Aplicação de algoritmos de aprendizado de máquina para classificação da qualidade das carcaças dos lotes de bovinos abatidos: um estudo de caso nos dados do programa Precoce MS

Rafael Rodrigues Marquesi

SERVIÇO DE PÓS-GRADUAÇÃO DA FACOM-UFMS

Data de Depósito:

Assinatura:_____

Orientador: *Prof. Dr. Rafael Geraldeli Rossi*
Coorientadora: *Dra. Thaís Basso Amaral*

Defesa do Mestrado Profissional em Computação Aplicada da Faculdade de Computação da Universidade Federal do Mato Grosso do Sul, como parte dos requisitos para obtenção do título de Mestre em Computação Aplicada.

UFMS - Campo Grande
Agosto/2023

A minha esposa,
Amanda,

Aos meus pais,
Aparecido e Maria Neide,

A minha irmã,
Thamires,

Ao meu orientador e coorientadora,
Rafael Rossi e Thaís Amaral.

Agradecimentos

Agradeço ao Instituto Federal de Educação, Ciência e Tecnologia de Mato Grosso (IFMT), e aos colegas de trabalho que seguraram as "pontas", pela oportunidade de proporcionar o afastamento para realização do programa de mestrado.

Agradeço minha esposa Amanda Cristina de Souza pela paciência e parceria, em todos os momentos que tivemos de deixar de realizar algum programa em família, com amigos, e até juntos, para me dedicar ao programa de mestrado. Como, também, em me apoiar nos momentos que pensei que não daria conta.

Agradeço ao meu orientador, o Prof. Dr. Rafael Geraldeli Rossi, por todas as orientações, pelas dicas, críticas, sugestões e esclarecimentos que permitiram a realização deste trabalho. Desde o início do projeto esteve disponível para me auxiliar, me ajudando muito em diversos desafios enfrentados.

Agradeço a minha coorientadora, a Dra. Thaís Basso Amaral, por estar sempre apoiando com parcerias para o projeto. Como, também, me auxiliando com dúvidas especificas do programa Precoce MS.

Agradeço ao pessoal do grupo de estudo, Madson Viana e Glaucilene Gonçalves, com o auxílio, dicas e validações durante o desenvolvimento da aplicação.

Agradeço a todos os professores, pela paciência, dedicação e o ótimo trabalho que desempenharam durante todos esses anos, contribuindo muito para a minha formação.

Resumo

Visando incentivar a produção de carne bovina de qualidade superior e buscando atender a critérios de um mercado cada vez mais exigente, o Governo do Mato Grosso do Sul provém incentivos fiscais, por meio do programa Precoce MS, para produtores que abatem animais com carcaça de qualidade superior e em idade jovem. O programa Precoce MS disponibiliza um conjunto de dados com informações relacionadas a características dos bovinos, sistemas de produção, e a qualidade da carcaça dos animais abatidos. Porém, a análise manual dos dados para encontrar fatores que podem estar relacionados à produção de uma carcaça de qualidade superior pode ser inviável. Neste cenário, técnicas de Mineração de Dados podem ser aplicadas para extrair conhecimento útil e construir modelos para predição da qualidade da carcaça. Trabalhos anteriores já aplicaram técnicas de Mineração de Dados no conjunto de dados do programa Precoce MS. No entanto, a performance de classificação era incerta em dados atuais, não foram utilizados algoritmos estado-da-arte para dados tabulares, não se tinha a utilização de animais separados em lote, e não foram utilizados outros atributos potencialmente importantes para predição da qualidade da carcaça, como atributos climáticos, nutricionais, e relacionados ao preço de *commodities*. Dado isso, o presente trabalho teve por objetivo utilizar técnicas de Mineração de Dados, mais especificamente, algoritmos para a construção de modelos de classificação para predição da qualidade do lote de carcaças, considerando: algoritmos estado-da-arte, de diferentes paradigmas e com diferentes conjuntos de parâmetros. Além disso, foram considerados os dados mais atuais do programa Precoce-MS (até a data de execução deste trabalho) e o conjunto de dados enriquecido com atributos meteorológicos, nutricionais e de precificação de *commodities*. Os resultados demonstraram que o algoritmo *Random Forest Classifier* apresentou a melhor performance de classificação (72.63% de Acurácia). Por fim, utilizando o melhor modelo, foi desenvolvida uma *API REST* para realizar a

classificação do lote de bovinos a serem abatidos.

Palavras-chave: mineração de dados, aprendizado de máquina, algoritmos supervisionados, classificação, pecuária de precisão, agropecuária.

Abstract

In order to encourage the production of superior quality beef and seeking to meet the criteria of an increasingly demanding market, the Government of Mato Grosso do Sul provides tax incentives, through the Precoce MS program, for producers who slaughter animals with higher quality carcasses and at a young age. The Precoce MS program provides a dataset with information related to the characteristics of cattle, production systems, and the quality of the carcass of the slaughtered animals. However, manually analyzing the data to find factors that may be related to the production of a higher quality carcass may be impractical. In this scenario, Data Mining techniques can be applied to extract useful knowledge and build models to predict carcass quality. Previous works have already applied Data Mining techniques to the Precoce MS program dataset. However, the classification performance was uncertain in current data, state-of-the-art algorithms were not used for tabular data, animals were not used in batches, and other potentially important attributes for predicting the quality of the carcass were not used, such as climatic, nutritional, and commodity price-related attributes. Given this, the present work aimed to use Data Mining techniques, more specifically, algorithms for the construction of classification models to predict the quality of the batch of carcasses, considering: state-of-the-art algorithms, from different paradigms and with different sets of parameters. Furthermore, the most current data from the Precoce-MS program was considered (up to the date of execution of this work) and the data set enriched with meteorological, nutritional and commodity pricing attributes. The results showed that the Random Forest Classifier algorithm presented the best classification performance (72.63% Accuracy). Finally, using the best model, a REST API was developed to classify the batch of cattle to be slaughtered.

Keywords: *data mining, machine learning, supervised algorithms, classification, precision livestock, agriculture.*

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AM Aprendizado de Máquina

ANN *Artificial Neural Network*

API *Application Programming Interface*

APPCC Análise de Perigos e Pontos Críticos de Controle

ASPNP Associação Sul-matogrossense de Produtores de Novilho Precoce

AUC *Area Under the Curve*

Bagging *Bootstrap aggregation*

BI *Business Intelligence*

BN *Batch Normalization*

CART *Classification And Regression Tree*

CNPTIA Embrapa Agricultura Digital

CSV *Comma-Separated Values*

CTLEAN *Computed Tomography Lean Meat Yield*

CWT *Carcase Weigh*

DBSCAN *Density-Based Spatial Clustering of Applications with Noise*

DL *Deep Learning*

DM *Data Mining*

DNNs *Deep Neural Networks*

DP Desvio Padrão

EMA *Eye Muscle Area*

Embrapa Empresa Brasileira de Pesquisa Agropecuária

EurepGap *European Retailers Produce Working Group - Good Agricultural Practices*

FC *Fully Connected*

FS *Feature Selection*

GA *Genetic Algorithm*

GBT *Gradient Boosting Tree*

GRFAT *Greville Rule Fat Depth*

HCW *Hot Carcass Weight*

HPC *High Performance Computing*

ICMS Imposto sobre Circulação de Mercadorias e Serviços

ID Identificador

ID Identificador

IMF *Intramuscular Fat*

JSON *Javascript Object Notation*

KDD *Knowledge Discovery in Databases*

KNN *k-Nearest Neighbor*

LW *Loin Weight*

MD Mineração de Dados

ML *Machine Learning*

MLP *Multilayer Perceptron*

MS Mato Grosso do Sul

MT *Model Tree*

OLAP *Online Analytical Processing*

PCA *Principal Componets Analysis*

RBF *Radial Basis Function*

REST *Representational State Transfer*

RF *Random Forests*

RNA Redes Neurais Artificias

ROC *Receiver Operating Characteristic*

SEFAZ Secretaria de Fazenda

SEPAF Secretaria de Estado de Produção e Agricultura Familiar

SIF Sistema de Inspeção Federal

SirLn *Weights of Sirloin*

SMO *Sequential Minimal Optimization*

SNP *Single Nucleotide Polymorphisms*

StrLn *Striploin*

SVM *Support Vector Machines*

SVR *Support Vector Regression*

TabNet *Interpretable Tabular Data Learning*

TIC Tecnologia da Informação e Comunicação

TndLn *Tenderloin*

WSGI *Web Server Gateway Interface*

XGBoost *Extreme Gradiente Boosting*

CAPÍTULO

1

Introdução

A exportação de carne brasileira tem grande impacto na economia do país. Por exemplo, de 2000 a 2020 as exportações renderam US$ 265 bilhões. Dentre os diferentes tipos de carne, a carne bovina vem ganhando cada vez maior notoriedade. Por exemplo, no ano de 2020, o maior rebanho bovino mundial foi o brasileiro, representando 14.3% do total mundial. No mesmo ano, o Brasil foi o maior exportador de carne bovina, com 2.2 milhões de toneladas sendo responsável por 14.4% do mercado internacional (Guaraldo, 2021).

O estado de Mato Grosso do Sul ocupa o quarto lugar em tamanho de rebanho no Brasil, com um total de 19 milhões de cabeças, representando cerca de 10% dos abates do país. Porém, quando se refere ao quesito qualidade, o estado está à frente dos demais, dado que cerca de 93% dos animais abatidos (dentre 4 milhões analisados pelo estudo) apresentaram padrão de qualidade desejável, enquanto os outros estados apresentam uma taxa de aproximadamente 30% (Amaral e Gomes, 2020).

O padrão de qualidade da carne bovina é norteado por questões relacionadas às exigências sanitárias e segurança alimentar impostas pelos países compradores. Desta forma, cada país possuí uma lista de exigências para importar a carne produzida no Brasil (Paixão e Almeida, 2020). Um resumo das principais exigências dos mercados externos é apresentado na Tabela 1.1.

Para estimular a produção de carne com qualidade no estado do Mato Grosso do Sul, foi desenvolvido o programa Precoce MS. Este programa foi instituído no decreto N°14.526, de 28 de julho de 2016 em conjunto com a resolução SEFAZ/SEPAF N°69 de 30 de agosto de 2016. Neste, o produtor poderá ganhar um incentivo fiscal equivalente a, no máximo, 67% do ICMS recolhido, utilizando técnicas que contribuam para produção de animais com qualidade

Tabela 1.1: Principais exigências dos mercados externos importadores de carne brasileira.

Mercado	**Exigências**
Países da União Europeia	Rastreabilidade, Sistema de Inspeção Federal (SIF), aprovação para comercialização, diferentes especificações de corte, selos de qualidade, Análise de Perigos e Pontos Críticos de Controle (APPCC), *EurepGap*, entre outros.
Países do Oriente Médio	APPCC, ritual religioso do *Halal*, para alguns países apenas SIF, outros países como a Arábia requerem habilitação e documentação específica.
Países da Ásia	Varia conforme o país. Os requisitos são basicamente SIF, APPCC e ritual religioso do *Halal*.
Rússia e Europa Central	SIF.

Fonte: Paixão e Almeida (2020).

de carcaça superior e que possuam sistemas de produção mais sustentáveis. Para concessão, são avaliados as seguintes dimensões: o processo produtivo, o produto obtido e a padronização do lote, onde, 30% do incentivo a ser pago ao produtor é resultante da dimensão processo produtivo e 70% do incentivo é resultante da dimensão do produto obtido (SEFAZ, 2016; Governo do Estado de Mato Grosso do Sul, 2016).

Uma das alternativas para auxiliar o produtor em diferentes desafios, como na produção de carne bovina que atenda os padrões de qualidade exigido pelo mercado, é o uso de Tecnologias da Informação e Comunicação (TICs). Por exemplo, a área da Pecuária de Precisão trabalha uma série de tecnologias para garantir boas práticas de produção de carne, como o manejo de rebanho com auxílio de dispositivos eletrônicos, simulações de crescimento de pastagens, previsão de preços de *commodities*, diagnóstico precoce de doenças, estudo de linhagens genéticas, classificação automática das carcaças, etc (Cáceres et al., 2011).

Além disso, pode-se também empregar técnicas de Mineração de Dados (MD) para melhorar o processo produtivo, por exemplo, para a detecção de doenças, como febre aftosa (Lee et al., 2017), na estimativa de biomassa em pastagem, sendo uma fonte de alimentação da pecuária (Ali et al., 2017), classificar o comportamento do gado usando dados coletados por sensores (Dutta et al., 2015), e predizer o peso da carcaça do bovino durante a trajetória de criação (Alonso et al., 2013).

Técnicas de MD também tem sido empregadas para a extração de padrões sobre a produção de carne de qualidade no estado do Mato Grosso do Sul.

Por exemplo, no trabalho de Mota (2016), são utilizadas técnicas de *Business Intelligence (BI)* e MD, aplicando algoritmos de classificação para descobrir padrões e relacionamentos vinculados ao grau de acabamento e ao rendimento da carcaça. Neste trabalho, foram utilizados diversos conjuntos de dados disponibilizados pela Associação Sul-Mato-Grossense de Produtores de Novilho Precoce (ASPNP) e de estudos realizados na área de ciência animal pela Embrapa Gado de Corte. Já em Nucci (2019), que teve como base as propostas de Mota (2016), são utilizados algoritmos de Aprendizado de Máquina (AM) para a construção de um modelo de classificação do acabamento da carcaça. Foi utilizado dados do programa Precoce MS no período de 09/02/2017 até 23/01/2019.

Vale ressaltar que estes trabalhos carecem de uma aplicação em uma base de dados mais atualizada, uma vez que as características que levam à qualidade podem mudar ao longo do tempo, há também a ausência de algoritmos estado-da-arte para a classificação de dados tabulares, e o não uso, por estes trabalhos, de outros atributos que podem influenciar a qualidade da carcaça, como a quantidade de chuva, temperatura, informações nutricionais da alimentação e de *commodities*, como milho e arroba do boi. Por fim, tais trabalhos mediram a qualidade do animal individualmente, enquanto não há uma mensuração da qualidade do lote, o que seria mais viável em uma aplicação real para o produtor.

1.1 Objetivos

O objetivo geral do trabalho é avaliar qual o algoritmo de aprendizado de máquina supervisionado possui melhor desempenho no conjunto de dados mais atual do programa Precoce MS e, então, construir um sistema de classificação para predizer a qualidade das carcaças dos lotes de bovinos a serem abatidos, dados parâmetros fornecidos pelo produtor. O sistema apoiará os produtores de gado de corte na tomada de decisão, mitigará os riscos de insucessos em regiões onde há maior necessidade, apoiará políticas públicas e incentivos a produção, visando aumentar a qualidade da carne produzida.

Já como objetivos específicos têm-se:

- Aplicar um maior número de algoritmos de AM e parâmetros desses algoritmos em comparação com os trabalhos Mota (2016) e Nucci (2019);
- Aplicar técnicas de *tuning* de parâmetros para obter os parâmetros que propiciam os melhores resultados para cada algoritmo;
- Avaliar a performance e comparar os resultados obtidos nos diferentes modelos de classificação utilizando técnicas de significância estatística;

e

- Desenvolvimento de uma *API REST* que conterá o modelo de classificação selecionado e retornará as previsões do mesmo.

Dessa forma, espera-se obter um modelo de classificação da qualidade das carcaças do lote de bovinos a ser produzido, que auxiliará produtores na tomada de decisão, além da extração de conhecimento quanto a quais fatores têm maior impacto na produção de lote de animais de qualidade de carcaça superior. Ademais, será implementada uma *API REST* de apoio à tomada de decisões contendo o melhor modelo de classificação.

1.2 Principais Contribuições

Como principais contribuições do trabalho temos:

- Geração de uma nova base de dados do programa Precoce MS com as informações dos animais abatidos agregados em lote;
- Execução de algoritmos de AM estado-da-arte, para classificação, não aplicados em trabalhos anteriores;
- Exploração de uma grande variação de parâmetros dos algoritmos analisados;
- Avaliação de performance e custo benefício dos algoritmos analisados;
- Análise de importância dos atributos utilizados na geração do melhor modelo selecionado, buscando entender melhor o que leva a criação de um animal de qualidade de carcaça superior;
- Desenvolvimento de uma *API REST* contendo o melhor modelo de classificação, o qual irá realizar a predição da qualidade do lote dadas as características informadas pelo produtor.

1.3 Organização do Texto

O restante do trabalho está dividido em 4 capítulos. No Capítulo 2 são apresentados o embasamento teórico acerca das etapas de pré-processamento, extração de padrões, e pós-processamento, de um processo de MD qual será utilizado neste trabalho, e também, os trabalhos relacionados. No Capítulo 3 são apresentados o método de pesquisa adotado, baseado na aplicação de cada etapa de um processo de MD no conjunto de dados do programa Precoce MS, desde sua aquisição até a utilização de conhecimento. No Capítulo 4 são

apresentados os resultados obtidos através da aplicação do processo de MD no conjunto de dados do programa Precoce MS. Por fim, no Capítulo 5 são apresentadas as conclusões do trabalho, destacando os principais resultados e contribuições, as limitações encontradas, e propostas de trabalhos futuros utilizando o conjunto de dados do programa Precoce MS.

CAPÍTULO

2

Embasamento Teórico

Neste capítulo são apresentadas as descrições e técnicas de cada etapa de um processo de MD empregadas neste projeto. Ainda, são apresentados os trabalhos relacionados ao uso da MD, tanto para a predição de qualidade das carcaças de bovinos, quanto em outras aplicações relacionados à pecuária.

2.1 Mineração de Dados

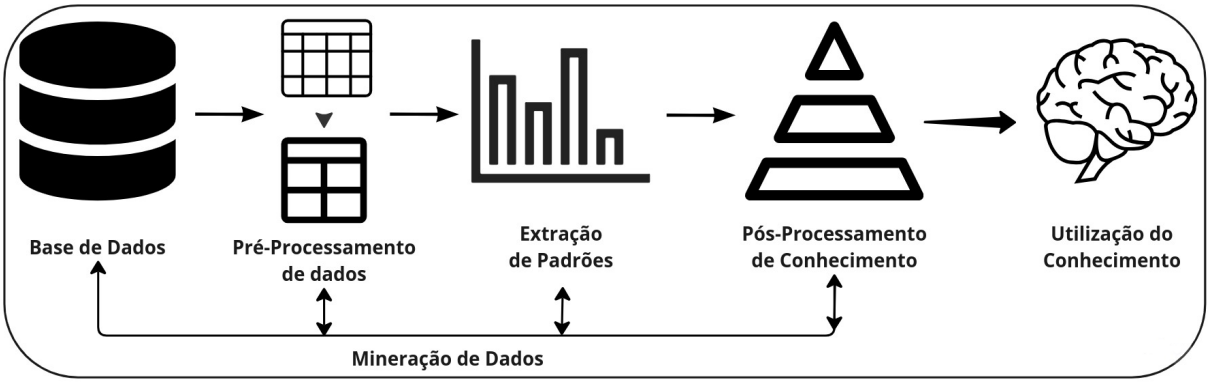
Mineração de Dados (MD), do inglês *Data Mining (DM)*, trata-se da solução de problemas analisando dados já existentes em uma determinada base de dados através do processo de descoberta de padrões nos dados, sendo que os padrões descobertos devem representar alguma vantagem para o problema proposto (Witten e Frank, 2005). A MD também pode ser definida como um processo que combina métodos tradicionais de análise de dados com algoritmos sofisticados para processar grandes volumes de dados, buscando explicar esses dados e fazer previsões a partir deles (Tan et al., 2014).

Na Figura 2.1 é apresentada uma ilustração das etapas de um processo de MD, que consistem em cinco etapas: obtenção da base de dados, pré-processamento, extração de padrões, pós-processamento, e a utilização do conhecimento. Os detalhes das etapas de pré-processamento, extração de padrões, e pós-processamento serão apresentadas nas próximas subseções.

2.1.1 Pré-processamento

Os dados que servem como matéria-prima para os algoritmos de MD podem conter ruídos (por exemplo, *outliers*), serem incompletos (por exemplo,

Figura 2.1: Etapas de um processo de Mineração de Dados.



Fonte: Adaptado de Parmezan et al. (2012).

valores ausentes), ou ainda possuírem atributos redundantes, ou espúrios, os quais podem impactar negativamente a qualidade dos resultados da MD (Chakrabarti et al., 2008; García et al., 2015). Dado isso, a etapa de pré-processamento visa a aplicação de técnicas para tornarem os dados mais adequados para a obtenção de resultados satisfatórios em um processo de MD, sendo, talvez, a etapa mais trabalhosa e demorada (Parmezan et al., 2012; Tan et al., 2014).

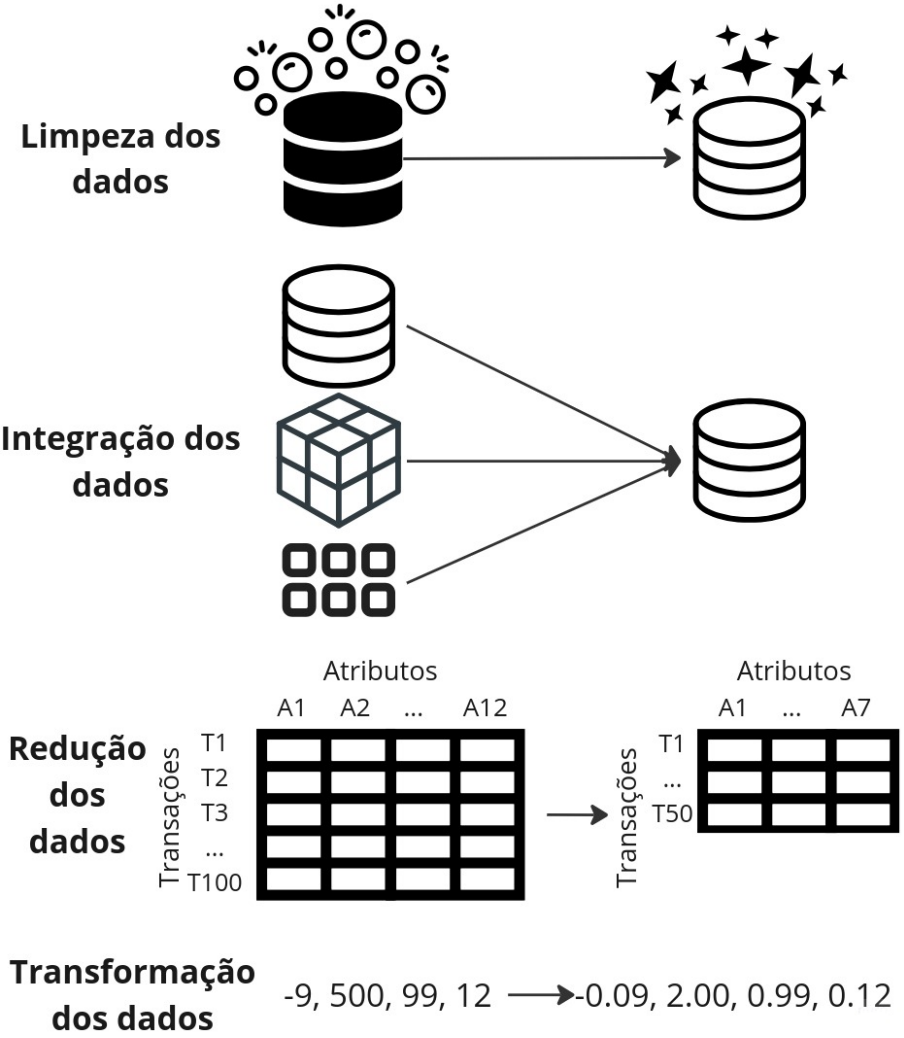
O pré-processamento pode ser dividido em algumas tarefas, como as apresentadas na Figura 2.2, na qual são aplicadas técnicas para limpeza de dados, integração de dados, redução de dados e transformação de dados (Han et al., 2011). As diversas técnicas desenvolvidas buscam superar os problemas nos conjuntos de dados reais e obter um conjunto de dados final, melhorando substancialmente a qualidade geral dos padrões extraídos ou o tempo necessário na etapa de extração de padrões (Chakrabarti et al., 2008). Os detalhes de cada uma das tarefas mencionadas serão apresentados nas próximas subseções.

Limpeza de Dados

A limpeza de dados busca corrigir e filtrar dados incorretos. Dados sujos, em geral, são dados ausentes, dados errôneos (ocasionados por algum erro na entrada dos dados), ou dados ruidosos (dados corrompidos ou distorcidos), os quais podem afetar negativamente no resultado do modelo na etapa de extração de padrões (García et al., 2015). As técnicas utilizadas tentam preencher valores ausentes, ou eliminar exemplos com valores ausentes, suavizar ruídos ao identificar discrepâncias e corrigir inconsistências nos dados (Han et al., 2011).

Existem alguns métodos para o tratamento de valores ausentes, como ig-

Figura 2.2: Exemplos de tarefas de pré-processamento dos dados.



Fonte: Adaptado de Han et al. (2011).

norar a tupla que contenha um ou mais de um valor ausente, preencher o valor ausente manualmente, usar uma constante global para preencher os valores ausentes, usar a média do atributo para preencher os valores ausentes, usar a média ou mediana para as amostras pertencentes a mesma classe da tupla analisada, ou usar o valor provável para preencher o valor ausente (Chakrabarti et al., 2008).

Utilizando algumas técnicas de descrição estatística (por exemplo, *boxplots*, *scatter plots* e histogramas) é possível conseguir identificar *outliers*, que pode representar dados com ruídos (Han et al., 2011). Dados com ruídos são problemas relevantes, especialmente na abordagem supervisionada, afetando na extração de conhecimento e na predição dos modelos obtidos em algoritmos de classificação e regressão (García et al., 2015).

Técnicas de suavização podem ser utilizadas para tratar os ruídos. Como o método *binning*, que suaviza um valor consultando sua "vizinhança", ou

seja, os valores mais próximos (Chakrabarti et al., 2008). A regressão também é utilizada como método para suavização, utilizando, por exemplo, a regressão linear, que envolve encontrar a melhor "reta" para ajustar dois atributos em que um atributo possa ser usado para prever o outro, e/ou regressão linear múltipla, no qual mais de dois atributos estão envolvidos e os dados são ajustados em uma superfície multidimensional (Chakrabarti et al., 2008; Han et al., 2011). Por fim, uma das técnicas que pode-se detectar *outliers* é por agrupamento, como, por exemplo, no algoritmo *Density-Based Spatial Clustering of Applications with Noise (DBSCAN)*, qual organiza valores semelhantes em grupos ou *"clusters"*, sendo que, intuitivamente, os valores que estão fora dos *"clusters"* podem ser considerados *outliers* (Han et al., 2011; Tan et al., 2014).

Integração dos Dados

A integração de dados compreende a mesclagem de várias fontes de armazenamento de dados, apresentando muitos desafios. O processo deve ser realizado com cuidado para evitar redundâncias e inconsistências no conjunto de dados resultante (García et al., 2015).

As técnicas tipicamente utilizadas na integração de dados são a identificação e unificação de variáveis e domínios, também conhecida como identificação de entidades, a detecção/eliminação de duplicação de tuplas, e a detecção e resolução de conflitos em valores de dados de diferentes fontes (Chakrabarti et al., 2008; García et al., 2015).

As tuplas duplicadas em um conjunto de dados podem ser removidas, pois, pode-se tornar uma fonte de inconsistência e causar desperdiço de espaço e tempo para construção de modelos. Portanto, deve-se evitar este tipo de situação (García et al., 2015).

Por fim, é possível ter uma mesma entidade do mundo real, mas os valores de atributos de diferentes origens podem diferir. Por exemplo, um atributo preço pode ser armazenado em moeda brasileira em um conjunto de dados e em moeda americana em outro, também pode-se ter um atributo peso armazenado em quilo grama (kg) em um conjunto de dados e em libras (lbs) em outro, tornando-se importante a detecção e resolução de conflitos de valor de dados durante a integração de dados (Chakrabarti et al., 2008).

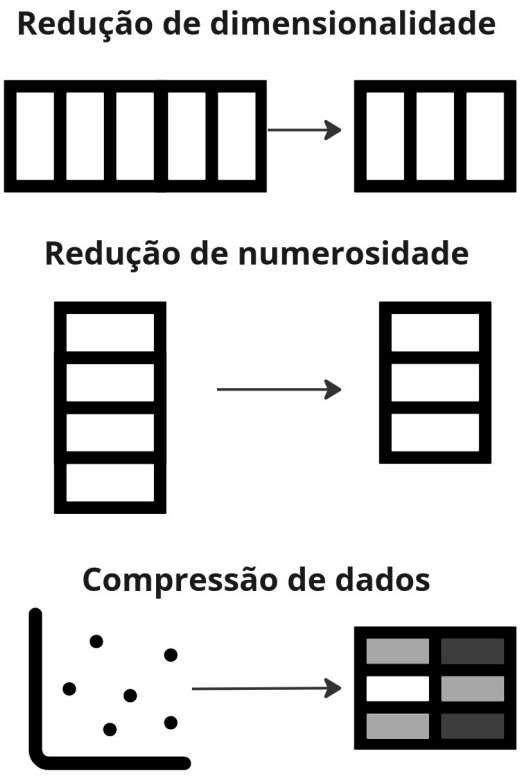
Redução dos Dados

A complexidade dos algoritmos de extração de padrões é baseada no número de exemplos e atributos. Portanto, a construção de modelos considerando grandes quantidades de dados pode muitas vezes inviabilizar a construção do modelo, seja por restrições de tempo ou espaço (García et al., 2015).

Dessa forma, as técnicas de redução de dados podem ser utilizadas para obter uma representação reduzida dos dados originais, tentando preservar as informações e padrões originalmente presentes no conjunto original (Han et al., 2011).

A redução de dados é composta por três estratégias principais, como pode ser observado na Figura 2.3. A redução de dimensionalidade visa reduzir o número de atributos, na qual pode-se utilizar o processo de *Feature Selection (FS)* para a escolha de um subconjunto ótimo de recursos de acordo com um determinado critério, e técnicas como *autoencoders*, que permite o processamento de dados de entrada de alta dimensão com mais eficiência, e análise de componentes principais, ou *PCA*. Já a redução de numerosidade pretende substituir o volume de dados original por formas alternativas e menores de representação dos dados, utilizando técnicas como amostragem de dados e agrupamento de dados. Por fim, a compressão de dados procura obter uma representação "comprimida" dos dados originais, por exemplo, reduzir o número de valores para um determinado atributo contínuo em um conjunto de intervalos, utilizando técnicas como discretização e *binning*, que também pode ser encontrada como estratégia para limpeza de dados abordada na seção 2.1.1 (Han et al., 2011).

Figura 2.3: Estratégias de redução de dados.



Fonte: Adaptado de García et al. (2015).

O processo de seleção de recursos, *Feature Selection (FS)*, para redução de dimensionalidade, é categorizado conforme o conjunto de dados de treinamento, como supervisionado, não supervisionado e semi-supervisionado (Aggarwal, 2014). A seleção de recursos supervisionada possui a categoria modelos de filtro, que se baseia em medidas das características gerais dos dados de treinamento, como a correlação (Aggarwal, 2014). Qual, é tipicamente utilizada para análise de redundância de atributos.

A redundância de atributos é um problema ao conjunto de dados, pois pode tornar o conjunto de dados muito maior do que realmente deveria ser, podendo, também, induzir a *overfitting* do modelo induzido. Por exemplo, um atributo, como receita anual, pode ser "derivado" de outro atributo ou conjunto de atributos. Utilizando a técnica de correlação pode-se medir quão forte é a implicação de um atributo para outro (Han et al., 2011; García et al., 2015).

Quando os dados são nominais, é utilizada a medida de correlação χ^2 (qui-quadrado). Suponha que dois atributos nominais, A e B, contenham c e r valores distintos cada um, a saber, $a_1,...,a_c$ e $b_1,...,b_r$. Para verificar a correlação entre eles, é criada uma tabela de contingência, com os valores c de A formando as colunas e os valores r de B formando as linhas. Seja (A_i,B_j) os eventos conjuntos em que o atributo A assume o valor a_i e o atributo B assume o valor b_j. Cada evento conjunto possível (A_i,B_j) tem sua própria entrada na tabela (Han et al., 2011; García et al., 2015). O valor χ^2 é definido da seguinte forma:

$$\chi^2 = \sum_{i=1}^{c} \sum_{j=1}^{r} \frac{(o_{ij} - e_{ij})^2}{e_{ij}}, \tag{2.1}$$

em que o_{ij} é a frequência observada do evento conjunto (A_i,B_j), e e_{ij} é a frequência esperada de (A_i,B_j) calculada da seguinte forma:

$$e_{ij} = \frac{count(A = a_i) \times count(B = b_j)}{m}, \tag{2.2}$$

em que m é o número de tuplas do conjunto de dados, $count(A = a_i)$ é o número de tuplas com valor a_i para A e $count(B = b_j)$ é o número de tuplas com valor b_j para B.

A medida estatística χ^2 (qui-quadrado) testa a hipótese de que A e B são independentes, ou seja, não há correlação entre eles, com um $(r-1) \times (c-1)$ graus de liberdade (Han et al., 2011; García et al., 2015).

Para atributos numéricos, pode-se usar coeficiente de correlação como o coeficiente de *Pearson*, dado por Han et al. (2011); García et al. (2015):

$$r_{A,B} = \frac{\sum_{i=1}^{m}(a_i - \overline{A})(b_i - \overline{B})}{m\sigma_A\sigma_B} = \frac{\sum_{i=1}^{m}(a_i b_i) - m\overline{AB})}{m\sigma_A\sigma_B}, \tag{2.3}$$

em que A e B são variáveis aleatórias, m é o número de tuplas, a_i e b_i são os respectivos valores de A e B na tupla i, $\overline{A}$ e $\overline{B}$ são os respectivos valores médios de A e B, e $\sigma_A\sigma_B$ são os respectivos desvios padrões de A e B (Han et al., 2011). O valor obtido da Equação 2.3 sempre será entre $-1 \leq r_{A,B} \leq +1$. Se o valor for: (i) igual a 0, não há correlação entre os atributos; (ii) menor que 0, há correlação negativa; e (iii) maior que 0, há correlação positiva. Quanto maior o valor, mais forte a correlação, ou seja, um atributo implica no outro mais. Dessa forma, um valor alto de $r_{A,B}$ pode indicar que o atributo A ou o atributo B pode ser removido (Han et al., 2011).

A técnica de amostragem de dados, utilizada na redução de numerosidade, permite que um grande conjunto de dados seja representado por uma amostra de dados (ou subconjunto) menor, facilitando a análise e modelagem (Chakrabarti et al., 2008; García et al., 2015). Para isso, suponha um grande conjunto de dados, $\mathcal{D}$, contendo m tuplas. Existem várias formas de amostragem de dados, no entanto, segundo Han et al. (2011) e García et al. (2015), as maneiras mais comuns de amostrar $\mathcal{D}$ são:

- Amostra aleatória simples sem reposição (AASSR) de tamanho s: criada extraindo s das m tuplas de $\mathcal{D}$, na qual a probabilidade extrair qualquer tupla em $\mathcal{D}$ é $1/m$, isto é, todas as tuplas têm igual chance de ser extraída;

- Amostra aleatória simples com reposição (AASCR) de tamanho s: análogo a AASSR, exceto que cada vez que uma tupla é extraída de $\mathcal{D}$, ela é registrada e substituída. Isto é, depois que uma tupla é tirada, ela é colocada de volta em $\mathcal{D}$ e pode ser retirada novamente;

- Amostra balanceada: a amostra é retirada de acordo com uma variável alvo, sendo forçada a ter uma determinada composição de acordo com um critério pré-definido. Por exemplo, 90% dos clientes mais velhos ou com 21 anos e 10% dos clientes com menos de 21 anos;

- Amostra de *cluster*: se as tuplas em $\mathcal{D}$ forem agrupadas em g grupos ou *clusters* mutuamente disjuntos, então uma amostra aleatória simples de s *clusters* pode ser obtido, no qual $s < g$. Por exemplo, em conjunto de dados espaciais, pode-se optar por definir *clusters* geograficamente baseados sobre quão próximas as diferentes áreas estão localizadas;

- Amostra estratificada: se $\mathcal{D}$ for dividido em partes mutuamente disjuntas chamadas estratos, uma amostra estratificada de $\mathcal{D}$ é gerada pela obtenção de uma amostra aleatória simples em cada estrato. Isto ajuda a garantir uma amostra representativa, especialmente quando os dados são distorcidos.

Uma das vantagens da utilização da amostragem de dados, na redução de numerosidade, é que o custo de obtenção de uma amostra é proporcional ao tamanho da amostra, s, em oposição a m, o tamanho do conjunto de dados (Chakrabarti et al., 2008). Assim, a complexidade amostral é sublinear ao tamanho do conjunto de dados e não há necessidade de realizar uma passagem completa de $\mathcal{D}$ para tomar decisões de modo a incluir ou não uma determinada tupla no subconjunto amostrado (García et al., 2015). Outras técnicas podem exigir pelo menos uma passagem completa por $\mathcal{D}$ (Chakrabarti et al., 2008).

A técnica *binning*, utilizada na compressão de dados, é o processo de converter uma variável contínua em um conjunto de intervalos (García et al., 2015). Nesta técnica, os valores selecionados são distribuídos em vários *bins*, compartimentos, de igual tamanho ou frequência, e, então, cada valor em um *bin* pode ser substituído pela média ou mediana dos valores do *bin*, ou cada valor em um *bin* pode ser substituído pela média ou mediana de cada *bin* (Chakrabarti et al., 2008). Ambas podem ser aplicadas recursivamente aos compartimentos resultantes para gerar conceitos de hierarquia (Chakrabarti et al., 2008). Então, cada compartimento pode ser tratado como categorias, com opção de impor ordem a eles (García et al., 2015). Por exemplo, pode-se aplicar a técnica a uma variável que representa a renda anual de um cliente em reais, onde pode ser gerado os intervalos 0 a 5.000, 5.001 a 10.000, 10.001 a 15.000 (García et al., 2015). Tais intervalos poderiam permitir a análise das faixas de renda, na qual os clientes da primeira faixa tem menos possibilidade de obter um empréstimo do que os clientes da última faixa (García et al., 2015).

Transformação dos Dados

Diferentes atributos podem ter diferentes intervalos de valores. Muitas vezes, a discrepância ou a magnitude dos intervalos de valores podem fazer com que os algoritmos de extração de padrões não convirjam ou obtenham padrões espúrios (Tan et al., 2014). Diante disso, podemos aplicar a transformações nos valores dos atributos.

Um tipo de transformação comumente utilizado é a normalização, a qual visa fazer com que um conjunto de valores tenha uma propriedade particular, como, por exemplo, suponha dois atributos, idade e renda, sendo possível notar diferença significativa em valores absolutos na renda entre as pessoas (em centena ou milhares de reais) (Tan et al., 2014). Se for realizado a comparação dos atributos, a comparação será dominada pelas diferenças de renda, isto é, os valores de renda dominarão os cálculos. Sendo assim, pode-se utilizar a fórmula $x' = (x - \bar{x})/\sigma_x$ para transformar o atributo renda e obter novos valores que terão média 0 e desvio padrão 1, em que $\bar{x}$ é a média dos valores do

atributo, σ_x é o desvio padrão, e x é o valor a ser transformado (Tan et al., 2014). Entretanto, vale ressaltar que a normalização pode obter efeitos indesejados se os atributos possuírem valores *outliers*, visto que a média e desvio padrão são afetadas por esses valores (Albon, 2018). Nesse cenário, a fórmula utilizada anteriormente na transformação é modificada, sendo a média substituída pela mediana, e o desvio padrão substituído pelo desvio padrão absoluto, isto é, $\sigma_A = \sum_{i=1}^{n} |x_i - \mu|$, em que x_i é o i-ésimo valor do atributo, n o número de amostras, μ a média ou mediana e A é uma variável aleatória (Tan et al., 2014).

Outro tipo de normalização comumente utilizado é a *min-max*, que consiste em realizar a transformação linear nos dados originais para ficarem em um pequeno intervalo, como [0.0, 1.0]. Segundo Chakrabarti et al. (2008), a técnica mapeia um valor numérico, v, de um atributo A para v' no intervalo $[new\_min_A, new\_max_A]$ utilizando a equação:

$$v' = \frac{v - min_A}{max_A - min_A}(new\_max_A - new\_min_A) + new\_min_A, \tag{2.4}$$

em que min_A e max_A são os valores originais de um atributo, A, mínimo e máximo, respectivamente (García et al., 2015). O intervalo $[-1.0, 1.0]$ também é tipicamente utilizado para dados já centrados em zero ou dados esparsos (Pedregosa et al., 2011). Para isso, basta substituir na Equação 2.4 o novo intervalo $[new\_min_A, new\_max_A]$ por $[-1.0, 1.0]$, respectivamente. Esse tipo de normalização é muito comum naqueles conjuntos de dados que estão sendo preparados para serem utilizados com métodos de aprendizado baseados em distâncias, qual evitará que atributos com uma grande diferença entre o mínimo e máximo dominem sobre os outros no cálculo da distância (García et al., 2015).

2.1.2 Extração de Padrões

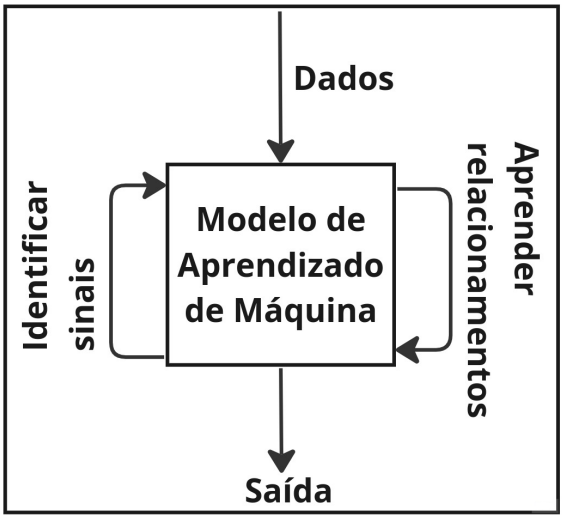
A extração de padrões visa construir modelos, para descoberta de padrões nos dados e representar o conhecimento embutido nos dados. Esta etapa é apoiada pela área de Aprendizado de Máquina (AM), do inglês *Machine Learning (ML)* (Witten e Frank, 2005; Parmezan et al., 2012). Na qual, são utilizadas ferramentas computacionais capazes de criar por si próprias uma indução de uma hipótese para um problema utilizando-se de experiências passadas. De acordo com Mitchell (1997):

> *"Diz-se que um programa de computador aprende com a experiência E no que diz respeito a alguma classe de tarefas T a medida de desempenho P, se seu desempenho nas tarefas em T, conforme medido por P, melhora com a experiência E."*

Segundo Ozdemir (2016), o AM preocupa-se com a capacidade de determinar certos padrões dos dados, mesmo que os dados tenham erros inerentes ou ruídos. Para isso, os modelos utilizados no AM não são informados sobre a melhor solução e, em vez disso, recebem vários exemplos do problema e procuram encontrar e/ou explorar relacionamentos entre os dados, para então usar essas relações na previsão de observações futuras ou generalizar os dados para revelar padrões interessantes.

Os distintos algoritmos de AM possuem suas especificidades, explorando diferentes partes da matemática e da ciência da computação (Ozdemir, 2016). No entanto, geralmente, é seguido um fluxo de aprendizado pelos algoritmos de AM, no qual se inicia com o recebimento dos dados, encontrando relacionamentos nos dados, e fornecendo como saída o modelo aprendido, conforme ilustrado na Figura 2.4:

Figura 2.4: Fluxo aprendizado de máquina.



Fonte: Adaptado de Ozdemir (2016).

O AM é separado em quatro tipos diferentes, conforme o critério do grau de supervisão presente nos dados: aprendizado supervisionado, no qual um conjunto de n exemplos (ou casos) $(a_1, c_1), ..., (a_n, c_n)$ são rotulados com os valores c; aprendizado não supervisionado, no qual não há nenhum rótulo definido para o conjunto de exemplos; aprendizado semi-supervisionado, em que apenas uma pequena parte do conjunto de exemplos está rotulado; e aprendizado por reforço, no qual não se utiliza de rótulo, mas sim de interações com o ambiente que melhore seu desempenho dado quão bem a ação foi medida por uma função de recompensa (Parmezan et al., 2012; Raschka, 2015). Dados os objetivos do trabalho e a disponibilidade de rótulos em todos os exemplos, o tipo de aprendizado utilizado neste trabalho foi o supervisionado, o qual será detalhado na próxima subseção.

Aprendizado Supervisionado

A supervisão, no aprendizado supervisionado, vem dos exemplos rotulados utilizados nos dados de treinamento (Han et al., 2011), isto é, os exemplos contêm uma classe de saída já conhecida. Dado que este trabalho fará uso exclusivo de aprendizado supervisionado, menções ao aprendizado de máquina daqui em diante irão se referir ao aprendizado de máquina supervisionado.

Os algoritmos de AM realizam buscas através do espaço de hipóteses visando encontrar o melhor modelo dadas restrições de espaço/tempo. Para tal, considere pares de exemplo $(\mathbf{x}_1, y_1), ...(\mathbf{x}_n, y_n)$, na qual $\mathbf{x}_i$ representa o vetor de características do i-ésimo exemplo, y_i representa o rótulo do i-ésimo exemplo, e considere que existe uma função f que tendo como entrada $\mathbf{x}_i$ consiga retornar y_i, ou seja, $y_i = f(\mathbf{x}_i)$. Dado isso, o objetivo do AM é encontrar uma função h (hipótese) que se aproxime da função f, isto é, $h \approx f$ (Russel e Norvig, 2010). Para medir a adequação da hipótese encontrada é utilizado um conjunto de exemplos de testes, sendo um conjunto distinto do utilizado no treinamento (Russel e Norvig, 2010).

Conforme o tipo do atributo y, pode-se ter: (i) a tarefa de classificação, caso os valores de y sejam discretos, e (ii) a tarefa de regressão, caso os valores de y sejam contínuos. Dados os objetivos e tipo do atributo referente ao rótulo utilizado neste trabalho, o tipo de tarefa a ser utilizada será a tarefa de classificação.

Classificação

Para uma tarefa em que podemos ter como entrada x e y, sendo x observações independentes e y uma classe pertencente a x, no qual y possui valores discretos, denominamos como classificação. Por exemplo, um funcionário de um banco precisa analisar os dados de um cliente para saber se é seguro ou arriscado realizar um empréstimo. Utilizando uma tarefa de classificação, é previsto para o funcionário se é "seguro" ou "arriscado" para o pedido de empréstimo (Han et al., 2011).

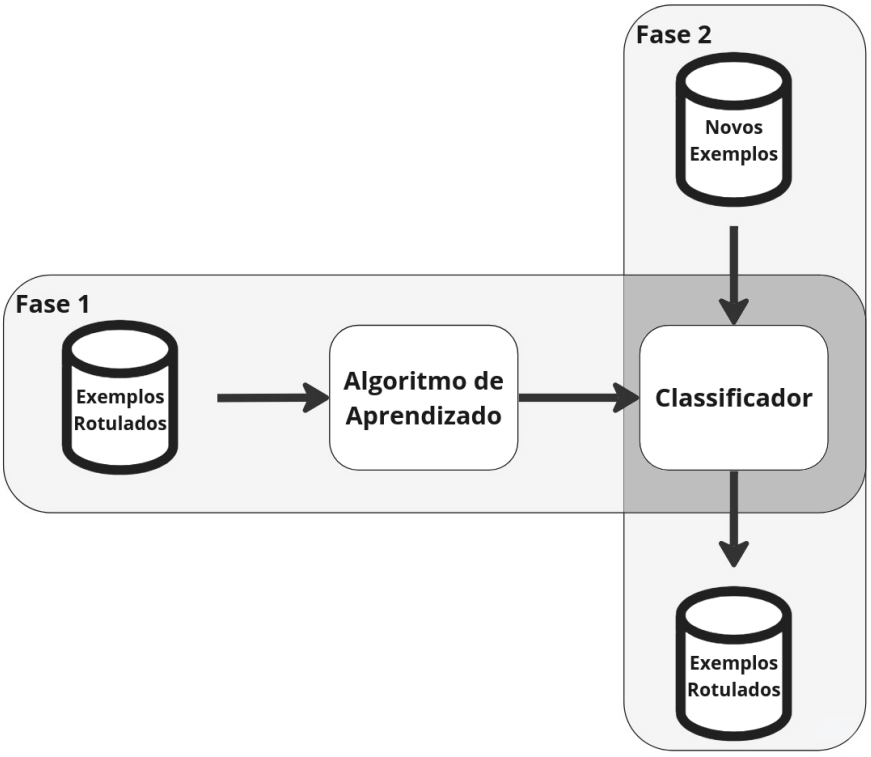
Formalmente, a classificação é defina por Zaki e Jr. (2014) como um modelo ou função f que prevê o rótulo da classe y para dado um exemplo de entrada $\mathbf{x}$, ou seja, $y = f(\mathbf{x})$, em que $y \in \{c_1, c_2, ..., c_k\}$ e cada c_i é um rótulo de classe (um valor de atributo categórico).

A classificação possui denominações, conforme o valor de y. Se y for um valor dentre apenas duas possibilidades, a classificação é denominada binária. Se possuir um valor dentre três ou mais valores, a classificação é denominada multi-classe. Por fim, se um valor de y for composto por um subconjunto de valores, a classificação é denominada multi-rótulo (Tan et al., 2014).

Os algoritmos de classificação, normalmente, realizam o processo em duas

fases, como demonstrado na Figura 2.5. A fase de treinamento (Fase 1), consiste na etapa de aprendizado na qual um algoritmo de classificação constrói um modelo (Classificador) a partir de exemplos de treinamento com seus rótulos de classes associados (Exemplos Rotulados); e a fase de teste (Fase 2), consiste na etapa de classificação na qual o modelo construído é utilizado para prever rótulos a exemplos de teste não rotulados (Novos Exemplos), que não foram usados na construção do modelo (Aggarwal, 2014; Han et al., 2011). Ainda, é na fase de teste (Fase 2) que algumas medidas preditivas do modelo são estimadas. Se os exemplos utilizados fossem os mesmos exemplos de treinamento, provavelmente o modelo sofreria de superajuste dos dados, também conhecido como *overfit*, no qual o modelo pode incorporar algumas anomalias particulares dos dados de treinamento que não estão presentes em todo conjunto de dados, influenciando diretamente nas medidas utilizadas (Han et al., 2011).

Figura 2.5: Fases do processo de classificação.



Fonte: Adaptado de Parmezan et al. (2012).

Como técnicas de classificação mais comumente utilizadas, podemos citar os classificadores baseados em instâncias, classificadores *naïve bayes*, classificadores de árvores de decisão, redes neurais, e máquinas de vetores de suporte (Tan et al., 2014; Aggarwal, 2014). Cada técnica possui sua especificidade de funcionamento, no entanto, possuem um objetivo chave de construir modelos com boa capacidade de generalização, ou seja, modelos que predizem

com precisão os rótulos de classe de registros anteriormente desconhecidos (Tan et al., 2014).

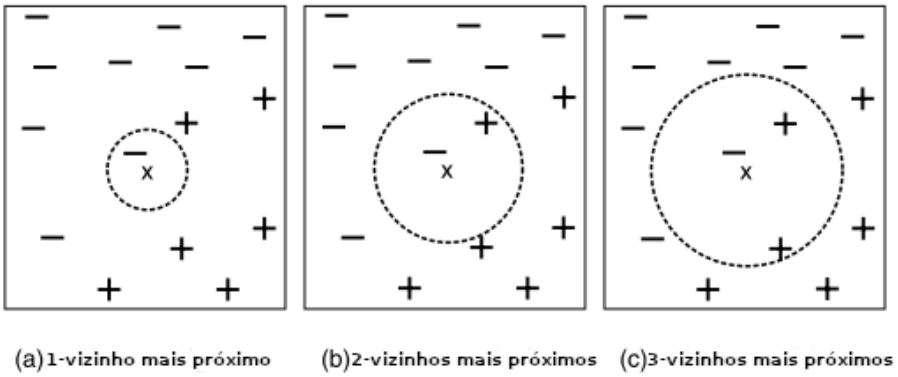
***k*-Vizinhos Próximos:** o algoritmo *k*-Vizinhos Próximos, do inglês *k-Nearest Neighbor (KNN)*, visa construir um modelo a partir do conjunto de vizinhos mais conhecidos para determinar o rótulo da classe (Aggarwal, 2014).

O *KNN* é um algoritmo baseado em instâncias que correspondem a pontos em um espaço n-dimensional $\Re^n$ (Mitchell, 1997). Apesar de simples e eficaz, é tido como um algoritmo lento, pois armazena os exemplos de treinamento para realizar a previsão de um novo exemplo desconhecido (Chakrabarti et al., 2008).

O k corresponde ao número de vizinhos mais próximos que serão consultados para classificar um novo exemplo. Que pode ser definido como: dada uma consulta x_q, encontrar os k exemplos mais próximos de x_q. Em que para a tarefa de classificação, é realizado a consulta por voto de pluralidade dos vizinhos mais próximos encontrados (Russel e Norvig, 2010).

A Figura 2.6 ilustra o exemplo para 1, 2, e 3 vizinhos mais próximos de um ponto de dados x localizado no centro do círculo. Na qual, podemos observar na Figura 2.6(a) que o 1 vizinho mais próximo do ponto de dados é um exemplo negativo, atribuído a classe negativa ao ponto de dados. Na Figura 2.6(b) os 2 vizinhos mais próximos do ponto de dados é um exemplo negativo e um exemplo positivo, sendo possível escolher aleatoriamente a classe do ponto de dados. Por fim, na Figura 2.6(c) os 3 vizinhos mais próximos do ponto de dados é um exemplo negativo e dois exemplos positivos, e utilizando a votação de pluralidade, é atribuído à classe positiva ao ponto de dados (Tan et al., 2014).

Figura 2.6: Exemplo de 1, 2, e 3 vizinhos mais próximos de uma instância.



Fonte: Adaptado de Tan et al. (2014).

Na votação de pluralidade, originalmente, todos os vizinhos têm o mesmo impacto na classificação, estando mais próximo ou mais distante. Tornando o algoritmo sensível à escolha de k. Uma maneira de reduzir o impacto de k é ponderar a influência de cada vizinho, de acordo com sua distância atribuindo um peso: $1/d$, em que d é a distância para o vizinho (Tan et al., 2014).

Para encontrar os vizinhos mais próximos de uma instância, normalmente, é utilizada a distância Euclidiana padrão. Formalmente, definida por Mitchell (1997) como: dada uma instância x que pode ser descrita por um vetor de características $\langle a_1(x), a_2(x), ...a_n(x)\rangle$ no qual $a_r(x)$ indica o valor r-ésimo atributo de x, então a distância entre duas instâncias x_i e x_j é dada por $d(x_i, x_j)$, em que

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}. \tag{2.5}$$

***Naïve Bayes*:** o classificador *Naïve Bayes*, ou *Bayes* ingênuo, é baseado no teorema de *Bayes* e é comumente utilizado para classificação binária e multiclasse na abordagem probabilística. É um classificador particularmente adequado quando a dimensionalidade das entradas é alta, e não tem dificuldade com dados ruidosos ou faltantes. Um dos seus pontos fortes é a suposição da independência dos atributos. Apesar de simples, pode muitas vezes obter desempenho comparável com alguns métodos de classificação sofisticados, como árvores de decisão e classificador de rede neural (Russel e Norvig, 2010; Aggarwal, 2014).

Segundo Mitchell (1997), a tarefa de aprendizagem é realizada pelo classificador, na qual cada instância x é descrita por uma conjunção de valores de atributos e na qual a função alvo $f(x)$ pode assumir qualquer valor de algum conjunto finito V. Em que, uma nova instância pode ser descrita por uma tupla de valores de atributo $\langle a_1, a_2...a_n\rangle$, e, para realizar a classificação, ou predição, é fornecido um conjunto de exemplos de treinamento a função alvo (Mitchell, 1997).

O classificador é dito ingênuo (*Naïve*), pois assume que o efeito de um valor de atributo de uma determinada classe alvo é independente dos valores de outros atributos (García et al., 2015). Assim, supõe-se que dado uma classe alvo da instância, a probabilidade de observar a conjunção $a_1, a_2...a_n$ é apenas o produto das probabilidades para os atributos individuais (Mitchell, 1997). Então a suposição de independência pode ser

formalmente declarada da seguinte forma:

$$\upsilon_{NB} = argmax_{\upsilon_j \in V} P(\upsilon_j) \Pi_i P(a_i|\upsilon_j) \tag{2.6}$$

em que V é o conjunto de classes, υ_j a j-ésima classe, a_i o valor do i-ésimo atributo, Π_i as multiplicações em ordem das i-probabilidades, $argmax$ a maior probabilidade prevista, e υ_{NB} a saída do valor alvo.

Uma característica importante a salientar sobre a suposição de independência, é que atributos correlacionados podem degradar o desempenho do classificador (Tan et al., 2014). Isto é, uma classificação entre dois atributos correlacionados pode ser feita de forma errônea. Isso pode acontecer porque a suposição de independência condicional dos atributos não é mais válida (Tan et al., 2014). No entanto, o classificador tem um desempenho surpreendentemente bom em dados em que essa suposição não se sustenta (Brownlee, 2018).

Para calcular a probabilidade $P(a_i|\upsilon_j)$ na equação 2.6, é necessário verificar se o a_n atributo é categórico ou contínuo. Lembrando que, a_i se refere ao i-ésimo valor do a_n atributo. Se o a_n atributo é categórico, então a $P(a_i|\upsilon_j)$ é o número de exemplos da classe υ_j no conjunto de dados de treinamento com o valor a_i para o a_n, dividido pelo número exemplos da classe υ_j no conjunto de treinamento, ou seja, é realizada a contagem da frequência das várias combinações possíveis nos dados de treinamento (Han et al., 2011). Já se o a_n atributo é contínuo, então é assumido como tendo uma distribuição Gaussiana com uma a média μ e desvio padrão σ, definido por Han et al. (2011) como

$$g(a, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(a-\mu)^2}{2\sigma^2}}, \tag{2.7}$$

de modo que

$$P(a_i|\upsilon_j) = g(a_i, \mu_{\upsilon_j}, \sigma_{\upsilon_j}) \tag{2.8}$$

em que a_i é o i-ésimo valor do atributo a_n, e a média μ_{υ_j} e o desvio padrão σ_{υ_j} dos valores do atributo a_n para os exemplos de treinamento da classe υ_j (Han et al., 2011). Isto posto, em posse dos três valores podemos estimar a probabilidade $P(a_i|\upsilon_j)$ utilizando a equação 2.7 para valores contínuos.

O cálculo de probabilidade na equação 2.6 pode ser comprometido, se um possível valor de um atributo não ocorrer nos dados de treinamento em conjunto com cada valor de classe. Isto é, se ocorrer uma probabilidade nula, toda multiplicação será penalizada pelo valor zero, independente de quão grande as outras probabilidades sejam, o resultado será zero (Wit-

ten e Frank, 2005). Uma forma de solucionar o problema é com a técnica estimador de *Laplace*, que consiste em adicionar um ao numerador e q ao denominador na contagem de frequência, sendo q a contagem de vezes que um é adicionado ao numerador, garantindo que um valor de atributo que ocorra zero vezes receba uma probabilidade diferente de zero (Witten e Frank, 2005; Han et al., 2011).

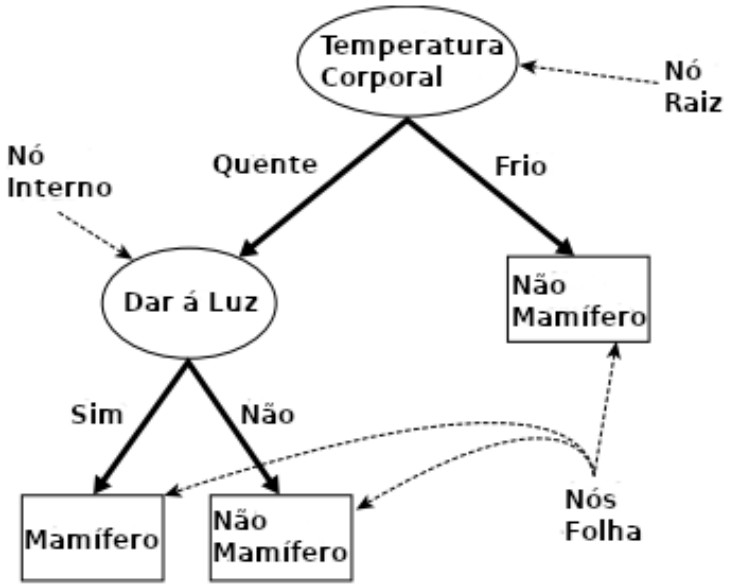
Árvores de Decisão: as árvores de decisão criam um particionamento hierárquico dos dados, que relaciona as diferentes partições no nível folha às diferentes classes. A abordagem geral é tentar dividir recursivamente os dados de treinamento de modo a maximizar a discriminação entre as diferentes classes em diferentes nós (Aggarwal, 2014). Ainda, segundo Mitchell (1997), uma árvore de decisão é um método para aproximar funções alvo de valores discretos, em que a função aprendida é representada por uma árvore de decisão. Estas árvores aprendidas também podem ser representadas como conjuntos de regres se-então para uma melhor representação e legibilidade humana (Mitchell, 1997).

Para exemplificar como funciona a classificação em uma árvore de decisão, considere que dado um animal vertebrado classificá-lo em mamífero e não mamífero (Tan et al., 2014). Utilizando o conjunto de regras da árvore de decisão ilustrada na Figura 2.7, para classificar um novo animal a primeira condição a ser analisada é a temperatura corporal, se é quente ou fria. Se for fria, o animal é classificado como não mamífero. Caso quente, outra condição é realizada verificando se o animal dá à luz, sim ou não. Se sim, o animal é classificado como mamífero, se não, o animal é classificado como não mamífero. Dessa forma, a classificação de um novo registro de teste é simples, visto que a árvore de decisão tenha sido construída.

Uma árvore de decisão possui estrutura como demonstrado na Figura 2.7: nó raiz, que não possui arestas de entrada e zero ou mais arestas de saída; nó interno, cada um com exatamente uma aresta de entrada e duas ou mais arestas de saída; e nós folha ou nós terminais, cada um com exatamente com uma aresta de entrada e nenhuma aresta de saída (Tan et al., 2014). São os nós terminais, ou nós folha, que recebem os rótulos das classes. Já nos nós não terminais, são quais contêm as condições de teste de atributo para separar registros que tenham diferentes características (Tan et al., 2014).

A construção de uma árvore de decisão é realizada a partir dos atributos do conjunto de dados de treinamento. Sendo assim, podem existir diversas árvores de decisão que podem ser induzidas a partir de seus

Figura 2.7: Representação de uma árvore de decisão para classificação de animais mamíferos.



Fonte: Adaptado de Tan et al. (2014).

atributos. Dessa forma, encontrar uma árvore ótima é computacionalmente inviável devido ao tamanho exponencial no espaço de busca (Tan et al., 2014). Entretanto, na busca de induzir uma árvore em tempo razoável, embora sub-ótima, alguns eficientes algoritmos utilizam a estratégia gulosa, que toma uma série de decisões localmente ótimas sobre qual atributo usar para particionar os dados (Tan et al., 2014).

Na busca de escolher o melhor atributo na construção de uma árvore de decisão, algumas medidas podem ser utilizadas para determinar a melhor maneira de se dividir os exemplos de treinamento. Uma medida utilizada para determinar o desempenho de um particionamento de dados é o ganho, do inglês *gain*. O ganho compara o grau de impureza do nó pai (antes da divisão) com o grau de impureza dos nós filhos (após a divisão) (Tan et al., 2014). Sendo que quanto maior a diferença, maior o ganho, melhor é o particionamento (Witten e Frank, 2005). A medida é dada pela seguinte equação:

$$\Delta = I(\text{pai}) - \sum_{j=1}^{k} \frac{N(\upsilon_j)}{N} I(\upsilon_j), \tag{2.9}$$

em que $I(\cdot)$ é a medida de impureza de um determinado nó, N o número total de exemplos no nó pai, k o número de valores de atributos e $N(\upsilon_j)$ é o número de exemplos associados ao nó filho, υ_j. O processo de escolha

de um atributo, para a construção de uma árvore de decisão, é aplicado recursivamente, até que todos os nós folhas sejam puros ou quando não for mais possível realizar a divisão dos dados (Witten e Frank, 2005)

As medidas de impureza, utilizada na Equação 2.9, fornecem uma maneira eficaz de avaliar a qualidade de um nó em seu nível de discriminação entre as diferentes classes (Aggarwal, 2014). Ainda, segundo Witten e Frank (2005) são utilizadas para calcular quais nós filhos de um atributo são mais puros. Quanto menor o grau da impureza, maior a diferença na distribuição das classes. Por exemplo, dado um nó com distribuição de classe (0, 1) tem zero de impureza, enquanto dado um nó com distribuição de classe uniforme (0.5, 0.5) tem a maior impureza (Tan et al., 2014). Alguns exemplos de medidas de impureza é dado por Tan et al. (2014):

$$\textit{Entropy(t)} = -\sum_{i=0}^{c-1} p(i|t)\log_2 p(i|t), \tag{2.10}$$

$$\textit{Gini(t)} = 1 - \sum_{i=0}^{c-1} [p(i|t)]^2, \tag{2.11}$$

$$\textit{Classification error(t)} = 1 - \max_i [p(i|t)], \tag{2.12}$$

em que $p(i|t)$ é a fração de exemplos pertencentes à classe i em um dado nó t, c é o número de classes e $0\log_2 0 = 0$ para o cálculo da entropia (*Entropy*).

Uma árvore de decisão pode aprender muito com os dados de treinamento, utilizados durante sua criação. Logo, a árvore pode sofrer de *overfitting* e não generalizar bem para dados não conhecidos. Uma estratégia adota para evitar *overfitting* é a pós-poda, que consiste em realizar a poda de uma árvore após sua construção. Duas operações podem ser utilizadas na pós-poda: substituição de subárvore, que consiste em selecionar algumas subárvores e substituir por uma folha simples; e elevação de subárvore, que consiste em "levantar" uma subárvore para substituir outra que está acima (Witten e Frank, 2005).

Florestas Aleatórias: as Florestas Aleatórias, em inglês *Random Forests (RF)*, é um tipo de *ensemble*, que aprende um conjunto de modelos e os usa em combinação buscando aumentar o desempenho preditivo (Witten e Frank, 2005). O algoritmo utiliza a abordagem *Bootstrap aggregation (Bagging)* para construir múltiplas árvores de decisão, descritas na seção 2.1.2, utilizadas como modelos-base, que irão formar a floresta de árvores. O *Bagging* é um método muito popular da técnica de *ensemble*, que adota o mecanismo de amostragem *Bootstrap* na construção dos

modelos-base.

O procedimento de amostragem *Bootstrap* gera novos conjuntos de dados por amostragem a partir do conjunto de dados original, aplicando substituição e treina os modelos-base (neste caso árvores de decisão) nos conjuntos de dados amostrados. Segundo Aggarwal (2014) o procedimento pode ser realizado da seguinte forma: dado um conjunto de treinamento $\mathcal{D} = \{x_i, y_i\}_{i=1}^{m}$, em que $x_i \in \mathbb{R}^n$, $y_i \in \mathcal{Y}$ e m é o tamanho do conjunto de dados de treinamento, os dados de $\mathcal{D}$ podem ser amostrados com substituição para formar um novo conjunto de dados $\mathcal{D}'$, mantendo o tamanho igual ao de $\mathcal{D}$. Em razão da amostragem, alguns dos exemplos aparecem mais de uma vez em $\mathcal{D}'$, enquanto alguns exemplos em $\mathcal{D}$ não aparecem em $\mathcal{D}'$ (Aggarwal, 2014).

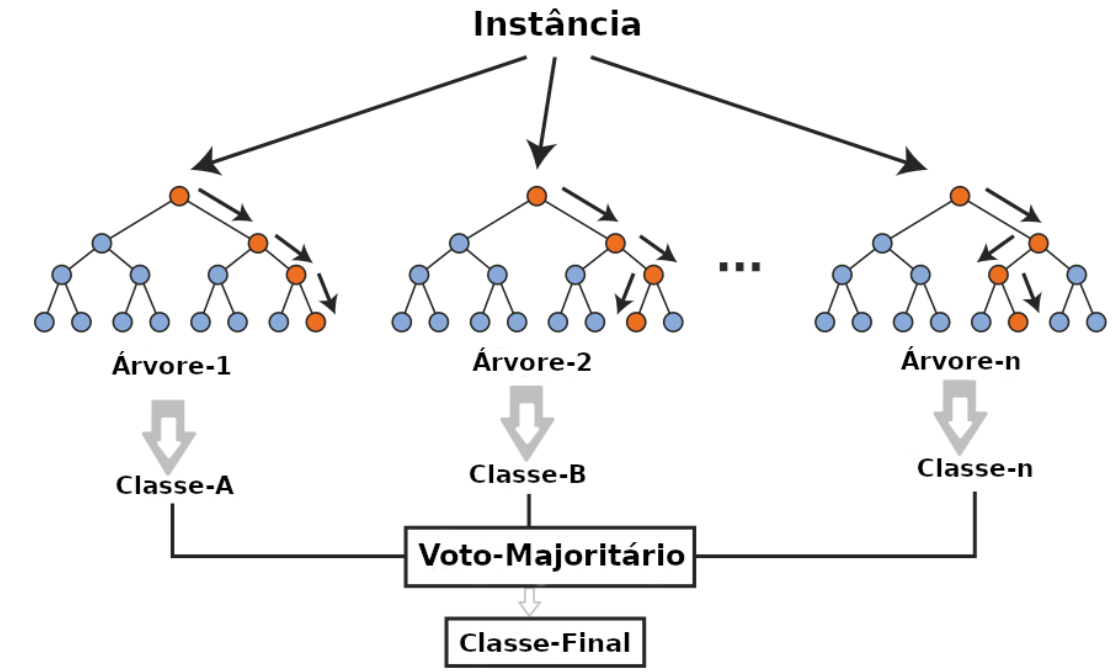
Para promover a diversidade entre as árvores de decisão, além de utilizar o *Bagging*, o *RF* incorpora a seleção aleatória do espaço de atributos na construção do conjunto de treinamento. Isto é, no processo de construção de uma árvore de decisão para selecionar o melhor atributo em cada nó, o *RF* considera apenas um subconjunto aleatório n' do espaço de atributos n (Aggarwal, 2014; Han et al., 2011).

Dessa forma, por exemplo, pode-se utilizar o *RF* para treinar um classificador em cada um dos conjuntos de dados $\mathcal{D}'$ e combinar as previsões feitas pelas múltiplas árvores de decisões para gerar a previsão final utilizando o esquema de votação majoritária, como pode-se observar Figura 2.8 que ilustra a estrutura de uma floresta aleatória para classificação (Tan et al., 2014). Sendo assim, para cada exemplo x_i, sua previsão final é o rótulo da classe com maior número de previsões feitas pelos classificadores (Aggarwal, 2014). Devido à diversidade e aleatoriedade do *RF*, o modelo tem um bom desempenho, pois possui baixo *overfitting*, consegue generalizar para novos dados, e é robusto a erros e a *outliers* (Aggarwal, 2014; Han et al., 2011).

***Extreme Gradient Boosting*:** o *Extreme Gradiente Boosting (XGBoost)* é baseado no conceito de *Boosting*, qual é um tipo de *ensemble*. A técnica de *Boosting* combina classificadores em sequência, de forma que ao utilizar a previsão de $n+1$ classificadores, a classificação seja mais acurada que a de n classificadores, ou seja, o próximo classificador adicionado na sequência visa corrigir os erros dos classificadores anteriores (Aggarwal, 2014).

O *XGBoost* utiliza o modelo de conjuntos de árvores de decisão, mais especificamente um conjunto de árvores de classificação e regressão (também conhecido como *CART*, em inglês *Classification And Regression Tree*),

Figura 2.8: Representação de uma floresta aleatória para classificação.



Fonte: Adaptado de Haroon (2017).

como classificadores base para a técnica de *Boosting*. Diferente das árvores de decisão, cada *CART* contém uma pontuação real associada a cada uma das folhas (Chen e Guestrin, 2016).

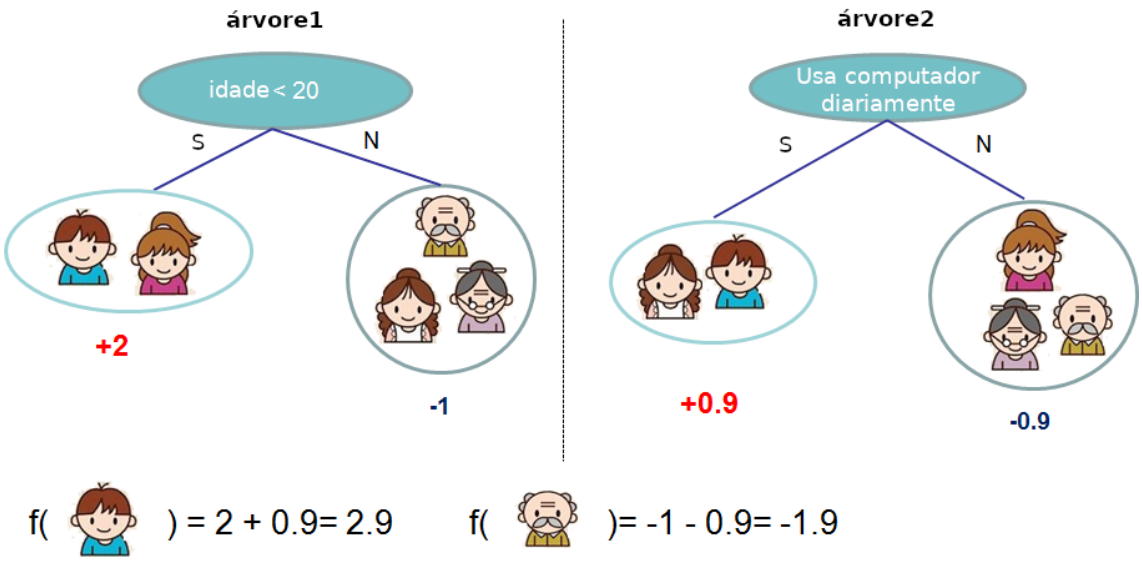
Por exemplo, dado um conjunto de dados de m exemplos e n atributos $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$ $(|D| = n, \mathbf{x}_i \in \mathbb{R}^m, y_i \in \mathbb{R})$, um modelo de conjuntos de árvores usa k funções aditivas para prever uma saída, como demonstrado na parte inferior da Figura 2.9, qual ilustra duas árvores *CART* com a respectiva pontuação de cada folha da árvore criada. Esta operação é matematicamente dada por Chen e Guestrin (2016) como:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), f_k \in \mathcal{F}, \tag{2.13}$$

em que $\mathcal{F} = \{f(\mathbf{x}) = w_{q(\mathbf{x})}\}$ $(q : \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ é o espaço das árvores de regressão (*CART*). Aqui, o q representa a estrutura de cada árvore que mapeia um exemplo até o índice folha correspondente, e o T é o número de folhas em uma árvore. Sendo que cada f_k corresponde a uma estrutura de árvore independente q e ao peso w, onde w_i representa a pontuação na i-ésima folha (Chen e Guestrin, 2016).

Para aprender o conjunto de funções, a função objetivo combina uma função de perda e um termo de regularização. A seguinte função regula-

Figura 2.9: Modelo de conjunto de árvore *CART*. A previsão final para cada exemplo da árvore é a soma das previsões de cada árvore.



Fonte: Adaptado de Chen e Guestrin (2016).

rizada é dada por Chen e Guestrin (2016) na seguinte forma:

$$\begin{aligned} \mathcal{L}(\phi) &= \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \\ \text{onde } \Omega(f) &= \gamma T + \frac{1}{2}\lambda ||w||^2, \end{aligned} \tag{2.14}$$

l é uma função de perda convexa diferenciável que mede a diferença entre a previsão $\hat{y}_i$ e o alvo y_i. Já o segundo termo, Ω, penaliza a complexidade do modelo, ajudando a suavizar os pesos finais aprendidos para evitar o ajuste excessivo. Dessa forma, o objetivo regularizado tenderá a selecionar um modelo empregando funções simples e preditivas (Chen e Guestrin, 2016).

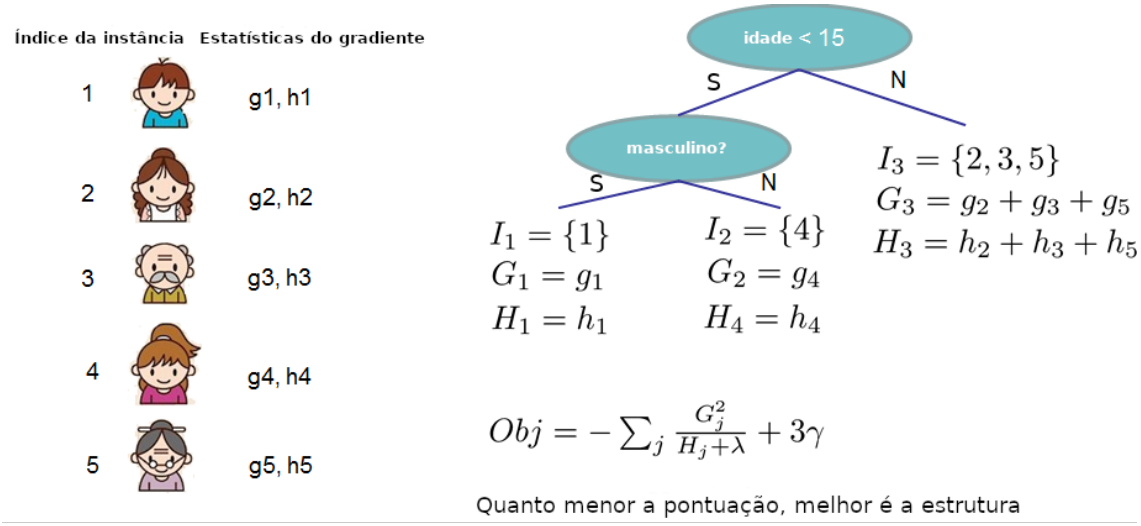
O modelo é treinado de forma aditiva, isto é, a cada iteração t é adicionada uma nova árvore de forma que otimize o objetivo. Para medir a qualidade de uma estrutura de árvore q é derivada uma pontuação, dada por Chen e Guestrin (2016) pela equação:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2}\sum_{j=1}^{T}\frac{(\sum_{i\in I_j} g_i)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T, \tag{2.15}$$

onde $g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ e $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ são a estatística do gradiente e do gradiente de segunda ordem, respectivamente. $I_j = \{i|q(x_i) = j\}$ é o conjunto de índices de pontos de dados atribuídos a j-ésima folha. Essa pontuação é como uma pontuação de impureza para avaliar ár-

vores de decisão, exceto pelo fato de ser derivada para uma gama mais ampla de funções objetivas. A Figura 2.10 ilustra como a pontuação pode ser calculada: sendo necessário apenas calcular a soma estatísticas dos exemplos em cada nó folha e, em seguida, aplicar a Equação 2.15 para obter a qualidade da árvore (Chen e Guestrin, 2015).

Figura 2.10: Estrutura para o cálculo da pontuação da qualidade da árvore.



Fonte: Adaptado de Chen e Guestrin (2016).

Tendo uma maneira de medir a qualidade de uma árvore q, o ideal seria enumerar todas as árvores possíveis e escolher a melhor. No entanto, na prática, isso é intratável, então é realizado a otimização de um nível de árvore por vez. Mais especificamente, tenta-se dividir uma folha em duas. Por exemplo, supondo que I_E e I_D sejam conjuntos de instâncias dos nós esquerdos e diretos após a divisão. Sabendo-se que $I = I_E \cup I_D$, então, a redução de perda após a divisão é dada por (Chen e Guestrin, 2016):

$$\mathcal{L}_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_E} g_i)^2}{\sum_{i\in I_E} h_i + \lambda} + \frac{(\sum_{i\in I_D} g_i)^2}{\sum_{i\in I_D} h_i + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma, \tag{2.16}$$

a pontuação obtida, na prática, é geralmente utilizada para avaliar os candidatos divididos. Pode-se observar que se a pontuação obtida for menor que γ, seria melhor não adicionar o ramo (Chen e Guestrin, 2016).

Por fim, outras duas importantes técnicas são utilizadas para evitar o *overfitting*. A primeira técnica dimensiona os pesos recém-adicionados por um fator η após cada etapa de aumento da árvore, que se assemelha à taxa de aprendizado utilizada na otimização estocástica, reduzindo a influência de cada árvore e deixando espaço para futuras árvores melhorarem o modelo. A segunda técnica é a subamostragem de atributos

(colunas), que além de evitar o *overfitting*, também acelera os cálculos do algoritmo (Chen e Guestrin, 2016).

Redes Neurais Artificiais: as Redes Neurais Artificias (RNA), em inglês *Artificial Neural Network (ANN)*, tentam simular sistemas naturais biológicos, correspondentes ao cérebro humano. A arquitetura mais básica de uma rede neural artificial é um perceptron, que contém um conjunto de nós de entrada, um nó de saída, com conexão entre eles (Aggarwal, 2014). Cada nó de entrada é conectado ao nó de saída por um link ponderado, utilizado para emular a força da conexão (Tan et al., 2014).

Um perceptron, ou também neurônio, é um mecanismo computacional que transforma um conjunto de entradas $x = (x_1, ..., x_n)$ em uma única saída $\hat{y}$ (Aggarwal, 2014). Qual, no contexto deste tópico de RNA, o $\hat{y}$ se refere a saída prevista pelo algoritmo. Para produzir a saída $\hat{y}$ é utilizado a composição de duas funções, como podemos observar na Figura 2.11. Primeiro é utilizada a função de valor líquido ξ, que utiliza os parâmetros ou pesos w da unidade para resumir os dados de entrada em um valor líquido v, dada como $v = \xi(x, w)$, e normalmente assume a forma de uma soma ponderada, uma distância ou *kernel* (Aggarwal, 2014). Depois é acionada a função de ativação ϕ, que transforma o valor líquido v no valor de saída da unidade $\hat{y}$, dada como $\hat{y} = \phi(v)$, sendo ela que decidi disparar ou inibir os sinais, dependendo da função escolhida (Aggarwal, 2014). Umas das funções de ativação mais utilizadas são: linear, degrau e sigmoide.

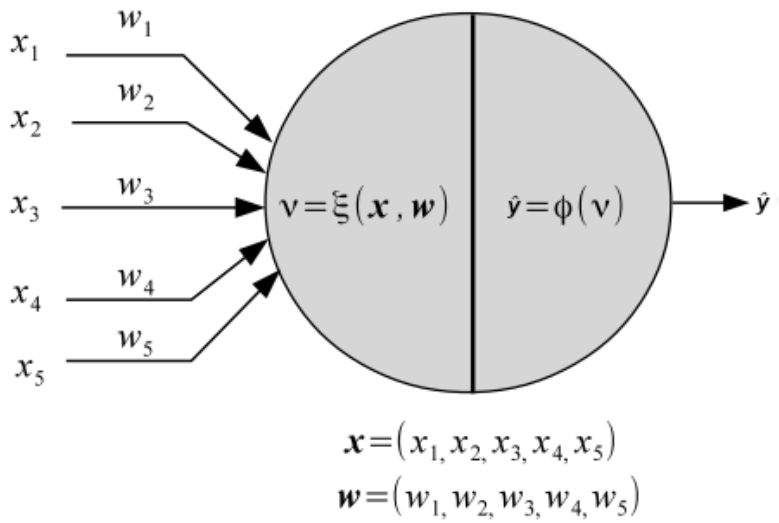
Durante a fase de treinamento de um modelo perceptron, os parâmetros de peso w são ajustados até que as saídas do perceptron se tornem consistentes com as saídas reais dos exemplos de treinamento (Tan et al., 2014). A fórmula para ajuste dos pesos é dada por Tan et al. (2014) como

$$w_j^{(k+1)} = w_j^{(k)} + \lambda(y_i - \hat{y}_i^{(k)})x_{ij}, \tag{2.17}$$

em que $w^{(k)}$ é o parâmetro peso associado com o i-ésimo link de entrada após a k-ésima iteração, λ é o parâmetro conhecido como taxa de aprendizado, do inglês *learning rate*, e x_{ij} é o valor do j-ésimo atributo do exemplo de treinamento x_i.

Intuitivamente, pode-se perceber na fórmula 2.17 que o novo peso, $w_j^{(k+1)}$, é uma combinação do antigo peso, $w_j^{(k)}$, e um termo proporcional ao erro de predição, $(y - \hat{y})$, que se refere a diferença entre a saída desejada e a saída real (Tan et al., 2014). Ainda, a taxa de aprendizado, λ, é utilizada para controlar a quantidade de ajustes dos pesos feitos nos links a cada iteração, para que os pesos não sejam alterados drasticamente a ponto

Figura 2.11: Modelo matemático de um perceptron.



Fonte: Adaptado de Aggarwal (2014).

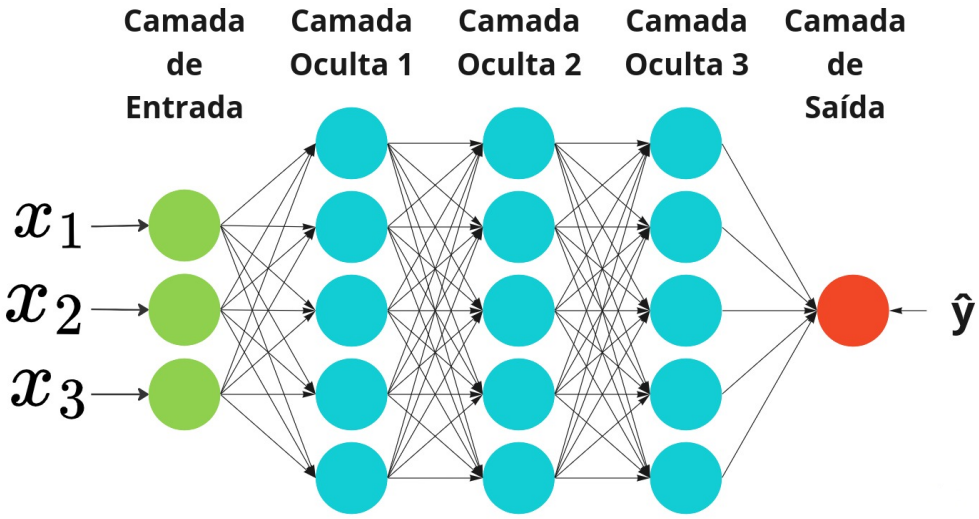
de modificar iterações anteriores (Tan et al., 2014).

O modelo perceptron são classificadores poderosos. No entanto, individualmente, é garantido a convergência em direção a uma solução ótima (desde que a taxa de aprendizado seja suficientemente pequena) se existir um hiperplano de separação, ou seja, para problemas de classificação linearmente separáveis (Witten e Frank, 2005; Tan et al., 2014). O modelo não consegue lidar com padrões não lineares ou mais complicados.

Para lidar com problemas não linearmente separáveis e relacionamentos mais complexos, entre as variáveis de entrada e saída, é utilizada a RNA com várias camadas de perceptrons (neurônios), entre a entrada e saída da rede. A Figura 2.12 demonstra um exemplo de uma estrutura de uma RNA multicamadas, composta pela camada de entrada, as camadas intermediárias, também conhecidas como camadas ocultas, e os nós embutidos nessas camadas são conhecidos como nós ocultos, e a camada de saída. Esta estrutura é chamada *feed-forward*, na qual os nós em uma camada são conectados apenas aos nós na próxima camada (Tan et al., 2014). Já em estrutura de RNA recorrente, os links podem conectar nós em uma mesma camada ou nós de uma camada às camadas anteriores (Tan et al., 2014).

A estrutura da RNA multicamadas apresentada é conhecida como um dos métodos de aprendizado profundo, em inglês *deep learning*, que visam aprender e explorar as habilidades das redes neurais para gerar

Figura 2.12: Exemplo de uma rede neural multicamadas.



Fonte: Adaptado de IZBICKI e SANTOS (2020).

representações das características internas dos dados dentro de suas camadas ocultas. Mais camadas ocultas e mais nós nessas camadas geram uma rede mais poderosa com a capacidade de gerar representações internas mais significativas dos dados (Aggarwal, 2014).

Em uma RNA multicamadas, não é possível utilizar a abordagem de treinamento de um perceptron, pois, não se tem conhecimento inicialmente sobre as verdadeiras saídas dos nós ocultos para determinar o cálculo de erro da predição, utilizado na fórmula 2.17, associado com cada nó oculto (Witten e Frank, 2005; Tan et al., 2014). Isto posto, temos como principal objetivo de uma RNA multicamada de determinar um conjunto de pesos w que minimize uma função de erro (Aggarwal, 2014). Uma função amplamente utilizada é o total da soma dos erros ao quadrado, dada por Tan et al. (2014) como

$$E(w) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2. \tag{2.18}$$

em que $\hat{y}_i$ é a previsão da rede obtida na i-ésima unidade de saída e y_i é o rótulo de classe da i-ésima instância. Sendo importante destacar que a soma dos erros ao quadrado depende do peso w porque a classe prevista $\hat{y}$ é uma função dos pesos atribuídos aos nós ocultos e de saída, sendo que, geralmente, a função de ativação utilizada na saída de uma RNA é não linear (Tan et al., 2014).

Como as saídas corretas para os nós das camadas ocultas são desconhecidas, uma solução, a grosso modo, é modificar os pesos das conexões

que levam aos nós ocultos com base na força de contribuição de cada nó para previsão final (Witten e Frank, 2005). Para isso, é utilizado o método gradiente descendente, que explora as informações fornecidas pela derivada da função que deve ser minimizada (neste caso, a função erro), até que um mínimo seja alcançado (Witten e Frank, 2005). Pensando nisso, algoritmos gulosos foram desenvolvidos baseados no método, na qual a fórmula utilizada para atualização de peso é dada da seguinte forma por Tan et al. (2014):

$$w_j \longleftarrow w_j - \lambda \frac{\partial E(w)}{\partial w_j}, \tag{2.19}$$

em que λ é a taxa de aprendizado. Como o gradiente descendente requer derivações, a função de ativação utilizada deve ser não linear (Witten e Frank, 2005).

O cálculo de atualização de pesos, para nós na camada oculta, não é trivial, pois é difícil avaliar o seu termo de erro sem saber quais devem ser seus valores de saída (Tan et al., 2014). Para isso, é utilizada uma técnica chamada de retropropagação, do inglês *backpropagation*, que consiste em duas etapas: a primeira etapa, é o avanço da rede, na qual as saídas dos neurônios na camada k são computados antes de computar as saídas na camada $k+1$; a segunda etapa, é o retrocesso da rede, é quando a fórmula de atualização do peso é aplicada no sentido inverso, isto é, no qual os pesos na camada $k+1$ são atualizados antes que os pesos da camada k (Tan et al., 2014). Permitindo utilizar os erros dos neurônios da camada $k+1$ para estimar os erros dos neurônios da camada k.

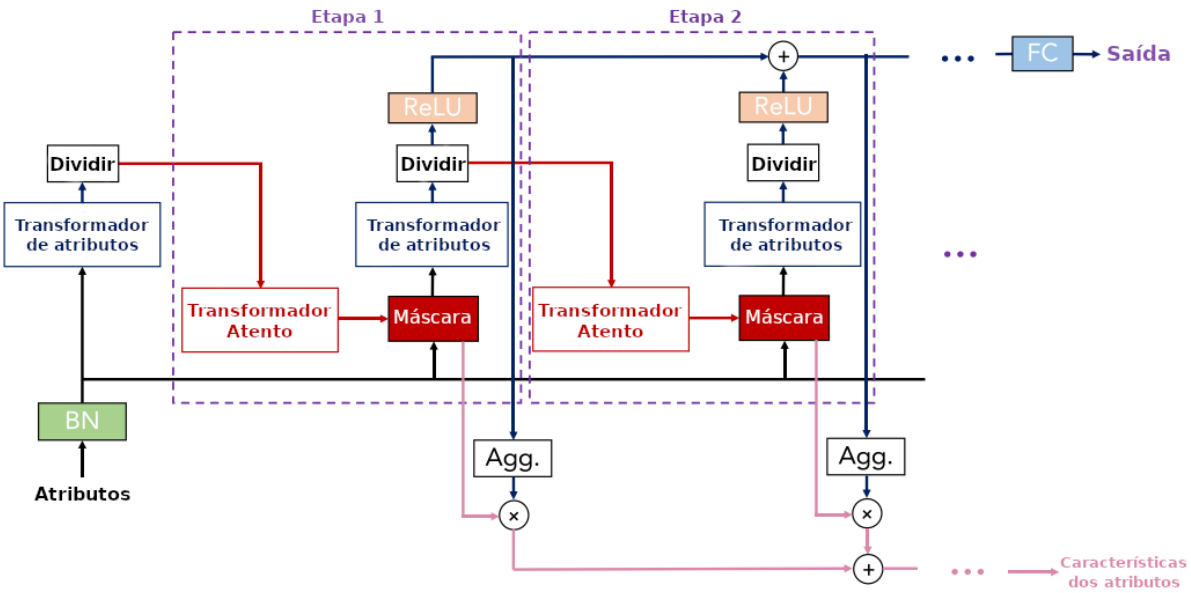
***Interpretable Tabular Data Learning*:** o algoritmo *Interpretable Tabular Data Learning (TabNet)*, que pode ser traduzido de forma direta como Aprendizagem de Dados Tabulares Interpretáveis, utiliza-se da arquitetura de redes neurais profundas, em inglês *Deep Neural Networks (DNNs)*, que são basicamente uma RNA multicamada, explicada na Seção 2.1.2.

A proposta da rede *TabNet* é de solucionar problemas que podem ocorrer ao treinar RNAs com grande número de camadas ocultas e nós, dos quais pode-se citar: a dissipação do gradiente e o alto tempo de processamento (Aggarwal, 2014). Para isso, segundo Arık e Pfister (2021), o *TabNet* é proposto como uma nova arquitetura *DNNs* canônica para dados tabulares, que combina a explicabilidade do modelo, semelhante a modelos baseados em árvores de decisão, e se beneficia do alto desempenho, semelhante a redes neurais profundas. A arquitetura permite a inserção de dados tabulares brutos, sem nenhum tipo de pré-processamento, e usa uma técnica chamada de atenção sequencial para selecionar os atributos relevantes a processar em cada etapa na camada de decisão, ajudando a

explicar a criação do modelo e a aprender modelos mais precisos (Arık e Pfister, 2021). Para isso, a arquitetura, é dividida em um codificador dos dados tabulares e um decodificador.

A codificação dos dados tabulares, como demonstrado na Figura 2.13, é composta principalmente por um transformador de atributos, um transformador atento e uma máscara de atributos. Também, como não é considerada nenhuma normalização global dos atributos, é aplicado apenas a normalização em lote, em inglês *batch normalization (BN)*. O bloco Dividir divide a saída da etapa de decisão e suas informações processadas para ser usada pelo transformador atento e pela etapa subsequente, bem como para a saída geral. Para cada etapa, a máscara de seleção de atributos fornece informações interpretáveis sobre a funcionalidade do modelo, que podem ser agregadas para obter informações importantes de atributos globais. Assim temos que a codificação é baseada no processamento sequencial de várias etapas com N_{etapas} de decisão, isto é, a i-ésima etapa insere as informações processadas da (i - 1)-ésima etapa para decidir quais atributos usar e gerar a representação dos atributos processados a ser agregada na decisão geral (Arık e Pfister, 2021).

Figura 2.13: Arquitetura de codificação do *TabNet*.



Fonte: Adaptado de Arık e Pfister (2021).

Um mecanismo importante utilizado na arquitetura é a seleção de atributos suaves com esparsidade controlável no aprendizado de ponta a ponta, onde um único modelo executa em conjunto a seleção de atributos e o mapeamento de saída, resultando em um desempenho superior com re-

presentações compactas. Isto é, a capacidade de aprendizado de uma etapa de decisão não é desperdiçada em etapas irrelevantes e, assim, o modelo torna-se mais eficiente em termos de parâmetros. Este trabalho é realizado pela máscara obtida através do transformador atento utilizando atributos processados pela etapa anterior, como se pode visualizar na Figura 2.13 (Arık e Pfister, 2021).

O transformador atento utiliza a normalização *Sparsemax* qual incentiva a dispersão ao mapear a projeção euclidiana no simplex probabilístico, observado como superior em desempenho e alinhado para seleção de atributos esparsos para explicabilidade (Arık e Pfister, 2021).

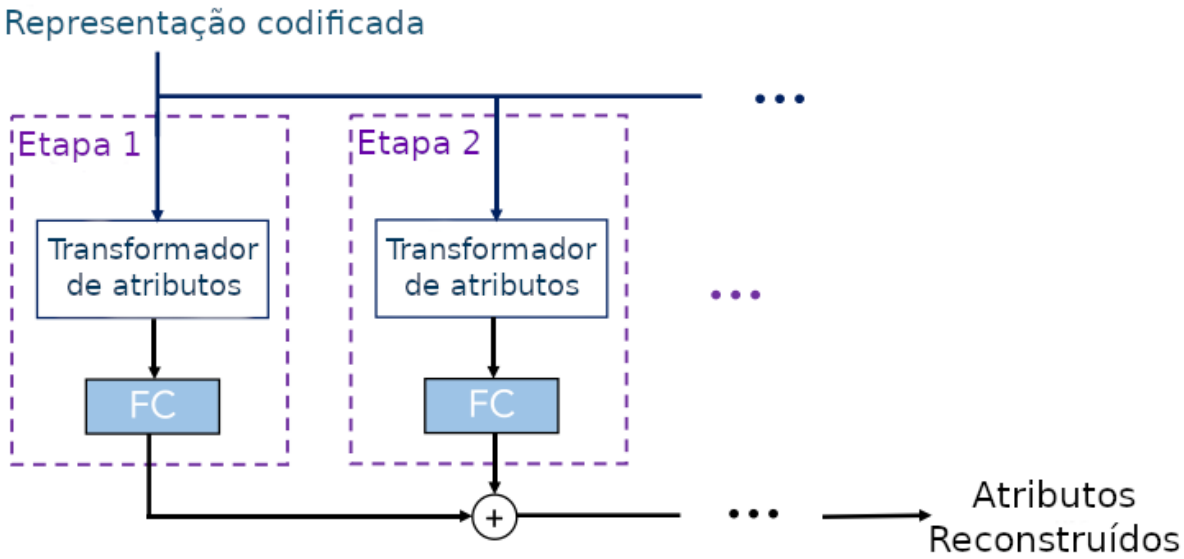
Para realizar o processamento dos atributos filtrados é utilizado o transformador de atributos. Ele é composto por quatro camadas, onde duas são compartilhadas em todas as etapas de decisão, já que os mesmos atributos são inseridos em diferentes etapas, e duas são dependentes da etapa de decisão atual. As quatro camadas são compostas por uma camada *fully connected (FC)*, seguida por *BN* e uma unidade linear fechada, em inglês *Gated Linear Unit (GLU)*, com capacidade não linear. Eventualmente, as camadas são conectadas a uma conexão residual normalizada com normalização de $\sqrt{0.5}$, que ajuda a estabilizar o aprendizado, garantindo que a variância em toda a rede não mude drasticamente (Arık e Pfister, 2021).

Por fim, a decodificação dos dados tabulares tem o objetivo de reconstruir os atributos a partir dos atributos codificados anteriormente. Seu funcionamento é representado na Figura 2.14. Sendo composta por um bloco transformador de atributos, seguido por camadas *FC*, em cada etapa de decisão. Assim, as saídas são somadas para obter os atributos reconstruídos (Arık e Pfister, 2021).

Máquinas de Vetores de Suporte: as Máquinas de Vetores de Suporte, em inglês *Support Vector Machine (SVM)*, usam hiperplanos lineares para separar as classes umas das outras. A ideia é usar uma condição linear que separe as classes, induzindo um hiperplano de margem máxima de separação entre as classes (Aggarwal, 2014). Está técnica tem raízes na teoria de aprendizagem estatística, e dentre seus aspectos, está o funcionamento ótimo com dados de alta dimensão, e ela representa o limite de decisão usando um subconjunto dos exemplos de treinamento, conhecidos como vetores de suporte (Tan et al., 2014).

A Figura 2.15, ilustra um conjunto de dados com exemplos pertencentes a duas classes distintas, representadas por quadrados e círculos, quais são linearmente separáveis. Ainda, como mostrado na Figura 2.15, há

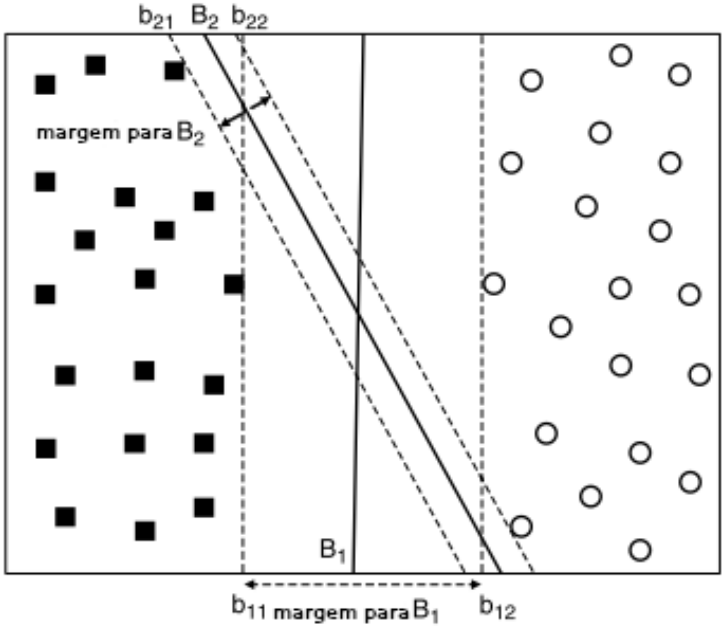
Figura 2.14: Arquitetura do decodificador do *TabNet*.



Fonte: Adaptado de Arık e Pfister (2021).

dois limites de decisão que separam às duas classes, representados pelos hiperplanos B_1 e B_2, com suas respectivas fronteiras de decisão, ilustradas pelos hiperpalnos b_{11}, b_{12} e b_{21}, b_{22} (Tan et al., 2014). Os hiperplanos utilizados como fronteiras são obtidos o movendo para longe do limite de decisão até tocar um exemplo de uma classe (Tan et al., 2014).

Figura 2.15: Exemplo de margem de limite de decisão.



Fonte: Adaptado de Tan et al. (2014).

A distância entre um hiperplano de limite de decisão e um hiperplano fronteira é conhecida como margem do classificador (Tan et al., 2014). Na Figura 2.15 é demonstrada as margens para os limites de decisão B_1 e B_2. Os limites de decisão com grandes margens tendem a ter melhores generalização, e os de margens pequenas tendem a sofrer de *overfitting* (Tan et al., 2014). Dessa forma, para problemas lineares, é desejável projetar classificadores que maximizem as margens de seus limites de decisão para garantir que seus erros de generalização sejam minimizados (Tan et al., 2014). Com isso em mente, para o exemplo ilustrado na Figura 2.15, o hiperplano escolhido para separar às duas classes é B_1, visto que sua margem de separação é a maior.

O *SVM* linear, também conhecido como classificador de margem máxima, procura um hiperplano com a maior margem (Tan et al., 2014). Por exemplo, para um problema de classificação binária com n exemplos de treinamento, em que cada exemplo é denotado por uma tupla (x_i, y_i) $(i = 1, 2, ..., n)$, em que $x_i = (x_{i1}, x_{i2}, ..., x_{id})^T$ corresponde ao i-ésimo atributo exemplo, e, por convenção, $y_i \in -1, 1$ denotando os rótulos das classes (Tan et al., 2014). Com isso, o limite de decisão de um classificador linear é dado por Tan et al. (2014) como:

$$w \cdot x + b = 0, \tag{2.20}$$

em que w e b são parâmetros do modelo. Já a margem pode ser medida de diferentes maneiras (Aggarwal, 2014). Entretanto, Aggarwal (2014) define formalmente a margem que mede o intervalo entre um limite de decisão H e a instância mais próxima como:

$$M \equiv \min_i \frac{w^T x_i + b}{\| w \|^*}, \tag{2.21}$$

em que $\| \cdot \|^*$ é a norma dual da norma que escolhermos na definição de distância.

No entanto, a formulação apresentada anteriormente constrói apenas limites de decisão livres de erros. A abordagem conhecida como margem suave, em inglês *soft margin*, consegue tolerar pequenos erros de treinamento, e lidar com situações em que as classes não são linearmente separáveis (Tan et al., 2014). Essa abordagem pode encontrar um limite de decisão com uma margem muito ampla, entretanto, classificar erroneamente muitos dos exemplos de treinamento (Tan et al., 2014). Para evitar este problema, é utilizado um parâmetro C que representa a penalidade aplicada aos exemplos de treinamento classificados incorretamente.

O parâmetro pode ser escolhido com base no desempenho do modelo no conjunto de validação (Tan et al., 2014).

Outra abordagem utilizada em um problema de conjunto de dados que possuem limites de decisão não lineares, que merece destaque, é a transformação dos dados. O objetivo dessa abordagem é transformar os dados de seu espaço de coordenadas original x em um novo espaço $\phi(x)$, para que um limite de decisão linear possa ser aplicado para separar os exemplos de treinamento do espaço transformado (Tan et al., 2014). A transformação de um conjunto de dados, para tornar-se linearmente separável, é dada por Aggarwal (2014) como:

$$\phi(x) = [x_1 x_1 \quad x_2 x_2]^T. \tag{2.22}$$

Porém, essa abordagem pode sofrer com o problema da maldição da dimensionalidade, muitas vezes associado a dados de alta dimensão (Tan et al., 2014).

Para tratar o problema da maldição da dimensionalidade, pode-se utilizar um método conhecido como truque do *kernel*, em inglês *kernel trick*. Este método calcula a similaridade no espaço transformado usando o conjunto de atributos original (Tan et al., 2014). O método funciona mesmo se não soubermos a qual superfície de separação ele corresponde no espaço dimensional infinito (Aggarwal, 2014). Duas funções *kernel* comumente utilizadas são a polinomial e a *Radial Basis Function (RBF)* (Aggarwal, 2014; Tan et al., 2014).

Ajuste de Parâmetros

Ajuste de parâmetro (também denominado como ajuste *off-line*), em inglês *parameter tuning*, é uma abordagem na qual bons valores de parâmetro (configuração de parâmetros) são identificados antes de aplicar o algoritmo para resolver os problemas em questão. Nesse caso, os resultados do ajuste de parâmetro, ou seja, a configuração de parâmetro ideal fundada pelo processo de ajuste, é usado na resolução de problemas e esses valores de parâmetros permanecem inalterados durante a execução (Huang et al., 2020).

Os modelos de classificação apresentados anteriormente, são compostos de diversos parâmetros em sua execução. Mesmo que a configuração de parâmetros padrão sejam fornecidas, os parâmetros encontrados através do ajuste de parâmetros podem resultar em melhoria de desempenho, pois as configurações padrão, geralmente, são determinadas com outros problemas (ou contextos diferentes) que diferem do problema em questão (Huang et al., 2020).

Logo, o principal propósito do ajuste de parâmetros é encontrar uma confi-

guração que maximize o desempenho de um algoritmo sobre a(s) instância(s) de um problema (Huang et al., 2020). Este problema de encontrar uma configuração é declarado formalmente por Hoos (2011), dado:

- um algoritmo A com diferentes parâmetros que afetam seu comportamento;
- um espaço de configurações de parâmetros C, na qual cada configuração $c \in C$ específica valores para os parâmetros de A;
- um conjunto de instâncias do problema I;
- uma métrica de desempenho m que mede o desempenho de A em I para uma dada configuração c;

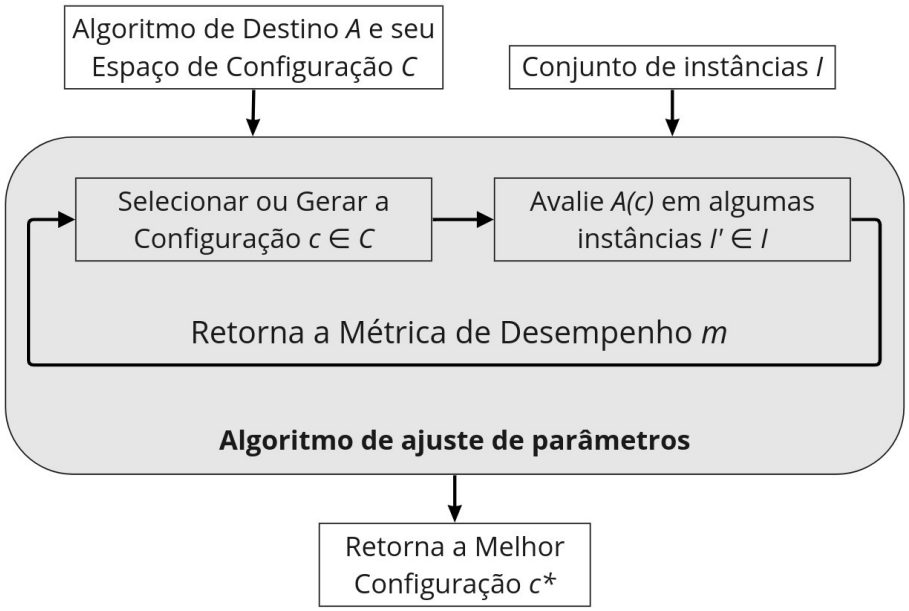
encontrar uma configuração $c^* \in C$ que resulte no desempenho ótimo de A sobre I conforme a métrica m (Hoos, 2011).

Neste contexto, temos uma forma automatizada de ajuste de parâmetros, representada na Figura 2.16. Em que, o método de ajuste automático de parâmetros é frequentemente chamado algoritmo de ajuste de parâmetros, e o algoritmo cujo desempenho deve ser maximizado é frequentemente chamado algoritmo de destino, e o $A(c)$ é utilizado para denotar o algoritmo de destino A sob uma configuração específica c (Hoos, 2011; Huang et al., 2020). Dependendo do algoritmo de destino, vários tipos de parâmetros podem ocorrer, como parâmetros categóricos, booleanos, numéricos, e condicionais, cada um com sua especificidade de utilização (Hoos, 2011). Sendo que, conforme a situação, faz-se útil colocar restrições nas configurações, por exemplo, para excluir certas combinações de valores de parâmetros que levariam a um comportamento mal definido de um certo algoritmo de destino (Hoos, 2011).

O processo de ajuste de parâmetros, geralmente, requer inúmeras execuções do algoritmo para analisar seu desempenho em uma instância ou conjunto de instâncias de problemas com diferentes configurações de parâmetros (Huang et al., 2020). O que torna o processo demorado, sendo sua principal desvantagem. Já sua principal vantagem é sua universalidade, ou seja, um bom método de ajuste de parâmetros é aplicável em diferentes algoritmos (Huang et al., 2020).

Um dos métodos mais simples para gerar e avaliar os parâmetros são os não iterativos. Primeiramente criam uma série de configurações de parâmetros (configurações candidatas) e, em seguida, avaliando cada uma delas para encontrar a melhor configuração (Huang et al., 2020). Na qual, a abordagem de força bruta se enquadra. Nesta abordagem, um conjunto de configurações de parâmetros é gerado, então o desempenho de cada configuração candidata

Figura 2.16: Processo automatizado de ajuste de parâmetros.



Fonte: Adaptado de Huang et al. (2020).

é estimado executando o mesmo número de execuções em instâncias de treinamento (Huang et al., 2020). Desse modo, a configuração com o melhor desempenho estimado é considerada a configuração de parâmetro ideal.

Baseada na abordagem de força bruta, uma poderosa técnica de otimização de hiperparâmetros, utilizada em aprendizado de máquina, é a pesquisa em grade, em inglês *grid search* (Raschka, 2015). A técnica pode ser utiliza com a validação cruzada *k-fold* aninhada. Isto é, um *loop* externo é utilizado para separar os dados em treino e teste, e um *loop* interno é utilizado para avaliar o desempenho do modelo (Raschka, 2015).

2.1.3 Pós-processamento

O pós-processamento visa avaliar, validar e consolidar os resultados obtidos. Esta etapa garante que somente resultados válidos e úteis serão incorporados a um sistema de apoio a decisão. Esta etapa tem apoio da área de visualização dos padrões extraídos, permitindo a exploração dos dados e da área de medidas estatísticas, sendo importante que os resultados avaliados sejam estatisticamente significantes e confiáveis (Parmezan et al., 2012; Tan et al., 2014).

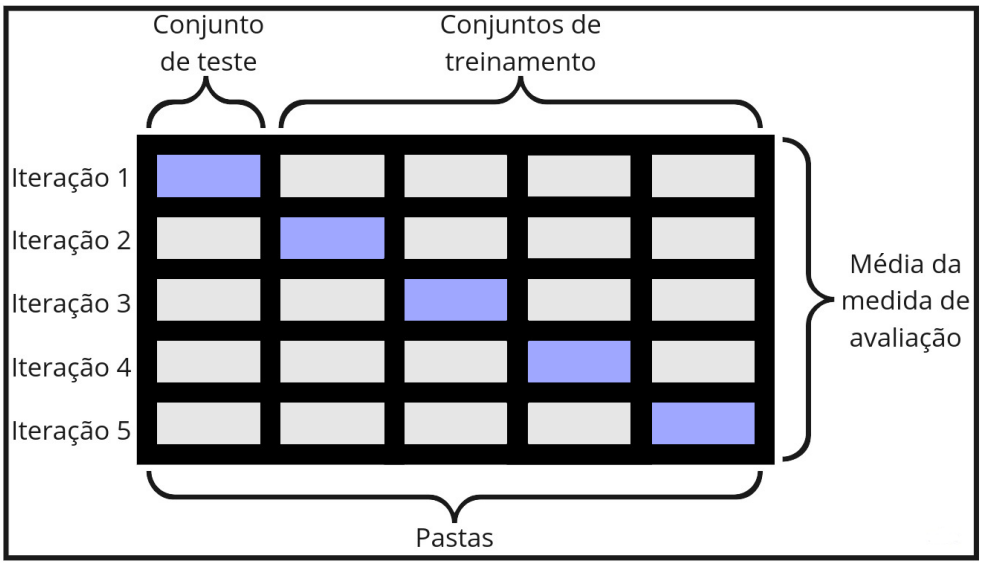
Validação Cruzada

Para estimar o comportamento de um modelo de classificação, na prática, normalmente constrói-se o classificador em um conjunto de treinamento, e

avalia-se o classificador em um conjunto de teste, tal que o conjunto de treino e teste são disjuntos, esta abordagem é chamada de *hold-out*. No entanto, esta estratégia de validação em somente dois conjuntos tem fraquezas, como a pontuação do método, com alta dependência de como os dados são divididos nos conjuntos de treino e teste, e o modelo não está sendo treinado e avaliado usando todos os dados disponíveis (Albon, 2018).

Na abordagem da validação cruzada em k pastas, em inglês *k-fold cross-validation*, cada registro do conjunto de dados é usado k vezes para treinamento e exatamente uma vez para teste (Tan et al., 2014). Para isso, os registros são divididos k pastas, sendo o valor de k definido pelo usuário (Witten e Frank, 2005). A validação é feita em k iterações, sendo que em cada iteração, $k-1$ pastas são utilizadas para treino e uma pasta é utilizada para teste. Ao final, tira-se uma média da medida de avaliação nas k iterações para ser utilizada como valor da validação, como é apresentado na Figura 2.17. As pastas podem ser particionadas com a porcentagem de cada classe diferente do conjunto de treinamento. A versão da validação cruzada utilizada para tratar o problema é conhecida como validação cruzada *k-fold* estratificada, em inglês *k-fold cross validation stratified* (Boschetti e Massaron, 2016).

Figura 2.17: Conjunto de dados utilizando a validação cruzada *k-fold*.



Fonte: Autoria própria.

Medidas de Avaliação

A avaliação do desempenho de um modelo de classificação é baseado nas contagens de registros de teste previstos correto e incorretamente (Tan et al., 2014). Para isso, é utilizada a matriz de confusão. A Tabela 2.1 descreve uma

matriz de confusão para classificação binária. Na qual temos Verdadeiros Positivos (VP) que foram preditos como 1 e na classe real são 1, Verdadeiros Negativos (VN) que foram preditos como 0 e na classe real são 0, Falsos Positivos (FP) que foram preditos como 1 e na classe real são 0, e Falsos Negativos (FN) que foram preditos como 0 e na classe real são 1 (Tan et al., 2014).

Tabela 2.1: Matriz de confusão para classificação binária.

		Classe predita	
		Positivo = 1	Negativo = 0
Classe real	Positivo = 1	VP	FP
	Negativo = 0	FN	VN

Fonte: Adaptado de Tan et al. (2014).

Alguns cálculos de medidas de avaliação, se utilizam da matriz de confusão. A cada iteração de um classificador são calculados (Aggarwal, 2014):

- **Acurácia**, definida como a proporção de instâncias classificadas corretamente (Aggarwal, 2014), dada pela equação:

$$\text{Acurácia} = \frac{VP+VN}{VP+FP+FN+VN}; \tag{2.23}$$

- **Taxa de erro**, definida como a proporção de instâncias classificadas incorretamente (Tan et al., 2014), dada pela equação:

$$\text{Taxa de erro} = \frac{FP+FN}{VP+FP+FN+VN}; \tag{2.24}$$

- **Revocação**, sendo os verdadeiros positivos em comparação com o número de casos verdadeiramente positivos (Aggarwal, 2014), dada pela equação:

$$\text{Revocação} = \frac{VP}{VP+FN}; \tag{2.25}$$

- **Precisão**, são os verdadeiros positivos em comparação com o número de instâncias positivas preditas (Aggarwal, 2014), dada pela equação:

$$\text{Precisão} = \frac{VP}{VP+FP}; \tag{2.26}$$

- **F1**, a F1-*measure* (F1-*score*) utiliza a precisão e a revocação para fazer a média harmônica dessas duas medidas (Aggarwal, 2014), dada pela equação:

$$\text{F1} = 2 * \frac{Precisão * Revocação}{Precisão + Revocação}; \tag{2.27}$$

- **Micro-*Averaging***, é uma estratégia utilizada para sumarizar os resultados da Precisão e Revocação para todas as classes de um conjunto de dados, no qual é calculado a soma dos termos individualmente de cada medida (Sokolova e Lapalme, 2009; Manning et al., 2008; Sebastiani, 2002), dada pelas equações:

$$\text{Precisão}^{Micro} = \frac{\sum_{c_i \in C} VP_{c_i}}{\sum_{c_i \in C} (VP_{c_i} + FP_{c_i})}, \tag{2.28}$$

e

$$\text{Revocação}^{Micro} = \frac{\sum_{c_i \in C} VP_{c_i}}{\sum_{c_i \in C} (VP_{c_i} + FN_{c_i})}, \tag{2.29}$$

em que c_i é a i-ésima classe pertencente a C;

- **Macro-*Averaging***, é uma estratégia utilizada para calcular a média dos valores de Precisão e Revocação considerando todas as classes de um conjunto de dados (Sokolova e Lapalme, 2009; Manning et al., 2008; Sebastiani, 2002), dada pelas equações:

$$\text{Precisão}^{Macro} = \frac{\sum_{c_i \in C} Precisão_{c_i}}{|\,C\,|}, \tag{2.30}$$

e

$$\text{Revocação}^{Macro} = \frac{\sum_{c_i \in C} Revocação_{c_i}}{|\,C\,|}, \tag{2.31}$$

em que c_i é a i-ésima classe pertencente a C.

Uma abordagem gráfica comumente utilizada para representar a performance de um modelo de classificação quando há duas classes é a curva *Receiver Operating Characteristic (ROC)*, a qual busca exibir graficamente a compensação entre a taxa de verdadeiro positivo, no eixo y, em relação à taxa de falso positivo, no eixo x, de um classificador (Tan et al., 2014). A taxa de verdadeiro positivo é a proporção de exemplos positivos reais previstos corretamente como exemplos positivos. Já a taxa de falso positivo é a proporção de exemplos negativos previstos incorretamente como exemplos positivos. Cada ponto da curva *ROC* corresponde a um limiar que pode ser usado para classificar exemplos em 2 classes (Shahinfar et al., 2014).

A *Area Under the Curve (AUC) ROC*, em português área sob a curva *ROC*, é definida como a área entre a curva *ROC* e o eixo horizontal (Shahinfar et al., 2014), sendo que, se o modelo de classificação for perfeito, então sua *AUC* seria igual a 1. Já se o modelo de classificação simplesmente realizasse suposições aleatórias, então sua *AUC* seria igual a 0.5 (Tan et al., 2014). Portanto, quanto maior a área sob a curva *ROC*, maior será o acerto da classificação. Ademais, uma área maior indica que a classificação é mais correta para os exemplos com maior confiança de classificação.

2.2 Trabalhos Relacionados

Nesta seção são apresentados os trabalhos relacionados ao tema de AM associado ao auxílio na produção animal. Foi dado enfoque em trabalhos que utilizaram dados tabulares e realizaram previsão e estimativa de alguma categoria de característica relacionada à qualidade da carcaça do animal, buscando auxiliar o produtor na tomada de decisão no manejo de sua produção e potencialização dos resultados econômicos.

O trabalho de Shahinfar et al. (2019) considera a necessidade dos criadores de ovelhas da Austrália de terem uma previsão das características de peso da carcaça quente *(HCW)*, gordura intramuscular *(IMF)*, profundidade de gordura de regra de *Greville (GRFAT)*, rendimento da carcaça magra com base em dados de tomografia computadorizada *(CTLEAN)* e peso do lombo *(LW)*, da qualidade da carcaça do ovino ainda em idade bem jovem. Tendo essas previsões no início de vida, permitiria aos criadores ajustarem suas práticas de manejo. Para realização das predições, o conjunto de dados utilizado foi composto por registros ambientais, carcaça e peso vivo, coletados ao longo de um período de cerca de 10 anos do *Sheep CRC Information Nucleus Flock*. Foram considerados 8 locais com diferentes características climáticas e de produção para a coleta de dados. Para realização da predição das características, foram escolhidos os algoritmos de AM *Deep Learning (DL)*, *Gradient Boosting Tree (GBT)*, *k-Nearest Neighbor (KNN)*, *Model Tree (MT)* e *Random Forest (RF)*. O conjunto de dados foi dividido em 75% para treinamento e 25% para teste. Foi utilizada a busca em grade para os ajustes dos hiperparâmetros dos algoritmos. Com isso, os modelos de aprendizado de máquina conseguiram prever as 5 características, mesmo que com a falta de dados, ampla gama de raças, idades, climas e sistemas de manejo. O algoritmo *Random Forest (RF)* superou todos os outros na predição de todas as características, com o *Model Tree (MT)* como segundo melhor desempenho (Shahinfar et al., 2019).

Em diversos países pelo mundo a qualidade da carcaça é a forma de remuneração do produtor de bovinos. Diante disso, Soulat et al. (2018) realizou um estudo cujo objetivo é investigar o impacto de informações coletadas de toda a vida do animal (do nascimento ao abate) em comparação com o período de engorda, e tentar identificar os fatores de criação que permitem um melhoramento e mantenham estáveis as características de qualidade. Para isso, foram utilizadas informações de 96 novilhas coletadas em 8 fazendas, com número variado de animais por fazenda. As informações sobre os fatores de criação foram coletadas por questionário, realizado em 3 pesquisas. Os algoritmos utilizados para predição foram a Regressão Logística Multinomial Cumulativa e a Regressão Logística Multinomial Clássica. Buscando um melhor entendi-

mento entre os períodos de vida das novilhas, foi desenvolvido dois modelos para predizer os agrupamentos de qualidade da carcaça. O primeiro modelo continha os fatores de criação selecionados durante o período de engorda. O segundo modelo continha todos os fatores de criação medidos durante a vida das novilhas (incluindo o período de engorda). A validação dos modelos de predição foi realizada por meio do procedimento de *bootstrap*, sendo um método de reamostragem com substituição. Com modelos que apresentaram acurácia de 62.8%, foi possível predizer o potencial da qualidade da carcaça a partir de fatores de criação com os animais ainda vivos. Também, foram demonstrados nos resultados que as práticas de criação aplicadas antes da engorda impactam nas características de qualidade da carcaça (Soulat et al., 2018).

O peso da carcaça é um dos principais fatores para composição do valor do animal. Alonso et al. (2013) utilizaram medidas zoométricas dos animais buscando predizer o peso da carcaça de bovinos de corte dias antes do abate. A abordagem utilizada foi baseada em procedimentos de aprendizado de máquina que objetivam aprender uma função e mapear o peso da carcaça a partir de uma coleção de medidas morfológicas dos animais e do número de dias até o abate. Para isso, foi utilizado um conjunto de dados de 390 descrições numéricas dos animais, correspondente a 144 bovinos, de ambos sexos, da raça Asturiana de *los Valles*, raça de corte do norte da Espanha. O algoritmo escolhido foi o *Support Vector Machines (SVM)* para Regressão *(SVR)*, utilizando o *kernel radial basis function (RBF)* e o *kernel linear*. Para validação, foi utilizada a validação cruzada, e para otimização dos hiperparâmetros, foi utilizada busca em grade. Com isso, foram realizadas as análises dos erros para os diferentes *kernel* utilizados. Utilizando o *Wilcoxon signed-rank test*, com $p < 0,01$, não houve diferenças significativas nos erros apresentados. O erro relativo foi próximo a 4%, pouco acima de 22 quilos. Portanto, as previsões foram bastante precisas. Os resultados mostram que usando uma função não linear é possível obter previsões precisas muito tempo antes do abate. No entanto, os erros são menores se as medidas dos animais forem tomadas próximo ao abate (Alonso et al., 2013).

Além da qualidade da carcaça e do peso, o marmoreio tem grande impacto na formação do preço do animal para o produtor. No estudo realizado por Shahinfar et al. (2020), utilizando métodos de AM para predição numérica, uma das características utilizadas é a pontuação do marmoreio. Ademais, as outras características exploradas na predição são o peso da carcaça *(CWT)*, a área do músculo ocular *(EMA)*, pesos do lombo *(SirLn)*, *striploin (StrLn)*, e filé *mignon (TndLn)*. Com base em uma ampla gama de medições biofísicas e de polimorfismos de nucleotídeo único *(SNP)*. Também, neste trabalho foram utilizados dados genéticos. A genotipagem foi feita por outro grupo de pesqui-

sadores que utilizaram o Algoritmo Genético *(GA)*. Para realização do estudo de caso foi utilizado um conjunto de dados com 3.989 bovinos *Hanwoo* criados pelo Sistema Nacional de Melhoramento de Gado *Hanwoo* da Coreia do Sul. Os algoritmos de AM utilizados foram *Multilayer Perceptron (MLP)*, *Model Tree (MT)*, *Random Forest (RF)* e *Support Vector Machine (SVM)* com *Sequential Minimal Optimization (SMO)*. O conjunto de dados foi dividido em 80% para treinamento e 20% para teste. Já para otimização dos hiperparâmetros foi utilizado a busca em grade. Todas as 6 características utilizadas nos modelos foram treinadas e testadas com sucesso. Os algoritmos que tiveram um desempenho melhor foi o *Support Vector Machine (SVM)* e a *Model Tree (MT)* (Shahinfar et al., 2020).

Levando o olhar para base de dados do programa Precoce MS, temos outros estudos realizados utilizando desta fonte. Como no trabalho de Mota (2016), que teve como principais objetivos a construção de um armazém de dados (*Data Warehouse*), a disponibilização de um portal *web* dos dados armazenados por consultas analíticas *online (OLAP)*, e a utilização da Mineração de Dados (MD) para aplicação de algoritmos de classificação para predição do momento de abate ideal de animais, observando os pré-requisitos de qualidade da carne. Porém, o autor não coloca quais algoritmos de classificação foram utilizados. Os conjuntos de dados utilizados foram fornecidos pela Associação Sul-matogrossense de Produtores de Novilho Precoce (ASPNP) e de estudos realizados na área de ciência animal pela Embrapa Gado de Corte. Dessa forma, foi construído um armazém de dados (*Data Warehouse*) contendo os dados de abate de gado de corte. Posteriormente, foi disponibilizado um portal Web para acesso aos dados do armazém de dados (*Data Warehouse*) utilizando consultas analíticas *online (OLAP)*. Na etapa final, foi aplicado e analisado os resultados dos algoritmos de MD. Utilizando técnicas de MD para aplicação dos algoritmos de classificação, quais não foram especificados pelo autor, buscou-se obter informações sobre o melhor momento para o abate conforme a qualidade da carcaça do animal. O autor evidência que o atributo mais relevante para o experimento, o peso vivo do animal, foi armazenado no banco de dados da ASPNP como média de todos os animais de um lote abatido, se tornando um dado não preciso e influenciado diretamente no resultado obtido na classificação. Como o peso individual do animal é um atributo importante no processo de aprendizagem, o autor, buscou outra fonte de dados com os pesos dos animais armazenados separadamente. Com os dados de estudos realizados na área de ciência animal pela Embrapa, após aplicar os algoritmos de classificação, utilizando 70% do conjunto de dados para treino e 30% para testes, foi obtido resultados significativamente melhores (Mota, 2016).

Baseado na proposta de Mota (2016), o trabalho de Nucci (2019) teve como

principal objetivo construir um modelo de classificação do grau de gordura da carcaça por meio de algoritmos de AM. O conjunto de dados utilizado era composto por dados de abate de bovinos cadastrados no programa estadual Precoce MS e do processo produtivo de cada estabelecimento rural correspondente, possuindo mais informações relacionadas a produção do animal. Na construção do modelo, o autor utilizou os algoritmos para AM *Support Vector Machines*, *k-Nearest Neighbors*, *Naive Bayes* e *Random Forest Classifier*. O conjunto de dados foi dividido em treino e teste, com o conjunto de treinamento recebendo 80% dos dados e testes 20%. A técnica de busca em grade foi utilizada para seleção de hiperparâmetros de forma automática. Desta forma, utilizando as métricas de *Precision*, *Recall* e *F1*, o modelo que obteve melhor acurácia foi o *Random Forest Classifier*. Uma das dificuldades apontada pelo autor, foi o desbalanceamento dos dados, o que o obrigou a estudar técnicas de balanceamento de dados, compará-las, e aplicar no conjunto de dados durante a construção do modelo, auxiliando na melhora dos resultados obtidos na classificação. É elencado, pelo autor, que as características mais relevantes para criação do modelo foram: o peso da carcaça, o sexo, o mês de abate, a maturidade do animal e a localização da fazenda (Nucci, 2019).

Pode-se observar que a utilização de algoritmos de AM para predizer características da qualidade de carcaça são utilizadas com bastante sucesso na literatura. Desta forma, é possível observar nos trabalhos pontos importantes elencados pelos autores. Por exemplo Mota (2016), demonstrou haver a necessidade de se utilizar informações mais precisas do animal e da fazenda, ou seja, como esses animais foram criados antes de serem abatidos. No trabalho de Soulat et al. (2018), também foi possível evidenciar que a utilização de dados de toda vida do animal pode trazer melhores resultados para predição da qualidade da carcaça. Alguns trabalhos relataram que a utilização de atributos referentes à quantidade de chuva, temperatura, e informações nutricionais da alimentação, podem ser úteis para discriminar ou determinar a qualidade da carcaça (Shahinfar et al., 2019). Outros trabalhos utilizaram dados genéticos dos animais no modelo de predição. Porém, essas categorias de dados são mais complexos de se obter (Shahinfar et al., 2020). Outro ponto importante abordado em Nucci (2019), foi a não utilização da área de AM chamada “aprendizado profundo” (*Deep Learning*). O autor acredita que utilizar esses algoritmos pode melhorar as métricas de performance de classificação para os dados do programa Precoce MS.

2.3 Considerações Finais

Este capítulo apresentou os conceitos e técnicas utilizadas neste projeto relacionadas a cada etapa de um processo de MD. No qual foi abordado a etapa de pré-processamento, com suas tarefas de limpeza de dados, integração de dados, redução de dados, e transformação dos dados. Na etapa posterior, foi apresentada a extração de padrões, apoiada pela área de AM. Sendo abordado os conceitos de aprendizado supervisionado, classificação, e o funcionamento dos algoritmos de classificação k-Vizinhos Próximos, *Naïve Bayes*, Árvores de Decisão, Florestas Aleatórias, *Extreme Gradiente Boosting*, Redes Neurais Artificiais, *Interpretable Tabular Data Learning* e Máquinas de Vetores de Suporte. Dando suporte para encontrar os melhores parâmetros para os algoritmos de classificação, foi apresentado o ajuste de parâmetros, também conhecido como *tuning* de parâmetros. Como última etapa do processo de MD, foi abordado o pós-processamento, com a apresentação dos temas de validação cruzada e medidas de avaliação. Por fim, foram apresentados trabalhos relacionados ao tema de AM associado ao auxílio na produção animal. Sendo observado na literatura que o AM é utilizado com sucesso para predizer características da qualidade da carcaça, e acredita-se que empregar diferentes abordagens nos dados do programa Precoce MS, utilizando atributos que podem influenciar a qualidade da carcaça, fazendo o uso de algoritmos estado-da-arte, aplicando técnica de *tuning* de parâmetros e separando os animais em lotes, pode-se melhorar as métricas de performance de classificação.

CAPÍTULO

3

Método de Pesquisa

Diante do proposto por este trabalho, de avaliar qual algoritmo de AM com melhor desempenho no conjunto de dados do programa Precoce MS e, então, construir um modelo de classificação para predizer a qualidade das carcaças dos lotes de bovinos a serem abatidos, o processo de MD, apresentado na Seção 2.1, foi o escolhido a ser seguido para construção do método de pesquisa. Os detalhes de cada etapa do processo de MD adotado nesse projeto, são detalhados nas próximas seções.

3.1 Base de Dados

O conjunto de dados do programa Precoce MS, com informações de abate individual dos animais, da fazenda, do clima, do processo produtivo, da classificação da qualidade do animal e das variáveis econômicas relacionadas ao preço de *commodities* da soja, do milho e da arroba do gado, foi disponibilizado por uma representante da Embrapa Gado de Corte, em formato *CSV (Comma-Separated Values)*. O conjunto de dados é composto por 112 atributos (colunas) e 3.153.593 de exemplos (registros) de animais abatidos referente aos anos de 2017, 2018, 2019 e 2020. Para demonstrar a composição do conjunto de dados, foi escolhido 6 atributos e 5 exemplos aleatórios que podem ser observados na Tabela 3.1. A representação completa do conjunto de dados pode ser visualizada na Tabela A.1.

O conjunto de dados foi explorado visando identificar a distribuição de frequência dos valores dos atributos. A Figura 3.1 exemplifica a distribuição para três atributos: (a) `Tipificacao`, composto pelos valores qualitativos nominais “Macho Inteiro”, “Macho Castrado” e “Fêmea”, qual indica o tipo dos

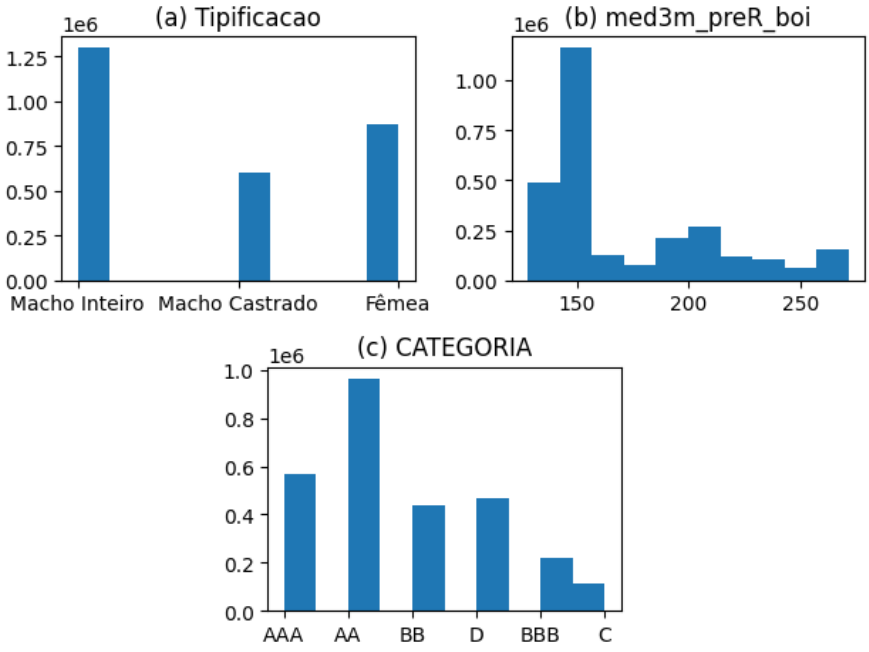
Tabela 3.1: Cinco exemplos aleatórios do conjunto de dados do programa Precoce MS.

ID_ANIMAL	Tipificacao	Peso	ANO	CATEGORIA	Classificacao
614677	Fêmea	193.20	2018	AA	SIM
3532447	Fêmea	175.50	2020	D	NÃO
137837	Macho Castrado	290.20	2017	BBB	SIM
3479832	Fêmea	197.40	2020	BB	SIM
2342444	Macho Castrado	254.40	2019	D	NÃO

Fonte: Autoria própria.

animais abatidos; (b) `Med3m_preR_boi`, composto pela média do preço da arroba do boi dos três meses que antecedem a data do abate dos animais; e (c) `CATEGORIA`, composto pelos valores qualitativos ordinais “AAA”, “AA”, “BBB”, “BB”, “C” e “D”, qual, na ordem exposta de melhor para pior, indica a categoria da carcaça dos animais abatidos, sendo utilizado posteriormente para gerar a classificação do lote conforme será apresentado na próxima seção. Ainda, pode-se observar que os atributos possuem diferentes distribuições, necessitando a utilização de técnicas específicas para cada tipo de atributo.

Figura 3.1: Distribuição da frequência dos atributos (a) `Tipificacao`, (b) `Med3m_preR_boi` e (c) `CATEGORIA`.



Fonte: Autoria própria.

3.2 Pré-Processamento de Dados

Nesta seção são apresentados passos utilizados para tornar a base de dados adequada para atingir os objetivos do projeto, além de aumentar a qualidade de dados para a construção de modelos mais acurados. Foram utilizados neste projeto as seguintes etapas de pré-processamento: eliminação de atributos correlacionados, eliminação de exemplos duplicados, tratamento de valores ausentes, agregação dos dados e técnicas de transformação dos dados.

O coeficiente de correlação de *rank* de *Spearman*, procedimento estatístico projetado para medir a relação entre duas variáveis em uma escala ordinal de medição que tem como principal característica a utilização de *rankings* e não faz nenhuma suposição sobre a distribuição dos dados (Haroon, 2017; Brownlee, 2019), foi utilizado para verificação de correlação dos atributos do conjunto de dados. O método utilizado retorna um valor entre -1, sendo uma correlação negativa perfeita, 1, sendo uma correlação positiva perfeita, e 0 quando não existe correlação entre os atributos. Assim, foi possível verificar que alguns atributos possuíam alta correlação, principalmente os relacionados às variáveis econômicas e às variáveis climáticas, como demonstrado na Tabela 3.2. Dessa forma, foi escolhido um dos atributos a se manter no conjunto de dados. Por exemplo, a correlação entre o atributo `Med3m_formITUmax` e `Med3m_formITUinst` foi de 0.99, sendo o atributo `Med3m_formITUmax` escolhido a se manter. Então o procedimento foi repetido até manter-se atributos com taxas de correlação abaixo de 0.95.

Tabela 3.2: Cinco exemplos da análise de correlação de *Spearman*.

Atributo 1	**Atributo 2**	**Correlação**
Med3m_formITUmax	Med3m_formITUinst	0.998
Med7d_formITUmax	Med7d_formITUinst	0.997
Med12m_TempMin	Med12m_TempInst	0.996
Med7d_TempMin	Med7d_TempInst	0.996
Med12m_preR_boi	Med6m_preR_boi	0.990

Fonte: Autoria própria.

Utilizando o atributo `ID_ANIMAL`, sendo a identificação individual sequencial de cada exemplo no conjunto de dados, foi identificado que havia um total de 162 exemplos duplicados. Então foi adotado a estratégia de manter o primeiro exemplo encontrado. Por exemplo, exemplos que tinham o `ID_ANIMAL` igual a 3636657, foi mantido o primeiro encontrado e excluído o restante.

Atributos utilizados para realizar a identificação individual sequencial de alguma informação podem ser prejudiciais para a modelagem. Dessa forma,

os atributos com tal característica foram removidos, a saber: ID_ANIMAL, Frigorifico_ID e Questionario_ID. Tais atributos não contribuem para o AM, conforme explanado na Seção 2.1.1 (Witten e Frank, 2005). Também foi realizada a remoção de atributos considerados irrelevantes para a modelagem, após conversa com a especialista do domínio da Embrapa Gado de Corte.

Dado que em uma situação prática, provavelmente, o produtor desejará colocar as informações referente ao lote de animais a serem abatidos, não de cada animal, pois, seria muito trabalhoso realizar a previsão para cada animal, foi gerado o conjunto de dados agregado pelos atributos DataAbate, EstabelecimentoIdentificador e Tipificacao, separando os animais em lotes para serem utilizados na etapa de extração de padrões. O atributo DataAbate refere-se a data em que o animal foi abatido, o atributo EstabelecimentoIdentificador refere-se ao identificador do produtor do animal abatido, e o atributo Tipificacao refere-se ao tipo do animal abatido, que pode assumir três valores: 'Macho Inteiro', 'Fêmea' e 'Macho Castrado'. Dessa forma, por exemplo, um lote pode ser composto por informações de animais abatidos do tipo 'Macho Castrado', na data de '01/01/2020', do produtor 999.

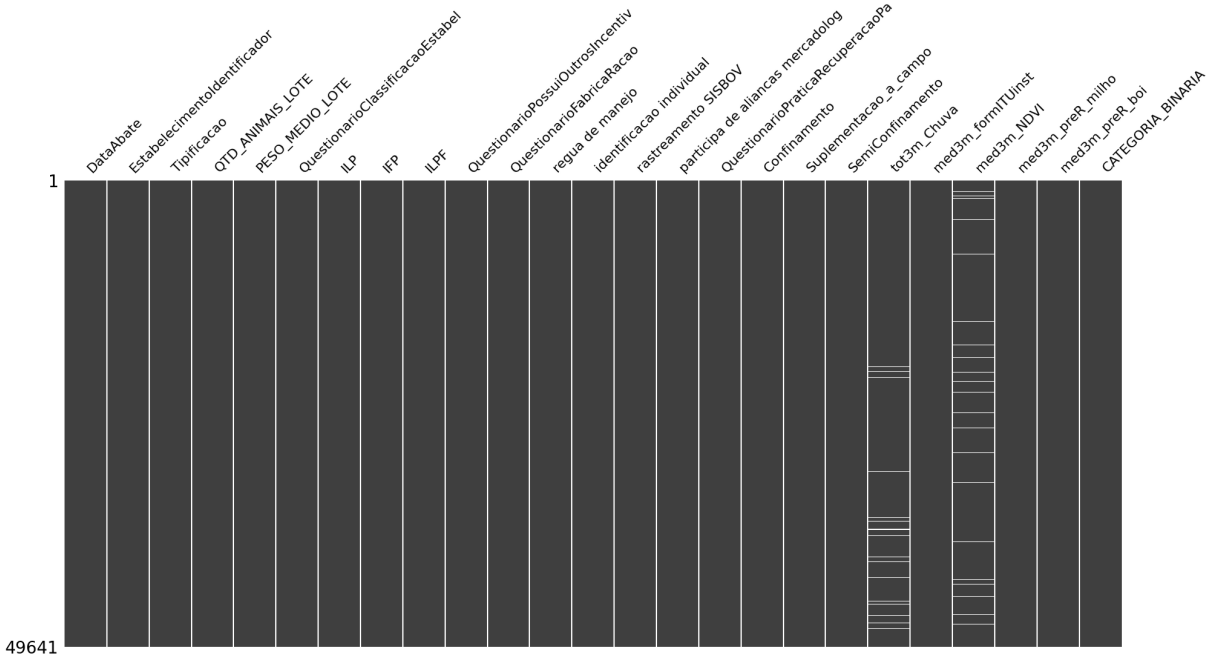
Na geração da agregação dos animais por lote, foram criados três novos atributos: QTD_ANIMAIS_LOTE, sendo a quantidade de animais abatidos no lote; PESO_MEDIO_LOTE, sendo a média simples dos pesos dos animais abatidos no lote; e CATEGORIA_BINARIA, que contêm se um lote é de 'Alta qualidade' ou 'Baixa qualidade'. O atributo binário CATEGORIA_BINARIA, escolhido como classe alvo, foi gerado a partir do atributo CATEGORIA, sendo que se um lote possuí 50% ou mais de animais categorizados como "AAA" e "AA" o lote tem 'Alta qualidade', senão o lote tem 'Baixa qualidade'[1]. Já para o restante dos atributos do conjunto de dados agregado, para atributos do tipo numérico foi utilizada a estratégia de obter a média do valor do lote, enquanto para atributos categóricos foi utilizada a estratégia de obter o valor mais frequente no lote.

Utilizando a técnica de verificação de valores ausentes no conjunto de dados, explicada com mais detalhes na Seção 2.1.1, foi averiguado que os atributos Med3m_NDVI e Tot3m_Chuva foram os que apresentaram a maior quantidade de valores ausentes, com aproximadamente 3.14% e 2.72% de valores ausentes, respectivamente. A Figura 3.2 ilustra uma representação destes atributos em contraste com os demais atributos do conjunto de dados. Sendo possível observar o preenchimento desde o primeiro exemplo (1) até o último exemplo (49.641), isto é, quanto mais área em branco, mais valores ausentes. Ademais, explorando os exemplos do conjunto de dados, verificou-se a presença de valores ausentes em outros atributos, em menor quantidade. Dessa

[1]As regras aqui aplicadas foram definidas pela especialista de domínio da Embrapa Gado de Corte, qual se embasou em seu trabalho realizado em Amaral et al. (2021).

forma, foi adotada a estratégia de remover todos os exemplos que apresentavam algum valor ausente em sua composição. Sendo removido um total de 2.905 exemplos, restando 46.736 exemplos no conjunto de dados.

Figura 3.2: Representação dos valores ausentes para cada atributo do conjunto de dados.



Fonte: Autoria própria.

Os atributos também podem apresentar valores discrepantes, podendo fazer com que os algoritmos utilizados para extração de padrões não convirjam ou obtenham padrões espúrios (Tan et al., 2014). Dessa forma, foi identificado atributos que apresentaram tal característica. Pode-se observar na Tabela 3.3 alguns exemplos de atributos que ocorrem a discrepância no conjunto de dados. Por exemplo, o atributo `Tot3m_Chuva` possuí valor mínimo de 0.00 e máximo de 1170.24, com um desvio padrão (dp) de 197.88, e uma média de 258.77. Uma das técnicas de transformação de dados utilizadas é a normalização *min-max*, explicada em mais detalhes na Seção 2.1.1. Após aplicada a técnica, os dados originais ficaram em um intervalo entre $[0.0, 1.0]$.

A maioria dos algoritmos de AM não sabem lidar com atributos não numéricos. Sendo assim, foi realizado o tratamento nos atributos categóricos com diferentes técnicas de codificação, procurando transmitir adequadamente as informações de cada tipo de categoria (nominal, ordinal, etc.) (Albon, 2018).

Dessa forma, foi aplicada uma conversão dos atributos categóricos para valores numéricos. Para os atributos categóricos nominais, como a `Tipificacao` ('Macho Inteiro', 'Fêmea' e 'Macho Castrado'), foi utilizada a técnica *One Hot Encoder*, ou seja, foi gerado um atributo binário para cada valor de `Tipifica-`

Tabela 3.3: Atributos que ocorrem discrepâncias de valores antes de aplicar técnica *min-max*.

	Tot3m_Chuva	**Med3m_formITUinst**	**Med3m_NDVI**	**Med3m_preR_milho**	**Med3m_preR_boi**
média	258.77	71.79	0.54	43.11	173.12
dp	197.88	3.51	0.09	10.84	38.42
min	0.00	61.45	0.30	26.57	128.17
máx	1170.24	80.51	0.75	75.66	271.42

Fonte: Autoria própria.

cao. Já para os atributos categóricos ordinais, como QuestionarioClassificacaoEstabel ('0', '21', '26' e '30'), foi utilizada a codificação de ordinal para inteiros, na qual o primeiro valor na ordem será mapeado para o valor 0, e o último da ordem será mapeado para $numero\_de\_valores\_do\_atributo - 1$.

Por fim, para o atributo classe CATEGORIA_BINARIA, composto pelos valores 'Alta qualidade' e 'Baixa qualidade', foi codificado utilizando *Label Encoder* a qual mapeia valores inteiros para os rótulos.

Após aplicada as técnicas de pré-processamento, o conjunto de dados preparado para modelagem ficou com um total de 24 atributos e 46.736 exemplos. A representação do conjunto de dados após pré-processamento, onde são apresentados todos os atributos com o total de exemplos, média, desvio padrão, e os valores mínimo e máximo, pode ser visualizada na Tabela A.2. Ademais, observou-se que o atributo classe, CATEGORIA_BINARIA, possuía as classes distribuídas de forma balanceada[2], com 23.440 exemplos de 'Alta qualidade' e 23.296 exemplos de 'Baixa qualidade'.

3.3 Extração de Padrões

Para extração de padrões dos dados, foram utilizados algoritmos de AM supervisionado para classificação. Os sete modelos de classificação analisados neste trabalho, apresentados na Seção 2.1.2, são das seguintes abordagens: *Naïve Bayes*, Árvores de Decisão, Redes Neurais Artificiais e Máquinas de Vetores de Suporte. Para isso, foram utilizados os seguintes algoritmos: GaussianNB[3], DecisionTreeClassifier[4], RandomForestClassifier[5], XGB-

[2]Antes da aplicação da regra utilizada na agregação, utilizando o atributo CATEGORIA como atributo classe, o conjunto de dados possuía as classes distribuídas de forma desbalanceada, ficando balanceado naturalmente após a agregação.

[3]Link para documentação do algoritmo GaussianNB: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html.

[4]Link para documentação do algoritmo DecisionTreeClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html.

[5]Link para documentação do algoritmo RandomForestClassifier: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

`Classifier`[6], `MLPClassifier`[7], `TabNetClassifier`[8] e `LinearSVC`[9].

Buscando executar os classificadores com os melhores parâmetros possíveis, foi utilizada a técnica de otimização de hiperparâmetros a busca em grade, em inglês *grid search*, qual consiste em uma pesquisa exaustiva sobre valores de parâmetros especificados para um classificador (Huang et al., 2020), detalhado na Seção 2.1.2. Sendo assim, os hiperparâmetros utilizados na busca em grade para os algoritmos aplicados neste trabalho são apresentados na Tabela 3.4. Na qual pode-se observar que cada algoritmo recebe um dicionário com os hiperparâmetros a serem executados. Note que o algoritmo `TabNetClassifier` teve mais combinações de parâmetros, com 768, e o `GaussianNB` teve menos, com 1.

Tabela 3.4: Hiperparâmetros utilizados na busca em grade para os algoritmos analisados.

Algoritmo	**Dicionário de Hiperparâmetros**	**Total de Combinações**
GaussianNB	Padrão[10]	1

DecisionTreeClassifier	`{`
	`splitter: ['best'],`
	`criterion: ['gini', 'entropy'],`
	`min_samples_split: [1, 2, 50, 100],`
	`min_samples_leaf: [1, 5, 10],`
	`max_depth: [1, 4, 7, 10, None],`
	`class_weight: ['balanced', None]`
	`}`
		240

RandomForestClassifier	`{`
	`criterion: ['entropy'],`
	`max_features: [0.75],`
	`n_estimators: [100, 1000],`
	`max_depth: [2, 7, None],`
	`class_weight: ['balanced', None]`
	`}`
		12

Contínua na próxima página

[6]Link para documentação do algoritmo `XGBClassifier`: `https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier`.

[7]Link para documentação do algoritmo `MLPClassifier`: `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`.

[8]Link para documentação do algoritmo `TabNetClassifier`: `https://dreamquark-ai.github.io/tabnet/`.

[9]Link para documentação do algoritmo `LinearSVC`: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html`.

[10]Utilização dos valores padrões dos parâmetros do algoritmo.

Algoritmo	**Dicionário de Hiperparâmetros**	**Total de Combinações**
XGBClassifier	{ tree_method: ['hist'], max_delta_step: [1.0], subsample: [0.75], colsample_bytree: [0.75], n_estimators: [100, 1000], learning_rate: [0.01, 0.1, 0.2], gamma: [0.05, 0.1, 1.0], max_depth: [2, 7, None], reg_lambda: [0.01, 1.0], reg_alpha: [0, 1.0] }	216
MLPClassifier	{ max_iter: [1000], early_stopping: [True], hidden_layer_sizes: [(50, 100, 50), (100,), (200, 100)], activation: ['logistic', 'relu'], solver: ['adam', 'sgd'], alpha: [0.0001, 0.05], learning_rate: ['constant', 'adaptive'], learning_rate_init: [0.0001, 0.001], momentum: [0.0, 0.3, 0.6, 0.8999] }	384
TabNetClassifier	{ optimizer_fn: [Adam], optimizer_params: [{'lr': 0.02}, {'lr': 0.01}, {'lr': 0.001}], scheduler_fn: [StepLR], scheduler_params: [{'step_size': 10, 'gamma': 0.95}], mask_type: ['sparsemax'], n_a: [8, 64], n_steps: [3, 10], gamma: [1.3, 2.0], cat_emb_dim: [10, 20], n_independent: [2, 5], n_shared: [2, 5], momentum: [0.02, 0.4], lambda_sparse: [0.001, 0.1] }	768

Contínua na próxima página

Algoritmo	**Dicionário de Hiperparâmetros**	**Total de Combinações**
LinearSVC	{ dual: [False], penalty: ['l1', 'l2'], C: [0.001, 0.01, 0.1, 1.0], max_iter: [100, 1000, 10000], class_weight: ['balanced', None] }	48

Fonte: Autoria própria.

3.4 Pós-Processamento do Conhecimento

A construção dos modelos de classificação iniciou-se com a separação do conjunto de dados em treinamento e teste. Com o conjunto de dados dividido em 80% para treinamento e 20% para teste, para realizar a escolha do melhor conjunto de parâmetros dos modelos foi utilizada a técnica *k-fold cross validation stratified*, ou validação cruzada estratificada *k-fold*, no qual, são particionados os dados de treinamento em k subconjuntos disjuntos, contendo aproximadamente a mesma porcentagem de amostras de cada classe alvo (o funcionamento detalhado da técnica é abordada na Seção 2.1.3). Neste trabalho foi utilizado $k = 10$.

As medidas de avaliação utilizadas neste trabalho fizeram o uso da matriz de confusão, explicadas em mais detalhes na Seção 2.1.3. Como o problema é de classificação binária, e o atributo classe está balanceado, foi utilizada como medida de avaliação principal a Acurácia.

Utilizando o conjunto de teste com o melhor classificador encontrado na busca em grade, foram extraídas as medidas de avaliação Acurácia, Revocação, Precisão e *F1-score*. Também, foi utilizada as técnicas de Macro-*Averaging* e Micro-*Averaging* para sumarizar os resultados das medidas para as diferentes classes. Ainda, foi criado o gráfico da curva *ROC*, onde foi possível obter o valor da *AUC* para o melhor classificador.

Para avaliar os resultados obtidos entre as diferentes métricas de avaliação dos classificadores, foram realizados testes estatísticos para verificar diferenças significativas entre os resultados.

3.5 Utilização do Conhecimento

O melhor modelo[11], com seus melhores parâmetros, foi escolhido após realizado os testes estatísticos para verificar as diferenças significativas dos resultados das medidas de avaliação. Com isso, o melhor modelo de classificação pode ser utilizado para realizar predição de novos dados.

Para auxiliar o produtor na tomada de decisões, foi desenvolvida uma *API REST*[12], utilizando o *framework web WSGI (Web Server Gateway Interface) Flask* desenvolvido em *Python* (Pallets, 2023), qual carrega o melhor modelo de classificação selecionado neste projeto.

Para realizar a predição as informações dos atributos preditores são recebidas em formato *JSON (Javascript Object Notation)*, como exemplificado na coluna Requisição *JSON* na Tabela 3.5. Então, é realizada a predição e retornada uma resposta em formato *JSON* com os seguintes atributos: `class_probabilities`, que é a probabilidade da classe prevista; `label`, que é o rótulo da classe prevista; e `prediction`, que é o valor numérico da classe prevista. A Tabela 3.5, na coluna Resposta *JSON*, exemplifica a resposta da predição.

Buscando simplificar e acelerar o fluxo de trabalho, a aplicação foi encapsulada em um contêiner *Docker*. O contêiner *Docker* é uma unidade padronizada de *software* que permite aos desenvolvedores isolar seu aplicativo de seu ambiente, facilitando a criação e compartilhamento em diferentes ambientes com múltiplas linguagens, estruturas, arquiteturas e interfaces (Docker, 2023).

3.6 Considerações Finais

Neste capítulo foram abordados temas relacionados ao método de pesquisa, no qual foram apresentadas os detalhes de cada etapa do processo de MD adotados nesse projeto, como a base de dados, as técnicas de pré-processamento aplicadas ao conjunto de dados, a extração de padrões via algoritmos de AM supervisionado para classificação, o pós-processamento do conhecimento via técnicas de avaliação e validação, e a utilização do conhecimento através do melhor modelo escolhido e disponibilizado via uma *API REST*. No próximo capítulo serão apresentados os resultados gerados a partir do método apresentado neste capítulo.

[11] Link para o código-fonte utilizado para treinamento, avaliação, e escolha do melhor modelo: `https://github.com/rafamarquesi/precoce-ms-classification`.

[12] Link para o código-fonte da *API REST*: `https://github.com/rafamarquesi/precoce-ms-classification-api`.

Tabela 3.5: Demonstração de uma requisição e uma resposta para predição utilizando o formato *JSON*.

Requisição *JSON*


```
{
   "tipificacao": "Femea",
   "qtd_animais_lote": 12,
   "peso_medio_lote": 203.26666259765625,
   "questionario_classificacao_estabel": 21,
   "ilp": 0,
   "ifp": 0,
   "ilpf": 0,
   "questionario_possui_outros_incentiv": 0,
   "questionario_fabrica_racao": 0,
   "regua_de_manejo": 1,
   "identificacao_individual": 0,
   "rastreamento_sisbov": 0,
   "participa_de_aliancas_mercadolog": 0,
   "questionario_pratica_recuperacao_pa": 0,
   "confinamento": 0,
   "suplementacao_a_campo": 0,
   "semi_confinamento": 0,
   "tot3m_chuva": 427.54998779296875,
   "med3m_form_itu_inst": 75.47000122070312,
   "med3m_ndvi": 0.5099999904632568,
   "med3m_prer_milho": 75.66000366210938,
   "med3m_prer_boi": 271.4200134277344
}
```


Resposta *JSON*


```
{
"class_probabilities":
[
   {
     "Alta qualidade": 0.769
   },
   {
     "Baixa qualidade": 0.231
   }
],
"label": "Alta qualidade",
"prediction": 0
}
```


Fonte: Autoria própria.

CAPÍTULO

4

Resultados

Neste capítulo será apresentado os resultados obtidos através da aplicação do processo de MD, empregue no conjunto de dados do programa Precoce MS, cujos detalhes foram apresentados no Capítulo 3. Para execução do treinamento dos modelos, foi disponibilizado pela Embrapa Agricultura Digital (CNPTIA) uma máquina em seu ambiente de computação de alto desempenho *High Performance Computing (HPC)*. A máquina possuía 37 processadores e 45.9 *GB* de memória *RAM*. Utilizando tais configurações, a execução levou 2 dias e 8 horas para ser finalizada.

4.1 Análise da Performance de Classificação

Na Tabela 4.1 são apresentadas as maiores médias da acurácia, e o seu respectivo desvio padrão, obtidas pela busca em grade. Pode-se observar que o melhor algoritmo de classificação foi o `RandomForestClassifier`, obtendo uma acurácia de 72.63% e desvio padrão de 0.00620. Os outros dois melhores classificadores foram o `XGBClassifier` e o `MLPClassifier`, porém, mesmo considerando o desvio padrão, não houve sobreposição em relação aos resultados do `RandomForestClassifier`. Os piores classificadores foram o `LinearSVC` e o `GaussianNB`, chegando a uma diferença de 5.846% e 9.171%, respectivamente, de acurácia em relação ao melhor algoritmo. Os hiperparâmetros que obtiveram tais resultados estão disponíveis na Tabela B.1.

Os resultados obtiveram desvio padrão muito baixo, como pode ser observado na Tabela 4.1, para todos os *folds* da validação cruzada aplicadas nos algoritmos. Isso mostra que os resultados para cada *fold* não estão viciados e os dados estão estratificados.

Tabela 4.1: Resultados da média da pontuação da acurácia para validação cruzada estratificada de 10 *folds*, seguidas pelo desvio padrão (±) entre parênteses, para cada algoritmo analisado.

Algoritmo	**Acurácia**
RandomForestClassifier	0.72630 (±0.00620)
XGBClassifier	0.71542 (±0.00950)
MLPClassifier	0.70485 (±0.00869)
TabNetClassifier	0.68902 (±0.00960)
DecisionTreeClassifier	0.68830 (±0.00768)
LinearSVC	0.66784 (±0.00887)
GaussianNB	0.63459 (±0.01147)

Fonte: Autoria própria.

Buscando verificar se houve diferença significativa dos resultados obtidos na validação cruzada entre cada algoritmo demonstrado na Tabela 4.1, foi aplicado o teste estatístico *t* pareado, cujas comparações são apresentadas na Tabela 4.2. Utilizando o nível de significância para o valor de *p* igual a 0.05, isto é, valores menores que isso sugere haver uma diferença significativa entre os algoritmos, é possível observar que apenas a comparação dos algoritmos `TabNetClassifier` e `DecisionTreeClassifier` não apresentaram diferença significativa entre os resultados obtidos nos modelos, com um valor de *p* igual a 0.78553. Ainda, vale ressaltar, que houve diferença significativa entre os dois algoritmos que ficaram melhor ranqueados, o `RandomForestClassifier` e o `XGBClassifier`, com um valor de *p* igual a 0.00084.

Tabela 4.2: Resultados do valor *p* para o teste *t* pareado entre cada algoritmo analisado.

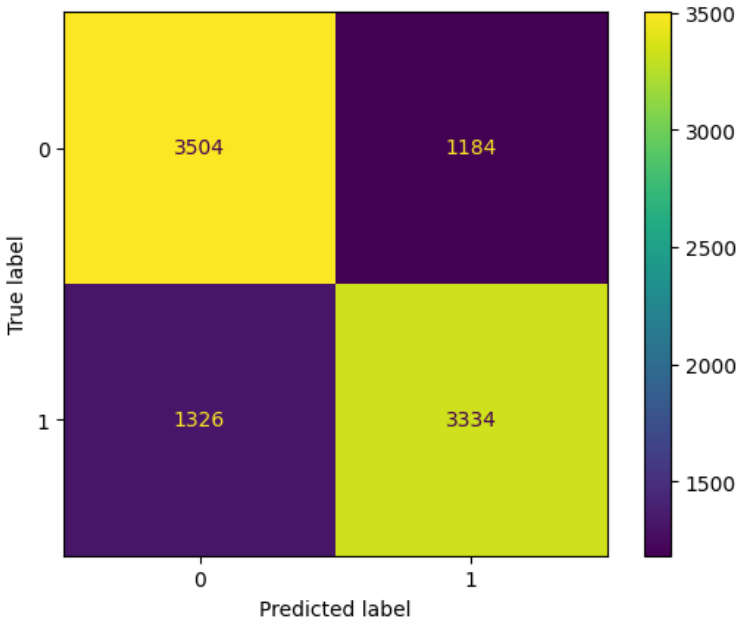
	Random Forest Classifier	**XGB Classifier**	**MLP Classifier**	**TabNet Classifier**	**Decision Tree Classifier**	**LinearSVC**	**GaussianNB**
Random Forest Classifier		0.00084	0.00001	0.00000	0.00000	0.00000	0.00000
XGB Classifier			0.00351	0.00001	0.00000	0.00000	0.00000
MLP Classifier				0.00006	0.00004	0.00000	0.00000
TabNet Classifier					0.78553	0.00003	0.00000
Decision Tree Classifier						0.00001	0.00000
LinearSVC							0.00000
GaussianNB							

Fonte: Autoria própria.

4.2 Resultados Detalhados Para o Algoritmo Selecionado

Vale observar que o `RandomForestClassifier` foi considerado o melhor algoritmo de classificação, com uma acurácia de 72.63%, e foi gerado um modelo, utilizando os melhores hiperparâmetros apresentados na Tabela B.1, a partir de todo o conjunto de dados de treinamento. Dessa forma, foi realizado a avaliação do modelo criado no conjunto de teste, onde foi computada a matriz de confusão demonstrada na Figura 4.1.

Figura 4.1: Matriz de confusão da avaliação do modelo gerado a partir do algoritmo `RandomForestClassifier`.



Fonte: Autoria própria.

A partir da matriz de confusão apresentada na Figura 4.1, foram realizados os cálculos de medidas de avaliação de Precisão, Revocação, F1-*score*, juntamente com suas Macro e Micro *averaging*, e Acurácia. Os resultados para as medidas estão disponíveis na Tabela 4.3, onde, pode-se observar, também, o valor obtido para as medidas para as classes separadamente, representadas por Alta qualidade e Baixa qualidade. Vale notar que os resultados, para as diferentes classes, ficaram bem próximos para todas as medidas calculadas. Demonstrando que o modelo conseguiu generalizar bem para ambas classes. Podendo-se destacar a maior diferença da medida de Revocação, qual Alta qualidade obteve 75% e Baixa qualidade obteve 72%. Já para as medidas utilizando a Macro e Micro *averaging* e a Acurácia, os resultados foram iguais a 73%. A coluna suporte, da Tabela 4.3, refere-se a quantidade de exemplos

utilizados no teste.

Tabela 4.3: Resultados das medidas de avaliação Precisão, Revocação, F1-*score*, Macro e Micro *averaging* e Acurácia para o modelo de classificação gerado com o algoritmo `RandomForestClassifier`.

	Precisão	**Revocação**	**F1-*score***	**suporte**
Alta qualidade	0.73	0.75	0.74	4688
Baixa qualidade	0.74	0.72	0.73	4660
Acurácia			0.73	9348
Macro *avg*	0.73	0.73	0.73	9348
Micro *avg*	0.73	0.73	0.73	9348

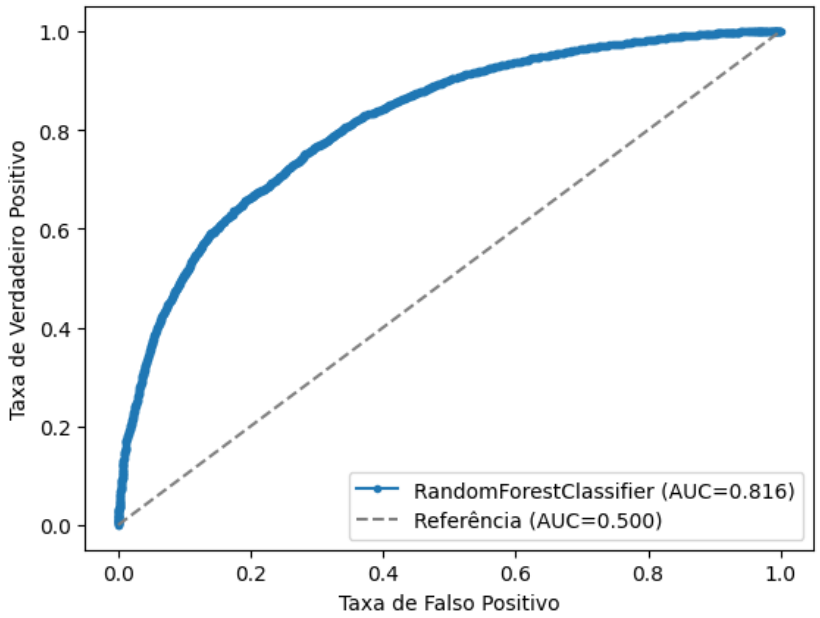
Fonte: Autoria própria.

Buscando verificar a compensação da taxa de verdadeiro positivo entre a taxa de falso positivo, para o modelo gerado a partir do algoritmo `RandomForestClassifier`, foi utilizado o gráfico da curva *ROC*, apresentado na Figura 4.2. Como pode-se observar, o modelo gerou uma curva com direção ao canto superior esquerdo, sendo que, quanto mais próxima à curva estiver do canto superior esquerdo do diagrama, melhor o seu desempenho. Ademais, a subida inicial no canto inferior direito também indica que o classificador acerta mais nos exemplos com maior confiança de classificação. O valor obtido para a *AUC* foi igual a 0.816.

Ainda utilizando o modelo final gerado, foi possível observar a importância dos atributos no treinamento do modelo. A medida de importância de cada atributo foi baseada em sua impureza, também conhecida como importância *Gini*. Sendo assim, os atributos vistos no treinamento, com sua respectiva porcentagem de importância, estão demonstrados na Figura 4.3. Como é possível observar, foram utilizados 24 atributos no treinamento do modelo final, sendo possível encontrar suas respectivas descrições na Tabela A.3. Destes, os que apresentarem maior porcentagem de importância foi o `ILPF`, `QuestionarioFabricaRacao`, `QuestionarioPossuiOutrosIncentiv` e `IFP`, com 16%, 12%, 11% e 11%, respectivamente. Portanto, pode-se observar que os atributos referentes a sistemas de produção sustentável e que integra atividades agrícolas, pecuárias e florestais em uma mesma área[1], a citar o `ILPF` (Integração lavoura-pecuária-floresta) e o `IFP` (Integração floresta-pecuária), como, também, a fabricação de ração na propriedade e possuir outros incentivos

[1] `https://www.embrapa.br/qualidade-da-carne/carne-bovina/producao-de-carne-bovina/sistemas-de-producao-integrados-ilpf`.

Figura 4.2: Curva *ROC* do modelo gerado a partir do algoritmo `RandomForestClassifier`, com sua respectiva *AUC*.



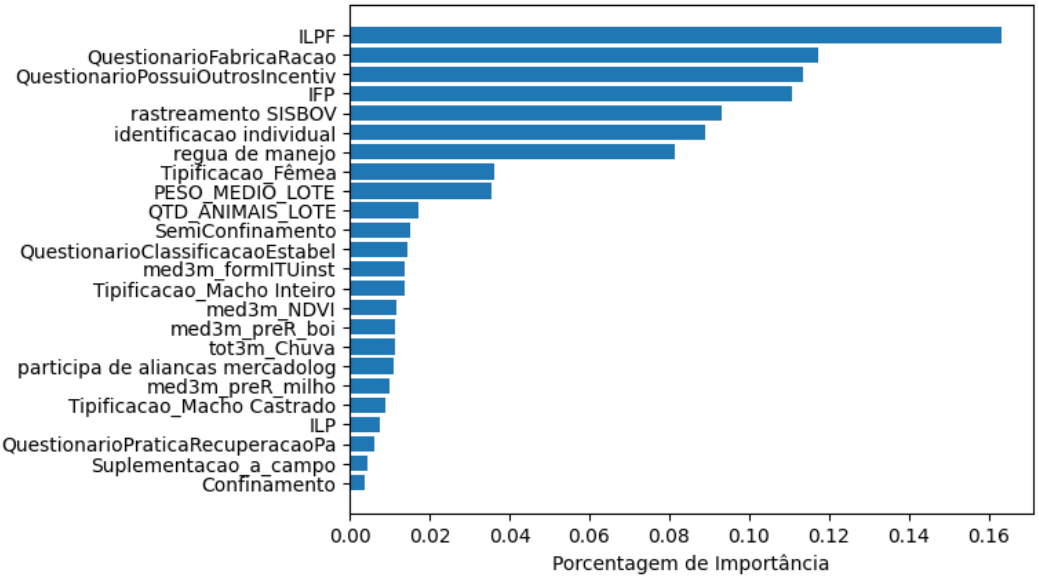
Fonte: Autoria própria.

além do Precoce MS, tiveram grande importância na predição da qualidade do lote de carcaças. Já os atributos que apresentaram a menor porcentagem foi o `Confinamento`, `Suplementacao_a_campo`, `QuestionarioPraticaRecuperacaoPa` e `ILP`, com valor igual ou abaixo de 0.007%. Desta forma, estes atributos são menos importantes na hora de distinguir a qualidade do lote de carcaças.

4.3 Análise de Tempo

Pensando no tempo de treinamento de cada algoritmo considerado neste projeto, são apresentados na Tabela 4.4 os tempos médios de treinamento e de avaliação dos exemplos de testes (em segundos) para validação cruzada de 10 *folds*, com seus respectivos desvios padrões. Vale ressaltar que os menores tempos de treinamento foram obtidos pelo `GaussianNB`, `DecisionTreeClassifier` e `LinearSVC`, porém, estes algoritmos obtiveram menor acurácia conforme a Tabela 4.1. Já os maiores tempos de treinamento ficaram por conta dos algoritmos `TabNetClassifier` e `RandomForestClassifier`. Ainda, utilizando uma média de 3.738 exemplos de testes, pode-se observar que os menores tempos de avaliação foram obtidos pelos algoritmos `DecisionTreeClassifier`, `LinearSVC` e `GaussianNB`, porém, também, estes algoritmos obtiveram menor acurácia conforme a Tabela 4.1. Já os maiores tempos para realizar a avaliação ficaram por conta dos algoritmos `TabNetClassifier` e

Figura 4.3: Importância dos atributos no modelo gerado a partir do algoritmo *Random Forest Classifier*.



Fonte: Autoria própria.

`RandomForestClassifier`.

Tabela 4.4: Resultados da média do tempo de treinamento e do tempo de avaliação em segundos (s), para validação cruzada de 10 *folds*, seguidas pelo desvio padrão (±) entre parênteses, para cada algoritmo analisado.

Algoritmo	**Média Tempo Treinamento**	**Média Tempo Avaliação**
`RandomForestClassifier`	238.48 s (±1.20)	1.39 s (±0.03)
`XGBClassifier`	8.20 s (±0.09)	0.12 s (±0.01)
`MLPClassifier`	15.66 s (±4.09)	0.03 s (±0.01)
`TabNetClassifier`[2]	891.99 s (±154.41)	0.58 s (±0.06)
`DecisionTreeClassifier`	0.34 s (±0.05)	0.03 s (±0.01)
`LinearSVC`	0.36 s (±0.07)	0.03 s (±0.00)
`GaussianNB`	0.10 s (±0.02)	0.04 s (±0.01)

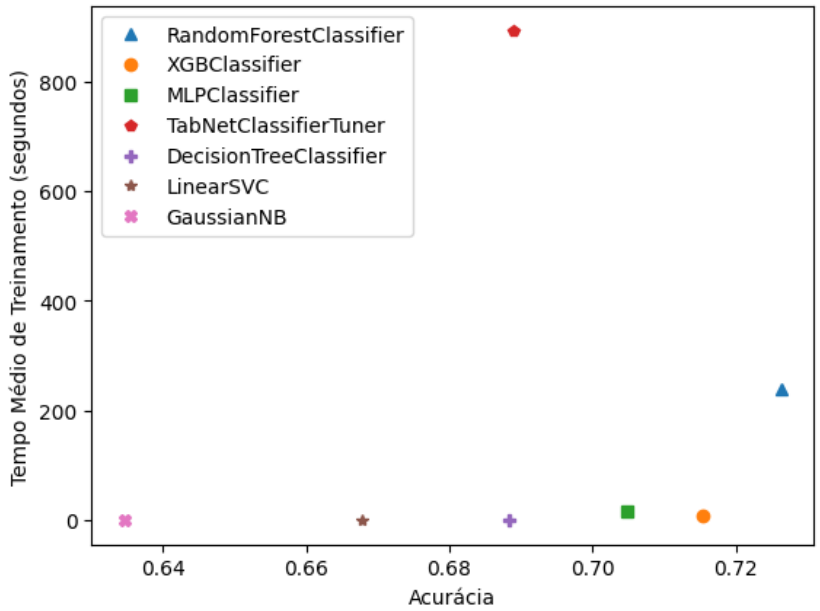
Fonte: Autoria própria.

Na Figura 4.4 é apresentado o gráfico da relação entre o tempo médio de treinamento, em segundos, e a acurácia obtida por cada algoritmo, no qual o eixo x reproduz a acurácia obtida e o eixo y o tempo médio de treinamento. Pode-se destacar que o algoritmo `RandomForestClassifier` obteve a maior acurácia, porém com um tempo de treinamento maior se comparado a se-

[2]Algoritmo executado utilizando *CPUs*.

gunda maior acurácia, obtida pelo algoritmo `XGBClassifier`.

Figura 4.4: Gráfico do tempo médio de treinamento pela acurácia.



Fonte: Autoria própria.

4.4 Considerações Finais

Neste capítulo foram apresentados os resultados para os diferentes algoritmos de classificação utilizados neste trabalho, aplicados ao conjunto de dados do programa Precoce MS, juntamente com seus respectivos tempo de execução, melhores hiperparâmetros, e melhor acurácia. Os algoritmos que obtiveram as melhores acurácias foram o `RandomForestClassifier`, `XGBClassifier` e o `MLPClassifier`. Sendo que foi possível observar que houve diferença significativa entre os resultados obtidos, através do teste *t* pareado. Como o objetivo era obter o algoritmo com resposta mais acurada possível, o `RandomForestClassifier` foi o que obteve a maior acurácia, sendo o escolhido para gerar o modelo a partir de todo o conjunto de treinamento. A partir do modelo gerado com o algoritmo, foram calculadas as medidas de avaliação Precisão, Revocação, F1-*score*, Macro e Micro *averaging* e Acurácia, a partir da matriz de confusão criada utilizando o conjunto de teste. Ademais, foi gerado o gráfico da curva *ROC* e obtido o valor da *AUC* para o modelo. Ainda, foi apresentado a importância de cada atributo no treinamento do modelo final, onde foi possível notar que produtores que praticam a recuperação de pastagem através da integração lavoura-pecuária-floresta e da integração pecuária-floresta, como, também, fabricam ração em sua propriedade e possuí outros

incentivos além do programa Precoce MS, tais características, irão exercer grande influência em determinar a qualidade do lote. Já para produtores que realizam o sistema de confinamento e a suplementação alimentar a campo dos animais, como, também, praticam a recuperação de pastagem, mais especificamente a recuperação de pastagem através da integração lavoura-pecuária, tais características, irão exercer pouca influência em determinar a qualidade do lote. Por fim, pensando no tempo de treinamento, foram apresentados os tempos médio de treinamento e de avaliação de cada algoritmo analisado, com destaque para os algoritmos que obtiveram as piores acurácia. Entretanto, ainda referente ao tempo de treinamento em relação com a acurácia, os algoritmos `XGBClassifier` e o `MLPClassifier` podem ser também utilizados para a finalidade deste projeto, porém com um menor tempo de treinamento.

CAPÍTULO

5

Conclusões

Na busca de estimular a produção de carne com qualidade, foi desenvolvido no estado do Mato Grosso do Sul o programa Precoce MS através do decreto N°14.526, de 28 de julho de 2016 em conjunto com a resolução SEFAZ/SEPAF N°69 de 30 de agosto de 2016. O produtor que produzir um animal com uma boa qualidade de carcaça poderá obter incentivo fiscal de até 67% do ICMS recolhido. Para se obter o incentivo, são avaliados o processo produtivo, o produto obtido, e a padronização do lote.

Buscando auxiliar os produtores na produção de um animal com boa qualidade de carcaça e na tomada de decisões, o objetivo do projeto de pesquisa foi de avaliar qual o algoritmo de aprendizado de máquina com melhor desempenho no conjunto de dados mais atual do programa Precoce MS e, então, construir um modelo de classificação para predizer a qualidade das carcaças do lote de bovinos a serem abatidos, dados parâmetros fornecidos pelo produtor por meio de uma *API REST*.

Desta forma, todas as etapas de um processo de mineração de dados foram aplicas neste projeto: (i) obtenção da base de dados, disponibilizado por uma representante da Embrapa Grado de Corte; (ii) pré-processamento, na qual foi aplicada técnicas para tratar exemplos e atributos que possam impactar negativamente a qualidade dos resultados na extração de padrões; (iii) extração de padrões, na qual foram aplicados uma grande grana de algoritmos e parâmetros; (iv) pós-processamento, na qual foram aplicadas medidas de avaliação, buscando avaliar, validar e consolidar os resultados; (v) utilização do conhecimento, onde foi construída uma *API REST*, para realização de predição da qualidade das carcaças do lote a ser abatido, com o melhor modelo selecionado.

5.1 Principais Resultados e Contribuições

- Geração de uma base de dados com as características necessárias para os experimentos que ficará disponível para outros estudos: onde foi realizado a eliminação de atributos altamente correlacionados, a remoção de exemplos duplicados, a remoção de valores ausentes, a agregação dos animais (exemplos) por lote, a seleção de um subconjunto de atributos junto à especialista de domínio da Embrapa, a aplicação da técnica de normalização *min-max*, e a aplicação das técnicas de codificação *Ordinal Encoder*, *One Hot Encoder*, e *Label Encoder*.

- Execução de algoritmos estado-da-arte e com uma grande variação de parâmetros: onde que os algoritmos `GaussianNB`, `RandomForestClassifier`, e `LinearSVC`, já foram utilizados em trabalhos anteriores, e os algoritmos `DecisionTreeClassifier`, `XGBClassifier`, `MLPClassifier`, e `TabNetClassifier`, foram utilizados pela primeira vez no conjunto de dados do programa Precoce MS.

- Análise de custo benefício: por obter a maior pontuação de acurácia, com 72.63%, o algoritmo `RandomForestClassifier` foi considerado o melhor algoritmo de classificação para o proposto do projeto. No entanto, seu tempo de treinamento foi de 238.48 segundos. Dessa forma, analisando o tempo de treinamento dos algoritmos, pode-se considerar a utilização dos algoritmos `XGBClassifier`, com um tempo de treinamento de 8.20 segundos, e acurácia de 71.54%, e do `MLPClassifier`, com um tempo de treinamento de 15.66 segundos, e acurácia de 70.48%, sem grande perca no poder de predição.

- Análise da importância dos atributos: utilizando o melhor algoritmo selecionado, o modelo foi treinado a partir de 24 atributos preditivos. Destes, os que apresentarem maior porcentagem de importância foi o `ILPF`, `QuestionarioFabricaRacao`, `QuestionarioPossuiOutrosIncentiv` e `IFP`, com 16%, 12%, 11% e 11%, respectivamente. Demonstrando que produtores que praticam a recuperação de pastagem através da integração lavoura-pecuária-floresta e da integração pecuária-floresta, como, também, fabricam ração em sua propriedade e possuí outros incentivos além do programa Precoce MS, tais características, irão exercer grande influência em determinar a qualidade do lote.

- Desenvolvimento de uma *API REST*: buscando auxiliar o produtor na tomada de decisões, foi desenvolvida uma *API REST* utilizando o *framework web WSGI Flask*, para disponibilização do modelo. Onde é possível realizar a predição utilizando os atributos preditores e retornado uma resposta com as informações do rótulo da classe prevista, do valor numérico da classe prevista, e a probabilidade da classe prevista. A *API* foi encapsulada utilizando *Docker*

e encontra-se em fase de implantação na infraestrutura da Embrapa[1].

5.2 Limitações

Houve algumas limitações iniciais na execução do projeto, onde o objetivo era classificar a carcaça de cada animal individualmente. Nesta situação inicial, o conjunto de dados possuía 3.153.593 exemplos e 112 atributos. Além disso, definiu-se duas tarefas de classificação: classificar a carcaça como alta ou baixa qualidade, ou entre as categorias de qualidade ("AAA", "AA", "BBB", "BB", "C" e "D").

Dado o grande volume de dados, a complexidade de alguns algoritmos, como o *TabNet* e a grande combinação dos parâmetros dos algoritmos, houve uma dificuldade em gerar os primeiros resultados devido à falta de recursos computacionais iniciais (neste momento estavam sendo utilizados equipamentos próprios e a máquina virtual disponibilizada pelo *Google Cloud*[2]). Vale ressaltar que foram posteriormente cedidas máquinas na nuvem da Embrapa, uma com 37 *CPUs* e 45.9 *GB* de memória e outra com 120 *CPUs* e 115 *GB* de memória. Entretanto, houve algumas limitações de memória e um alto tempo de execução.

Por fim, decidiu-se por eliminar atributos julgados desnecessários pela especialistas de domínio da Embrapa Gado de Corte e optou-se por fazer a classificação do lote, onde foram gerados dados agregados por lote de animais e com isso diminuiu-se o volume de dados. Também decidiu-se utilizar uma combinação menor de parâmetros em relação à proposta original, o que permitiu que fossem apresentados todos os resultados reportados nesse projeto.

5.3 Trabalhos Futuros

Pensando na melhoria do modelo gerado a partir do conjunto de dados do programa Precoce MS, pode-se realizar em trabalhos futuros uma nova avaliação dos algoritmos com dados mais atualizados e verificar se novos atributos foram criados ao longo do tempo, como, também, examinar métodos de aprendizado incremental, buscando ampliar o conhecimento do modelo, e de inteligência artificial explicável (IAE), buscando explicar o motivo da predição do modelo.

Outro ponto, é possível explorar mais os parâmetros do algoritmo de rede neural profunda utilizada neste projeto, o `TabNetClassifier`. Também, pode-se utilizar outras arquiteturas de redes neurais profundas, como o *Neural*

[1]Link para a página do ambiente de teste *alpha*: `https://bovclass.alpha.cnpgc.embrapa.br/`.

[2]Link para o *Google Cloud*: `https://cloud.google.com/`.

Oblivious Decision Ensembles (NODE). Pensando em dados tabulares, é interessante utilizar outros algoritmos não explorados neste projeto, como o *CatBoost*. Por fim, algumas outras técnicas, como de seleção de atributos e seleção de instâncias, e diferente técnicas de otimização de hiperparâmetros podem ser adotadas.

APÊNDICE

A

Conjunto de Dados

A primeira etapa de um processo de MD é obter o conjunto de dados. Estes dados servem como matéria-prima para um processo de MD. Durante a execução das etapas de pré-processamento, extração de padrões, e pós-processamento do conhecimento, estes dados serão submetidos a diferentes técnicas até serem utilizados para extração de conhecimento. Tais etapas estão explicadas com mais detalhes no Capítulo 2.

O conjunto de dados do programa Precoce MS foi utilizado como matéria-prima para construção deste projeto. Sua representação pode ser visualizada na Tabela A.1, onde se tem os atributos do conjunto de dados, com seu total de exemplos, total de valores ausentes, e cinco exemplos aleatórios.

Tabela A.1: Representação dos atributos do conjunto de dados da base completa, com o total de exemplos, total de valores ausentes e cinco exemplos aleatórios.

Nome Atributo	**Total Exemplos**	**Total Valores Ausentes**	**Cinco Exemplos**
ID_ANIMAL	3153593	0	1133893, 271691, 4197854, 4150478, 2582382,
EstabelecimentoMunicipio	3153593	0	LAGUNA CARAPA, ALCINOPOLIS, SONORA, PEDRO GOMES, RIBAS DO RIO PARDO,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
DataAbate	3153593	0	2018-01-09T00:00:00.000000000, 2020-12-16T00:00:00.000000000, 2020-01-22T00:00:00.000000000, 2018-01-18T00:00:00.000000000, 2020-07-10T00:00:00.000000000,
Frigorifico_ID	3153593	0	26, 40, 15, 53, 33,
Frigorifico_CNPJ	3153593	0	13837014000106, 2916265007768, 2916265007768, 2916265007768, 2916265023968,
Frigorifico_RazaoSocial	3153593	0	MARFRIG GLOBAL FOODS SA, JBS SA, AGROINDUSTRIAL IGUATEMI EIRELI, MARFRIG GLOBAL FOODS SA, JBS S/A,
Municipio_Frigorifico	3153593	0	CAMPO GRANDE, NAVIRAI, CAMPO GRANDE, CAMPO GRANDE, NAVIRAI,
Tipificacao	3153593	0	Macho Inteiro, Fêmea, Fêmea, Macho Inteiro, Macho Inteiro,
Maturidade	3153593	0	2, 2, 2, d, 4,
Acabamento	3153593	0	Gordura Mediana - Acima De 3 A Até 6 Mm De Espessura, Gordura Mediana - Acima De 3 A Até 6 Mm De Espessura, Gordura Mediana - Acima De 3 A Até 6 Mm De Espessura, Gordura Mediana - Acima De 3 A Até 6 Mm De Espessura, Gordura Mediana - Acima De 3 A Até 6 Mm De Espessura,
Peso	3153593	0	250.0, 300.0, 285.8, 187.4, 203.0,
EstabelecimentoIdentificador	3153593	0	1283, 5279, 4181, 1736, 1264,
Data_homol	3153593	0	2018-01-09T00:00:00.000000000, 2019-03-01T00:00:00.000000000, 2017-08-22T00:00:00.000000000, 2017-04-10T00:00:00.000000000, 2017-07-27T00:00:00.000000000,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
Questionario_ID	3153593	0	3872, 1260, 9636, 7034, 12486,
QuestionarioClassificacaoEstabel	3153593	0	21, 21, 30, 21, 21,
FERTIIRRIGACAO	3153593	0	0, 0, 0, 0, 1,
ILP	3153593	0	0, 0, 1, 0, 0,
IFP	3153593	0	0, 0, 0, 0, 0,
ILPF	3153593	0	0, 0, 0, 0, 0,
CONCEN_VOLUM	3153593	464	0, 0, 0, 0, 0,
CREEPFEEDING	3153593	464	0, 0, 0, 0, 0,
FORN_ESTRAT_SILAGEM	3153593	464	0, 0, 0, 0, 0,
PROTEICO	3153593	464	1, 0, 1, 1, 0,
PROTEICO_ENERGETICO	3153593	464	1, 0, 0, 0, 0,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
RACAO_BAL_CONS_INFERIOR	3153593	464	0, 0, 0, 1, 0,
SAL_MINERAL	3153593	464	0, 0, 1, 1, 1,
SALMINERAL_UREIA	3153593	464	0, 0, 0, 0, 0,
RACAOO_BAL_CONSUMO_IG	3153593	464	1, 1, 0, 1, 0,
GRAO_INTEIRO	3153593	464	0, 0, 0, 0, 0,
ALTO_CONCENTR_VOLUM	3153593	464	0, 0, 0, 1, 0,
ALTO_CONCENTRADO	3153593	464	0, 0, 0, 0, 0,
QuestionarioPossuiOutrosIncentiv	3153593	0	0, 0, 0, 0, 0,
QuestionarioFabricaRacao	3153593	0	1, 0, 0, 1, 0,
area so confinamento	3153593	2991235	<NA>, <NA>, <NA>, <NA>, <NA>,
regua de manejo	3153593	464	1, 1, 1, 1, 0,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
boa cobertura vegetal, com baixa	3153593	610286	1, 1, 1, 1, 1,
erosaoo laminar ou em sulco igua	3153593	610286	0, <NA>, 0, 0, 0,
identificacao individual	3153593	464	1, 1, 0, 1, 1,
rastreamento SISBOV	3153593	920	0, 0, 0, 0, 1,
Lista Trace	3153593	1911149	<NA>, 1, 0, 1, <NA>,
BPA	3153593	464	0, 0, 0, 0, 0,
participa de aliancas mercadolog	3153593	464	0, 0, 0, 0, 0,
QuestionarioPraticaRecuperacaoPa	3153593	0	1, 0, 0, 1, 1,
Confinamento	3153593	464	1, 1, 1, 0, 0,
Suplementacao_a_campo	3153593	464	0, 1, 1, 1, 1,
SemiConfinamento	3153593	464	0, 0, 1, 0, 0,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
dif_datas	3153593	0	173, 447, 427, 418, 486,
DataAbate_6m_ANT	3153593	0	2018-11-22T00:00:00.000000000, 2018-11-07T00:00:00.000000000, 2019-10-02T00:00:00.000000000, 2018-07-29T00:00:00.000000000, 2019-01-16T00:00:00.000000000,
data_homol_select	3153593	0	2017-03-30T00:00:00.000000000, 2018-10-10T00:00:00.000000000, 2017-04-27T00:00:00.000000000, 2017-09-15T00:00:00.000000000, 2020-03-11T00:00:00.000000000,
data12m	3153593	0	2019-09-10T00:00:00.000000000, 2018-01-23T00:00:00.000000000, 2019-06-11T00:00:00.000000000, 2019-10-22T00:00:00.000000000, 2019-10-23T00:00:00.000000000,
data6m	3153593	0	2020-04-19T00:00:00.000000000, 2020-03-09T00:00:00.000000000, 2018-05-09T00:00:00.000000000, 2017-05-02T00:00:00.000000000, 2020-02-20T00:00:00.000000000,
data3m	3153593	0	2019-06-10T00:00:00.000000000, 2019-10-17T00:00:00.000000000, 2018-05-14T00:00:00.000000000, 2019-02-04T00:00:00.000000000, 2019-12-11T00:00:00.000000000,
data1m	3153593	0	2019-09-20T00:00:00.000000000, 2020-04-11T00:00:00.000000000, 2018-10-21T00:00:00.000000000, 2019-07-28T00:00:00.000000000, 2017-10-27T00:00:00.000000000,
data7d	3153593	0	2017-10-10T00:00:00.000000000, 2017-08-03T00:00:00.000000000, 2018-11-13T00:00:00.000000000, 2020-06-14T00:00:00.000000000, 2019-11-11T00:00:00.000000000,
tot7d_Chuva	3153593	297982	92.22, 35.88, 0.0, 0.53, nan,
med7d_TempInst	3153593	527	24.67, 27.73, 27.44, 26.64, 16.87,
med7d_TempMin	3153593	527	25.03, 27.09, 24.72, 27.21, 23.46,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
med7d_UmidInst	3153593	527	79.96, 84.88, 53.07, 73.89, 71.85,
med7d_formITUinst	3153593	527	73.48, 67.57, 73.33, 70.11, 69.28,
med7d_formITUmax	3153593	527	74.83, 82.28, 79.16, 77.13, 68.43,
med7d_NDVI	3153593	109552	0.62, 0.54, 0.63, 0.56, 0.5,
med7d_EVI	3153593	109552	0.41, 0.45, 0.2, 0.44, 0.25,
med7d_preR_soja	3153593	527	104.86, 101.29, 147.99, 75.32, 79.32,
med7d_preR_milho	3153593	527	47.09, 36.65, 71.79, 76.42, 37.46,
med7d_preR_boi	3153593	527	226.62, 285.81, 249.67, 149.97, 127.77,
tot1m_Chuva	3153593	211967	23.23, 142.84, 76.26, 36.97, 55.94,
med1m_TempInst	3153593	527	33.96, 25.98, 19.98, 20.54, 26.48,
med1m_UmidInst	3153593	527	74.87, 76.88, 49.86, 67.85, 74.11,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
med1m_formITUinst	3153593	527	68.86, 72.99, 75.63, 67.65, 72.82,
med1m_NDVI	3153593	109552	0.59, 0.58, 0.51, 0.53, 0.37,
med1m_EVI	3153593	109552	0.27, 0.41, 0.33, 0.45, 0.42,
med1m_preR_soja	3153593	527	134.87, 71.45, 76.7, 115.58, 88.25,
med1m_preR_milho	3153593	527	49.76, 49.79, 38.05, 47.83, 37.44,
med1m_preR_boi	3153593	527	145.9, 251.79, 205.33, 156.92, 140.28,
tot3m_Chuva	3153593	70611	541.37, 160.85, 433.82, 208.14, 567.63,
med3m_TempInst	3153593	527	21.21, 25.81, 22.64, 23.8, 20.51,
med3m_UmidInst	3153593	527	70.18, 65.3, 66.34, 65.95, 60.48,
med3m_formITUinst	3153593	527	77.08, 75.61, 66.98, 74.9, 74.72,
med3m_formITUmax	3153593	527	76.8, 78.33, 70.22, 67.55, 73.75,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
med3m_NDVI	3153593	109552	nan, 0.55, 0.53, 0.4, 0.48,
med3m_EVI	3153593	109552	0.21, 0.46, 0.29, 0.28, nan,
med3m_preR_soja	3153593	527	70.56, 70.52, 125.93, 88.57, 77.59,
med3m_preR_milho	3153593	527	49.02, 32.18, 41.19, 37.91, 37.66,
med3m_preR_boi	3153593	527	157.02, 263.36, 159.2, 134.83, 153.81,
tot6m_Chuva	3153593	22724	364.4, 818.7, 1090.9, 243.39, 164.89,
med6m_TempInst	3153593	527	24.48, 22.92, 24.58, 26.0, 19.85,
med6m_UmidInst	3153593	527	60.05, 49.32, 67.17, 61.36, 58.86,
med6m_formITUinst	3153593	527	70.4, 74.38, 70.13, 76.03, 74.18,
med6m_NDVI	3153593	109552	0.44, 0.56, 0.5, 0.58, 0.64,
med6m_EVI	3153593	109552	nan, 0.3, 0.36, 0.4, 0.3,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
med6m_preR_soja	3153593	527	78.41, 115.85, 88.47, 91.95, 70.15,
med6m_preR_milho	3153593	527	35.99, 37.1, 35.91, 55.84, 37.35,
med6m_preR_boi	3153593	527	185.04, 143.54, 199.86, 157.71, 154.88,
tot12m_Chuva	3153593	4424	1128.91, 1139.48, 1540.78, 957.69, 1107.99,
med12m_TempInst	3153593	527	25.2, 23.95, 22.68, 22.78, 23.18,
med12m_TempMin	3153593	527	21.78, 23.85, 23.75, 24.02, 21.73,
med12m_UmidInst	3153593	527	69.16, 63.02, 66.52, 69.31, 66.63,
med12m_formITUinst	3153593	527	72.07, 72.53, 72.7, 69.77, 70.9,
med12m_NDVI	3153593	109552	0.52, 0.55, 0.55, 0.58, 0.52,
med12m_EVI	3153593	109552	0.4, 0.39, 0.39, 0.34, 0.38,
med12m_preR_soja	3153593	527	71.0, 81.96, 106.49, 99.88, 84.43,

Contínua na próxima página

Nome Atributo	Total Exemplos	Total Valores Ausentes	Cinco Exemplos
med12m_preR_milho	3153593	527	52.55, 49.1, 52.91, 37.62, 30.07,
med12m_preR_boi	3153593	527	224.4, 196.01, 145.46, 161.35, 144.17,
cnt7d_CL_ITUinst	3153593	527	0.5, 0.0, 0.0, 0.0, 0.0,
cnt1m_CL_ITUinst	3153593	527	0.8065, 0.0968, 0.0323, 0.0, 0.0,
cnt3m_CL_ITUinst	3153593	527	0.0, 0.0, 0.2688, 0.0, 0.129,
cnt6m_CL_ITUinst	3153593	527	0.0, 0.0383, 0.0328, 0.6284, 0.1913,
cnt12m_CL_ITUinst	3153593	527	0.0, 0.1011, 0.1612, 0.0, 0.388,
ANO	3153593	0	2019, 2020, 2018, 2020, 2018,
CATEGORIA	3153593	0	AA, BB, BBB, D, BB,
classificacao	3153593	0	NÃO, SIM, SIM, SIM, SIM,
Motivo	3153593	0	Classificado, Classificado, Classificado, Classificado, Classificado,

Fonte: Autoria própria.

Como dito anteriormente, uma das etapas de um processo de MD é o pré-processamento. Após a finalização desta etapa, o conjunto de dados pode ter sofrido muitas modificações. Pode-se visualizar na Tabela A.2 a representação do conjunto de dados do programa Precoce MS após a etapa de pré-processamento, onde é demonstrado os atributos, com seu total de exemplos, sua média, seu desvio padrão, seu valor mínimo, e seu valor máximo.

Tabela A.2: Representação dos atributos do conjunto de dados após pré-processamento, com o total de exemplos, média, desvio padrão, valor mínimo e valor máximo.

Nome Atributo	**Total Exemplos**	**Média**	**Desvio Padrão**	**Mínimo**	**Máximo**
QTD_ANIMAIS_LOTE	46736	0.06	0.08	0.00	1.00
PESO_MEDIO_LOTE	46736	0.23	0.09	0.00	1.00
QuestionarioClassificacaoEstabel	46736	1.60	0.74	0.00	3.00
ILP	46736	0.39	0.49	0.00	1.00
IFP	46736	0.05	0.22	0.00	1.00
ILPF	46736	0.03	0.18	0.00	1.00
QuestionarioPossuiOutrosIncentiv	46736	0.03	0.18	0.00	1.00
QuestionarioFabricaRacao	46736	0.54	0.50	0.00	1.00
regua de manejo	46736	0.80	0.40	0.00	1.00
identificacao individual	46736	0.66	0.47	0.00	1.00
rastreamento SISBOV	46736	0.28	0.45	0.00	1.00
participa de aliancas mercadolog	46736	0.26	0.44	0.00	1.00
QuestionarioPraticaRecuperacaoPa	46736	0.55	0.50	0.00	1.00
Confinamento	46736	0.42	0.49	0.00	1.00
Suplementacao_a_campo	46736	0.90	0.30	0.00	1.00
SemiConfinamento	46736	0.57	0.50	0.00	1.00
tot3m_Chuva	46736	0.06	0.05	0.00	1.00
med3m_formITUinst	46736	0.55	0.19	0.00	1.00
med3m_NDVI	46736	0.55	0.20	0.00	1.00
med3m_preR_milho	46736	0.35	0.21	0.00	1.00
med3m_preR_boi	46736	0.33	0.26	0.00	1.00
Tipificacao_Fêmea	46736	0.33	0.47	0.00	1.00
Tipificacao_Macho Castrado	46736	0.34	0.47	0.00	1.00
Tipificacao_Macho Inteiro	46736	0.32	0.47	0.00	1.00
CATEGORIA_BINARIA	46736	0.50	0.50	0.00	1.00

Fonte: Autoria própria.

Buscando elucidar dúvidas sobre os atributos, após aplicada a etapa de pré-processamento, é possível visualizar na Tabela A.3 os atributos com suas respectivas descrições.

Tabela A.3: Descrição dos atributos do conjunto de dados após pré-processamento.

Nome Atributo	**Descrição**
QTD_ANIMAIS_LOTE	Quantidade de animais abatido no lote.
PESO_MEDIO_LOTE	Peso médio do lote de animais abatido.
QuestionarioClassificacaoEstabel	Classificação do produtor em relação ao sistema produtivo, podendo ser considerado sem classificação (0), simples (21), intermediário (26) e avançado (30).
ILP	Prática recuperação de pastagem através da Integração Lavoura-Pecuária.
IFP	Prática recuperação de pastagem através da Integração Pecuária-Floresta.
ILPF	Prática recuperação de pastagem através da Integração Lavoura-Pecuária-Floresta.
QuestionarioPossuiOutrosIncentiv	Produtor possuí outros incentivos além do Precoce MS.
QuestionarioFabricaRacao	O produtor fabrica ração na propriedade.
regua de manejo	O produtor utiliza a régua de manejo desenvolvida pela Embrapa para controle de entrada e saída dos animais.
identificacao individual	O produtor realiza a identificação individual dos animais desde o nascimento até a comercialização do produto final.
rastreamento SISBOV	O produtor faz a rastreabilidade dos animais pelo SISBOV.
participa de aliancas mercadolog	Participação em associações de produtores visando à produção comercial sistematizada e organizada conforme padrões pré-estabelecidos para atendimento de acordos comerciais (acordos mercadológicos).
QuestionarioPraticaRecuperacaoPa	Pratica recuperação de pastagem.
Confinamento	Pratica o sistema de confinamento.
Suplementacao_a_campo	Pratica suplementação alimentar a campo dos animais.
SemiConfinamento	Pratica o sistema de semi confinamento.
tot3m_Chuva	Média do total de precipitação em milímetros durante os 3 últimos meses que antecederam o abate do animal.

Contínua na próxima página

Nome Atributo	Descrição
med3m_formITUinst	Média do total do índice de temperatura e umidade instantânea durante os 3 últimos meses que antecederam o abate do animal.
med3m_NDVI	Média do total do índice de vegetação da diferença normalizada, que serve para analisar a condição da vegetação natural ou agrícola nas imagens geradas por sensores remotos, durante os 3 últimos meses que antecederam o abate do animal.
med3m_preR_milho	Média do total do *commodities* do preço do milho durante os 3 últimos meses que antecederam o abate do animal.
med3m_preR_boi	Média do total do *commodities* do boi gordo durante os 3 últimos meses que antecederam o abate do animal.
Tipificacao_Fêmea	Animal do tipo fêmea.
Tipificacao_Macho Castrado	Animal do tipo macho castrado.
Tipificacao_Macho Inteiro	Animal do tipo macho inteiro.
CATEGORIA_BINARIA	Categoria do lote, alta qualidade ou baixa qualidade.

Fonte: Autoria própria.

APÊNDICE

B

Parâmetros Algoritmos

O ajuste de parâmetros é uma etapa importante executada durante um processo de MD. Encontrar os melhores hiperparâmetros, dos algoritmos, durante a construção de um modelo pode resultar em melhoria no desempenho e na sua avaliação, como explicado em mais detalhes na Seção 2.1.2.

Após aplicada a técnica de ajuste de parâmetros foi obtido os hiperparâmetros que obtiveram a melhor pontuação de acurácia, para cada algoritmo analisado neste trabalho. Na Tabela B.1 é possível observar cada algoritmo com seus respectivos melhores hiperparâmetros.

Tabela B.1: Hiperparâmetros que obtiveram melhor pontuação de acurácia, para cada algoritmo analisado.

Algoritmo	**Hiperparâmetros**
`RandomForestClassifier`	[criterion: 'entropy', max_features: 0.75, n_estimators: 1000, max_depth: None, class_weight: None]
`XGBClassifier`	[tree_method: 'hist', max_delta_step: 1.0, subsample: 0.75, colsample_bytree: 0.75, n_estimators: 1000, learning_rate: 0.1, gamma: 1.0, max_depth: 7, reg_lambda: 0.01, reg_alpha: 1.0]

Contínua na próxima página

Algoritmo	Hiperparâmetros
MLPClassifier	[max_iter: 1000, early_stopping: True, hidden_layer_sizes: (200, 100), activation: 'relu', solver: 'adam', alpha: 0.0001, learning_rate: 'adaptive', learning_rate_init: 0.001, momentum: 0.6]
TabNetClassifier	[optimizer_fn: Adam, optimizer_params: 'lr': 0.02, scheduler_fn: StepLR, scheduler_params: 'step_size': 10, 'gamma': 0.95, mask_type: 'sparsemax', n_a: 64, n_steps: 3, gamma: 2.0, cat_emb_dim: 10, n_independent: 5, n_shared: 5, momentum: 0.4, lambda_sparse: 0.001]
DecisionTreeClassifier	[splitter: 'best', criterion: 'gini', min_samples_split: 100, min_samples_leaf: 10, max_depth: 10, class_weight: None]
LinearSVC	[dual: False, penalty: 'l1', C: 0.01, max_iter: 100, class_weight: None]
GaussianNB	Padrão[1]

Fonte: Autoria própria.

[1] Utilização dos valores padrões dos parâmetros do algoritmo.

Referências Bibliográficas

Aggarwal, C. C. (2014). *Data Classification - algorithms and applications*. CRC Press, New York, NY, 1thª edição. Citado nas páginas 12, 18, 19, 20, 22, 24, 25, 29, 30, 31, 32, 34, 36, 37, e 41.

Albon, C. (2018). *Python Machine Learning CookBook - Practical Solutions from Preprocessing to Deep Learning*. O'Reilly Media, Inc., 1thª edição. Citado nas páginas 15, 40, e 53.

Ali, I., Cawkwell, F., Dwyer, E., e Green, S. (2017). Modeling managed grassland biomass estimation by using multitemporal remote sensing data—a machine learning approach. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(7):3254–3264. Citado na página 2.

Alonso, J., Ángel Rodríguez Castañón, e Bahamonde, A. (2013). Support vector regression to predict carcass weight in beef cattle in advance of the slaughter. *Computers and Electronics in Agriculture*, 91:116–120. Citado nas páginas 2 e 44.

Amaral, T. B. e Gomes, R. (2020). Artigo: Potencial de produção de novilho precoce no estado de ms. `https://www.embrapa.br/busca-de-noticias/-/noticia/53407929/artigo-potencial-de-producao-de-novilho-precoce-no-estado-de-ms`. Acessado em: 19/06/2021. Citado na página 1.

Amaral, T. B., Gomes, R. C., e Rosa, G. J. M. (2021). Produção de Novilho Precoce no Estado de Mato Grosso do Sul: Análise exploratória de dados do Programa PROAPE-Precoce MS. página 28. Citado na página 52.

Arık, S. e Pfister, T. (2021). TabNet: Attentive Interpretable Tabular Learning. *35th AAAI Conference on Artificial Intelligence, AAAI 2021*, 8A:6679–6687. Citado nas páginas 32, 33, 34, e 35.

Boschetti, A. e Massaron, L. (2016). *Python Data Science Essentials*. Packt Publishing Ltd., Birmingham, UK, 2thª edição. Citado na página 40.

Brownlee, J. (2018). *MACHINE LEARNING ALGORITHMS FROM SCRATCH - With Python*. v1.7ª edição. Citado na página 21.

Brownlee, J. (2019). *Statistical Methods for Machine Learning*. v1.4ª edição. Citado na página 51.

Cáceres, E. N., Pistori, H., Turine, M. A. S., Pires, P. P., Soares, C. O., e Carromeu, C. (2011). Computational precision livestock-position paper. *II Workshop of the Brazilian Institute for Web Science Research*, (02-03):9. Citado na página 2.

Chakrabarti, S., Cox, E., Frank, E., Güting, R. H., Han, J., Refaat, M., Pyle, D., Schneider, M., Jiang, X., Teorey, T. J., Kamber, M., Lightstone, S. S., Nadeau, T. P., Neapolitan, R. E., e Witten, I. H. (2008). *DATA MINING: Know it all*. Morgan Kaufmann, 1thª edição. Citado nas páginas 8, 9, 10, 13, 14, 15, e 19.

Chen, T. e Guestrin, C. (2015). Xgboost: Reliable large-scale tree boosting system. In *Proceedings of the 22nd SIGKDD Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA*, páginas 13–17. Citado na página 28.

Chen, T. e Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 13-17-August-2016:785–794. Citado nas páginas 26, 27, 28, e 29.

Docker (2023). Why docker | docker. `https://www.docker.com/why-docker/`. Acessado em: 04/04/2023. Citado na página 58.

Dutta, R., Smith, D., Rawnsley, R., Bishop-Hurley, G., Hills, J., Timms, G., e Henry, D. (2015). Dynamic cattle behavioural classification using supervised ensemble classifiers. *Computers and Electronics in Agriculture*, 111:18–28. Citado na página 2.

García, S., Luengo, J., e Herrera, F. (2015). *Data Preprocessing in Data Mining*, volume 72. Springer International Publishing Switzerland, 1thª edição. Citado nas páginas 8, 9, 10, 11, 12, 13, 14, 15, e 20.

Governo do Estado de Mato Grosso do Sul (2016). Decreto n°14.526, de 28 de julho de 2016. `https://www.spdo.ms.gov.br/diariodoe/Index/Download/42493`. Acessado em: 20/06/2021. Citado na página 2.

Guaraldo, M. C. (2021). Brasil é o quarto maior produtor de grãos e o maior exportador de carne bovina do mundo, diz estudo. https://www.embrapa.br/busca-de-noticias/-/noticia/62619259/brasil-e-o-quarto-maior-produtor-de-graos-e-o-maior-exportador-de-carne-bovina-do-mundo-diz-estudo. Acessado em: 19/06/2021. Citado na página 1.

Han, J., Pei, J., e Kamber, M. (2011). *Data mining: Concepts and Techniques*. Morgan Kaufmann, 3thª edição. Citado nas páginas 8, 9, 10, 11, 12, 13, 17, 18, 21, 22, e 25.

Haroon, D. (2017). *Python Machine Learning Case Studies: Five Case Studies for the Data Scientist*. Apress Berkeley, CA, 1thª edição. Citado nas páginas 26 e 51.

Hoos, H. H. (2011). Automated algorithm configuration and parameter tuning. In *Autonomous search*, páginas 37–71. Springer. Citado na página 38.

Huang, C., Li, Y., e Yao, X. (2020). A survey of automatic parameter tuning methods for metaheuristics. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 24:201–216. Citado nas páginas 37, 38, 39, e 55.

IZBICKI, R. e SANTOS, T. M. D. (2020). *Aprendizado de máquina : uma abordagem estatística*. São Carlos, SP, 1thª edição. Citado na página 31.

Lee, W., Kim, S. H., Ryu, J., e Ban, T.-W. (2017). Fast detection of disease in livestock based on deep learning. *Journal of the Korea Institute of Information and Communication Engineering*, 21(5):1009–1015. Citado na página 2.

Manning, C. D., Raghavan, P., e Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, England. Citado na página 42.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/-Math, 1thª edição. Citado nas páginas 15, 19, 20, e 22.

Mota, F. M. d. (2016). Uma abordagem de análises olap e de mineração de dados para suporte à tomada de decisão no setor da pecuária de corte do brasil. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul. Citado nas páginas 3, 45, e 46.

Nucci, H. H. P. (2019). Classificação do grau de acabamento de gordura da carcaça de bovinos de corte usando aprendizado de máquina. Dissertação de Mestrado, Universidade Federal de Mato Grosso do Sul. Citado nas páginas 3, 45, e 46.

Ozdemir, S. (2016). *Three Principles of Data Science*. Packt Publishing Ltd. Citado na página 16.

Paixão, L. C. A. e Almeida, M. M. Y. d. (2020). Carne bovina : expansão nas exportações e exigências internacionais. *Revista Interface Tecnológica*, 17(2):556–566. Citado nas páginas 1 e 2.

Pallets (2023). Flask | the pallets projects. `https://palletsprojects.com/p/flask/`. Acessado em: 04/04/2023. Citado na página 58.

Parmezan, A. R. S., Lee, H. D., Spolaôr, N., e Chung, W. F. (2012). Avaliação de métodos para seleção de atributos importantes para aprendizado de máquina supervisionado no processo de mineração de dados. Citado nas páginas 8, 15, 16, 18, e 39.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., e Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. Citado na página 15.

Raschka, S. (2015). *Python ML*. Packt Publishing Ltd. Citado nas páginas 16 e 39.

Russel, S. e Norvig, P. (2010). *Artificial Intelligence, A Modern Approach*. Prentice Hall, 3thª edição. Citado nas páginas 17, 19, e 20.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys*, 34(1):1–47. Citado na página 42.

SEFAZ (2016). Resolução conjunta sefaz/sepaf n°69 de 30 de agosto de 2016. `https://www.legisweb.com.br/legislacao/?id=328328`. Acessado em: 20/06/2021. Citado na página 2.

Shahinfar, S., Al-Mamun, H. A., Park, B., Kim, S., e Gondro, C. (2020). Prediction of marbling score and carcass traits in korean hanwoo beef cattle using machine learning methods and synthetic minority oversampling technique. *Meat science*, 161:107997. Citado nas páginas 44, 45, e 46.

Shahinfar, S., Kelman, K., e Kahn, L. (2019). Prediction of sheep carcass traits from early-life records using machine learning. *Computers and electronics in agriculture*, 156:159–177. Citado nas páginas 43 e 46.

Shahinfar, S., Page, D., Guenther, J., Cabrera, V., Fricke, P., e Weigel, K. (2014). Prediction of insemination outcomes in Holstein dairy cattle using alternative machine learning algorithms. *Journal of Dairy Science*, 97(2):731–742. Citado na página 42.

Sokolova, M. e Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 45(4):427–437. Citado na página 42.

Soulat, J., Picard, B., Léger, S., e Monteils, V. (2018). Prediction of beef carcass and meat quality traits from factors characterising the rearing management system applied during the whole life of heifers. *Meat science*, 140:88–100. Citado nas páginas 43, 44, e 46.

Tan, P.-N., Steinbach, M., e Kumar, V. (2014). *Introduction to Data Mining*. Pearson, 1thª edição. Citado nas páginas 7, 8, 10, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 25, 29, 30, 31, 32, 34, 35, 36, 37, 39, 40, 41, 42, e 53.

Witten, I. H. e Frank, E. (2005). *Data mining : practical machine learning tools and techniques*. Morgan Kaufmann Publishers, 2thª edição. Citado nas páginas 7, 15, 21, 22, 23, 24, 30, 31, 32, 40, e 52.

Zaki, M. J. e Jr., W. M. (2014). *DATA MINING AND ANALYSIS Fundamental Concepts and Algorithms*. Cambridge University Press, New York, NY. Citado na página 17.