PC2 das variáveis de amostras de vinho tinto com grande sobreposição dos agrupamentos das qualidades



Fonte: O autor, 2020.

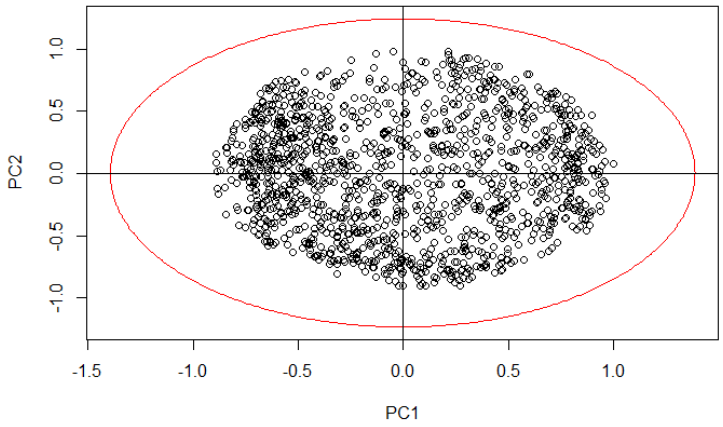
Na Figura 35, confirma-se que com o pré-processamento, não há presença de *outliers*, visto que não há pontos fora dos limites das distâncias dos *scores* e ortogonal. Na Figura 36, verifica-se o bom agrupamento dos pontos, todos contidos na elipse de 95% de confiança, constatando que o PCA robusto é uma ferramenta estatística eficaz para diagnóstico de pontos atípicos.

Figura 35 - O mapa de *outliers* das qualidade de vinho tinto, baseados em dois componentes principais obtidos com o ROBPCA



Fonte: O autor, 2020.

Figura 36 - *Scores* de PC1 versus PC2 para amostras de vinho tinto, delimitados pela elipse de 95% de confiança.



Fonte: O autor, 2020.

4.3.3. Seleção de amostras

Antes da aplicação efetiva dos modelos quimiométricos, as amostras são subdivididas em conjuntos de treinamento e teste, de acordo com a Tabela 14, garantindo a predição dos modelos aplicando o conjunto de teste.

Tabela 14 - Dimensão dos conjuntos de treino e teste para os conjuntos de vinho tinto

Conjunto de dados	**Frequência**	
	Treino	**Teste**
Qualidade de Vinhos Tinto	909	450

Fonte: O autor, 2020.

4.3.4. Modelos preditivos de regressão

Assim como o conjunto de vinhos branco, o problema é modelado de forma a prever o número da qualidade do vinho tinto para qualquer amostra. Este estudo investiga a influência de diferentes fatores no sabor dos vinhos. Consequentemente, um produtor pode definir estrategicamente o preço de sua classe de vinhos. O objetivo é avaliar empiricamente como a acidez, o açúcar residual, o álcool e outros produtos químicos, afetam o sabor do vinho, portanto, a qualidade do vinho.

Os modelos de regressão utilizados apesentam parâmetros de mérito descritos nas Tabelas 15 e 16. Tanto as métricas MAE e RMSE estão acima do esperado, demonstrando que tais modelos não são capazes de predizer a qualidade de forma efetiva. A métrica R^2 também não apresenta bons resultados, sugerindo baixa qualidade do ajuste obtido, indicados na Tabela 17.

Tsiliki et al. (2015) buscaram modelos de regressão preditiva através de abordagens de modelagem diferentes nesse mesmo conjunto, usando diversos modelos quimiométricos como o SVM e o RF, que alcança R^2 de 0,396 e 0,500, respectivamente e RMSE de 0,002 para todos os casos.

Tabela 15 - MAE referente aos modelos de regressão aplicados ao conjunto de Vinho Tinto

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.4347	0.5017	0.5141	0.5078	0.5201	0.5540	0
svmLinear	0.4351	0.5034	0.5074	0.5052	0.5221	0.5532	0
svmRadial	0.3990	0.4819	0.4998	0.4874	0.5078	0.5226	0
knn	0.4481	0.4948	0.5192	0.5111	0.5320	0.5383	0
cart	0.5308	0.5538	0.5649	0.5737	0.5737	0.6414	0
bagging	0.4685	0.4937	0.5113	0.5136	0.5314	0.5604	0
rf	0.4430	0.4854	0.5120	0.5013	0.5159	0.5330	0
gbm	0.4462	0.4783	0.5121	0.5052	0.5223	0.5626	0

Fonte: O autor, 2020.

Tabela 16 - RMSE referente aos modelos de regressão aplicados ao conjunto de Vinho Tinto

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.5493	0.6460	0.6592	0.6578	0.6704	0.7472	0
svmLinear	0.5590	0.6464	0.6590	0.6620	0.6758	0.7533	0
svmRadial	0.5497	0.6370	0.6467	0.6491	0.6786	0.6966	0
knn	0.5782	0.6444	0.6590	0.6629	0.6922	0.7395	0
cart	0.6496	0.6747	0.6901	0.7008	0.7230	0.7806	0
bagging	0.5844	0.6363	0.6525	0.6551	0.6877	0.7085	0
rf	0.5656	0.6331	0.6448	0.6401	0.6563	0.7025	0
gbm	0.5584	0.6209	0.6443	0.6451	0.6665	0.7155	0

Fonte: O autor, 2020.

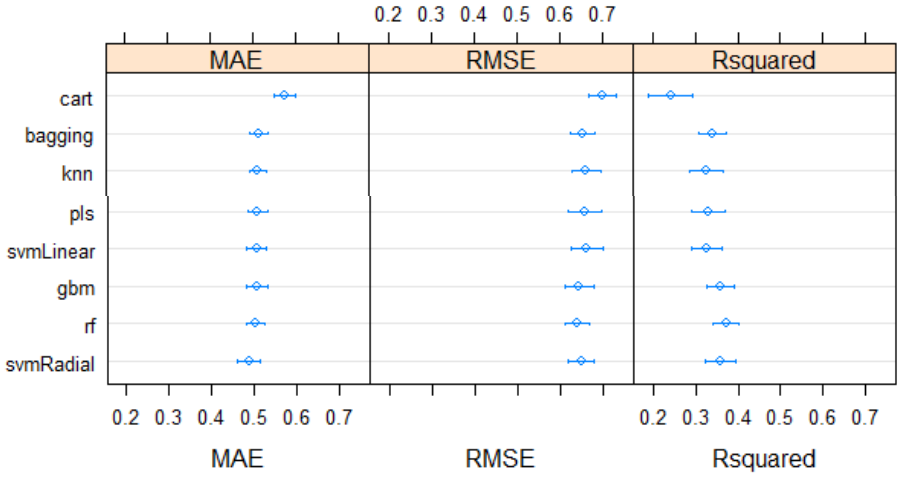
Tabela 17 - R^2 referente aos modelos de regressão aplicados ao conjunto de Vinho Tinto

Modelo	Mínimo	1º Quartil	Mediana	Média	3º Quartil	Máximo	NA's
pls	0.23583	0.3054	0.3318	0.3301	0.3445	0.4600	0
svmLinear	0.23016	0.3101	0.3218	0.3267	0.3445	0.4377	0
svmRadial	0.29102	0.3258	0.3528	0.3592	0.3859	0.4603	0
knn	0.22751	0.2942	0.3229	0.3262	0.3726	0.4058	0
cart	0.09941	0.2088	0.2689	0.2437	0.3032	0.3057	0
bagging	0.23730	0.3220	0.3450	0.3395	0.3657	0.3924	0
rf	0.31482	0.3403	0.3709	0.3719	0.3998	0.4378	0
gbm	0.29760	0.3369	0.3549	0.3586	0.3870	0.4374	0

Fonte: O autor, 2020.

A baixa correlação da variável qualidade em relação as outras variáveis está diretamente relacionada com esses resultados insatisfatórios. Os mesmos estão dispostos na Figura 37, com nível de confiança de 95%.

Figura 37 - Gráfico de pontos das métricas dos modelos de regressão para o conjunto de vinho tinto



Fonte: O autor, 2020.

Após a criação das funções para cada modelo quimiométrico, procedeu com a predição dos dados visando a classificação dos dados gerados por essas funções. A exatidão e o índice Kappa estão demonstrados na Tabela 18. O GBM apresentou melhores resultados, sendo de qualquer forma, valores não adequados para uma boa classificação. Na literatura, diversos trabalhos apresentam valores superiores aos alcançados (BHATTACHARJEE, 2016; CORTEZ et al., 2009).

Tabela 18 - Exatidão e índice Kappa da classificação do conjunto de vinhos tinto

Modelos	Exatidão (%)	Índice Kappa
CART	52,00	0,1980
BAGGING	59,10	0,3410
KNN	55,80	0,2880
PLS	58,60	0,3210
SVM LINEAR	58,20	0,3090
GBM	60,40	0,3530
RF	58,20	0,3240
SVM RADIAL	59,50	0,3390

Fonte: O autor, 2020.

Assim como o conjunto de vinhos branco, os resultados não satisfatórios indicam que a aplicação dos modelos de regressão não é recomendada ao aplicar a

metodologia desenvolvida, visto a baixa relação entre as variáveis dependentes e a variável preditora.

4.3.5. Modelos preditivos de Classificação

Para os vinhos tintos, os classificadores lineares também não apresentam índices adequados, com média de 58,62% e para as árvores classificatórias, média de 57,82%. Mais uma vez, o OLVQ1 se mostra como modelo mais adequado, com exatidão de 99,33% e índice kappa de 0,9896, como demonstrado na Tabela 19.

Tabela 19 - Desempenho da exatidão e do índice Kappa para os modelos submetidos no conjunto de dados de Qualidade de Vinhos Tinto

Tipos de Modelos	**Modelos**	**Exatidão (%)**	**Índice Kappa**
Classificadores Lineares	PLS-DA	59,78%	0,3167
	MDA	52,44%	0,2427
	LDA	60,89%	0,3565
	RDA	58,67%	0,3259
Árvores classificatórias e baseadas em Regras	CART	55,78%	0,2463
	C5.0	55,56%	0,2874
	Bagging	56,89%	0,2943
	GBM	59,33%	0,3320
	RF	61,56%	0,3637
Classificadores não lineares	ANN	59,11%	0,3131
	SVM	60.22%	0,3347
	K-NN	59,56%	0,3360
	NB	55,56%	0,3151
	OLVQ1	99,33%	0,9896

Fonte: O autor, 2020.

Para os conjuntos de Vinho tinto e Vinho branco, poucos trabalhos estão contidos na literatura com intuito de classificar tais amostras. Omatu, Yano e Ikeda (2015) e Cortez et al. (2009) alcançam exatidãos inferiores utilizando diferentes modelos quimiométricos em relação ao valor encontrado no presente trabalho, como descrito na Tabela 20.

Tabela 20 – Exatidão encontrada na literatura para diferentes modelos quimiométricos para os conjuntos de Qualidade de Vinho

Vinho Tinto

Método	**Exatidão %**	**Referência**
Regressão Múltipla	88,6	(CORTEZ et al., 2009)
Perceptron Multicamada	88,8	(CORTEZ et al., 2009)
SVM	89,0	(CORTEZ et al., 2009)
LVQ	83,4	(OMATU; YANO; IKEDA, 2015)
OLVQ1	99,3	Presente trabalho

Vinho Branco

Método	**Exatidão %**	**Referência**
Regressão Múltipla	84,3	(CORTEZ et al., 2009)
Perceptron Multicamada	84,7	(CORTEZ et al., 2009)
SVM	86,8	(CORTEZ et al., 2009)
OLVQ1	99.7	Presente trabalho

Fonte: O autor, 2020.

Assim, considerando os resultados obtidos e os resultados contidos na literatura para o mesmo banco de dados, a rede neural OLVQ1 proposta é uma boa abordagem para resolver problemas de classificação.

5. PERSPECTIVAS FUTURAS

Compreensivelmente, essa visão geral sistemática de LVQ pode servir apenas como ponto de partida para pesquisas e aplicações adicionais na quimiometria. De forma abrangente, os estudos propostos para o futuro são:

- Implantação e comparação do LVQ e suas variantes na solução de problemas químicos reais;

- Exploração abrangente dos parâmetros de configurações possíveis do LVQ;

- Testar outros modelos de regressão multivariada para estimar a Qualidade do Vinho que sejam capazes de produzir resultados satisfatórios;

CONCLUSÕES FINAIS

Métodos de pré-processamento e transformação de dados foram aplicados em conjuntos de dados previamente conhecidos para reduzir as diferenças de magnitude e variabilidade de cada variável. Os mesmos foram divididos em subconjuntos de treinamento e validação.

Nos métodos de classificação, os modelos são construídos no conjunto de dados de treinamento e o desempenho dos classificadores é obtido nos dados de teste. Esse desempenho foi avaliado a partir dos parâmetros de mérito: exatidão e índice Kappa. Entre todos os classificadores, o OLVQ1 teve melhor desempenho, com exatidão acima de 99% e Kappa maior que 0,99 em todos os conjuntos. Esses resultados superam valores encontrados na literatura para os mesmos conjuntos de dados.

O LVQ apresenta vantagens claras: é simples, intuitivo e fácil de implementar, enquanto ainda produz um desempenho superior. No entanto, em problemas mais complexos, a distância Euclideana pode causar interferências se os dados tiverem muitas dimensões ou forem ruidosos. O pré-processamento e transformação dos dados foram essenciais para bons resultados. É um modelo ainda muito pouco aplicado na quimiometria, mas é tratado como competitivo e promissor.

Para este modelo, o teste de permutação foi realizado no conjunto de identificação de vidro para 72 ensaios diferentes contendo todas as amostras, e nenhum deles apresentou resultados semelhantes ao resultado do conjunto original, indicando que não havia sobreajuste no modelo construído.

Para avaliar a aplicabilidade de modelos de regressão nos conjuntos de qualidade do vinho tinto e branco, foram empregados os parâmetros de mérito MAE, RMSE e R^2, sendo os resultados obtidos insatisfatórios. A variável preditora qualidade é estimada por entrevista e tem dependência fraca com as variáveis físico-químicas. Foram criadas as funções para cada modelo quimiométricos com o intuito de proceder a classificação. A exatidão e índice Kappa foram os parâmetros de mérito usados neste estudo de classificação. Mais uma vez, o OLVQ1 apresentou-se como o método de classificação com melhor desempenho para os dois bancos de dados.

REFERÊNCIAS

ADAMS, M. J. **Chemometrics in Analytical Spectroscopy**. Cambridge: The Royal Society of Chemistry, 1995.

AL-DAOUD, E. A comparison between three neural network models for classification problems. **Journal of Artificial Intelligence**,v. 2, n. 2, p. 56-64, 2009.

ALDAYEL, M. S. **K-Nearest Neighbor classification for glass identification problem**. International Conference on Computer Systems and Industrial Informatics, ICCSII. **Anais**. 2012

ALLOUCHE, O.; TSOAR, A.; KADMON, R. Assessing the accuracy of species distribution models: Prevalence, kappa and the true skill statistic (TSS). **Journal of Applied Ecology**, v. 43, n. 6, p. 1223–1232, 2006.

ALTMAN, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. **American Statistician**, v. 46, n. 3, p. 175–185, 1992.

ATHITSOS, V.; SCLAROFF, S. **Boosting Nearest Neighbor Classi ers for Multiclass Recognition**. Computer Vision and Pattern Recognition. **Anais**...IEEE Computer Society Conference, 2004.

BARATLOO, A. et al. Part 1: Simple Definition and Calculation of Accuracy, Sensitivity and Specificity. **Emergency (Tehran, Iran)**, v. 3, n. 2, p. 48–9, 2015.

BHATTACHARJEE, S. An analysis of classification techniques in data mining for an analysis of classification techniques in data mining for determining quality of wine product: red & white. **Vidyaniketan Journal of Management and Research**, v. 4, n. 1, p. 161–177, 2016.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

BLAKE, C. L.; MERZ, C. **UCI repository of machine learning databases. University of California**. Disponível em: <http://www.ics.uci.edu/~mlearn/ MLRepository.html.>. Acesso em: 07 jun. 2019.

BOX, G. E.; COX, D. R. An analysis of transformations revisited, rebutted. **Journal of the American Statistical Association**, v. 77, n. 377, p. 209–210, 1982.

BREIMAN, L. Bagging predictors: Technical Report n. 421. **Department of Statistics University of California**, n. 2, p. 19, 1994.

BREIMAN, L. Random forests. **Machine Learning**, v. 45, n. 1, p. 5–32, 2001.

BRERETON, R. G. Pattern recognition in chemometrics. **Chemometrics and Intelligent Laboratory Systems**, v. 149, p. 90–96, 2015.

BRO, R. Multivariate calibration: What is in chemometrics for the analytical chemist?

Analytica Chimica Acta, v. 500, n. 1–2, p. 185–194, 2003.

BROWNLEE, J. **Bagging and Random Forest Ensemble Algorithms for Machine Learning**. Disponível em: <https://machinelearningmastery.com/bagging-and-random-forest-ensemble-algorithms-for-machine-learning/>. Acesso em: 18 nov. 2019.

BUSCAGLIA, J. A. Elemental analysis of small glass fragments in forensic science. **Analytica Chimica Acta**, v. 288, n. 1–2, p. 17–24, 1994.

ÇALIS, N. ET AL. Self organizing mixture network in mixture discriminant analysis: An experimental study. **International Journal of Computational and Mathematical Sciences**, v. 5, n. 3, p. 141–144, 2011.

CARONI, C. Outlier detection by example - ProQuest. **Communications in Statistics: Simulation and Computation**, v. 29, n. 1, p. 139–151, 2000.

CASTRO, A. A. M.; PRADO, P. P. L. Algoritmos para Reconhecimento de padrões. **Rev. Ciênc. Exata**, v. 5, n. 5, p. 129–145, 2001.

CHAMBERS, J. M. **Programming with data: A guide to the S language**. New York: Springer-Verlag, 1998.

CHITSAZAN, N.; NADIRI, A. A.; TSAI, F. T. C. **Prediction and structural uncertainty analyses of artificial neural networks using hierarchical Bayesian model averaging**. [S.l.]: Elsevier B.V., 2015. v. 528

CORREIA, P. R. M.; FERREIRA, M. M. C. Reconhecimento de padrões por métodos não supervisionados: Explorando procedimentos quimiométricos para tratamento de dados analíticos. **Química Nova**, v. 30, n. 2, p. 481–487, 2007.

CORTEZ, P. et al. Modeling wine preferences by data mining from physicochemical properties. **Decision Support Systems**, v. 47, n. 4, p. 547–553, 2009.

CRUZ, J. F. M. **Caracterização de gasolinas por espectroscopia FT- Raman.** 2003. 215 f. Tese (Doutorado em Química) - Programa de Pós-Graduação em Química, PUC-Rio, Rio de Janeiro, 2003. Disponível em: <http://www.dbd.puc-rio.br/pergamum/tesesabertas/9716243_03_pretextual.pdf>. Acesso em: 9 nov. 2019.

DASGUPTA, A. et al. A Study and Analysis of a Feature Subset Selection Technique Using Penguin Search Optimization Algorithm. **Data Science**, p. 51–68, 2019.

SENA, M. M. et al. Avaliação do uso de métodos quimiométricos em análise de solos. **Química Nova**, v. 23, n. 4, p. 547–556, 2000.

DEBSKA, B.; GUZOWSKA-ŚWIDER, B. Application of artificial neural network in food classification. **Analytica Chimica Acta**, v. 705, n. 1–2, p. 283–291, 2011.

DEVANGAD, P. et al. Quantification of Mn in glass matrices using laser induced breakdown spectroscopy (LIBS) combined with chemometric approaches. **Analytical Methods**, v. 8, n. 39, p. 7177–7184, 2016.

DOMENICONI, C.; GUNOPULOS, D. An Adaptive Metric Machine for Pattern

Classification. **Advances in Neural Information Processing Systems**, v. 13, p. 458, 2001.

EL-KHATIB, M. J.; ABU-NASSER, B. S.; ABU-NASER, S. S. Classification Using Artificial Neural Network. **International Journal of Academic Pedagogical Research**, v. 3, n. 2, p. 25–31, 2019.

ER, Y.; ATASOY, A. The Classification of White Wine and Red Wine According to Their Physicochemical Qualities. **International Journal of Intelligent Systems and Applications in Engineering**, v. 4, n. Special Issue, p. 23–26, 2016.

FERREIRA, M. M. C. et al. Chemometrics i: Multivariate calibration, a tutorial. **Quimica Nova**, v. 22, n. 5, p. 724–731, 1999.

FLUMIGNAN, D. L. **Caracterização da qualidade e previsão dos parâmetros físico-químicos de gasolinas comerciais brasileiras através da aplicação de métodos quimiométricos em perfis (fingerprintings) espectroscópicos de ressonância magnética nuclear.** 2010. 225 f. Tese (Doutorado em Química) - Programa de Pós-Graduação em Química, Universidade Estadual Paulista, Araraquara, 2010.

FRIEDMAN, J. H. Regularized discriminant analysis. **Journal of the American Statistical Association**, v. 84, n. 405, p. 165–175, 1989.

FRIENDLY, M. Corrgrams: Exploratory displays for correlatigon matrices. **American Statistician**, v. 56, n. 4, p. 316–324, 2002.

GOMES, O. F. M. **Microscopia Co-Localizada: Novas Possibilidades na Caracterização de Minérios.** 2007. 105 f. Tese (Doutorado em Ciência dos Materiais e Metalurgia) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2007.

GUPTA, A. Classification Of Complex UCI Datasets Using Machine Learning And Evolutionary Algorithms. **International Journal of Scientific & Technology Research**, v. 4, n. 5, p. 85–94, 2015.

GUPTA, Y. Selection of important features and predicting wine quality using machine learning techniques. **Procedia Computer Science**, v. 125, p. 305–312, 2018.

HAWKINS, D. M. The Problem of Overfitting. **Journal of Chemical Information and Computer Sciences**, v. 44, n. 1, p. 1–12, 2004.

HU, Y.; LIU, W. Generalized Learning Vector Quantization Algorithm with Feature Data Range Taken into Account. **Journal of Convergence Information Technology**, v. 8, n. 6, p. 1058–1067, 2013.

IHAKA, R.; GENTLEMAN, R. R: A Language for Data Analysis and Graphics. **Journal of Computational and Graphical Statistics**, v. 5, n. 3, p. 299–314, 1996.

IZBICKI, R.; SANTOS, T. dos. **Machine Learning sob a ótica estatística- Uma abordagem preditivista para a estatística com exemplos em R.** 2018. [versão em desenvolvimento] Disponível em: <http://www. rizbicki.ufscar.br/sml.pdf>. Acesso em: 23 out. 2019.

JAIN, A. K.; DUIN, R. P. W.; MAO, J. Statistical pattern recognition: A review. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 22, n. 1, p. 4–37, 2000.

JENSEN, R.; SHEN, Q. Feature selection for aiding glass forensic evidence analysis. **Intelligent Data Analysis**, v. 13, n. 5, p. 703–723, 2009.

JEUNE, W. et al. Multinomial logistic regression and random forest classifiers in digital mapping of soil classes in western Haiti. **Revista Brasileira de Ciência do Solo**, v. 42, p. 1–20, 2018.

KIL, D. H.; SHIN, F. B. **Pattern Recognition and Prediction with Applications to Signal Processing**. Woodbury: American Institute of Physics, 1996.

KOHONEN, T. The self-organizing map. **Proc. IEEE**, v. 78, p. 1464–1480, [s.d.].

KOHONEN, T. Improved versions of learning vector quantization. Proceedings of the International Joint Conference on Neural Networks. **Anais**...San Diego, CA: 1990

KOHONEN, T. New developments of learning vector quantization and the self-organizing map. Symposium on Neural Networks: Alliances and Perspectives in Senri. **Anais**...Osaka, Japan: 1992

KUHN, M. Building Predictive Models in R Using the caret Package. **Journal of Statistical Software**, v. 28, n. 5, p. 1–26, 2008.

KUHN, M.; JOHNSON, K. **Applied predictive modeling**. New York: Springer, 2013.

KUMAR, R.; SHARMA, V. Chemometrics in forensic science. **TrAC - Trends in Analytical Chemistry**, v. 105, p. 191–201, 2018.

LANTZ, B. **Machine learning with R**. 2 nd ed. Birmingham: Packt, 2015.

LAVINE, B. K. Critical Reviews in Analytical Chemistry Pattern Recognition Pattern Recognition. **Critical Reviews in Analytical Chemistry**, v. 36, n. 36, p. 3–4, 2006.

LEE, L. C. et al. Forensic differentiation of paper by ATR-FTIR spectroscopy technique and partial least-squares-discriminant analysis (PLS-DA). AIP Conference Proceedings. **Anais**...2016

LEISCH, F.; DIMITRIADOU, E. **Package "mlbench"R package version**. [S.l: s.n.]. Disponível em: <https://scholar.google.com/scholar?q=Leisch+and+Dimitriadou,+2010&btnG=&hl=en&as_sdt=0,36#0>. Acesso em: 18 nov. 2019.

LORENA, A. C.; CARVALHO, A. C. P. L. F. DE. Uma Introdução às Support Vector Machines. **RITA**, v. 16, n. 2, p. 43–67, 2007.

LUXBURG, U. VON; SCHÖLKOPF, B. Statistical Learning Theory: Models, Concepts, and Results. **Handbook of the History of Logic**, v. 10, p. 651–706, 2011.

MCHUGH, M. L. Interrater reliability: the kappa statistic. **Biochemic Med**, v. 22, n. 3,

p. 276–282, 2012.

MEHMOOD, T.; AHMED, B. The diversity in the applications of partial least squares: An overview. **Journal of Chemometrics**, v. 30, n. 1, p. 4–17, 2016.

MITCHELL, T. **Machine Learning**. 2. ed. Maidenhead: McGraw-Hill, 1997.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. . **Foundations of Machine Learning**. 2 nd ed. [S.l.]: MIT press, 2018.

MORAIS, E. C. Reconhecimento De Padrões E Redes Neurais Artificiais Em Predição De Estruturas Secundárias De Proteínas. **Coppe/Ufrj**, p. 135, 2010.

NABABAN, A. A.; SITOMPUL, O. S.; TULUS. Attribute Weighting Based K-Nearest Neighbor Using Gain Ratio. **Journal of Physics**: Conference Series, v. 1007, p. 1–6, 2018.

NAPIER, G.; NEOCLEOUS, T.; NOBILE, A. A composite Bayesian hierarchical model of compositional data with zeros. **Journal of Chemometrics**, v. 29, n. 2, p. 96–108, 2015.

NODO C5.O. Disponível em: <https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/clementine/c50node_general.html>. Acesso em: 19 nov. 2019.

OMATU, S.; YANO, M.; IKEDA, Y. Smell classification of wines by the learning vector quantization method. Proceedings of the ACM Symposium on Applied Computing. **Anais**...2015

PARREIRA, T. F. **Utilização de métodos quimiométricos em dados de natureza multivariada.** Campinas State University, 2003.

PEARSON, K. On lines and planes of closest fit to systems of points in space. **Philosophical Magazine**, v. 2, n. 6, p. 559–572, 1901.

RAGHU, P. P.; POONGODI, R.; YEGNANARAYANA, B. Combined Neural Network Approach for Texture Classification. **Neural Networks**, v. 8, n. 6, p. 975–987, 1995.

RAUSCH, J. R.; KELLEY, K. A comparison of linear and mixture models for discriminant analysis under nonnormality. **Behavior Research Methods**, v. 41, n. 1, p. 85–98, 2009.

RIBEIRO, J. et al. Wine Vinification prediction using Data Mining tools. **Computing and Computational Intelligence, Proceedings**, p. 78, 2009.

RIDGEWAY, G. Generalized Boosted Models: A guide to the gbm package. **Compute**, v. 1, n. 4, p. 1–12, 2007.

RIMAL, R.; ALMØY, T.; SÆBØ, S. Comparison of multi-response prediction methods. **Chemometrics and Intelligent Laboratory Systems**, v. 190, p. 10–21, 2019.

SALIM, A. **Aplicação de técnicas de reconhecimento de padrões usando os**

descritores estruturais de proteínas da base de dados do software STING para discriminação do sítio catalítico de enzimas. 2015. 238 f. Dissertação (Mestrado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas, 2015. Disponível em: <http://repositorio.unicamp.br/handle/REPOSIP/259089 >. Acesso em : 02 out. 2019.

SCHNEIDER, M. et al. Métodos de Análise Exploratória Aplicadas ao Ensino de Espectroscopia na Região do Infravermelho. **Revista Virtual de Química**, v. 10, n. 2, p. 229–243, 2018.

SCHÖLKOPF, B.; SMOLA, A. J. **Learning with Kernels– Support Vector Machines, Regularization, Optimization, and Beyond**. MIT Press, 2002.

SERNEELS, S.; DE NOLF, E.; VAN ESPEN, P. J. Spatial sign preprocessing: A simple way to impart moderate robustness to multivariate estimators. **Journal of Chemical Information and Modeling**, v. 46, n. 3, p. 1402–1409, 2006.

STACKLIES, W.; REDESTIG, H. Handling of data containing outliers. **CAS-MPG Partner Institute for Computational Biology (PICB)**, p. 1–3, 2019.

R CORE TEAM. **A language and environment for statistical computing.** R Foundation for Statistical Computing: Vienna, 2019. Disponível em: <https://www.R-project.org/>. Acesso em: 08 ago. 2019.

THARWAT, A. et al. Linear discriminant analysis: a detailed tutorial. **AI Communications**, v. 30, n. 2, p. 169–190, 2017.

TODOROV, V.; FILZMOSER, P. An Object-Oriented Framework for Robust Multivariate Analysis. **Journal of Statistical Software**, v. 32, n. 3, p. 1–47, 2009.

TSILIKI, G. et al. RRegrs: An R package for computer-aided model selection with multiple regression models. **Journal of Cheminformatics**, v. 7, n. 1, p. 1–16, 2015.

UNIYAL, X.; BARTHWAL, P. Wine Quality Evaluation Using Machine Learning Algorithms. **Asia-Pacific Journal of Convergent Research Interchange**, v. 3, n. 4, p. 1–9, 2017.

VAPNIK, V.; GOLOWICH, S. E.; SMOLA, A. Support vector method for function approximation, regression estimation, and signal processing. **Advances in Neural Information Processing Systems**, p. 281–287, 1997.

VENABLES, W.; RIPLEY, B. **Modern Applied Statistics with S.** 4.ed. New York: Springer, 2002.

VINING, G. G.; MONTGOMERY, D. C.; PECK, E. A. I*ntroduction to Linear Regression Analysis*. 3. ed. New York: Wiley, 2001.

VURAL, M. S.; GÖK, M. Criminal prediction using Naive Bayes theory. **Neural Computing and Applications**, v. 28, n. 9, p. 2581–2592, 2017.

WITTEN, I. H.; FRANK, E.; HALL, M. A. **Data Mining**: Practical Machine Learning Tools and Techniques. 3.ed. San Francisco: Morgan Kaufmann, 2011.

WOLD, S.; SJÖSTRÖM, M.; ERIKSSON, L. PLS-regression: A basic tool of chemometrics. **Chemometrics and Intelligent Laboratory Systems**, v. 58, n. 2, p. 109–130, 2001.

WU, W. et al. Comparison of regularized discriminant analysis, linear discriminant analysis and quadratic discriminant analysis, applied to NIR data. **Analytica Chimica Acta**, v. 329, n. 3, p. 257–265, 1996.

YANG, W.; XU, H. A unified framework for outlier-robust PCA-like algorithms. 32nd International Conference on Machine Learning, ICML 2015. **Anais**...2015

YOUSSEF, A. M. et al. Erratum to: Landslide susceptibility mapping using random forest, boosted regression tree, classification and regression tree, and general linear models and comparison of their performance at Wadi Tayyah Basin, Asir Region, Saudi Arabia (Landslides, 10.10.) **Landslides**, v. 13, n. 5, p. 1315–1318, 2016.

ZADORA, G. Glass analysis for forensic purposes—a comparison of classification methods. **Journal of Chemometrics**, v. 21, p. 174–186, 2007.

ZEIDAT, N.; WANG, S.; EICK, C. F. **Dataset editing techniques** : a comparative study. Huston (USA): Department of Computer Science, University of Houston, 2005. Disponível em: <http://www.cse.unr.edu/bebis/cs679/PaperPresentations/ZWE05.pdf.>. Acesso em: 12 out. 2019.

ANEXO A - Roteiro do conjunto de dados de identificação de vidros aplicado no software R.

```
library(mlbench)
data(Glass)
dim(Glass)
str(Glass)
sapply(Glass, class)
summary(Glass)

library(psych)
describeBy(Glass[-10])

levels(Glass$Type)
percentage <- round(prop.table(table(Glass$Type)) * 100, 1)
cbind(freq=table(Glass$Type), percentage=percentage)
par(mfrow = c(1, 1))
corDegrade   <-   colorRampPalette(c("brown1","deepskyblue4",   "paleturquoise",
"gold", "lightsteelblue4", "paleturquoise"))
distribuiton<- barplot(percentage, horiz = F, col=corDegrade(10), xlab= "Glass
Type", ylab="Percentage (%)", las=2 )
text(x=distribuiton, y=percentage, labels=percentage, pos=3, xpd=NA)

x <- Glass [,1:9]
y <- Glass [,10]

#Histogram
par(mfrow = c(3, 3))
for (i in 1:ncol(x)) {
  hist(x[ ,i], xlab = names(x[i]), main = " ",
       col="cornsilk3", border = "ghostwhite" )
}

#Box plot
par(mfrow=c(3,3))
for (i in 1:ncol(x))
{
  boxplot(x[ ,i], ylab = names(x[i]), horizontal=F,main = paste(names(x[i]),
"Boxplot"),
          col="ghostwhite")
}

#Correlation Matrix
library(corrgram)
corrgram(x, lower.panel = panel.pts, upper.panel= panel.conf, diag.panel =
panel.density)

#Pre-processing
library(caret)
set.seed(1)
glass.trans <- preProcess(Glass[,-10], method=c("center", "scale",
                                                "BoxCox", "spatialSign"))
print(glass.trans)
transf <- predict(glass.trans,Glass[,-10])
glass.proc <- as.data.frame(transf)
par(mfrow=c(1,1))
boxplot(glass.proc, horizontal=F )
summary(glass.proc)

#ROBPCA
library(rrcov)
set.seed(1)
glass.HubPCA<- PcaHubert(glass.proc)
summary(glass.HubPCA)
par(mfrow=c(1,1))
plot(glass.HubPCA, main= " ")
scorePlot(glass.HubPCA, col="red", main= " ")
```

```
biplot(glass.HubPCA, main= "Loadings")
library(rgl)
plot3d(glass.HubPCA$scores[,1:3], xlab = "PC1  40.79%", ylab = " PC2  28.34%",
zlab = "PC3  17.53%",
       type = c("p"),
       add = FALSE, col = Glass$Type)

#Add the column type to the glass.proc matrix
library(stringr)
glass.new <- glass.proc%>%
  cbind(Type=Glass[,10])%>%
  na.omit()
# Create column with the predictor variable without outliers
glass.class<- glass.new[,10]

library(ggfortify)
PCAscores <- glass.HubPCA$scores
autoplot(prcomp(PCAscores), data = glass.new, colour = 'Type',
         frame = T, frame.type = 'norm') +
  geom_vline(xintercept=c(-0,0), linetype="dashed", size=0.25) +
  geom_hline(yintercept=c(-0,0), linetype="dashed", size=0.25)

#Train and test
set.seed(1)
inTrain <- createDataPartition(y = glass.class  , p = 0.75, list = F)
train.glass <- glass.new[inTrain,]
test.glass <- glass.new[-inTrain,]

nrow(train.glass)
levels(train.glass$Type)
percentage <- round(prop.table(table(train.glass$Type)) * 100, 1)
cbind(freq=table((train.glass$Type)), percentage=percentage)
nrow(test.glass)
levels(test.glass$Type)
percentage <- round(prop.table(table(test.glass$Type)) * 100, 1)
cbind(freq=table((test.glass$Type)), percentage=percentage)

#Chemometric Models
library(caret)

# ANN: Artificial Neural Networks
set.seed(1)
nnet = train( Type~., data=train.glass, method = "nnet",
              metric="Accuracy",
              preProc=c("center", "scale"),
              trControl = trainControl(method="cv", number=10))
nnetPredict<- predict(nnet, test.glass)
confusionMatrix(test.glass$Type, nnetPredict)

# Bagging
set.seed(1)
treebag<- train(Type~., data=train.glass, method="treebag",
                metric="Accuracy",
                preProc=c("center", "scale"),
                trControl=trainControl(method="cv", number=10))
treebagPredict<- predict(treebag, test.glass)
confusionMatrix(test.glass$Type, treebagPredict)

#C5.0
set.seed(1)
c5.0 <- train(Type~., data=train.glass, method="C5.0",
              metric="Accuracy",
              preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=10))
c5.0Predict<- predict(c5.0, test.glass)
confusionMatrix(test.glass$Type, c5.0Predict)

#CART
```

```
set.seed(1)
cart <- train(Type~., data=train.glass, method="rpart",
              metric="Accuracy",
              preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=10))
cartPredict<- predict(cart, test.glass)
confusionMatrix(test.glass$Type, cartPredict)

#GBM: Generalized Boosted Regression-
set.seed(1)
gbm <- train(Type ~ .,train.glass, method='gbm',
             verbose=FALSE,
             preProcess = c("center","scale"),
             metric = "Accuracy",
             trControl=trainControl(method="cv", number=10))
gbmPredict <- predict(gbm, test.glass)
confusionMatrix(test.glass$Type, gbmPredict)

# K-Nearest Neighbours
set.seed(1)
knn<- train(Type~., train.glass, method = "knn",
            preProcess = c("center","scale"),
            tuneLength = 10,
            trControl = trainControl(method = "cv", number = 10))
knnPredict <- predict(knn, test.glass)
confusionMatrix(knnPredict, test.glass$Type)

#LVQ:Learning Vector Quantization
library(class)
set.seed(1)
train_label <- factor(train.glass$Type)
test_label <- test.glass$Type

codeBook<- lvqinit(train.glass, train_label, size = 100)
buildCodeBook<- olvq1(train.glass, train_label, codeBook)
lvq.predict<- lvqtest(buildCodeBook, test.glass)
confusionMatrix(test_label, lvq.predict)

# LDA
set.seed(1)
lda<-  train(Type~., data=train.glass, method="lda",
             metric="Accuracy",
             preProc=c("center", "scale"),
             trControl=trainControl(method="cv", number=10))

ldaPredict<- predict(lda, test.glass)
confusionMatrix(test.glass$Type, ldaPredict)

#MDA: Mixture Discriminant Analysis
set.seed(1)
mda = train( Type~., data=train.glass, method = "mda",
             metric="Accuracy",
             preProc=c("center", "scale"),
             trControl = trainControl(method="cv", number=10))
mdaPredict<- predict(mda, test.glass)
confusionMatrix(test.glass$Type, mdaPredict)

#Naive Bayes
set.seed(1)
nb <- train(Type~., data=train.glass, method="nb",
            metric="Accuracy",
            preProc=c("center", "scale"),
            trControl=trainControl(method="cv", number=10))
nbPredict<- predict(nb, test.glass)
confusionMatrix(test.glass$Type, nbPredict)

#PLS: Partial Least Squares Discriminant Analysis using Cross-Validation and
Bootstrap
```

```
set.seed(1)
pls= train( Type~., data=train.glass, method = "pls",
            metric="Accuracy",
            preProc=c("center", "scale"),
            trControl = trainControl(method="cv", number=10))
plsPredict<- predict(pls, test.glass)
confusionMatrix(test.glass$Type, plsPredict)

# Random Forest
set.seed(1)
rf <- train(Type~., data = train.glass,method = 'rf',
            preProcess = c("center","scale"),
            metric = "Accuracy",
            ntree = 300,
            tuneGrid = expand.grid(.mtry = 1:9),
            trControl = trainControl(method = "cv", number = 10))
rfPredict <- predict(rf, test.glass)
confusionMatrix(rfPredict, test.glass$Type)

# RDA: Regularized Discriminant Analysis
set.seed(1)
rda.grid = train( Type~., data=train.glass, method = "rda",
                  metric="Accuracy",
                  preProc=c("center", "scale"),
                  trControl = trainControl(method="cv", number=10))
rda.gridPredict<- predict(rda.grid, test.glass)
confusionMatrix(test.glass$Type, rda.gridPredict)

#SVM Radial
set.seed(1)
svm <- train(Type~., train.glass, method = "svmRadial",
             preProcess = c("center","scale"),
             metric = "Accuracy",
             tuneLength = 5,
             tuneGrid = expand.grid(.sigma = 0.5, .C = 10),
             trControl = trainControl(method = "cv", number = 10))
svmPredict<- predict(svm, test.glass)
confusionMatrix(test.glass$Type, svmPredict)

#Overfitting analysis
# Creation of classes vector
vector <- c("1","2","3","5","6","7")
class(vector)
# Random Samples and Permutations
set.seed(1)
sample(x = vector, replace = FALSE)
#Setting the number of lines
n_lines <- 0.1*factorial(6)
n_lines
# Creating the zero matrix
data1 <- matrix(0, nrow = n_lines, ncol = length(vector))

#Random classes 72 times
for (i in 1:n_lines) {
  set.seed(i)
  data1[i,] <- sample(x = vector, replace = FALSE)
}
data1
class(data1)
data1 <- as.data.frame(data1)
names(data1)
names(data1) <- c("1","2","3","5","6","7")
class(data1)
View(data1)

#Apply to dataset
class1<- glass.new[,10]
class1
```

```
for (i in 1:(nrow(data1))) {
  perm<-c(rep(as.character(data1[i,1]), 70),
          rep(as.character(data1[i,2]), 76),
          rep(as.character(data1[i,3]), 17),
          rep(as.character(data1[i,5]), 13),
          rep(as.character(data1[i,6], 9)),
          rep(as.character(data1[i,7]), 29))
  class1<- cbind(class1, perm)
}
colnames(class1) <- c(1:73)
class(class1)
View(class1)

#Sample 2
library(stringr)
amostra.2 <- glass.proc%>%
  cbind(classe[,2])%>%
  na.omit()
class.2<- amostra.2[,10]
set.seed(1)
inTrain.2 <- createDataPartition(y = class.2, p = 0.75, list = F)
train.2<- amostra.2[inTrain.2,]
test.2 <- amostra.2[-inTrain.2,]
set.seed(1)
lvq.predict.2<- lvqtest(buildCodeBook, test.2 )
conf.1<-confusionMatrix(test_label, lvq.predict.2)

############################
#Sample 3
library(stringr)
amostra.2 <- glass.proc%>%
  cbind(classe[,3])%>%
  na.omit()
class.2<- amostra.2[,10]
set.seed(1)
inTrain.2 <- createDataPartition(y = class.2, p = 0.75, list = F)
train.2<- amostra.2[inTrain.2,]
test.2 <- amostra.2[-inTrain.2,]
set.seed(1)
lvq.predict.2<- lvqtest(buildCodeBook, test.2 )
conf.2<- confusionMatrix(test_label, lvq.predict.2)

#Sample 4
library(stringr)
amostra.3 <- glass.proc%>%
  cbind(classe[,4])%>%
  na.omit()
class.3<- amostra.3[,10]
set.seed(1)
inTrain.3 <- createDataPartition(y = class.3, p = 0.75, list = F)
train.3<- amostra.3[inTrain.3,]
test.3 <- amostra.3[-inTrain.3,]
set.seed(1)
lvq.predict.3<- lvqtest(buildCodeBook, test.3 )
conf.3<-confusionMatrix(test_label, lvq.predict.3)

# Sample 5
library(stringr)
amostra.4 <- glass.proc%>%
  cbind(classe[,5])%>%
  na.omit()
class.4<- amostra.4[,10]
set.seed(1)
inTrain.4<- createDataPartition(y = class.4, p = 0.75, list = F)
train.4<- amostra.4[inTrain.4,]
test.4 <- amostra.4[-inTrain.4,]
set.seed(1)
```

```
lvq.predict.4<- lvqtest(buildCodeBook, test.4 )
conf.4<-confusionMatrix(test_label, lvq.predict.4)

# Sample 6
library(stringr)
amostra.5 <- glass.proc%>%
  cbind(classe[,6])%>%
  na.omit()
class.5<- amostra.5[,10]
set.seed(1)
inTrain.5 <- createDataPartition(y = class.5, p = 0.75, list = F)
train.5<- amostra.5[inTrain.5,]
test.5 <- amostra.5[-inTrain.5,]
set.seed(1)
lvq.predict.5<- lvqtest(buildCodeBook, test.2 )
conf.5<-confusionMatrix(test_label, lvq.predict.5)

# Sample 7
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,7])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.6<-confusionMatrix(test_label, lvq.predict)

# Sample 8
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,8])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.7<-confusionMatrix(test_label, lvq.predict)

# Sample 9
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,9])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.8<-confusionMatrix(test_label, lvq.predict)

# Sample 10
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,10])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
```

```
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.9<-confusionMatrix(test_label, lvq.predict)

# Sample 11
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,11])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.10<-confusionMatrix(test_label, lvq.predict)

# Sample 12
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,12])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.11<-confusionMatrix(test_label, lvq.predict)

# Sample 13
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,13])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.12<-confusionMatrix(test_label, lvq.predict)

# Sample 14
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,14])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.13<-confusionMatrix(test_label, lvq.predict)

# Sample 15
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,15])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
```

```
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.14<-confusionMatrix(test_label, lvq.predict)

# Sample 16
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,16])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.15<-confusionMatrix(test_label, lvq.predict)

# Sample 17
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,17])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.16<-confusionMatrix(test_label, lvq.predict)

# Sample 18
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,18])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.17<-confusionMatrix(test_label, lvq.predict)

# Sample 19
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,19])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.18<-confusionMatrix(test_label, lvq.predict)

# Sample 20
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,20])%>%
  na.omit()
```

```
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.19<-confusionMatrix(test_label, lvq.predict)

# Sample 21
library(stringr)
amostra<- glass.proc%>%
  cbind(classe[,21])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.20<-confusionMatrix(test_label, lvq.predict)

# Sample 22
amostra<- glass.proc%>%
  cbind(classe[,22])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.21<-confusionMatrix(test_label, lvq.predict)

# Sample 23
amostra<- glass.proc%>%
  cbind(classe[,23])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.22<-confusionMatrix(test_label, lvq.predict)

# Sample 24
amostra<- glass.proc%>%
  cbind(classe[,24])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.23<-confusionMatrix(test_label, lvq.predict)

# Sample 25
amostra<- glass.proc%>%
  cbind(classe[,25])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
```

```
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.24<-confusionMatrix(test_label, lvq.predict)

# Sample 26
amostra<- glass.proc%>%
  cbind(classe[,26])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.25<-confusionMatrix(test_label, lvq.predict)

# Sample 27
amostra<- glass.proc%>%
  cbind(classe[,27])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.26<-confusionMatrix(test_label, lvq.predict)

# Sample 28
amostra<- glass.proc%>%
  cbind(classe[,28])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.27<-confusionMatrix(test_label, lvq.predict)

# Sample 29
amostra<- glass.proc%>%
  cbind(classe[,29])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.28<-confusionMatrix(test_label, lvq.predict)

# Sample 30
amostra<- glass.proc%>%
  cbind(classe[,30])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
```

```
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.29<-confusionMatrix(test_label, lvq.predict)

# Sample 31
amostra<- glass.proc%>%
  cbind(classe[,31])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.30<-confusionMatrix(test_label, lvq.predict)

# Sample 32
amostra<- glass.proc%>%
  cbind(classe[,32])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.31<-confusionMatrix(test_label, lvq.predict)

# Sample 33
amostra<- glass.proc%>%
  cbind(classe[,33])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.32<-confusionMatrix(test_label, lvq.predict)

# Sample 34
amostra<- glass.proc%>%
  cbind(classe[,34])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.33<-confusionMatrix(test_label, lvq.predict)

# Sample 35
amostra<- glass.proc%>%
  cbind(classe[,35])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.34<-confusionMatrix(test_label, lvq.predict)
```

```
# Sample 36
amostra<- glass.proc%>%
  cbind(classe[,36])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.35<-confusionMatrix(test_label, lvq.predict)

# Sample 37
amostra<- glass.proc%>%
  cbind(classe[,37])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.36<-confusionMatrix(test_label, lvq.predict)

# Sample 38
amostra<- glass.proc%>%
  cbind(classe[,38])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.37<-confusionMatrix(test_label, lvq.predict)

# Sample 39
amostra<- glass.proc%>%
  cbind(classe[,39])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.38<-confusionMatrix(test_label, lvq.predict)

# Sample 40
amostra<- glass.proc%>%
  cbind(classe[,40])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.39<-confusionMatrix(test_label, lvq.predict)

# Sample 41
amostra<- glass.proc%>%
```

```
  cbind(classe[,41])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.40<-confusionMatrix(test_label, lvq.predict)

# Sample 42
amostra<- glass.proc%>%
  cbind(classe[,42])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.41<-confusionMatrix(test_label, lvq.predict)

# Sample 43
amostra<- glass.proc%>%
  cbind(classe[,43])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.42<-confusionMatrix(test_label, lvq.predict)

# Sample 44
amostra<- glass.proc%>%
  cbind(classe[,44])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.43<-confusionMatrix(test_label, lvq.predict)

# Sample 45
amostra<- glass.proc%>%
  cbind(classe[,45])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.44<-confusionMatrix(test_label, lvq.predict)

# Sample 46
amostra<- glass.proc%>%
  cbind(classe[,46])%>%
  na.omit()
class<- amostra[,10]
```

```
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.45<-confusionMatrix(test_label, lvq.predict)

# Sample 47
amostra<- glass.proc%>%
  cbind(classe[,47])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.46<-confusionMatrix(test_label, lvq.predict)

# Sample 48
amostra<- glass.proc%>%
  cbind(classe[,48])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.47<-confusionMatrix(test_label, lvq.predict)

# Sample 49
amostra<- glass.proc%>%
  cbind(classe[,49])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.48<-confusionMatrix(test_label, lvq.predict)

# Sample 50
amostra<- glass.proc%>%
  cbind(classe[,50])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.49<-confusionMatrix(test_label, lvq.predict)

# Sample 51
amostra<- glass.proc%>%
  cbind(classe[,51])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
```

```
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.50<-confusionMatrix(test_label, lvq.predict)

# Sample 52
amostra<- glass.proc%>%
  cbind(classe[,52])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.51<-confusionMatrix(test_label, lvq.predict)

# Sample 53
amostra<- glass.proc%>%
  cbind(classe[,53])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.52<-confusionMatrix(test_label, lvq.predict)

# Sample 54
amostra<- glass.proc%>%
  cbind(classe[,54])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.53<-confusionMatrix(test_label, lvq.predict)

# Sample 55
amostra<- glass.proc%>%
  cbind(classe[,55])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.54<-confusionMatrix(test_label, lvq.predict)

# Sample 56
amostra<- glass.proc%>%
  cbind(classe[,56])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
```

```
conf.55<-confusionMatrix(test_label, lvq.predict)

# Sample 57
amostra<- glass.proc%>%
  cbind(classe[,57])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.56<-confusionMatrix(test_label, lvq.predict)

# Sample 58
amostra<- glass.proc%>%
  cbind(classe[,58])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.57<-confusionMatrix(test_label, lvq.predict)

# Sample 59
amostra<- glass.proc%>%
  cbind(classe[,59])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.58<-confusionMatrix(test_label, lvq.predict)

# Sample 60
amostra<- glass.proc%>%
  cbind(classe[,60])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.59<-confusionMatrix(test_label, lvq.predict)

# Sample 61
amostra<- glass.proc%>%
  cbind(classe[,61])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.60<-confusionMatrix(test_label, lvq.predict)

# Sample 62
```

```
amostra<- glass.proc%>%
  cbind(classe[,62])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.61<-confusionMatrix(test_label, lvq.predict)

# Sample 63
amostra<- glass.proc%>%
  cbind(classe[,63])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.62<-confusionMatrix(test_label, lvq.predict)

# Sample 64
amostra<- glass.proc%>%
  cbind(classe[,64])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.63<-confusionMatrix(test_label, lvq.predict)

# Sample 65
amostra<- glass.proc%>%
  cbind(classe[,65])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.64<-confusionMatrix(test_label, lvq.predict)

# Sample 66
amostra<- glass.proc%>%
  cbind(classe[,66])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.65<-confusionMatrix(test_label, lvq.predict)

# Sample 67
amostra<- glass.proc%>%
  cbind(classe[,67])%>%
  na.omit()
```

```
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.66<-confusionMatrix(test_label, lvq.predict)

# Sample 68
amostra<- glass.proc%>%
  cbind(classe[,68])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.67<-confusionMatrix(test_label, lvq.predict)

# Sample 69
amostra<- glass.proc%>%
  cbind(classe[,69])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.68<-confusionMatrix(test_label, lvq.predict)

# Sample 70
amostra<- glass.proc%>%
  cbind(classe[,70])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.69<-confusionMatrix(test_label, lvq.predict)

# Sample 71
amostra<- glass.proc%>%
  cbind(classe[,71])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.70<-confusionMatrix(test_label, lvq.predict)

# Sample 72
amostra<- glass.proc%>%
  cbind(classe[,72])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
```

```
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.71<-confusionMatrix(test_label, lvq.predict)

# Sample 73
amostra<- glass.proc%>%
  cbind(classe[,73])%>%
  na.omit()
class<- amostra[,10]
set.seed(1)
inTrain.amostra <- createDataPartition(y = class, p = 0.75, list = F)
train<- amostra[inTrain.amostra,]
test<- amostra[-inTrain.amostra,]
set.seed(1)
lvq.predict<- lvqtest(buildCodeBook, test )
conf.72<-confusionMatrix(test_label, lvq.predict)

##########
# Results from the Accuracy
Accuracy<-
c(conf.1$overall[1],conf.2$overall[1],conf.3$overall[1],conf.4$overall[1],conf
.5$overall[1],conf.6$overall[1],conf.7$overall[1],conf.8$overall[1],conf.9$ove
rall[1],conf.10$overall[1],

conf.11$overall[1],conf.12$overall[1],conf.13$overall[1],conf.14$overall[1],co
nf.15$overall[1],conf.16$overall[1],conf.17$overall[1],conf.18$overall[1],conf
.19$overall[1],conf.20$overall[1],

conf.21$overall[1],conf.22$overall[1],conf.23$overall[1],conf.24$overall[1],co
nf.25$overall[1],conf.26$overall[1],conf.27$overall[1],conf.28$overall[1],conf
.29$overall[1],conf.30$overall[1],

conf.31$overall[1],conf.32$overall[1],conf.33$overall[1],conf.34$overall[1],co
nf.35$overall[1],conf.36$overall[1],conf.37$overall[1],conf.38$overall[1],conf
.39$overall[1],conf.40$overall[1],

conf.41$overall[1],conf.42$overall[1],conf.43$overall[1],conf.44$overall[1],co
nf.45$overall[1],conf.46$overall[1],conf.47$overall[1],conf.48$overall[1],conf
.49$overall[1],conf.50$overall[1],

conf.51$overall[1],conf.52$overall[1],conf.53$overall[1],conf.54$overall[1],co
nf.55$overall[1],conf.56$overall[1],conf.57$overall[1],conf.58$overall[1],conf
.59$overall[1],conf.60$overall[1],

conf.61$overall[1],conf.62$overall[1],conf.63$overall[1],conf.64$overall[1],co
nf.65$overall[1],conf.66$overall[1],conf.67$overall[1],conf.68$overall[1],conf
.69$overall[1],conf.70$overall[1],
            conf.71$overall[1],conf.72$overall[1])
Accuracy


# Results from the Kappa Index
Kappa<-
c(conf.1$overall[2],conf.2$overall[2],conf.3$overall[2],conf.4$overall[2],conf
.5$overall[2],conf.6$overall[2],conf.7$overall[2],conf.8$overall[2],conf.9$ove
rall[2],conf.10$overall[2],

conf.11$overall[2],conf.12$overall[2],conf.13$overall[2],conf.14$overall[2],co
nf.15$overall[2],conf.16$overall[2],conf.17$overall[2],conf.18$overall[2],conf
.19$overall[2],conf.20$overall[2],

conf.21$overall[2],conf.22$overall[2],conf.23$overall[2],conf.24$overall[2],co
nf.25$overall[2],conf.26$overall[2],conf.27$overall[2],conf.28$overall[2],conf
.29$overall[2],conf.30$overall[2],

conf.31$overall[2],conf.32$overall[2],conf.33$overall[2],conf.34$overall[2],co
```

```
nf.35$overall[2],conf.36$overall[2],conf.37$overall[2],conf.38$overall[2],conf
.39$overall[2],conf.40$overall[2],

conf.41$overall[2],conf.42$overall[2],conf.43$overall[2],conf.44$overall[2],co
nf.45$overall[2],conf.46$overall[2],conf.47$overall[2],conf.48$overall[2],conf
.49$overall[2],conf.50$overall[2],

conf.51$overall[2],conf.52$overall[2],conf.53$overall[2],conf.54$overall[2],co
nf.55$overall[2],conf.56$overall[2],conf.57$overall[2],conf.58$overall[2],conf
.59$overall[2],conf.60$overall[2],

conf.61$overall[2],conf.62$overall[2],conf.63$overall[2],conf.64$overall[2],co
nf.65$overall[2],conf.66$overall[2],conf.67$overall[2],conf.68$overall[2],conf
.69$overall[2],conf.70$overall[2],
         conf.71$overall[2],conf.72$overall[2])
Kappa

# Summary
summary(Accuracy)
summary(Kappa)

# Boxplot
library(graphics)
par(mfrow=c(1,2))
boxplot(Accuracy,xlab="Accuracy", ylab="Values")
boxplot(Kappa,xlab="Kappa ", ylab="Values")

# Test of normality
shapiro.test(Accuracy); shapiro.test(Kappa)

# Wilcoxon signed rank test
res1 <- wilcox.test(Accuracy, mu = 1)
res2 <-wilcox.test(Kappa,mu=1)
# Printing the results
res1;res2
# The Accuracy and Kappa index do not equal to 1.

# Histogram
# Histogram
par(mfrow=c(1,1))
hist(Accuracy,  border="black",prob  =  TRUE,  xlab  =  "Accuracy",  ylab  =
"Density")
lines(density(Accuracy),lwd = 2, col = "red")
hist(Kappa, border="black",prob = TRUE, xlab = "Kappa", ylab = "Density")
lines(density(Kappa),lwd = 2, col = "red")
```

ANEXO B - Roteiro do conjunto de dados de qualidade do vinho branco aplicado no software r.

```
white.url     <-     "https://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-white.csv"
white.raw <- read.csv(white.url, header = TRUE, sep = ";")
white <- white.raw
dim(white)
str(white)
white$quality <- as.factor(white$quality)
sapply(white, class)
white <- white[!duplicated(white), ]
dim(white)
summary(white)
library(psych)
describeBy(white[-12])

levels(white$quality)
percentage <- round(prop.table(table(white$quality)) * 100, 1)
cbind(freq=table(white$quality), percentage=percentage)
par(mfrow = c(1, 1))
corDegrade   <-   colorRampPalette(c("orangered","lightblue4","pink2",
"palegreen4", "gold","purple4","orchid4"))
distribuiton<- barplot(percentage, horiz = F, col=corDegrade(5), xlab=
"Quality of White Wine", ylab="Percentage (%)", las=2 )
text(x=distribuiton, y=percentage, labels=percentage, pos=3, xpd=NA)

a <- white [,1:11]
b<- white [,12]

#Histogram
par(mfrow = c(3, 3))
for (i in 1:ncol(a)) {
  hist(a[ ,i], xlab = names(a[i]), main = " ",
       col="cornsilk3", border = "cornsilk4")
}

#Box plot
par(mfrow=c(3,2))
for (i in 1:ncol(a))
{
  boxplot(a[   ,i],   ylab   =   names(a[i]),   horizontal=F,main   =
paste(names(a[i]),

"Boxplot"), col="cornsilk3")
}

#Correlation matrix
library(corrgram)
corrgram(white,  lower.panel  =  panel.pts,  upper.panel=  panel.conf,
diag.panel = panel.density)

#Pre-processing
library(caret)
set.seed(1)
white.trans  <-  preProcess(a,  method=c("center",  "scale",  "BoxCox",
"spatialSign"))
print(white.trans)
transf <- predict(white.trans,a)
white.proc <- as.data.frame(transf)
colnames(white.proc)
```

```
par(mfrow=c(1,1))
boxplot(white.proc, horizontal=F )
summary(white.proc)

#ROBPCA
library(rrcov)
set.seed(1)
white.HubPCA<- PcaHubert(white.proc)
summary(white.HubPCA)
par(mfrow=c(1,1))
plot(white.HubPCA, main= " ")
scorePlot(white.HubPCA, col="red", main= " ")

#Add the Quality column to the white.proc matrix
library(stringr)
white.new <- white.proc%>%
  cbind(Quality=white[,12])%>%
  na.omit()
white.class<- white.new[,12]

PCAscores <- white.HubPCA$scores
library(ggfortify)
autoplot(prcomp(PCAscores), data = white.new, colour = 'Quality',
         frame = T, frame.type = 'norm') +
  geom_vline(xintercept=c(-0,0), linetype="dashed", size=0.25) +
  geom_hline(yintercept=c(-0,0), linetype="dashed", size=0.25)


#Train and test
inTrain <- createDataPartition(white.class, p = 2/3, list = F)
train.white <- white.new[inTrain,]
dim(train.white)
test.white <- white.new[-inTrain,]
dim(test.white)

# Chemometric models

#linear Models

# LDA
set.seed(1)
lda<-  train(Quality~., data=train.white, method="lda",
             metric="Accuracy",preProc=c("center", "scale"),
             trControl=trainControl(method="cv", number=5))

ldaPredict<- predict(lda, test.white)
confusionMatrix(test.white$Quality, ldaPredict)

#MDA
set.seed(1)
library(mda)
mda = train( Quality~., data=train.white, method="mda",
             metric="Accuracy", preProc=c("center", "scale"),
             tuneGrid = expand.grid(subclasses= 10),
             trControl=trainControl(method="cv", number=5))
mdaPredict<- predict(mda, test.white)
confusionMatrix(test.white$Quality, mdaPredict)


#PLS-DA
set.seed(1)
```

```
library(pls)
pls= train( Quality~., data=train.white, method = "pls",
            metric="Accuracy", preProc=c("center", "scale"),
            tuneGrid = expand.grid(ncomp=10),
            trControl = trainControl(method="cv", number=5))
plsPredict<- predict(pls, test.white)
confusionMatrix(test.white$Quality, plsPredict)

# RDA
set.seed(1)
library(klaR)
rda.grid = train( Quality~., data=train.white, method = "rda",
                  metric="Accuracy",preProc=c("center", "scale"),
                  trControl = trainControl(method="cv", number=5))
rda.gridPredict<- predict(rda.grid, test.white)
confusionMatrix(test.white$Quality, rda.gridPredict)

#SVM Linear
set.seed(1)
svm.linear<- train(Quality~., data=train.white, method = "svmLinear",
                   metric = "Accuracy", preProc=c("center", "scale"),
                   tuneLength = 5,
                   tuneGrid = expand.grid(.C= 10),
                   trControl = trainControl(method = "cv", number =
5))
svm.linearPredict<- predict(svm.linear, test.white)
confusionMatrix(test.white$Quality, svm.linearPredict)

#Nonlinear Models
# ANN
set.seed(1)
nnet = train( Quality~., data=train.white, method = "nnet",
              metric="Accuracy", preProc=c("center", "scale"),
              trControl = trainControl(method="cv", number=5))
nnetPredict<- predict(nnet, test.white)
confusionMatrix(test.white$Quality, nnetPredict)

# K-Nearest Neighbours
set.seed(1)
knn<- train(Quality~., data=train.white, method = "knn",
            tuneLength = 10,
            preProc=c("center", "scale"),
            trControl = trainControl(method = "cv", number = 5))
knnPredict <- predict(knn, test.white)
confusionMatrix(test.white$Quality, knnPredict)

#LVQ
library(class)
set.seed(1)
train_label <- factor(train.white$Quality)
test_label <- test.white$Quality

codeBook<- lvqinit(train.white, train_label, size = 100)
buildCodeBook<- olvq1(train.white, train_label, codeBook)
lvq.predict<- lvqtest(buildCodeBook, test.white)
confusionMatrix(test_label, lvq.predict)

set.seed(1)
lvq<- train(Quality~., data=train.white, method="lvq",
            metric="Accuracy",
            preProc=c("center", "scale"),
```

```
            tuneGrid = expand.grid(size=100, k= 10),
            trControl=trainControl(method="cv", number=5))
lvqPredict<- predict(lvq, test.white)
confusionMatrix(test.white$Quality, lvqPredict)

#Naive Bayes
set.seed(1)
nb <- train(Quality~., data=train.white, method="nb",
            metric="Accuracy",
            preProc=c("center", "scale"),
            trControl=trainControl(method="cv", number=5))
nbPredict<- predict(nb, test.white)
confusionMatrix(test.white$Quality, nbPredict)

# Tree Based Models
#Bagging
set.seed(1)
treebag<- train(Quality~., data=train.white, method="treebag",
                metric="Accuracy",preProc=c("center", "scale"),
                trControl=trainControl(method="cv", number=5))
treebagPredict<- predict(treebag, test.white)
confusionMatrix(test.white$Quality, treebagPredict)

#C5.0
set.seed(1)
c5.0 <- train(Quality~., data=train.white, method="C5.0",
              metric="Accuracy",preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=5))
c5.0Predict<- predict(c5.0, test.white)
confusionMatrix(test.white$Quality, c5.0Predict)

#Cart
set.seed(1)
cart <- train(Quality~., data=train.white, method="rpart",
              metric="Accuracy",preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=5))
cartPredict<- predict(cart, test.white)
confusionMatrix(test.white$Quality, cartPredict)

#GBM
set.seed(1)
gbm <- train(Quality~., data=train.white, method='gbm',
             verbose=FALSE,preProcess  =  c("center","scale"),metric  =
"Accuracy",
             trControl=trainControl(method="cv", number=5))
gbmPredict <- predict(gbm, test.white)
confusionMatrix(test.white$Quality, gbmPredict)

#Random forest
set.seed(1)
rf.Random <- train(Quality~., data=train.white,
                   method = "rf",
                   preProcess = c("center","scale"),
                   metric ="Accuracy",
                   tuneLength = 15,
                   trControl = trainControl(method = "cv", number = 5,
                                            search = "random"))
rf.RandomPredict <- predict(rf.Random, test.white)
confusionMatrix(rf.RandomPredict, test.white$Quality)

#APLICAÇÃO DE MODELOS DE REGRESSÃO
```

```
white.url     <-     "https://archive.ics.uci.edu/ml/machine-learning-
databases/wine-quality/winequality-white.csv"
white.raw <- read.csv(white.url, header = TRUE, sep = ";")
white <- white.raw
white <- white[!duplicated(white), ]
dim(white)
a <- white [,1:11]
b<- white [,12]

library(caret) ###PRÉ PROCESSAMENTO###
set.seed(1)
white.trans  <-  preProcess(a,  method=c("center",  "scale",  "BoxCox",
"spatialSign"))
print(white.trans)

library(rrcov) ###PCA ROBUSTO###
set.seed(1)
white.HubPCA<- PcaHubert(white.proc)
scorePlot(white.HubPCA, col="red", main= " ")

#Adicionar a coluna Quality a matriz white.proc
library(stringr)
white.new <- white.proc%>%
  cbind(Quality=white[,12])%>%
  na.omit()
white.class<- white.new[,12]

###Treinamento e teste###
inTrain <- createDataPartition(white.class, p = 2/3, list = F)
train.white <- white.new[inTrain,]
test.white <- white.new[-inTrain,]

dim(train.white)
dim(test.white)

### Algoritmos
###Modelos Lineares

control <- trainControl(method="cv", number=10)
seed <- 7
metric <- "RMSE"
preProcess=c("center", "scale")



#PLS
set.seed(seed)
fit.pls  <-  train(Quality~.,  data=train.white,  method="pls",tuneLength
= 10, metric=metric, preProc=c("center", "scale"), trControl=control)
summary(fit.pls)
coef(fit.pls$finalModel, fit.pls$bestTune$lambda)

# SVM Linear
set.seed(seed)
fit.svmLinear       <-       train(Quality~.,       data=train.white,
method="svmLinear",   metric=metric,   preProc=c("center",   "scale"),
trControl=control, fit=FALSE)

# kNN
```

```
set.seed(seed)
fit.knn    <-    train(Quality~.,    data=train.white,    method="knn",
metric=metric, preProc=c("center", "scale"), trControl=control)

# SVM Radial
set.seed(seed)
fit.svmRadial       <-       train(Quality~.,       data=train.white,
method="svmRadial",   metric=metric,   preProc=c("center",   "scale"),
trControl=control, fit=FALSE)

##Árvore de classificação

# Bagged CART
set.seed(seed)
fit.treebag  <-  train(Quality~.,  data=train.white,  method="treebag",
metric=metric, trControl=control)

# CARt
set.seed(seed)
fit.cart   <-   train(Quality~.,   data=train.white,   method="rpart",
metric=metric, trControl=control)

# Stochastic Gradient Boosting (Generalized Boosted Modeling)
set.seed(seed)
fit.gbm    <-    train(Quality~.,    data=train.white,    method="gbm",
metric=metric, trControl=control, verbose=FALSE)

# Random Forest
set.seed(seed)
fit.rf    <-    train(Quality~.,    data=train.white,    method="rf",
metric=metric, trControl=control)

results <- resamples(list( pls=fit.pls,
                     svmLinear= fit.svmLinear,svmRadial=fit.svmRadial,
                           knn=fit.knn,  cart=fit.cart,
                           bagging=fit.treebag,rf=fit.rf,gbm=fit.gbm))
results

# Table comparison
summary(results)

# Gráfico de caixas comparison
bwplot(results)
dev.off()

# Gráfico de pontos comparison
dotplot(results)
options(digits=4)

################################################
#  predict new data
################################################


pred.pls <-predict(fit.pls, newdata = test.white)
print("PLS")
postResample(pred = pred.pls, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.pls)),
as.factor(test.white$Quality))

pred.svmLinear <-predict(fit.svmLinear, newdata = test.white)
```

```
print("SVMLinear")
postResample(pred = pred.svmLinear, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.svmLinear)),
as.factor(test.white$Quality))

pred.svmRadial <-predict(fit.svmRadial, newdata = test.white)
print("SVMRadial")
postResample(pred = pred.svmRadial, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.svmRadial)),
as.factor(test.white$Quality))

pred.knn <-predict(fit.knn, newdata = test.white)
print("knn")
postResample(pred =pred.knn, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.knn)),
as.factor(test.white$Quality))

pred.cart <-predict(fit.cart, newdata = test.white)
print("cart")
postResample(pred = pred.cart, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.cart)),
as.factor(test.white$Quality))

pred.treebag<-predict(fit.treebag, newdata = test.white)
print("treebag")
postResample(pred = pred.treebag, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.treebag)),
as.factor(test.white$Quality))

pred.rf <-predict(fit.rf, newdata = test.white)
print("RF")
postResample(pred = pred.rf, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.rf)),
as.factor(test.white$Quality))

pred.gbm  <-predict(fit.gbm , newdata = test.white)
print("GBM")
postResample(pred = pred.gbm, obs = test.white$Quality)
confusionMatrix(as.factor(round(pred.gbm)),
as.factor(test.white$Quality))
```

ANEXO C- Roteiro do conjunto de dados de qualidade do vinho tinto aplicado no *software* r.

```
red.url   <-   "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
red.raw <- read.csv(red.url, header = TRUE, sep = ";")
red <- red.raw
str(red)
red$quality <- as.factor(red$quality)
sapply(red, class)
red <- red[!duplicated(red), ]
dim(red)
summary(red)
library(psych)
describeBy(red[-12])

levels(red$quality)
percentage <- round(prop.table(table(red$quality)) * 100, 1)
cbind(freq=table(red$quality), percentage=percentage)
par(mfrow = c(1, 1))
corDegrade    <-    colorRampPalette(c("orangered","lightblue4",    "orchid4",
"palegreen4", "gold","purple4", "salmon" ))
distribuiton<-  barplot(percentage,  horiz  =  F,  col=corDegrade(5),  xlab=
"Quality of Red Wine", ylab="Percentage (%)", las=2 )
text(x=distribuiton, y=percentage, labels=percentage, pos=3, xpd=NA)

d <- red[,1:11]
e <- red[,12]

#Histogram
par(mfrow = c(3, 3))
for (i in 1:ncol(d)) {
  hist(d[ ,i], xlab = names(d[i]), main = " ",
       col="cornsilk3", border = "cornsilk4")
}

#Box plot
par(mfrow=c(3,2))
for (i in 1:ncol(d)) {
  boxplot(d[ ,i], ylab = names(d[i]), horizontal=F,main = paste(names(d[i]),
                                                                "Boxplot"),
col="cornsilk3")
}

#Correlation matrix
library(corrgram)
corrgram(d,  lower.panel  =  panel.pts,  upper.panel=  panel.conf,  diag.panel  =
panel.density)

#Pre-processing
library(caret)
set.seed(1)
red.trans    <-    preProcess(d,    method=c("center",    "scale",    "BoxCox",
"spatialSign"))
print(red.trans)
transf <- predict(red.trans,d)
red.proc <- as.data.frame(transf)
colnames(red.proc)
par(mfrow=c(1,1))
boxplot(red.proc, horizontal=F )
summary(red.proc)

#ROBPCA
library(rrcov)
set.seed(1)
red.HubPCA<- PcaHubert(red.proc)
summary(red.HubPCA)
```

```
par(mfrow=c(1,1))
plot(red.HubPCA, main= " ")
scorePlot(red.HubPCA, col="red", main= " ")

#Add the Quality column to the red.proc matrix
library(stringr)
red.new <- red.proc%>%
  cbind(Quality=red[,12])%>%
  na.omit()
red.class<- red.new[,12]

PCAscores <- red.HubPCA$scores
library(ggfortify)
autoplot(prcomp(PCAscores), data = red.new, colour = 'Quality',
         frame = T, frame.type = 'norm') +
  geom_vline(xintercept=c(-0,0), linetype="dashed", size=0.25) +
  geom_hline(yintercept=c(-0,0), linetype="dashed", size=0.25)

#Train and Test
inTrain <- createDataPartition(red.class, p = 2/3, list = F)
train.red <- red.new[inTrain,]
dim(train.red)
test.red <- red.new[-inTrain,]
dim(test.red)

# Chemometric models

#linear Models

# LDA
set.seed(1)
lda<-  train(Quality~., data=train.red, method="lda",
             metric="Accuracy",preProc=c("center", "scale"),
             trControl=trainControl(method="cv", number=5))

ldaPredict<- predict(lda, test.red)
confusionMatrix(test.red$Quality, ldaPredict)

#MDA: Mixture Discriminant Analysis
set.seed(1)
library(mda)
mda<- train( Quality~., data=train.red, method="mda",
             metric="Accuracy", preProc=c("center", "scale"),
             tuneGrid = expand.grid(subclasses= 10),
             trControl=trainControl(method="cv", number=5))
mdaPredict<- predict(mda, test.red)
confusionMatrix(test.red$Quality, mdaPredict)

#Regressão logistica Multinomial
set.seed(1)
mlr <- train(Quality~., data=train.red, method = "glmnet",
             metric = "Accuracy",preProc=c("center", "scale"),
             tuneGrid = expand.grid(alpha = 0.1, lambda = 0.0049),
             trControl = trainControl(method = "cv", number = 5))
mlrPredict <- predict(mlr, test.red)
confusionMatrix(test.red$Quality, mlrPredict)

#PLS: Partial Least Squares Discriminant Analysis using Cross-Validation and
Bootstrap
set.seed(1)
library(pls)
pls<-  train( Quality~., data=train.red, method = "pls",
              metric="Accuracy", preProc=c("center", "scale"),
              tuneGrid = expand.grid(ncomp=10),
              trControl = trainControl(method="cv", number=5))
plsPredict<- predict(pls, test.red)
confusionMatrix(test.red$Quality, plsPredict)
```

```
# RDA: Regularized Discriminant Analysis
set.seed(1)
library(klaR)
rda.grid<- train( Quality~., data=train.red, method = "rda",
                  metric="Accuracy",preProc=c("center", "scale"),
                  trControl = trainControl(method="cv", number=5))
rda.gridPredict<- predict(rda.grid, test.red)
confusionMatrix(test.red$Quality, rda.gridPredict)

#SVM Linear
set.seed(1)
svm.linear<- train(Quality~., data=train.red, method = "svmLinear",
                   metric = "Accuracy", preProc=c("center", "scale"),
                   tuneLength = 5,
                   tuneGrid = expand.grid(.C= 10),
                   trControl = trainControl(method = "cv", number = 5))
svm.linearPredict<- predict(svm.linear, test.red)
confusionMatrix(test.red$Quality, svm.linearPredict)

#Nonlinear Models
# ANN
set.seed(1)
nnet <- train( Quality~., data=train.red, method = "nnet",
               metric="Accuracy", preProc=c("center", "scale"),
               trControl = trainControl(method="cv", number=5))
nnetPredict<- predict(nnet, test.red)
confusionMatrix(test.red$Quality, nnetPredict)

# K-Nearest Neighbours
set.seed(1)
knn<- train(Quality~., data=train.red, method = "knn",
            tuneLength = 10,
            preProc=c("center", "scale"),
            trControl = trainControl(method = "cv", number = 5))
knnPredict <- predict(knn, test.red)
confusionMatrix(test.red$Quality, knnPredict)

#LVQ
library(class)
set.seed(1)
train_label <- factor(train.red$Quality)
test_label <- test.red$Quality

codeBook<- lvqinit(train.red, train_label, size = 100)
buildCodeBook<- olvq1(train.red, train_label, codeBook)
lvq.predict<- lvqtest(buildCodeBook, test.red)
confusionMatrix(test_label, lvq.predict)

set.seed(1)
lvq<- train(Quality~., data=train.red, method="lvq",
            metric="Accuracy",
            preProc=c("center", "scale"),
            tuneGrid = expand.grid(size=100, k= 10),
            trControl=trainControl(method="cv", number=5))
lvqPredict<- predict(lvq, test.red)
confusionMatrix(test.red$Quality, lvqPredict)

#Naive Bayes
set.seed(1)
nb <- train(Quality~., data=train.red, method="nb",
            metric="Accuracy",
            preProc=c("center", "scale"),
            trControl=trainControl(method="cv", number=5))
nbPredict<- predict(nb, test.red)
confusionMatrix(test.red$Quality, nbPredict)

# Tree Based Models
#Bagging
```

```
set.seed(1)
treebag<- train(Quality~., data=train.red, method="treebag",
                metric="Accuracy",preProc=c("center", "scale"),
                trControl=trainControl(method="cv", number=5))
treebagPredict<- predict(treebag, test.red)
confusionMatrix(test.red$Quality, treebagPredict)

#C5.0
set.seed(1)
c5.0 <- train(Quality~., data=train.red, method="C5.0",
              metric="Accuracy",preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=5))
c5.0Predict<- predict(c5.0, test.red)
confusionMatrix(test.red$Quality, c5.0Predict)

#Cart
set.seed(1)
cart <- train(Quality~., data=train.red, method="rpart",
              metric="Accuracy",preProc=c("center", "scale"),
              trControl=trainControl(method="cv", number=5))
cartPredict<- predict(cart, test.red)
confusionMatrix(test.red$Quality, cartPredict)

#GBM
set.seed(1)
gbm <- train(Quality~., data=train.red, method='gbm',
             verbose=FALSE,preProcess    =    c("center","scale"),metric    =
"Accuracy",
             trControl=trainControl(method="cv", number=5))
gbmPredict <- predict(gbm, test.red)
confusionMatrix(test.red$Quality, gbmPredict)


#Random forest
set.seed(1)
rf.Random <- train(Quality~., data=train.red,
                   method = "rf",
                   preProcess = c("center","scale"),
                   metric ="Accuracy",
                   tuneLength = 15,
                   trControl = trainControl(method = "cv", number = 5,
                                            search = "random"))
rf.RandomPredict <- predict(rf.Random, test.red)
confusionMatrix(rf.RandomPredict, test.red$Quality)

## Regressão
red.url   <-   "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
red.raw <- read.csv(red.url, header = TRUE, sep = ";")
red <- red.raw
str(red)
str(red)
red <- red[!duplicated(red), ]
dim(red)
summary(red)

###PRÉ PROCESSAMENTO###
m <- red[,1:11]
n <- red[,12]
set.seed(1)
red.trans    <-    preProcess(m,    method=c("center",    "scale",    "BoxCox",
"spatialSign"))
print(red.trans)

library(rrcov) ##PCA ROBUSTO###
set.seed(1)
red.HubPCA<- PcaHubert(red.proc)
```

```
summary(red.HubPCA)
par(mfrow=c(1,1))
plot(red.HubPCA, main= " ") #Sem outliers
library(stringr)
red.new <- red.proc%>%
  cbind(Quality=red[,12])%>%
  na.omit()
is.na(red.proc)
red.class<- red.new[,12]

inTrain <- createDataPartition(red.class, p = 2/3, list = F) ###treinamento e
teste
train.red <- red.new[inTrain,]
test.red <- red.new[-inTrain,]
str(test.red)

###ALGORITMOS
###Modelos Lineares

control <- trainControl(method="cv", number=10)
seed <- 7
metric <- "RMSE"
preProcess=c("center", "scale")


#PLS
set.seed(seed)
fit.pls <- train(Quality~., data=train.red, method="pls",tuneLength = 10,
metric=metric, preProc=c("center", "scale"), trControl=control)
summary(fit.pls)
coef(fit.pls$finalModel, fit.pls$bestTune$lambda)

# SVM Linear
set.seed(seed)
fit.svmLinear <- train(Quality~., data=train.red, method="svmLinear",
metric=metric, preProc=c("center", "scale"), trControl=control, fit=FALSE)


### Não linear
# kNN
set.seed(seed)
fit.knn <- train(Quality~., data=train.red, method="knn", metric=metric,
preProc=c("center", "scale"), trControl=control)

# SVM Radial
set.seed(seed)
fit.svmRadial <- train(Quality~., data=train.red, method="svmRadial",
metric=metric, preProc=c("center", "scale"), trControl=control, fit=FALSE)

##Árvore de classificação

# Bagged CART
set.seed(seed)
fit.treebag <- train(Quality~., data=train.red, method="treebag",
metric=metric, trControl=control)

# CART
set.seed(seed)
fit.cart <- train(Quality~., data=train.red, method="rpart", metric=metric,
trControl=control)

# Stochastic Gradient Boosting (Generalized Boosted Modeling)
set.seed(seed)
fit.gbm <- train(Quality~., data=train.red, method="gbm", metric=metric,
trControl=control, verbose=FALSE)

# Random Forest
set.seed(seed)
```

```
fit.rf   <-   train(Quality~.,   data=train.red,   method="rf",   metric=metric,
trControl=control)
results <- resamples(list(pls=fit.pls,
                     svmLinear= fit.svmLinear,svmRadial=fit.svmRadial,
                           knn=fit.knn,  cart=fit.cart,
                           bagging=fit.treebag,rf=fit.rf,gbm=fit.gbm))
results

# Table comparison
summary(results)
jpeg("Plotinered.jpeg",width = 7, height = 8, units = 'in', res = 1900)
jpeg("Plotwinered.jpg",width = 960)

# Gráfico de caixas comparison
bwplot(results)
dev.off()

# Gráfico de pontos comparison
dotplot(results)
options(digits=4)

###############################################
#  predict new data
###############################################

pred.pls <-predict(fit.pls, newdata = test.red)
print("PLS")
postResample(pred = pred.pls, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.pls)), as.factor(test.red$Quality))

pred.svmLinear <-predict(fit.svmLinear, newdata = test.red)
print("SVMLinear")
postResample(pred = pred.svmLinear, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.svmLinear)), as.factor(test.red$Quality))

pred.svmRadial <-predict(fit.svmRadial, newdata = test.red)
print("SVMRadial")
postResample(pred = pred.svmRadial, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.svmRadial)), as.factor(test.red$Quality))

pred.knn <-predict(fit.knn, newdata = test.red)
print("knn")
postResample(pred =pred.knn, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.knn)), as.factor(test.red$Quality))

pred.cart <-predict(fit.cart, newdata = test.red)
print("cart")
postResample(pred = pred.cart, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.cart)), as.factor(test.red$Quality))

pred.treebag<-predict(fit.treebag, newdata = test.red)
print("treebag")
postResample(pred = pred.treebag, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.treebag)), as.factor(test.red$Quality))

pred.rf <-predict(fit.rf, newdata = test.red)
print("RF")
postResample(pred = pred.rf, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.rf)), as.factor(test.red$Quality))

pred.gbm  <-predict(fit.gbm , newdata = test.red)
print("GBM")
postResample(pred = pred.gbm, obs = test.red$Quality)
confusionMatrix(as.factor(round(pred.gbm)), as.factor(test.red$Quality))
```