DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Aprendizagem de máquina para apoio à tomada de decisão em vendas do varejo utilizando registros de vendas

Relatório submetido à Universidade Federal de Santa Catarina
como requisito para a aprovação da disciplina:
DAS 5511: Projeto de Fim de Curso

Felippe Schmoeller da Roza

Florianópolis, Setembro de 2016

Aprendizagem de máquina para apoio à tomada de decisão em vendas do varejo utilizando registros de vendas

Felippe Schmoeller da Roza

Esta monografia foi julgada no contexto da disciplina

DAS 5511: Projeto de Fim de Curso

e aprovada na sua forma final pelo

Curso de Engenharia de Controle e Automação

Prof. Eduardo Camponogara

Banca Examinadora:

Fabiano Tristão
Orientador na Empresa

Prof. Eduardo Camponogara
Orientador no Curso

Prof. Hector Bessa Silveira
Responsável pela disciplina

Maicon Rafael Zatelli, Avaliador

Andrio Renan Gonzatti Frizon, Debatedor

Hugo Gutierrez Sigolo, Debatedor

For even the very wise cannot
see all ends.

J. R. R. Tolkien

Resumo

O setor de varejo é de grande importância para a economia global e estratégias que permitam aumentar a eficiência de seus processos podem garantir seu equilíbrio e crescimento. Dentro desse contexto, empregar técnicas de aprendizado de máquina é uma possibilidade que auxilia empresas a tomar melhores decisões no seu negócio, como já acontece em algumas das grandes empresas do setor. Em redes de lojas físicas, porém, pouco é realizado na aplicação de inteligência para entender a relação entre os produtos e clientes, sendo tal prática já consolidada no *e-commerce*. Um dos grandes empecilhos é o fato de que a quantidade e qualidade de dados disponíveis para essa análise são inferiores em relação às lojas virtuais, onde rastrear as ações dos clientes é mais fácil. Para atingir bons resultados, o condicionamento dos dados é essencial para essa tarefa. Com o presente trabalho é proposto utilizar algoritmos de aprendizado de máquina para criar modelos que representem essas relações. Para atingir os objetivos, dois problemas que relacionam os produtos com os clientes que os compram foram desenvolvidos em parceria com uma empresa que atua no setor do varejo. Para criar os modelos, foram disponibilizados dados de cadastro de clientes e de vendas dos anos 2015 e meados de 2016 de cerca de 100 de suas lojas. Os algoritmos utilizados para criar os modelos de classificação foram a Árvore de Decisão e o *k-Nearest Neighbors*. Para agrupar os produtos, uma solução utilizando algoritmo genético é apresentada. Técnicas para avaliação de modelos para classificação são apresentadas, as quais permitiram validar os resultados obtidos.

Palavras-chave: aprendizado de máquina, varejo, agrupamento, árvore de decisão, algoritmo genético

Abstract

Retail is a sector of great importance for the global economy and techniques to increase the efficiency of some of its processes can ensure its balance and growth. In this context, applying machine learning techniques is a possibility that can help companies to make better business decisions, as it has been already applied in large retail companies. However, little has been done in applying intelligence to understand the relationship between the products and the clients at physical retail stores, in contrast with e-commerce initiatives. One of the greatest difficulties is the fact that the available data has lower quality and smaller volume than the virtual stores, because tracking the client's actions is far easier on a web store. In order to reach good results, processing the data is essential in this task. The present report proposes to use machine learning algorithms to create models that represent these relationships. To achieve the established objectives, two problems that relate the products with the clients were developed in partnership with a retail company. To create the models, a database containing the customer base and the sales records of the years of 2015 and 2016 of around 100 of its stores were made available. To generate the classification models, the Decision Tree and k-Nearest Neighbors algorithms were applied. In order to cluster the products, a genetic algorithm based solution is shown. Techniques to evaluate the classification models are also shown, allowing the validation of the obtained results.

Keywords: machine learning, retail, clustering, decision tree, genetic algorithm

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Nas últimas duas décadas, o aprendizado de máquina, notoriamente conhecido pelo termo em inglês *Machine Learning*, tornou-se um dos pilares da tecnologia da informação, e, consequentemente, traz uma atuação muito importante nas nossas vidas [1]. O crescente volume de dados disponíveis só aumenta a necessidade de estudar e aplicar técnicas para análise de dados que empreguem uma inteligência cada vez maior, sendo esse um ingrediente imprescindível para o progresso tecnológico e científico na atualidade.

Estamos cercados por dispositivos que utilizam tecnologias baseadas em aprendizado de máquina, direta ou indiretamente. Motores de busca aprendem para nos trazer os melhores resultados, softwares aprendem com transações de cartão de crédito para encontrar fraudes, sistemas embarcados em câmeras digitais aprendem para reconhecer faces e carros são equipados com sistemas de prevenção de acidentes que são construídos utilizando algoritmos de aprendizado de máquina [2].

1.1 Aprendizado de Máquina para construção de modelos

Em geral, o aprendizado de máquina busca resolver problemas do mundo real, que apresentem relevância e possuam bases de dados contendo informações que possibilitem alcançar a solução. Além disso, é necessário que o volume de dados disponível seja adequado.

Para a resolução desses problemas, é necessário abstrair os processos através da construção de modelos que sejam válidos para a zona de operação com que se deseja trabalhar, a qual consiste de um conjunto de entradas e saídas possíveis e relevantes. Logo, para um mesmo processo real, pode-se construir diversos modelos para se obter as soluções de um conjunto de problemas de interesse.

Já em se tratando da construção de um modelo, é importante conhecer o processo onde os problemas se encontram com um nível de profundidade que permita essa modelagem. Outro aspecto importante é conhecer os dados a que se tem acesso, para poder então se determinar quais tipos de problemas podem de fato ser solucionados.

Visando garantir que os modelos descrevam o fenômeno de estudo com nível adequado de precisão, cada um deles deve ser avaliado e validado. Esse processo de avaliação ocorre conforme critério inerente ao problema em questão, buscando atender às necessidades envolvidas para atingir a solução.

Para cada modelo construído, um ou mais algoritmos podem ser desenvolvidos de forma a resolver esses problemas. A Figura 1 ilustra como os modelos se relacionam com o mundo real e com esses algoritmos.

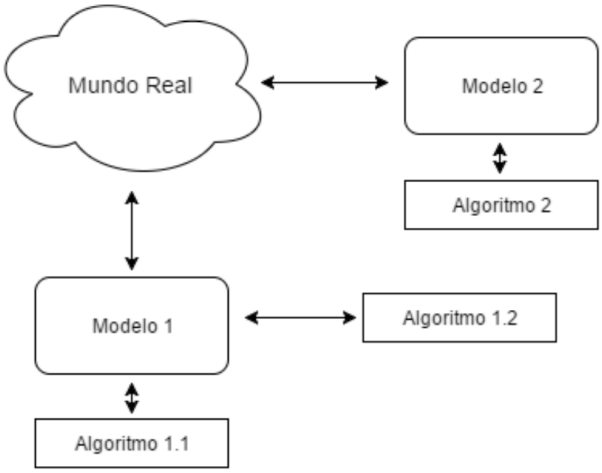

Figura 1 – Modelagem de fenômenos reais.

1.2 Estrutura do Documento

Este documento será organizado como segue. A apresentação dos problemas abordados é dada no Capítulo 2. O Capítulo 3 traz a fundamentação necessária para a compreensão das técnicas envolvidas e implementação das soluções. As soluções propostas, juntamente com os resultados obtidos, são apresentados no Capítulo 4. Por fim, as conclusões são apresentadas no capítulo final.

2 Problemas Abordados

O processo a ser estudado está inserido no setor do varejo, que é o modelo de negócio em que a comercialização de produtos se dá em quantidades usualmente pequenas, diretamente com o comprador final. O modelo de negócio se aperfeiçoou de tal modo que, atualmente, praticamente todos os fabricantes dependem de sua existência para conseguir oferecer seus produtos ao mercado [3].

O varejo representa um setor gigantesco que movimenta a economia global e, segundo projeções dadas pela *eMarketer* (companhia especializada em pesquisas e estudos de mercado), deve movimentar mais de 23 trilhões de dólares no mercado mundial no ano de 2016 [4].

No Brasil o setor do varejo é também muito importante, sendo o que mais emprega no país. Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE), referentes a 2013, o varejo emprega 9,2 milhões de pessoas, o que equivale a 19,1% da força de trabalho total. Além disso, de acordo com o Cempre (Cadastro Central de Empresas), mais de 40% das empresas brasileiras desempenham atividades no comércio, reforçando que, de fato, o varejo apresenta grande participação e importância na economia nacional.

Empregar técnicas que aumentem a eficiência de empresas desse setor é, então, essencial para a manutenção do sistema econômico do Brasil e demais países do mundo, garantindo empregos e crescimento econômico.

As técnicas de aprendizado de máquina já são amplamente utilizadas em sistemas de comércio eletrônico, o *e-commerce*, para resolver diversos problemas, como gerenciamento de estoque, logística de entrega dos produtos, estratégias de *marketing* e principalmente para prever o comportamento dos clientes, os oferecendo produtos que são de seu interesse.

Já para as lojas físicas, a maioria das ferramentas existentes são para auxiliar o gerenciamento estratégico e são aplicáveis apenas a grandes empresas devido ao alto custo de implantação. Mesmo para as grandes corporações, em se tratando de agregar inteligência à experiência do cliente enquanto na loja, muito pouco é realizado.

Para possibilitar a implementação de ferramentas de aprendizado de máquina em empresas do varejo, é indispensável a existência de uma base de dados que possua informações relevantes para os problemas a serem solucionados.

Muitas redes de lojas possuem cadastros de clientes que permitem um levantamento do perfil dos clientes que realizam compras, constituindo uma base de dados valiosa se for utilizada de forma correta. É possível, por exemplo, antecipar o aniversário de cada cliente e, com base no seu histórico de compras, oferecer descontos em produtos de seu

interesse. Ações desse tipo, empregadas em clientes que possuam um perfil de quem busca essa facilidade, são passíveis de fidelizar clientes além de representar um diferencial que pode garantir o sucesso de uma rede de lojas atuando no setor.

Para identificar o perfil dos clientes, é importante possuir na base de dados informações de cadastro como o sexo, faixa etária, estado civil, região de residência, etc. Outras informações, ligadas ao perfil econômico, como faixa de renda e classe social, são também importantes (ou até mais importantes) para identificar e agrupar perfis de clientes, mas são dados delicados para se obter diretamente no cadastramento por serem muito pessoais. Quando os dados disponíveis são insuficientes para alguma análise ou para gerar algum modelo com a qualidade desejada, pode-se enriquecer o banco de dados com informações obtidas em outros meios, como redes sociais, preenchendo lacunas de informações a que não se tem acesso diretamente.

Além disso, em se tratando de indicação de produtos, é importante manter um histórico com todas as vendas realizadas pelas lojas, com informações referentes aos produtos vendidos, como valor da venda, hora e data da venda, o vendedor que realizou a venda e o identificador do cliente que realizou a compra. Informações descritivas dos produtos comercializados pela rede, do estoque de cada loja e dados das lojas da rede, entre outras informações relacionadas, podem apresentar relevância dependendo do problema a ser abordado.

Por fim, em alguns casos, é interessante estudar indicadores sociais e econômicos da região em que a empresa atua, como o crescimento da economia, expectativas quanto ao mercado e projeções quanto ao poder aquisitivo das diversas classes econômicas. Essas informações podem ser utilizadas para enriquecer o modelo para casos em que a demanda exija um modelo que reaja a mudanças de mercado ou aspectos da economia da região de interesse, ou as antecipe.

2.1 Objetivos

O objetivo principal do trabalho realizado consiste em criar modelos, através de técnicas de aprendizado de máquina, utilizando o cadastro de clientes e o histórico de vendas das lojas de uma rede que atue no mercado do varejo. Espera-se que os resultados possibilitem um melhor entendimento quanto à relação entre os perfis de seus clientes e os produtos que esses clientes consomem.

Outro objetivo importante consiste em apresentar técnicas e algoritmos que possibilitem atingir a solução de diversos problemas, além de exemplificar a sua aplicação em um problema real. Para tanto, é necessário estudar algoritmos de aprendizado de máquina que permitam obter modelos que descrevam uma base de dados de interesse, implementá-los e avaliá-los.

Os dados utilizados para a análise consistem de uma tabela de clientes com mais de um milhão de registros, provenientes de cerca de 100 lojas de uma rede de lojas que atua no setor do varejo, em todas as regiões do Brasil.

O registo de vendas realizadas nas lojas também foi disponibilizado. Os dados de vendas são referentes a todas as vendas efetuadas por essas lojas nos anos de 2015 e meados do ano de 2016.

Uma questão a ser contornada para a resolução dos problemas e obtenção dos modelos é que a maior parte dos registros de cadastro de clientes não está completo, ou seja, a maior parte deles possui apenas o nome, sendo que os campos de interesse não estão presentes, enquanto outros possuem alguns desses campos preenchidos. Esse é um problema bastante recorrente no cadastramento de clientes em empresas desse setor. Os motivos que levam a esse fato são diversas, como o cliente se negar a dar alguma informação requisitada ou o vendedor não indagar todas as questões envolvidas no cadastro. Algumas soluções para contornar esse problema são apresentadas na seção 3.6.

Buscando atingir os objetivos traçados e atender algumas demandas dessa rede de lojas, definiu-se dois problemas a serem solucionados.

2.1.1 Problema 1

O primeiro problema a ser solucionado busca ajudar a entender qual perfil de cliente é mais propenso a comprar determinado tipo de produto. Para tanto, deve-se utilizar técnicas de aprendizado de máquina para criar um ou mais modelos que representem essa relação.

Como atributos a serem utilizados na entrada estão o grupo do produto, o qual agrupa produtos segundo algum critério de similaridade, o subgrupo, que traz alguma descrição adicional do produto e o fabricante, que representa o fabricante desse produto.

Como saída, busca-se obter o sexo, a faixa etária e a região, que são informações que representam um perfil de cliente.

Apresentando o problema de modo mais formal:

- Dado um produto, representado pelo seu grupo, subgrupo e fabricante, definir qual perfil de cliente, representado por seu sexo, faixa etária e região, é mais propenso a comprar esse produto.

2.1.2 Problema 2

O segundo problema busca trazer o tipo de produto mais adequado para determinado perfil de cliente, a partir de um ou mais modelos que representem essa relação,

obtidos utilizando técnicas de aprendizado de máquina. De certa forma é o problema inverso ao anterior.

Como atributos a serem utilizados na entrada estão o sexo, a faixa etária e a região, representando um perfil de cliente.

Já como saída do modelo deve-se obter o grupo do produto o qual esse perfil de cliente está mais propenso a comprar.

Apresentando o problema de modo mais formal:

- Dada a estimativa de um perfil de cliente, a partir do seu sexo, faixa etária e região (sendo que um ou mais desses itens podem não ser estimados), definir qual grupo de produtos deve ser oferecido a esse cliente.

3 Fundamentação

Neste capítulo são apresentados os fundamentos teóricos que permitem entender como e em que ocasiões se deve aplicar as ferramentas de aprendizado de máquina, algumas técnicas de aprendizado com seus respectivos algoritmos e técnicas necessárias para o condicionamento dos dados, permitindo a sua manipulação pelos algoritmos.

3.1 Tomada de decisão

A tomada de decisão é um processo cognitivo que resulta na escolha de uma determinada ação a partir de um conjunto de possibilidades. O processo de raciocínio percorrido para se atingir essa escolha pode ser classificado dentre dois grupos, podendo ser de inferência indutiva ou inferência dedutiva [5], [6]

A *inferência indutiva* é aquela em que se utiliza do raciocínio indutivo, no qual generalizações são obtidas a partir de premissas mais particulares. Esse é o processo mais utilizado nas pesquisas de cunho experimental, justamente por elas terem como objetivo obter conclusões que extrapolam as condições em que o experimento foi realizado [7].

Já a *inferência dedutiva* se utiliza do processo de raciocínio dedutivo, em que se parte de um conjunto de proposições gerais e se obtém um conjunto de proposições específicas. Em outras palavras, é o processo em que se obtém conhecimento de um membro específico de uma classe a partir do conhecimento geral, referente a todos os membros da mesma classe.

O objetivo da utilização das técnicas que serão apresentadas a seguir é de, através de algoritmos que implementem processos de inferência indutiva, obter a solução para os problemas pertinentes ao presente trabalho.

3.2 Aprendizado de máquina

O aprendizado de máquina é um sub-campo da *inteligência artificial*, um campo que surgiu dentro da ciência da computação, e aborda a questão sobre como tornar as máquinas aptas a aprender. No contexto em que se insere, o aprendizado se refere à inferência indutiva [8].

O objetivo principal do aprendizado de máquina é generalizar além dos exemplos existentes no conjunto de treinamento, pois independente da quantidade de dados existentes é muito improvável que, durante os testes, exatamente os mesmos exemplos apareçam.

Ainda não é possível fazer computadores aprenderem tão bem quanto as pessoas, porém algoritmos criados são eficientes para várias tarefas de aprendizado, e os estudos teóricos sobre aprendizado estão permitindo que novas técnicas sejam desenvolvidas [9].

Um dos aspectos que tornam esse um tema de importância é a dimensão dos problemas a serem resolvidos. A intuição humana é treinada em um universo tri-dimensional, logo existe uma dificuldade natural de se resolver problemas de dimensões maiores sem a utilização de ferramentas adequadas [10]. Além disso, a capacidade de armazenamento dos computadores e a quantidade de dados disponíveis cresce vertiginosamente e há muito o ser humano não consegue processar esse grande volume sem o auxílio de ferramentas de *software*.

Apesar da grande dimensão, inerente a certos problemas, ser uma das principais motivações de utilizar algoritmos de aprendizado de máquina, ela traz uma dificuldade para a avaliação dos resultados obtidos com os modelo gerados por esses algoritmos, já que muitas vezes a intuição não é suficiente avaliar certos aspectos. Para tanto, técnicas para avaliação dos algoritmos e modelos gerados são muito estudadas e necessárias para permitir entender o grau de confiança que se pode empregar sobre os resultados obtidos.

O aprendizado de máquina pode ser dividido em dois importantes sub-grupos de algoritmos: os que compõe a *aprendizagem não-supervisionada* e a *aprendizagem supervisionada*.

3.2.1 Aprendizagem não-supervisionada

A aprendizagem não-supervisionada, por vezes chamada *modelagem descritiva*, não utiliza informações das variáveis de saída. Os dados de entradas são analisados e agrupados conforme a proximidade dos seus valores. Para cada grupo, ou *cluster*, um rótulo é utilizado, permitindo indicar a qual grupo cada registro pertence [11].

3.2.2 Aprendizagem supervisionada

Na aprendizagem supervisionada, deve existir um *supervisor*, que é dado pelo registro dos valores das variáveis de saída, ou variáveis-objetivo, que são as variáveis que se deseja predizer a partir dos dados existentes. Essas variáveis devem ser escolhidas de modo a representar a resposta para algum problema que se deseje resolver [12]. Como resultado, obtém-se um modelo que descreva o conjunto de dados utilizados e espera-se que ele permita prever o comportamento da saída para novas entradas.

Os principais algoritmos de aprendizagem supervisionada criam modelos que permitem a classificação, para variáveis categóricas, e a regressão, para variáveis contínuas. A classificação consiste em atribuir um rótulo para a saída a partir de determinada entrada. A regressão retorna um valor que pertence a um espectro contínuo de valores [13].

Para avaliar o modelo obtido, normalmente se separa uma parte dos exemplos existentes para realizar testes, de preferência não os utilizando durante o processo de treinamento para evitar o sobre-ajuste, ou *overfitting*. Esse fenômeno ocorre quando o modelo descreve muito bem o conjunto utilizado para treinamento, mas tem baixa capacidade de prever novas entradas. É importante ressaltar que os dados inerentemente apresentam desvios causados por fatores aleatórios, e os algoritmos devem minimizar a influência desses desvios na geração dos modelos.

Para classificadores, a técnica mais utilizada para a avaliação consiste em comparar os valores de saída obtidos com o modelo treinado, utilizando os exemplos de teste como entrada, com os valores de saída existentes nesses exemplos. A taxa de acerto, ou *accuracy*, é dada por:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \mathbb{1}(\hat{y}_i = y_i) \tag{3.1}$$

onde y é a saída existente nos exemplos de teste, $\hat{y}$ os valores obtidos com o modelo treinado, $n_{samples}$ o número de exemplos de teste e $\mathbb{1}(x \in A)$ é a função indicadora, dada por:

$$\mathbb{1}(x \in A) = \begin{cases} 1 & if \quad x \in A, \\ 0 & if \quad x \notin A. \end{cases} \tag{3.2}$$

Já para modelos de regressão, o erro pode ser contabilizado utilizando o erro quadrático médio, dado por:

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\hat{y}_i - y_i)^2 \tag{3.3}$$

Um diagrama que representa o processo de treinamento e avaliação é apresentado na Figura 2.

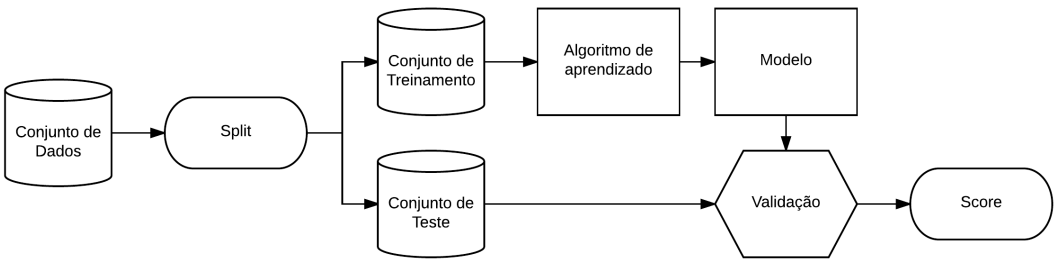

Figura 2 – Processo de treinamento e avaliação em aprendizagem supervisionada

Uma questão importante que surge é como avaliar se os resultados obtidos com as Equações (3.1) e (3.3) são bons. Na prática, não existe uma convenção quanto aos valores mínimos para estabelecer a qualidade do modelo, sendo esse um critério totalmente dependente do problema a ser resolvido e dos objetivos a serem atendidos por esse modelo. Porém, em se tratando de aprendizado de máquina, sempre se espera que o modelo treinado apresente uma taxa de acerto superior a um conjunto de escolhas aleatórias, o que demonstra que o modelo aprendeu com os exemplos fornecidos.

3.3 Árvores de decisão

O aprendizado por Árvore de Decisão (DT) é um método que permite a modelagem de sistemas discretos em que o modelo obtido é uma Árvore de Decisão, sendo um dos métodos mais utilizados para a inferência indutiva por ser robusto a ruído nos dados, além de ser de simples compreensão, permitindo uma representação gráfica do modelo gerado [14].

Outras razões motivam a popularidade desse método, como a simplicidade em se montar as árvores, com algoritmos eficientes que escalam bem com o aumento do tamanho da base de dados utilizada. Outro ponto é que a maioria dos algoritmos podem trabalhar tanto com variáveis de entrada contínuas quanto categóricas. Além disso, as árvores de decisão são não-paramétricas, ou seja, não necessitam de nenhuma informação *a priori* quanto à distribuição dos atributos de entrada ou da função-objetivo.

A sua representação consiste em uma estrutura em que cada nodo interno corresponde a um teste sobre um dado atributo, cada ramo descendente representa uma possibilidade para esse teste e cada folha, contendo a classe respectiva às instâncias por ela classificadas, é a decisão obtida após testar os atributos de forma sequencial. O caminho percorrido para chegar à classe corresponde a uma regra de classificação.

3.3.1 Entropia

A entropia caracteriza a organização de uma coleção de exemplos [15]. Dado um conjunto S, contendo valores positivos e negativos para um atributo-alvo, a entropia de S relativa à essa classificação é dada por:

$$Entropy(S) = -p_+ \log_2 p_+ - p_- \log_2 p_- \tag{3.4}$$

onde p_+ é a proporção de exemplos em que a saída é positiva e p_- a proporção de exemplos em que a saída é negativa. Para os cálculos envolvendo entropia se assumirá $0 \log_2 0$ como sendo 0.

Ao analisar a Equação (3.4), nota-se que a entropia é 0 quando todos os elementos de S pertencem à mesma classe, e 1 quando a coleção contém um mesmo número de

elementos cuja saída é positiva e negativa. Além disso, o resultado da soma entre p_+ e p_- sempre será 1, de forma que pode-se escrever $p_- = 1 - p_+$. A Figura 3 mostra a relação entre a proporção p_+ e a entropia.

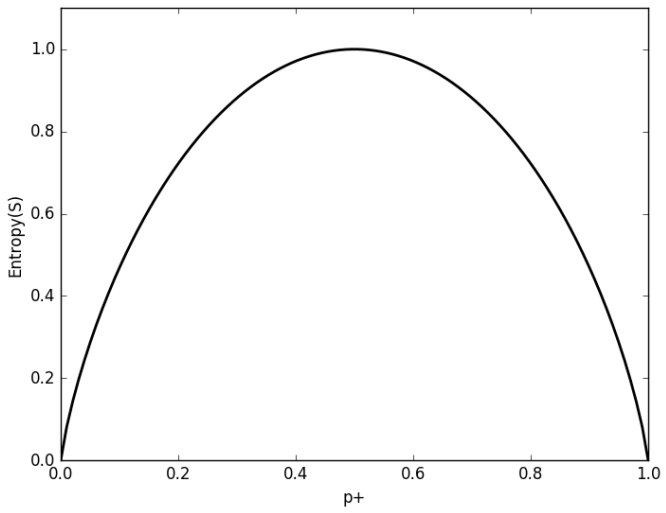

Figura 3 – Entropia de S em função de p_+

Generalizando o cálculo da entropia para casos em que o atributo-objetivo possui mais que dois valores possíveis, define-se a entropia como:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i \tag{3.5}$$

onde p_i é a proporção de S pertencente à classe i e c é o número de classes do atributo-objetivo.

3.3.2 Ganho de Informação

Utilizando a entropia, que representa a medida da impureza de um conjunto de exemplos de treinamento, pode-se definir uma medida para a eficácia de um atributo em classificar esses exemplos [16], [17]. Essa medida é chamada *ganho de informação*, e é definida como:

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{3.6}$$

onde $Values(A)$ são todos os valores possíveis para o atributo A e S_v é o subconjunto de S em que o atributo A tem valor v, i.e.:

$$S_v = \{s \in S | A(s) = v\} \tag{3.7}$$

O ganho de informação mede, então, a redução esperada da entropia ao classificar um conjunto utilizando determinado atributo.

3.3.3 Algoritmo ID3

O algoritmo ID3 é um dos mais conhecidos para a implementação de uma Árvore de Decisão [18]. Ele é um algoritmo não-incremental, ou seja, a sua estrutura é construída a partir de um conjunto de dados e espera-se que as instâncias desconhecidas sigam a mesma distribuição do conjunto de treinamento, de modo que para o modelo incluir novos exemplos, deve-se re-aplicar o algoritmo para o novo conjunto de exemplos.

O algoritmo ID3 está apresentado no Algoritmo 1.

Algorithm 1 Algoritmo ID3

ID3 (*Exemplos*, *Atributo_objetivo*, *Atributos*)
Exemplos: Os exemplos de treinamento
Atributo_ objetivo: O atributo cujo valor deve ser predito pela árvore
Atributos: Lista com os atributos que serão utilizados para classificar os exemplos

1: Crie um nodo Raiz para a árvore
2: **if** todos os exemplos são positivos **then**
3: **return** Raiz da árvore, com o rótulo '+'
4: **else if** todos os exemplos são negativos **then**
5: **return** Raiz da árvore, com o rótulo '-'
6: **else if** Atributos for vazio **then**
7: **return** Raiz com o rótulo = valor mais comum do Atributo-objetivo em Exemplos
8: **else**
9: A ← o atributo de Atributos que melhor classifica Exemplos[1]
10: Raiz ← A (rótulo = atributo de decisão A)
11: **for** cada possível valor v_i de A **do**
12: Acrescenta um novo arco abaixo da Raiz, correspondendo à resposta $A = v_i$
13: Seja $Exemplos_{v_i}$, o subconjunto de Exemplos que tem valor v_i para A
14: **if** $Exemplos_{v_i}$ for vazio **then**
15: Acrescenta na extremidade do arco um nodo folha com rótulo = valor mais comum do Atributo-objetivo em Exemplos
16: **else** acrescenta na extremidade do arco a sub-árvore:
17: ID3($Exemplos_{v_i}$, Atributo-objetivo, Atributos -{A})
return Raiz

[1]O melhor atributo é aquele que apresenta o maior ganho de informação, definido na Equação (3.6)

3.3.4 Exemplo prático

Para exemplificar o funcionamento do algoritmo ID3, utilizar-se-á uma tabela que contém dados reais, porém com dimensões reduzidas, permitindo assim a visualização da árvore gerada. Os dados a serem utilizados estão representados na Tabela 1. A tabela possui como atributos de entrada as colunas Região, Sexo e Estado Civil, que caracterizam os clientes que realizaram as compras. O atributo-objetivo é representado na coluna Grupo

Produto, que representa a categoria em que o produto que determinado cliente comprou pertence. Esse conjunto de dados será tratado como conjunto S.

Tabela 1 – Tabela com dados para exemplo

Região	Sexo	Estado Civil	Grupo Produto
Nordeste	Feminino	Casado	C1
Sul	Masculino	Solteiro	C1
Sul	Feminino	Outros	C2
Sul	Feminino	Outros	C1
Sudeste	Feminino	Solteiro	C2
Nordeste	Masculino	Casado	C1
Nordeste	Feminino	Outros	C2
Sul	Feminino	Casado	C2
Sul	Feminino	Casado	C1
Sudeste	Masculino	Solteiro	C2
Sul	Feminino	Outros	C2
Nordeste	Feminino	Casado	C1
Sudeste	Feminino	Solteiro	C2
Sudeste	Masculino	Outros	C2
Sudeste	Feminino	Outros	C1
Sudeste	Feminino	Casado	C1
Sudeste	Masculino	Solteiro	C1
Nordeste	Feminino	Outros	C2
Sul	Feminino	Solteiro	C1
Sudeste	Masculino	Casado	C1

Aplicando o Algoritmo 1 nos dados da Tabela 1, obtém-se o modelo de árvore que representa a coleção de dados S e permite a classificação de novos registros, ilustrado na Figura 4.

O Apêndice A apresenta um detalhamento passo-a-passo da obtenção desse modelo.

Para utilizar o modelo obtido para classificar novas entradas, basta percorrer a árvore realizando os testes sobre cada um dos atributos até chegar à folha respectiva que possui a classe relativa. A fim de exemplificar essa prática, será realizada a classificação de alguns exemplos hipotéticos, apresentados na Tabela 2.

Tabela 2 – Instâncias a serem classificadas

Instância	Região	Sexo	Estado Civil
x_{q1}	Nordeste	Masculino	Outros
x_{q2}	Sul	Masculino	Casado
x_{q3}	Sudeste	Feminino	Solteiro

Utilizando o modelo apresentado na Figura 4 para classificar os exemplos presentes na Tabela 2, obtém-se os resultados apresentados na Tabela 3.

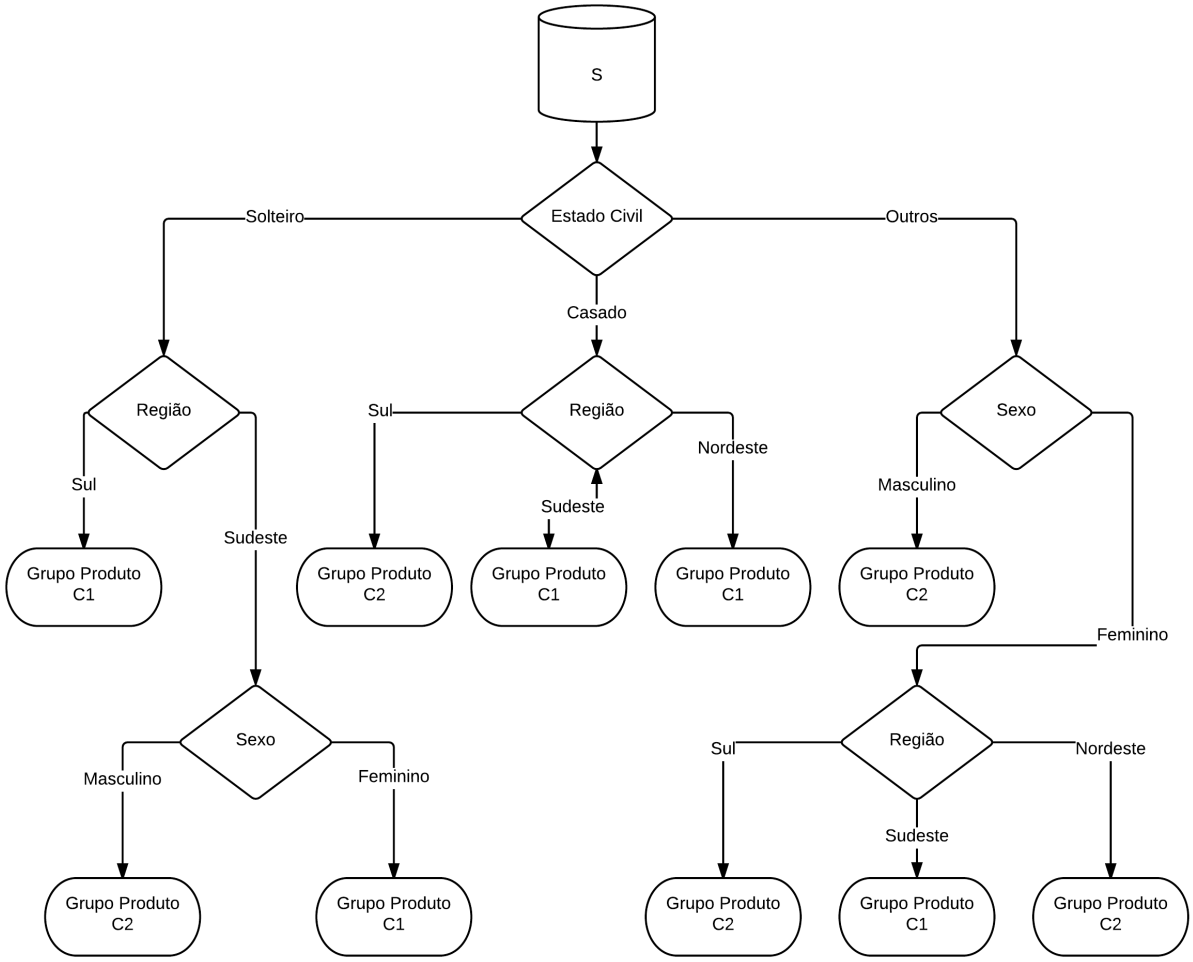

Figura 4 – Diagrama da árvore obtida após a classificação dos exemplos do conjunto S.

Tabela 3 – Classificação utilizando modelo obtido

Instância	Classificação
x_{q1}	C2
x_{q2}	C2
x_{q3}	C1

3.4 k-Nearest Neighbors

O algoritmo *k-Nearest Neighbors* (kNN) é um dos algoritmos mais populares para reconhecimento de padrões [19]. É um algoritmo não-paramétrico, sendo primeiramente descrito por E. Fix e L. L. Hodges em 1951 [20]. A categoria em que o método se insere é chamada de métodos de aprendizado baseados em instâncias. A diferença entre esses métodos e os outros utilizados para aprendizado de máquina é que eles não constroem modelos que representam o conjunto de treinamento durante o aprendizado, mas simples-

mente armazenam os dados para serem consultados na etapa de classificação, por isso sendo também chamados de algoritmos de aprendizado preguiçosos.

Um aspecto importante nesse tipo de abordagem é que inerentemente os algoritmos podem aferir aproximações diferentes da função-objetivo para cada instância que deve ser classificada, característica que pode ser uma vantagem significativa quando a função-alvo é muito complexa, onde mínimos ou máximos locais podem facilmente ser suprimidos em simplificações geradas ao se construir o modelo.

O algoritmo assume que todas as instâncias correspondem a pontos no espaço n-dimensional $\Re^n$. Pode-se então definir uma instância x como sendo um vetor no espaço:

$$\langle a_1(x), a_2(x), ..., a_n(x) \rangle \tag{3.8}$$

onde $a_r(x)$ representa o valor do r-ésimo atributo de x.

Para classificar os vizinhos mais próximos a certo ponto no espaço, é necessário definir uma métrica que represente essa distância. No caso se utiliza a distância Euclidiana. Pode-se definir a distância entre duas instancia x_i e x_j como sendo:

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2} \tag{3.9}$$

A partir da Equação (3.9) pode-se obter as k instâncias mais próximas de um ponto desejado no espaço. A escolha do parâmetro k é de grande relevância, pois seu valor afeta diretamente os resultados obtidos. Valores grandes de k apresentam maior atenuação dos ruídos presentes nos dados, já que um maior número de exemplos será analisado, porém pode suprimir características e tendências presentes em pequenos grupos de dados. Usualmente se avalia a qualidade do algoritmo para diferentes valores de k, e então o que apresenta melhor desempenho é escolhido. Alguns outros métodos eurísticos podem ser utilizados para obter um bom valor de k, como o algoritmo *Bayesian optimization* [21].

A utilização do algoritmo *k-Nearest Neighbors* é válida para problemas em que a função-objetivo é tanto real quanto categórica. Já em se tratando dos atributos, devido à definição dada na Equação (3.9), a sua representação deve estar no domínio real, para permitir o cálculo das distâncias Euclidianas. Considerando casos em que o problema abordado utiliza atributos categóricos, uma solução para a adequação dos dados seria representar cada valor de cada atributo como um equivalente numérico. Em alguns casos essa solução pode não ser recomendada, já que, em se tratando de variáveis qualitativas nominais, não se espera que as variáveis sejam passíveis de ordenação em seus valores. Essa aproximação, porém, permite atingir resultados interessantes para certos problemas.

3.4.1 O algoritmo

O Algoritmo 2 aproxima a função-objetivo utilizando o algoritmo *k-Nearest Neighbors* para os casos em que a função-objetivo pode ser escrita na forma $f : \Re \rightarrow V$, sendo V o conjunto finito $\{v_1, ..., v_s\}$.

Algorithm 2 Algoritmo *k-Nearest Neighbors*

Algoritmo de treinamento

1: • Cada exemplo de treinamento $\langle x, f(x) \rangle$ deve ser adicionado à lista exemplos_treinamento

Algoritmo de classificação:

2: • Dada uma instância x_q a ser classificada:

3: • Com $x_1, \ldots, x_k$ representando as k instâncias da lista exemplos_treinamento que são mais próximas de x_q

4: • **return**

$$\widehat{f}(x_q) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(x_i))$$

onde $\delta(a, b) = 1$ se $a = b$ e $\delta(a, b) = 0$ se $a \neq b$

Como mostrado no Algoritmo 2, o valor aproximado de $f(x_q)$ retornado pelo método, dado por $\widehat{f}(x_q)$, é o valor mais comum da função $f(x)$ dentre os k exemplos de treinamento mais próximos de x_q.

3.4.2 Exemplo prático

Para exemplificar a atuação prática desse método, será realizada a modelagem e classificação dos dados contidos na Tabela 1, previamente utilizada na Seção 3.3.4. Como os valores dos atributos utilizados são categóricos, torna-se necessário realizar uma conversão para valores equivalentes numéricos. Para tanto, será utilizará a seguinte conversão:

- Na coluna Região, o valor Sul receberá valor 0, Sudeste 1 e Nordeste 2.
- Na coluna Sexo, o valor Masculino receberá valor 0 e Feminino 1.
- Na coluna Estado Civil, o valor Casado receberá valor 0, Solteiro 1 e Outros 2.

Aplicando essa conversão se obtém o conjunto apresentado na Tabela 4, que representa a lista com os exemplos de treinamento já devidamente convertidos.

A Figura 5 representa os dados da Tabela 4 dispostos em um gráfico de dispersão. Nota-se que os dados ficam todos dispostos em dois planos, já que o atributo Sexo possui apenas dois valores possíveis. Além disso, como é de se esperar, a sobreposição de alguns pontos pode ocorrer.

Tabela 4 – Tabela com dados convertidos para equivalentes numéricos

Instância	Região	Sexo	Estado Civil	Grupo Produto
x_1	2	1	0	C1
x_2	0	0	1	C1
x_3	0	1	2	C2
x_4	0	1	2	C1
x_5	1	1	1	C2
x_6	2	0	0	C1
x_7	2	1	2	C2
x_8	0	1	0	C2
x_9	0	1	0	C1
x_{10}	1	0	1	C2
x_{11}	0	1	2	C2
x_{12}	2	1	0	C1
x_{13}	1	1	1	C2
x_{14}	1	0	2	C2
x_{15}	1	1	2	C1
x_{16}	1	1	0	C1
x_{17}	1	0	1	C1
x_{18}	2	1	2	C2
x_{19}	0	1	1	C1
x_{20}	1	0	0	C1

Para a classificação é necessário ter novas instâncias a serem classificadas. A Tabela 5 traz os mesmos exemplos hipotéticos de instâncias a serem classificadas apresentados na Tabela 2, com a devida conversão dos valores, possibilitando obter uma classificação utilizando esse algoritmo. Se realizará a classificação utilizando dois valores para k, $k = 3$ e $k = 5$.

Tabela 5 – Instâncias a serem classificadas

Instância	Região	Sexo	Estado Civil
x_{q1}	2	0	2
x_{q2}	0	0	0
x_{q3}	1	1	1

Os valores classificados estão apresentados na Tabela 6. Um detalhamento dos passos necessários para a classificação das instâncias desse exemplo está no Apêndice B. É interessante notar que para diferentes valores de k, as classificações obtidas pelo algoritmo podem mudar, de modo que a escolha desse parâmetro é de grande importância para a aplicação desse método.

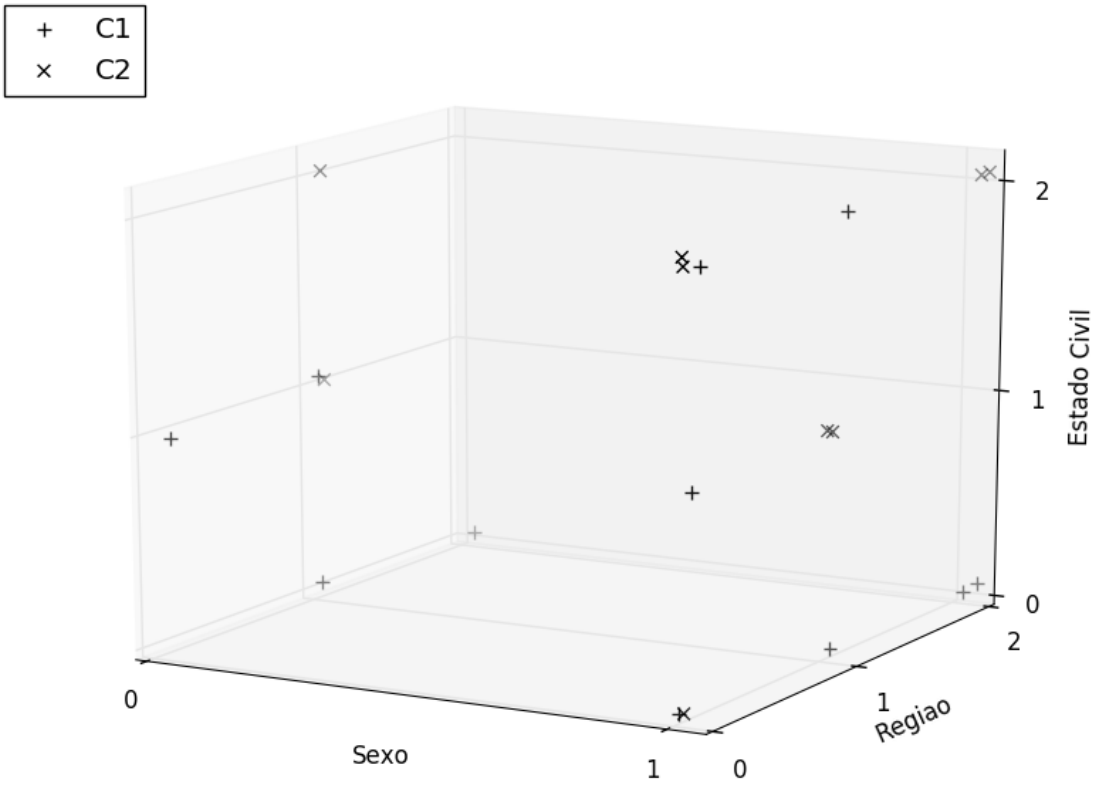

Figura 5 – Gráfico representando conjunto de exemplos de treinamento

Tabela 6 – Classificação utilizando k-Nearest Neighbors para valores diferentes de k

Instância	k=3	k=5
x_{q1}	C2	C2
x_{q2}	C2	C1
x_{q3}	C1	C1

Não existe uma convenção para a escolha do parâmetro k. O importante é conhecer a distribuição dos dados e o problema que se busca solucionar, para entender o impacto da escolha do parâmetro para essa solução. Pode-se também realizar testes para diferentes valores de k e avaliar os resultados utilizando a Equação (3.1), para então escolher o valor mais apropriado. Em alguns casos, quando se deseja buscar um k ótimo segundo algum critério de interesse, pode-se aplicar algum outro algoritmo que permita buscar essa otimização.

3.5 Algoritmo Genético

Os algoritmos genéticos são métodos de aprendizado que pertencem a uma classe de algoritmos que utilizam técnicas motivadas pela biologia evolutiva, chamados algoritmos

evolutivos. Esses algoritmos utilizam operadores que replicam fenômenos evolutivos como *crossover*, mutação, seleção natural, etc [22]. Os algoritmos genéticos buscam as melhores soluções dentro de um espaço de hipóteses, avaliadas a partir de uma função-objetivo.

A estrutura de um algoritmo genético consiste em modificar iterativamente um conjunto de hipóteses, chamado de população. Em cada iteração, os indivíduos da população são avaliados segundo a função-objetivo que se busca otimizar. Uma nova população é então gerada selecionando os indivíduos mais aptos dessa população de forma probabilística, sendo que alguns desses são inseridos diretamente na nova população enquanto outros são utilizados como base para gerar novos indivíduos, utilizando operadores genéticos, principalmente o *crossover* e a mutação [23].

3.5.1 Representação das hipóteses

Usualmente, as hipóteses, que são os indivíduos da população, são representadas por um número binário, de forma que os operadores genéticos possam ser facilmente aplicados.

Uma forma de obter uma representação para atributos categóricos é utilizar a codificação para o formato *One-hot*. Essa é uma representação em que todas as possibilidades possuem um único algarismo 1, sendo que os demais algarismos são todos 0 [24]. Por exemplo, para o atributo Estado_civil possuindo 3 possíveis categorias, Casado, Solteiro e Outros, pode-se representar a categoria Casado como 001, Solteiro como 010 e Outros como 100.

Além dessa possibilidade, pode-se representar as categorias como números inteiros ou até mesmo *strings*, sendo que as operações se tornam mais complexas e custosas computacionalmente.

3.5.2 Operadores genéticos

Como supracitado, os principais operadores genéticos empregados em algoritmos genéticos são o *crossover* e a mutação, estando presentes na maioria das implementações do algoritmo. Algumas implementações utilizam operadores complementares, pertinentes para sua a aplicação específica.

3.5.2.1 *Crossover*

Para garantir a variabilidade dos indivíduos testados e assim expandir o espaço de hipóteses avaliado, uma recombinação entre os indivíduos de uma população é realizada através do operador *crossover* [25]. Esse operador imita o processo biológico que ocorre na reprodução sexuada e que possui mesmo nome, onde descendentes são gerados a partir da combinação do material genético dos genitores. Quando o indivíduo é representado

por uma sequência de bits, o operador trata de gerar descendentes copiando grupos de bits de um dos genitores e os alocando na mesma posição para os descendentes. Em geral, esse operador trata de gerar dois descendentes a partir de dois genitores, de forma que os próximos exemplos seguirão essa proporção.

Para codificar os bits que contribuirão para cada um dos descendentes, uma máscara binária é utilizada, chamada de máscara de *crossover*, que possui mesmo número de bits que os indivíduos da população e indica quais bits serão provenientes de cada um dos genitores para cada descendente. As equações (3.10) e (3.11) trazem as expressões que permitem obter os descendentes, chamados de $offspring1$ e $offspring2$, a partir dos genitores, chamados de $parent1$ e $parent2$.

$$offspring1 = (parent1 \wedge Crossover\_mask) \vee (parent2 \wedge \neg Crossover\_mask) \quad (3.10)$$

$$offspring2 = (parent1 \wedge \neg Crossover\_mask) \vee (parent2 \wedge Crossover\_mask) \quad (3.11)$$

Existem três variações principais para esse operador, que se diferenciam pela forma que a máscara de *crossover* é gerada, sendo eles [26]:

- *Crossover* de ponto único: Nesse operador a máscara de *crossover* é sempre construída de forma a possuir n bits contínuos com valor 1, seguidos pela quantidade necessária de bits com valor 0 para completar a máscara. Como resultado, para cada descendente, os primeiros n bits são contribuição de um dos genitores e o restante do outro. Cada vez que essa operação é realizada, o valor n é escolhido de forma aleatória;
- *Crossover* de dois pontos: Nesse operador a máscara de *crossover* é formada por n_0 bits com valor 0, seguidos de bits com valor 1 até o bit de posição n_1, sendo completada por bits com valor 0. Cada vez que a operação é realizada, os valores n_0 e n_1 são escolhidos de forma aleatória, sendo que $n_1 > n_0$;
- *Crossover* uniforme: Nesse operador cada bit que forma a máscara possui valor 0 ou 1 escolhido de forma aleatória.

3.5.2.2 Mutação

A mutação é responsável por introduzir pequenas alterações aleatórias nos indivíduos, sendo geralmente aplicada após o operador *Crossover*, com uma probabilidade pequena de ocorrer para cada indivíduo. O operador consiste em escolher um bit de forma aleatória e trocar o seu valor.

3.5.3 O algoritmo

O Algoritmo 3 apresenta uma implementação para o algoritmo genético.

Algorithm 3 Algoritmo Genético

GA($Fitness, Fitness\_threshold, p, r, m$)
Fitness: Uma função que retorna um valor que avalie uma hipótese dada
Fitness_threshold: Um limiar que especifique um critério de parada
p: O tamanho da população
r: A fração da população a sofrer o crossover a cada iteração
m: A taxa de mutação

1: • *Inicializar população:* P ← Gerar p indivíduos aleatoriamente.
2: • *Avaliar:* Para cada h em P, calcular $Fitness(h)$
3: **while** [$\max_h Fitness(h)$] < $Fitness\_threshold$ **do**:
4: Criar uma nova geração P_S:
5: *Seleção:* Probabilisticamente selecionar $(1 - r)p$ indivíduos de P para adicionar a P_S. A probabilidade $Pr(h_i)$ de selecionar o indivíduo h_i de P é dada por:

$$Pr(h_i) = \frac{Fitness(h_i)}{\sum_{j=i}^{p} Fitness(h_j)}$$

6: *Crossover* : Probabilisticamente selecionar $\frac{rp}{2}$ pares de indivíduos de P, de acordo com $Pr(h_i)$ definido acima. Para cada par $\langle h_1, h_2 \rangle$, produzir dois descendentes aplicando o operador $Crossover$. Adicionar todos os descendentes a P_S.
7: *Mutação:* Escolher m porcento dos indivíduos de P_S aleatoriamente. Para cada um, aplicar o operador de Mutação.
8: *Atualização:* $P \leftarrow P_S$
9: *Avaliação:* Para cada h em P, calcular $Fitness(h)$
return indivíduo de P mais apto.

3.6 Pré-processamento dos dados

Dados provenientes de fontes reais geralmente necessitam de um pré-processamento a fim de serem utilizados pelas técnicas de aprendizado de máquina previamente apresentadas. As etapas mais importantes nessa etapa são a limpeza e seleção de variáveis e a amostragem [27], [28].

3.6.1 Valores incorretos

Valores incorretos podem causar distorções nos modelos obtidos, pois, em geral, os algoritmos de aprendizado de máquina assumem que todos os valores utilizados são completamente corretos. A verificação buscando esse tipo de erro deve ser realizada caso os resultados estejam muito diferentes do esperado, ou quando se supõe *a priori* que os dados possam ser incorretos.

Para encontrar valores potencialmente incorretos em variáveis categóricas, primeiramente se deve examinar as frequências. Utilizar um histograma, por exemplo, permite

buscar valores não usuais. No caso de serem poucos os exemplos contendo essas discrepâncias, pode-se examinar um por vez, para verificar se os valores são factíveis. Para casos em que a quantidade de dados analisada é muito grande, deve-se buscar uma ferramenta que realize essa verificação e correção (ou simplesmente uma sinalização para o desenvolvedor) de forma automática.

Essa análise pode requerer familiaridade e experiência com o tipo de dados estudado, e ainda assim, para alguns valores é difícil determinar qual o valor correto, ou esperado, para aquele campo. Quando dados supostamente incorretos ocorrem em pouca frequência e não podem ser interpretados ou substituídos, pode-se considerá-los como dados faltantes.

Já para variáveis contínuas, valores incorretos podem ser detectados como picos em distribuições, representando assim uma discrepância. Caso esses dados estejam prejudicando os resultados, podem ser tratados com a aplicação de filtros. Em casos onde os dados contendo valores incorretos são tão infrequentes que não podem ser detectados facilmente, pode-se mantê-los, pois provavelmente eles não apresentarão relevância no processo de modelagem.

3.6.2 Falta de dados

A falta de dados é o maior dos problemas relacionados ao condicionamento dos dados. Geralmente, dados faltantes são representados como valores nulos ou como células vazias. Uma análise, porém, sempre deve ser realizada nos dados a serem utilizados pois, em alguns casos, os dados faltantes são representados com valores não nulos, podendo levar o sistema a interpretá-los erroneamente como registros válidos [29].

As causas para a falta de dados são variadas. Em alguns casos pode ser um problema na etapa de aquisição dos dados. Outras vezes podem ser simplesmente dados cujos valores são desconhecidos. Em certas ocasiões também pode ocorrer uma perda de dados armazenados em uma base. Em outros casos, porém, os dados faltantes são omitidos propositalmente pelas fontes durante o processo de coleta. Em uma pesquisa, por exemplo, alguns dos entrevistados podem se recusar a responder algumas das questões propostas.

Cabe um estudo nos dados utilizados em cada aplicação para obter a codificação correta dos valores nulos para cada um dos campos que compõe a base de dados, e entender como esses valores podem impactar os modelos gerados pelas técnicas de aprendizado de máquina.

3.6.3 Resolvendo o problema da falta de dados

Para resolver problemas gerados pela falta de dados pode-se utilizar uma série de abordagens diferentes. A seguir serão apresentadas algumas delas.

3.6.3.1 *Listwise* e *Column Deletion*

A técnica mais simples e uma das mais utilizadas é chamada *listwise deletion*, que consiste em remover todos os registros, que são as linhas das tabelas, com qualquer dado faltante, restando apenas os registros com todos os valores a serem utilizados preenchidos. Em muitos casos essa é uma abordagem ideal, porém muitos conjuntos possuem dados faltantes em grande parte dos registros (potencialmente em todos eles), de forma que pode restar pouca informação para gerar os modelos.

Uma alternativa para evitar esse problema é utilizar a técnica chamada *column deletion*, onde se remove as variáveis, ou colunas da tabela, que tenham algum dado faltante. Desse modo, caso os dados faltantes se concentrem em poucas variáveis, o que é o típico, após a remoção ainda restará um número de registros suficiente para a construção dos modelos [30].

Ambas técnicas podem ser muito restritivas, de modo que muita informação importante pode ser perdida dependendo das características do conjunto de dados e do problema a ser resolvido. Nesses casos pode-se recorrer a alguma técnica mais avançada, como as que serão apresentadas a seguir.

3.6.3.2 Preenchimento com uma constante

Essa é outra técnica largamente utilizada, e consiste em preencher todos os campos faltantes com algum valor que indique essa inexistência de dado. Para variáveis do tipo *string*, por exemplo, pode-se preencher os campos faltantes com o valor "U". Já para variáveis numéricas, geralmente se utiliza o número 0.

Antes de aplicar essa técnica, uma análise dos dados e do problema deve ser realizada. Em muitos casos o valor utilizado pode prejudicar o modelo construído. Por exemplo, se um dos campos do conjunto é referente à idade da pessoa e todos os campos faltantes forem preenchidos com 0, o modelo provavelmente introduzirá um *bias* para um valor mais baixo que o esperado ao considerar esse campo em sua construção.

Para contornar esse problema, deve-se buscar utilizar uma constante que caracterize unicamente o valor como um valor faltante. Dessa forma, se necessário, será possível levar essa informação em consideração durante a construção dos modelos, evitando que esses valores se confundam com os valores corretamente preenchidos.

3.6.3.3 Preenchimento com média ou mediana para variáveis contínuas

Técnicas que acrescentam um pouco mais de sofisticação em relação às previamente apresentadas buscam preencher os campos faltantes com valores que não são pré-definidos. Dentre as possibilidades, a mais utilizada é o preenchimento utilizando o valor médio. A popularidade se dá principalmente pela facilidade de aplicação, já que a maioria das

linguagens e ferramentas possibilitam realizar o cálculo da média de uma série de forma simples e eficiente. Além disso essa é uma estratégia interessante conceitualmente, pois se espera que os valores faltantes sigam a distribuição dos demais valores, de modo que as médias devem se aproximar.

Apesar de ser uma boa abordagem, também existem problemas associados com a sua utilização. Quanto mais valores faltantes são preenchidos com o valor médio, mais valores ocupam essa faixa de valores na distribuição de frequências, de modo que surge um pico cada vez mais acentuado na distribuição. Como consequência, o desvio padrão fica cada vez menor e a tendência é que modelos gerados utilizando esse conjunto fiquem menos sensíveis a variações nos valores de entrada.

Em casos em que a média e mediana diferem, pode-se optar por preencher os valores faltantes utilizando a mediana, pois a mediana representa melhor o valor mais comum para aquela variável.

3.6.3.4 Preenchimento com distribuição

Quando grandes quantidades de dados estão faltando, o comportamento estatístico dos dados pode ser prejudicado pelo preenchimento por média, mediana ou constante, como explicitado anteriormente. Uma alternativa para minimizar esse problema é utilizar um valor selecionado aleatoriamente dentre uma distribuição conhecida, de modo que como resultado os dados tenderão a preservar melhor a distribuição original e também a distribuição real dos dados, já que se espera que ambas sejam próximas.

3.6.4 Amostragem

A amostragem é a principal técnica utilizada para selecionar um subconjunto com dados relevantes a partir de um conjunto maior de dados. Em algumas aplicações a amostragem é necessária devido ao custo computacional em processar todos os dados disponíveis.

Outra aplicação é para dividir o conjunto de exemplos em conjuntos para treinamento e para teste. Nesse caso, o conjunto de treinamento é utilizado para construir os modelos utilizando as técnicas de aprendizado de máquina e o conjunto de teste é usado para avaliar o modelo gerado, utilizando dados desconhecidos para o modelo.

Em se tratando de amostragem, a grande questão é encontrar um conjunto de dados que seja representativo, ou seja, que tenha as mesmas propriedades de interesse que o conjunto original. A técnica mais simples é a *amostragem aleatória*, onde a probabilidade de selecionar um exemplo é a mesma para todo o conjunto. Existem, porém, técnicas mais sofisticadas, como a *amostragem estratificada*, onde o conjunto de dados é dividido em

diversas partições de acordo com características dos dados, e então exemplos são escolhidos aleatoriamente dentre cada uma das partições.

Outro aspecto importante é quanto à remoção dos dados do conjunto. A abordagem mais comum é de retirar os exemplos escolhidos do conjunto, de forma que um mesmo exemplo não pode ser escolhido mais de uma vez. Pode-se também optar por não remover os exemplos escolhidos, com possibilidade de serem escolhidos repetidamente.

A prática comumente utilizada é de usar a amostragem aleatória, retirando os exemplos escolhidos do conjunto e utilizando uma proporção de 80/20 para exemplos de treinamento e teste, respectivamente. Variações com proporções entre 80/20 e 60/40 são apropriadas para a maioria das aplicações [31].

4 Soluções Propostas e Resultados

Para atingir os objetivos traçados no Capítulo 2 e implementar os métodos apresentados no Capítulo 3 utilizou-se a linguagem de programação Python [32], que permite implementar as técnicas de aprendizado de máquina com uma sintaxe simples e documentação vasta, juntamente com as bibliotecas Pandas [33], que auxilia no tratamento dos dados, e Scikit-learn [34], que auxilia na implementação dos algoritmos.

Este capítulo apresenta o condicionamento dos dados que foi realizado e logo após as soluções empregadas para resolver os problemas juntamente com os resultados respectivamente obtidos.

4.1 Dados

Os dados disponíveis no cadastro de clientes em bancos de dados de empresas do varejo possuem, em geral, poucas informações que podem ser utilizadas para criar sistemas de recomendação, já que, diferente de sistemas de *e-commerce*, é muito difícil rastrear as ações de cada cliente e cruzar os seus dados com outras fontes, como históricos de navegação ou até mesmo dados provenientes de redes sociais.

Outro fator é que os clientes geralmente não estão dispostos a despender muito tempo para o registro no cadastro, ou, em muitos casos, se sentem pouco à vontade em fornecer certos tipos de informação, de modo que os dados de cadastro se limitam, em geral, a poucas informações que permitem segmentar e classificar os clientes.

É imprescindível, então, realizar uma limpeza e condicionamento para tornar os dados aptos a serem utilizados pelos algoritmos estudados.

4.1.1 Dados disponíveis

Os dados utilizados para implementar e testar as soluções propostas são provenientes de um sistema de ERP (*Enterprise Resource Planning*) utilizado pela empresa. Os dados estão disponibilizados em um banco de dados relacional e as informações disponíveis são referentes ao histórico de compra e venda, cadastro de clientes, informações das filiais, estoque, etc.

Para segmentação dos clientes, os seguintes campos estão disponíveis:

- Cidade - município de residência do cliente;
- UF - estado de residência do cliente;

- região - região em que se situa a loja, podendo ser SUL, SUDESTE, CENTRO-OESTE, NORTE e NORDESTE;
- Faixa etária - faixa etária do cliente;
- Sexo - gênero do cliente, podendo ser MASCULINO, FEMININO ou OUTROS;
- Data de cadastro - data em que o cadastro foi realizado.

Já para a caracterização dos produtos, os seguintes campos estão disponíveis:

- Descrição Produto - texto que traz uma descrição sobre o produto, com nível de detalhamento elevado;
- Tipo Produto - tipo do produto em questão;
- Grupo Produto - agrupamento que traz produtos relacionados por algum critério de similaridade;
- Subgrupo Produto - traz alguma característica adicional ao grupo do produto.

4.1.2 Condicionamento dos dados

Para buscar os dados relevantes teve-se que realizar consultas utilizando linguagem SQL (*Structured Query Language*). Após extrair os dados dos campos pertinentes, foi realizada uma limpeza utilizando a técnica *Listwise Deletion*, sendo mantidos apenas os registros de venda cujo comprador possuísse o cadastro com ambos os atributos Sexo, Faixa Etária, Estado Civil e Região preenchidos.

Antes da limpeza, o volume de dados consistia de 5.521.955 registros de vendas de produtos associadas a clientes cadastrados. Ao fim da limpeza, restaram 130.193 registros.

4.2 Problema 1

O primeiro problema, apresentado na Seção 2.1.1, consiste em buscar o perfil de cliente mais propenso a comprar um determinado produto. A solução desse problema se resume em treinar um classificador utilizando o histórico de vendas das lojas. Para tanto se decidiu utilizar os atributos Região, Sexo e Faixa etária como variáveis de saída. Analisando o conjunto de exemplos disponíveis se percebeu que o campo Faixa etária possuía muitas classes, sendo que algumas apresentavam representatividade muito baixa. A Tabela 7 mostra a distribuição dos dados para esse atributo.

Tabela 7 – Distribuição dos dados para o atributo *Faixa etária*

Classe	Quantidade de exemplos
25 a 34	45120
35 a 44	28061
15 a 24	27600
45 a 55	17846
55 a 65	9610
65+	1856

Para adequar o problema às necessidades da empresa, e trazer uma maior uniformidade quanto à distribuição das faixas etárias, decidiu-se por agrupar as classes. A distribuição dos dados após esse agrupamento está apresentada na Tabela 8.

Tabela 8 – Distribuição dos dados para o atributo *Faixa etária* após agrupamento

Classe	Quantidade de exemplos
35+	57373
25 a 34	45120
24-	27600

A Tabela 9 apresenta os atributos utilizados como variáveis de saída com o respectivo número de classes existente em cada uma, após a adequação apresentada. A Tabela 10 apresenta os atributos utilizados como variáveis de entrada.

Tabela 9 – Classes por atributo de saída para o problema 1

Atributo	Número de classes
Sexo	2
Faixa Etária	3
Região	5

Tabela 10 – Classes por atributo de entrada para o problema 1

Atributo	Número de classes
Fabricante Produto	21
Grupo Produto	35
Subgrupo Produto	106

Com os dados devidamente condicionados se treinou modelos utilizando os algoritmos Árvore de Decisão e *k-Nearest Neighbors*. Se optou por treinar um modelo para cada um dos atributos de saída, já que a necessidade da empresa é de realizar as análises separadamente, de modo que se obteve 3 modelos para cada algoritmo utilizado.

O conjunto de exemplos foi então dividido em dois subconjuntos, um para treinamento e um para teste, sendo que a proporção utilizada foi de 80/20, respectivamente, totalizando 104.074 exemplos para treinamento e 26.019 exemplos para teste.

4.2.1 Resultados

Após o treinamento, os modelos obtidos foram avaliados utilizando a métrica descrita na Equação (3.1). Para o algoritmo *k-Nearest Neighbors* foram realizados testes utilizando diferentes valores de k. A Tabela 11 apresenta os resultados obtidos.

Tabela 11 – Taxas de acerto obtidos com k-Nearest Neighbors para diferentes valores de k

Atributo	$k = 10$	$k = 50$	$k = 100$	$k = 200$
Sexo	0.816941	0.824464	0.835966	0.836465
Região	0.638995	0.680149	0.681579	0.681298
Faixa Etária	0.423998	0.451401	0.458088	0.456666

Analisando a Tabela 11 percebe-se que os melhores resultados são dados para k igual a 100. Esses resultados foram utilizados para comparação com os resultados obtidos com a modelagem utilizando Árvore de Decisão. A Tabela 12 dispõe esses resultados.

Tabela 12 – Resultados obtidos para o problema 1

Atributo	Acerto DT	Acerto KNN
Sexo	0.835889	0.835966
Região	0.681733	0.681579
Faixa Etária	0.458011	0.458088

A partir da Tabela 9, se infere que as probabilidades de acerto para uma escolha aleatória seriam de 50%, 20% e 33% para os atributos Sexo, Faixa Etária e Região, respectivamente. Os resultados da Tabela 12 mostram que, tanto para os modelos obtidos com Árvore de Decisão quanto com *k-Nearest Neighbors*, todas as taxas de acerto foram superiores à uma escolha aleatória entre as classes possíveis.

4.3 Problema 2

O segundo problema também objetiva treinar um classificador utilizando dados do histórico de vendas das lojas. Os atributos utilizados como variáveis de entrada foram justamente os atributos apresentados na Tabela 9 como variáveis de saída para o Problema 1. Já a variável de saída escolhida foi o atributo Grupo Produto.

Uma dificuldade surge ao tentar aplicar diretamente os algoritmos previamente apresentados para solucionar esse problema, relacionada à quantidade de classes que

possuem os atributos de entrada e saída. Fica claro que a quantidade de informações disponíveis na entrada não é suficiente para mapear as 35 possíveis classes de saída, de modo que um classificador treinado para esse problema possuiria uma taxa de acerto muito pequena.

Para contornar essa dificuldade deve-se reduzir o problema, com duas principais soluções possíveis. Uma é remover algumas das classes, caso elas não apresentem muita relevância no contexto do problema. Outra é agrupar as classes de saída de alguma forma.

Optou-se por realizar um agrupamento utilizando algoritmo genético. Os indivíduos representam uma regra de agrupamento e o objetivo é otimizar a taxa de acerto de um modelo treinado com os dados do atributo de saída agrupados seguindo as regras de agrupamento de cada indivíduo.

4.3.1 *Clustering* utilizando Algoritmo Genético

Para encontrar um agrupamento que possibilitasse que o modelo treinado para classificação obtivesse um rendimento melhor o algoritmo genético apresentado no Algoritmo 3 teve que ser modificado.

As principais adaptações necessárias dizem respeito à representação do indivíduo, aos operadores de *crossover* e de mutação e à função *Fitness*.

4.3.1.1 Representação dos indivíduos

O indivíduo implementado é uma representação da regra de agrupamento, ou seja, é codificado de forma que as q classes existentes para o atributo Grupo Produto estejam mapeadas em n *clusters*. A Equação 4.1 ilustra a representação de um indivíduo da população, para o caso hipotético de se agrupar 6 produtos em 3 *clusters*.

$$\left\{ \overbrace{[PROD6, PROD4]}^{n1}, \overbrace{[PROD1, PROD3]}^{n2}, \overbrace{[PROD2, PROD5]}^{n3} \right\} \tag{4.1}$$

4.3.1.2 *Crossover*

O operador *crossover* também teve que sofrer adaptações, já que o indivíduo não é representado como um número binário. Optou-se por utilizar o *crossover* de dois pontos. Depois de realizar a operação deve-se regularizar as classes que se repetem, já que cada classe deve estar em apenas um *cluster*, e todas as classes devem estar em algum dos *clusters*. Essa adequação é realizada verificando a lista de classes da esquerda para a direita. Caso alguma classe já tenha sido utilizada, ela é substituída pela primeira classe não utilizada, em ordem crescente.

Um exemplo de aplicação prática do operador *Crossover* adaptado pode ser vista no Apêndice C.

4.3.1.3 Mutação

O operador de mutação também teve que ser levemente alterado já que a representação dos indivíduos não é binária e as classes representadas não podem se repetir. Assim como no operador apresentado na seção 3.5.2.2, um bit é escolhido aleatoriamente, nesse caso uma classe, mas então é alterado para uma classe escolhida aleatoriamente entre as classes existentes. Após essa alteração a classe alterada será uma repetição de outra já existente, logo uma adequação, como a apresentada na Seção 4.3.1.2, deve ser utilizada.

4.3.1.4 Função *Fitness*

A escolha da função *Fitness* varia para cada aplicação desejada do algoritmo genético, dependendo do que se pretende otimizar. Para o presente problema o objetivo é obter um agrupamento que permita treinar um modelo com maior taxa de acerto possível. Para tanto, a função *Fitness* consiste em avaliar p modelos de classificação, um para cada indivíduo da população, onde cada indivíduo representa uma regra de agrupamento. Para atender essa especificação, a função *Fitness* foi definida como:

$$Fitness(h_p) = accuracy(y, \hat{y})^2 \tag{4.2}$$

onde h_p é o modelo treinado para o indivíduo p, y são os valores de saída dos exemplos de validação, $\hat{y}$ são os valores previstos pelo modelo h_p para as entradas dos exemplos de validação e $accuracy(y, \hat{y})$ é definida pela Equação (3.1).

Para o treinamento dos modelos associados a cada indivíduo se optou por utilizar o aprendizado por Árvore de Decisão. O principal fator que levou à escolha desse algoritmo foi o fato de não ser necessário ajustar nenhum parâmetro adicional em sua implementação, como para o *k-Nearest Neighbors*. Considerando que se espera um tempo de processamento muito elevado para a execução do algoritmo genético, em vista às dimensões do problema em questão, a abordagem utilizada para o Problema 1, que consiste em testar diversos valores para k para atingir um resultado satisfatório, se tornaria inviável.

4.3.1.5 Implementação

Para implementar o algoritmo genético, teve-se que reduzir a quantidade de dados utilizados para treinamento e validação dentro da função *Fitness*, visando possibilitar o processamento do algoritmo em tempo factível. Esse é um parâmetro importante para a implementação e foi inserido na implementação do algoritmo como $sample\_size$.

Outro aspecto importante é a divisão dos dados, já que a cada iteração do algoritmo genético, p treinamentos são realizados. Além disso, a avaliação utilizando a função $Fitness$ escolhida deve ser aplicada a todos os indivíduos, preferencialmente a partir dos mesmos exemplos de validação. Assim como no Problema 1, o conjunto de exemplos foi dividido em conjuntos de treinamento e de teste, respectivamente S_{train} e S_{test}, com a proporção 80/20.

Para o treinamento e avaliação de cada indivíduo é utilizado um subconjunto, S_i, que possui tamanho $sample\_size$, com exemplos escolhidos aleatoriamente dentro do conjunto de treinamento S_{train}. O subconjunto S_i é então dividido em um conjunto para treinamento, S_{itrain}, e outro para a avaliação, S_{itest}, utilizados para construção e avaliação de modelos para todos os indivíduos da população. A divisão do conjunto S_i segue também a proporção 80/20.

Um novo subconjunto S_i, com sua divisão entre S_{itrain} e S_{itest}, é obtido para cada iteração do algoritmo genético, sendo mantido para todos os indivíduos testados na mesma iteração. Já o conjunto S é dividido em S_{train} e S_{test} no início do algoritmo, com a divisão dos dados sendo mantida inalterada durante toda a execução. A Figura 6 representa graficamente a divisão aplicada nos dados.

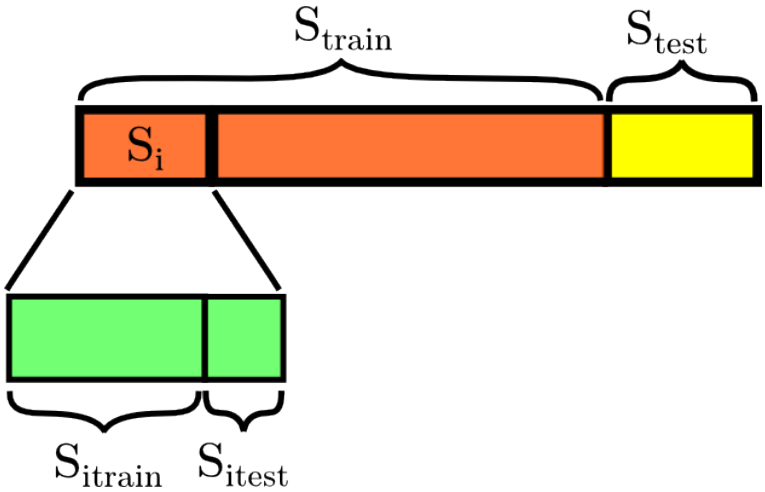

Figura 6 – Divisão dos dados do conjunto de exemplos S para o algoritmo genético

Para cada um dos p indivíduos da população, uma Árvore de Decisão é treinada utilizando o subconjunto S_{itrain}. O indivíduo é, então, avaliado utilizando os dados do subconjunto S_{itest}, segundo a Equação (3.1).

O critério de parada utilizado foi o número de iterações, j.

4.3.1.6 Resultado *Clustering*

A Tabela 13 apresenta os parâmetros utilizados para a execução do algoritmo genético.

Tabela 13 – Parâmetros utilizados pelo algoritmo genético

Parâmetro	Valor
n	5
p	100
r	0.6
m	0.001
j	400
$sample\_size$	1000

A Figura 7 apresenta a evolução do valor máximo da função *Fitness* dentre os p indivíduos através das j iterações. Analisando esse gráfico, percebe-se que para as primeiras iterações existe uma evolução mais rápida da função $Fitness$, que converge assintoticamente para o resultado final, de cerca de 0,46. Não é possível garantir que a convergência obtida é global, porém se espera, ao utilizar os operadores genéticos, que o espaço de hipóteses tenha sido bem explorado.

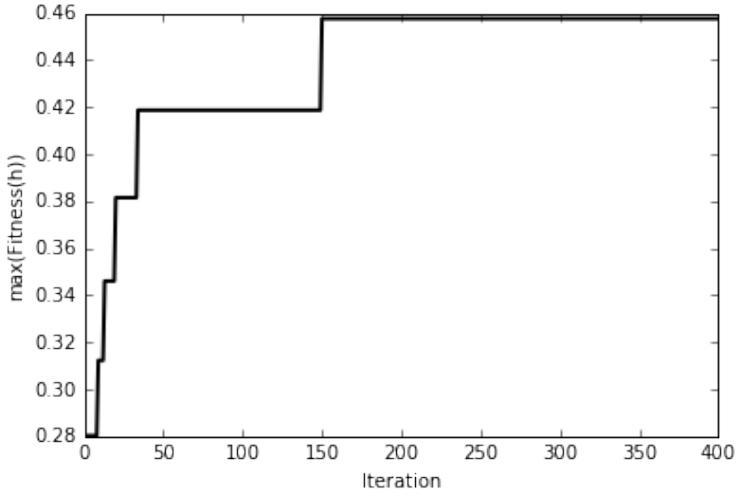

Figura 7 – Evolução da função *Fitness*

Após a execução das j iterações do algoritmo genético, foi obtido um agrupamento que maximiza a taxa de acerto para uma Árvore de Decisão, dado pelo indivíduo mais apto. O agrupamento resultante está apresentado na Tabela 26, no Anexo A.

4.3.1.7 Resultado classificação

Utilizando o agrupamento apresentado na Tabela 26, todos os exemplos do conjunto S foram agrupados, de modo que o número de classes do atributo de saída foi reduzido para 5.

Após o agrupamento, se treinou um classificador utilizando os exemplos do conjunto S_{train}. Após o treinamento, o modelo obtido foi avaliado utilizando a métrica descrita na

Equação (3.1). Os resultados estão dispostos na Tabela 14.

Tabela 14 – Resultados obtidos para o problema 2

Atributo	Acerto DT
Grupo Produto	0.448308870453

Considerando os 5 *clusters* obtidos com o agrupamento, para uma escolha aleatória a probabilidade de acerto seria de 20%. O resultado obtido é, então, superior ao esperado por um classificador aleatório.

5 Conclusões

Esse documento apresentou uma aplicação de técnicas de aprendizado de máquina visando resolver problemas relevantes para empresas do setor do varejo. Poucas são as iniciativas no mercado nacional que utilizam essas técnicas com um direcionamento para esse setor, de modo que o mercado brasileiro pode ganhar muito em produtividade e eficiência com a utilização dessas tecnologias, fortalecendo as empresas nacionais e a economia do país.

Foram apresentados algoritmos que possibilitam a modelagem de processos descritos por variáveis categóricas e, para solucionar um dos problemas, se utilizou uma adaptação do algoritmo genético para realizar o agrupamento dos dados. Além disso, algumas questões que surgem quanto ao condicionamento dos dados durante o desenvolvimento e implementação dos algoritmos foram abordadas, e técnicas para contornar esses problemas foram apresentadas.

Concluiu-se também que a complexidade dos problemas a serem abordados e a qualidade dos resultados depende totalmente dos dados a que se tem acesso, de modo que não se deve exigir dos algoritmos resolver problemas os quais os dados não possibilitam a solução. Em tais casos é mais eficaz reduzir a complexidade do problema ou enriquecer os dados com informações suplementares do que buscar algoritmos mais complexos.

Como resultado, obteve-se modelos que permitem a solução de dois problemas de interesse. A partir dos modelos gerados para solução do primeiro problema, que consiste em obter informações de perfil de cliente a partir de determinado produto para venda, é possível comparar o desempenho das técnicas Árvore de Decisão e *k-Nearest Neighbors*. Ambas as técnicas utilizadas apresentaram taxas de acerto muito semelhantes para os modelos gerados.

Já para avaliar a qualidade dos modelos gerados, se espera que a taxa de acerto seja superior a uma escolha aleatória, representando que existiu algum aprendizado. Para o primeiro problema, o modelo que utiliza o atributo Sexo como saída teve um resultado bastante satisfatório, com acerto superior a 80%, significativamente acima do acerto para uma escolha aleatória entre as duas classes, que seria de 50%. O modelo para o atributo Região também apresentou um bom resultado, com 68% de acerto para as cinco classes possíveis, onde se espera um acerto de 20% para escolhas aleatórias. O modelo para o atributo Faixa Etária não apresentou um resultado tão expressivo, com 45% de acerto, ainda assim superior à probabilidade de acerto em se escolher aleatoriamente dentre as três possíveis classes, que seria de 33%.

Para o segundo problema, que consiste em indicar um produto para ser vendido dado um perfil de cliente, obteve-se um agrupamento de produtos, formado por 5 *clusters*, que maximiza o acerto para uma Árvore de Decisão treinada. Treinando uma Árvore de Decisão utilizando esse agrupamento resultou em um acerto de 44%, valor bastante significativo para as cinco possíveis classes, onde, para escolhas aleatórias dentre os valores, se espera um acerto de 20%.

Conclui-se então que, dentro das limitações referentes aos dados disponíveis, foi possível treinar modelos utilizando técnicas de aprendizado de máquina que descrevem bem a base de dados de uma rede de lojas de varejo.

Como possibilidade de trabalho futuro, para o problema de recomendação de produtos para clientes de lojas do varejo, é interessante trabalhar o histórico de compras dos clientes, permitindo então utilizar as compras anteriores como informações para prever as suas ações futuras. Para tanto, pode-se buscar técnicas e algoritmos que já são amplamente utilizados em sistemas de *e-commerce*. Além disso, pode-se estudar ferramentas e técnicas para enriquecer a base de dados dos clientes utilizando cruzamento com informações de outras fontes, como redes sociais e dados provenientes de cadastros em outros sistemas.

O agrupamento obtido pode, também, ser melhor estudado. Uma aplicação de interesse seria obter um agrupamento que represente itens que comumente são vendidos juntos, permitindo organizar a disposição dos produtos fisicamente na loja, por exemplo. Para atingir esse resultado, pode-se utilizar a solução baseada em algoritmo genético apresentada, porém uma avaliação, e potencial mudança, da função $Fitness$ deve ser realizada.

Para permitir a manipulação dos modelos obtidos e possibilitar a classificação de novas entradas, uma interface deve ser desenvolvida. Diferentes abordagens podem ser utilizadas para tal. Pode-se, por exemplo, integrar o sistema aos computadores utilizados no caixa das lojas, de modo que no momento da venda o vendedor já tenha acesso a recomendações de produtos para aquele cliente. Outra opção consiste em utilizar os modelos em um abordagem mais estratégica, com análises que entreguem aos gestores um melhor entendimento das relações entre clientes e produtos vendidos, permitindo canalizar esforços de marketing para a rede, por exemplo.

Referências

1 SMOLA, A.; VISHWANATHAN, S. Introduction to machine learning. *Cambridge University, UK*, v. 32, p. 34, 2008. Citado na página 10.

2 SHALEV-SHWARTZ, S.; BEN-DAVID, S. *Understanding Machine Learning: From Theory to Algorithms*. [S.l.]: Cambridge University Press, 2014. Citado na página 10.

3 NIZZA, C. L. A influência da comunicação na busca de informações do consumidor de varejo. *Revista Pretexto*, v. 4, n. 1, 2003. Citado na página 12.

4 eMarketer, Inc. *Worldwide retail ecommerce sales: emarketers updated estimates and forecast through 2019*. [S.l.]: eMarketer, 2015. Citado na página 12.

5 TRIANTAPHYLLOU, E. *Multi-criteria decision making methods: a comparative study*. [S.l.]: Springer Science & Business Media, 2013. v. 44. Citado na página 16.

6 PLOUS, S. *The psychology of judgment and decision making*. [S.l.]: Mcgraw-Hill Book Company, 1993. Citado na página 16.

7 SOLOMONOFF, R. J. A formal theory of inductive inference. part i. *Information and control*, Elsevier, v. 7, n. 1, p. 1–22, 1964. Citado na página 16.

8 RÄTSCH, G. A brief introduction into machine learning. In: *21st Chaos Communication Congress*. [S.l.: s.n.], 2004. Citado na página 16.

9 MITCHELL, T. M. *The discipline of machine learning. July 2006 CMU-ML-06-108 School of Computer Science*. [S.l.]: Carnegie Mellon University, Pittsburgh, PA, 2006. Citado na página 17.

10 DOMINGOS, P. A few useful things to know about machine learning. *Communications of the ACM*, ACM, v. 55, n. 10, p. 78–87, 2012. Citado na página 17.

11 DUDA, R. O.; HART, P. E.; STORK, D. G. Unsupervised learning and clustering. *Pattern classification*, Wiley New York; Chichester, p. 519–598, 2001. Citado na página 17.

12 SATHYA, R.; ABRAHAM, A. Comparison of supervised and unsupervised learning algorithms for pattern classification. *Int. J. Adv. Res. Artificial Intell*, Citeseer, v. 2, n. 2, p. 34–38, 2013. Citado na página 17.

13 MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. *Foundations of machine learning*. [S.l.]: MIT press, 2012. Citado na página 17.

14 SU, J.; ZHANG, H. A fast decision tree learning algorithm. In: *AAAI*. [S.l.: s.n.], 2006. v. 6, p. 500–505. Citado na página 19.

15 BORDA, M. Statistical and informational model of an its. In: *Fundamentals in Information Theory and Coding*. [S.l.]: Springer, 2011. p. 7–52. Citado na página 19.

16 QUINLAN, J. R. Induction of decision trees. *Machine Learning*, Springer, v. 1, n. 1, p. 81–106, 1986. Citado na página 20.

17 ROKACH, L.; MAIMON, O. Decision trees. In: *Data Mining and Knowledge Discovery Handbook*. [S.l.]: Springer, 2005. p. 165–192. Citado na página 20.

18 MITCHELL, T. M. et al. *Machine learning*. [S.l.]: McGraw-Hill New York, 1997. Citado na página 21.

19 SUGUNA, N.; THANUSHKODI, K. An improved k-nearest neighbor classification using genetic algorithm. *International Journal of Computer Science Issues*, v. 7, n. 2, p. 18–21, 2010. Citado na página 23.

20 ABBOTT, D. *Applied Predictive Analytics: Principles and Techniques for the Professional Data Analyst*. [S.l.]: John Wiley & Sons, 2014. Citado na página 23.

21 THORNTON, C. et al. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In: ACM. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2013. p. 847–855. Citado na página 24.

22 MITCHELL, M. *An Introduction to Genetic Algorithms*. [S.l.]: MIT press, 1998. Citado na página 28.

23 WHITLEY, D. A genetic algorithm tutorial. *Statistics and Computing*, Springer, v. 4, n. 2, p. 65–85, 1994. Citado na página 28.

24 CUMMINGS, C. E. et al. State machine coding styles for synthesis. In: *SNUG'98 (Synopsys Users Group San Jose, CA, 1998) Proceedings*. [S.l.: s.n.], 1998. Citado na página 28.

25 HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* [S.l.]: U Michigan Press, 1975. Citado na página 28.

26 MAGALHAES-MENDES, J. A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers*, v. 12, n. 4, p. 164–173, 2013. Citado na página 29.

27 KOTSIANTIS, S.; KANELLOPOULOS, D.; PINTELAS, P. Data preprocessing for supervised leaning. *International Journal of Computer Science*, v. 1, n. 2, p. 111–117, 2006. Citado na página 30.

28 MÜLLER, H.; FREYTAG, J.-C. *Problems, methods, and challenges in comprehensive data cleansing*. [S.l.]: Professoren des Inst. Für Informatik, 2005. Citado na página 30.

29 RUBIN, D. B.; LITTLE, R. J. Statistical analysis with missing data. *Hoboken, NJ: J Wiley & Sons*, 2002. Citado na página 31.

30 ALLISON, P. D. *Multiple imputation for missing data: A cautionary tale*. [S.l.]: Philadelphia, 1999. Citado na página 32.

31 RICCI, F.; ROKACH, L.; SHAPIRA, B. *Introduction to Recommender Systems Handbook*. [S.l.]: Springer, 2011. Citado na página 34.

32 ROSSUM, G. van et al. *Python language website*. 2007. Citado na página 35.

33 MCKINNEY, W. Pandas, python data analysis library. 2015. *Reference Source*, 2014. Citado na página 35.

34 PEDREGOSA, F. et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, v. 12, n. Oct, p. 2825–2830, 2011. Citado na página 35.

Apêndices

APÊNDICE A – Desenvolvimento do exemplo com Árvore de Decisão

Essa resolução utiliza os dados presentes na Tabela 1, que serão tratados nessa seção como conjunto S.

Primeiramente deve-se verificar se todos os exemplos possuem os mesmos valores para o atributo-objetivo. No caso do conjunto S têm-se exemplos cujo Grupo Produto é C1 e outros em que é C2. Além disso a lista de atributos não é vazia, logo pode-se prosseguir no algoritmo. Em seguida é necessário definir qual atributo melhor classifica a coleção de exemplos S. Para tanto se calcula a entropia da coleção de exemplos, utilizando a equação 3.5.

$$Entropy(S) = -\frac{11}{20}\log_2\left(\frac{11}{20}\right) - \frac{9}{20}\log_2\left(\frac{9}{20}\right) = 0,993 \tag{A.1}$$

Com o valor de entropia é possível calcular o ganho de informação, utilizando a equação 3.6, para cada um dos atributos:

$$\begin{aligned} Gain(S, Região) = Entropy(S) - \frac{|S_{Sul}|}{|S|}Entropy(S_{Sul}) - \frac{|S_{Sudeste}|}{|S|}Entropy(S_{Sudeste}) \\ -\frac{|S_{Nordeste}|}{|S|}Entropy(S_{Nordeste}) = 0,993 - \frac{7}{20}\left[-\frac{4}{7}log_2\left(\frac{4}{7}\right) - \frac{3}{7}\log_2\left(\frac{3}{7}\right)\right] \\ -\frac{8}{20}\left[-\frac{4}{8}\log_2\left(\frac{4}{8}\right) - \frac{4}{8}\log_2\left(\frac{4}{8}\right)\right] - \frac{5}{20}\left[-\frac{3}{5}\log_2\left(\frac{3}{5}\right) - \frac{2}{5}\log_2\left(\frac{2}{5}\right)\right] = 0,00521 \end{aligned} \tag{A.2}$$

$$\begin{aligned} Gain(S, Estado\_civil) = Entropy(S) - \frac{|S_{Casado}|}{|S|}Entropy(S_{Casado}) \\ -\frac{|S_{Solteiro}|}{|S|}Entropy(S_{Solteiro}) - \frac{|S_{Outros}|}{|S|}Entropy(S_{Outros}) = \\ 0,993 - \frac{7}{20}\left[-\frac{6}{7}\log_2\left(\frac{6}{7}\right) - \frac{1}{7}\log_2\left(\frac{1}{7}\right)\right] - \frac{6}{20}\left[-\frac{3}{6}\log_2\left(\frac{3}{6}\right) - \frac{3}{6}\log_2\left(\frac{3}{6}\right)\right] \\ -\frac{7}{20}\left[-\frac{2}{7}\log_2\left(\frac{2}{7}\right) - \frac{5}{7}\log_2\left(\frac{5}{7}\right)\right] = 0,183 \end{aligned} \tag{A.3}$$

$$\begin{aligned} Gain(S, Sexo) = 0,993 - \frac{14}{20}\left[-\frac{7}{14}\log_2\left(\frac{7}{14}\right) - \frac{7}{14}\log_2\left(\frac{7}{14}\right)\right] \\ -\frac{6}{20}\left[-\frac{4}{6}\log_2\left(\frac{4}{6}\right) - \frac{2}{6}\log_2\left(\frac{2}{6}\right)\right] = 0,0173 \end{aligned} \tag{A.4}$$

Por ter o maior valor de ganho de informação, o atributo Estado Civil é o que melhor classifica a coleção de exemplos. A partir do atributo escolhido, deve-se acrescentar, a partir da raiz, um arco para cada valor possível desse atributo. Cada arco deve direcionar para um subconjunto em que o atributo escolhido é indicado pelo respectivo arco. A Figura 8 apresenta o diagrama da árvore gerada nessa iteração.

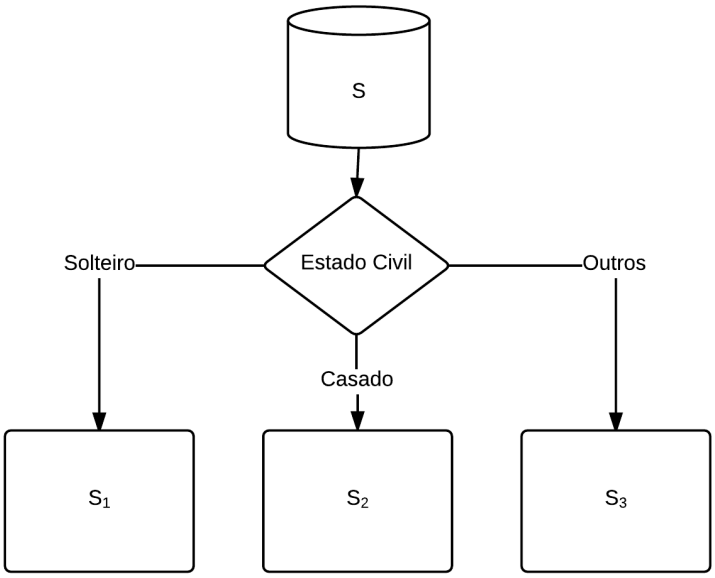

Figura 8 – Diagrama da árvore após escolha do primeiro atributo de classificação

A Tabela 15 representa o subconjunto S_1 formada por exemplos em que o valor para o Estado Civil é Solteiro. As Tabelas 16 e 17 representam os subconjuntos S_2 e S_3, cujos valores para o Estado Civil são Casado e Outros, respectivamente.

Região	Sexo	Grupo Produto
Sul	Masculino	C1
Sudeste	Feminino	C2
Sudeste	Masculino	C2
Sudeste	Feminino	C2
Sudeste	Masculino	C1
Sul	Masculino	C1

Tabela 15 – Coleção S_1

Região	Sexo	Grupo Produto
Nordeste	Feminino	C1
Nordeste	Masculino	C1
Sul	Feminino	C2
Sul	Feminino	C1
Nordeste	Feminino	C1
Nordeste	Feminino	C1
Sudeste	Masculino	C1

Tabela 16 – Coleção S_2

Região	Sexo	Grupo Produto
Sul	Feminino	C2
Sul	Feminino	C1
Nordeste	Feminino	C2
Sul	Feminino	C2
Sudeste	Masculino	C2
Sudeste	Feminino	C1
Nordeste	Feminino	C2

Tabela 17 – Coleção S_3

Para cada um dos subconjuntos obtidos deve-se repetir o mesmo processo realizado no conjunto S, ou seja, verificar se todos os exemplos possuem apenas um valor para o atributo-alvo ou se a lista de atributos está vazia, e em caso negativo calcular o ganho de informação para cada atributo a fim de selecionar o atributo que melhor classifica o subconjunto. A Tabela 18 apresenta os ganhos de informação calculados para os atributos de cada um dos subconjuntos S_1, S_2 e S_3, permitindo assim escolher o atributo que melhor classifica cada um desses subconjuntos.

Tabela 18 – Valores de ganho de informação para os atributos dos subconjuntos S_1, S_2 e S_3

Atributo	S_1	S_2	S_3
Região	0,459	0,577	0,0304
Sexo	0,0817	0,347	0,076

Para os subconjuntos S_1 e S_2, o atributo Região é o que melhor os classifica. Já para o subconjunto S_3 é o atributo Sexo que o melhor classifica.

A Figura 9 apresenta a estrutura da árvore após a classificação dos conjuntos S_1, S_2 e S_3.

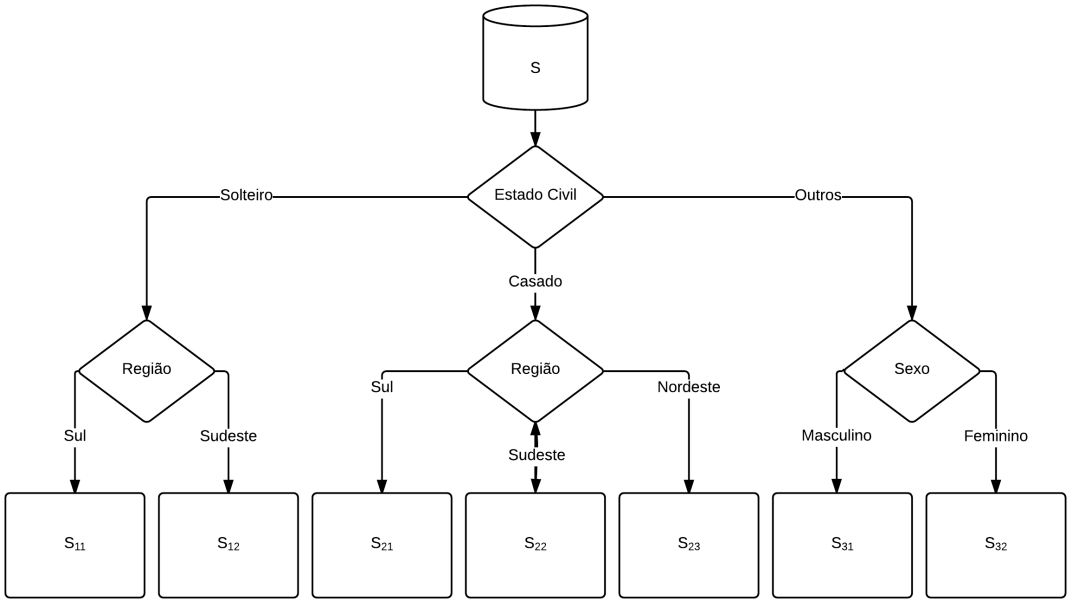

Figura 9 – Diagrama da árvore após escolha do segundo atributo de classificação

As Tabelas 19 a 25 apresentam os dados contidos em cada um dos subconjuntos obtidos.

Sexo	Grupo Prod.
Masc.	C1
Fem.	C1

Tabela 19 – Coleção S_{11}

Sexo	Grupo Prod.
Fem.	C2
Masc.	C2
Fem.	C2
Masc.	C1

Tabela 20 – Coleção S_{12}

Sexo	Grupo Prod.
Fem.	C1
Fem.	C2

Tabela 21 – Coleção S_{21}

Sexo	Grupo Prod.
Fem.	C1
Masc.	C1

Tabela 22 – Coleção S_{22}

Sexo	Grupo Prod.
Fem.	C1
Masc.	C1
Fem.	C1

Tabela 23 – Coleção S_{23}

Região	Grupo Prod.
Sudeste	C2

Tabela 24 – Coleção S_{31}

Região	Grupo Produto
Sul	C2
Sul	C1
Nordeste	C2
Sul	C2
Sudeste	C1
Nordeste	C2

Tabela 25 – Coleção S_{32}

Os conjuntos S_{11}, S_{22}, S_{23} e S_{31} possuem valor único para o atributo-alvo em todos os exemplos, logo para eles não é mais necessário nenhum refinamento, ou seja, apesar

de ainda terem atributos para classificação já podem ser representados como folhas a fim de evitar processamento desnecessário. Para os outros conjuntos mais uma iteração é necessária a fim de completar a classificação dos exemplos. Como apenas um atributo restou para cada um dos conjuntos restantes, não é mais necessário calcular os ganhos de informação.

Para classificar os conjuntos obtidos por seus atributos deve-se atribuir o valor mais comum do atributo alvo para cada folha da árvore. Para casos em que não existe valor mais comum único (como ocorre ao classificar o conjunto S_{21}), a escolha se torna arbitrária.

A partir dessa última iteração para refinar o modelo ilustrado na Figura 9 se obtém o modelo final da árvore, ilustrado na Figura 4 da subseção 3.3.4.

APÊNDICE B – Desenvolvimento do exemplo com k-Nearest Neighbors

Para facilitar a escolha das instâncias mais próximas, pode-se acrescentar as instâncias a serem classificadas em um gráfico juntamente com os exemplos de treinamento. Essa representação está ilustrada na Figura 10. Pode-se buscar as k instâncias mais próximas visualmente com auxílio desse gráfico.

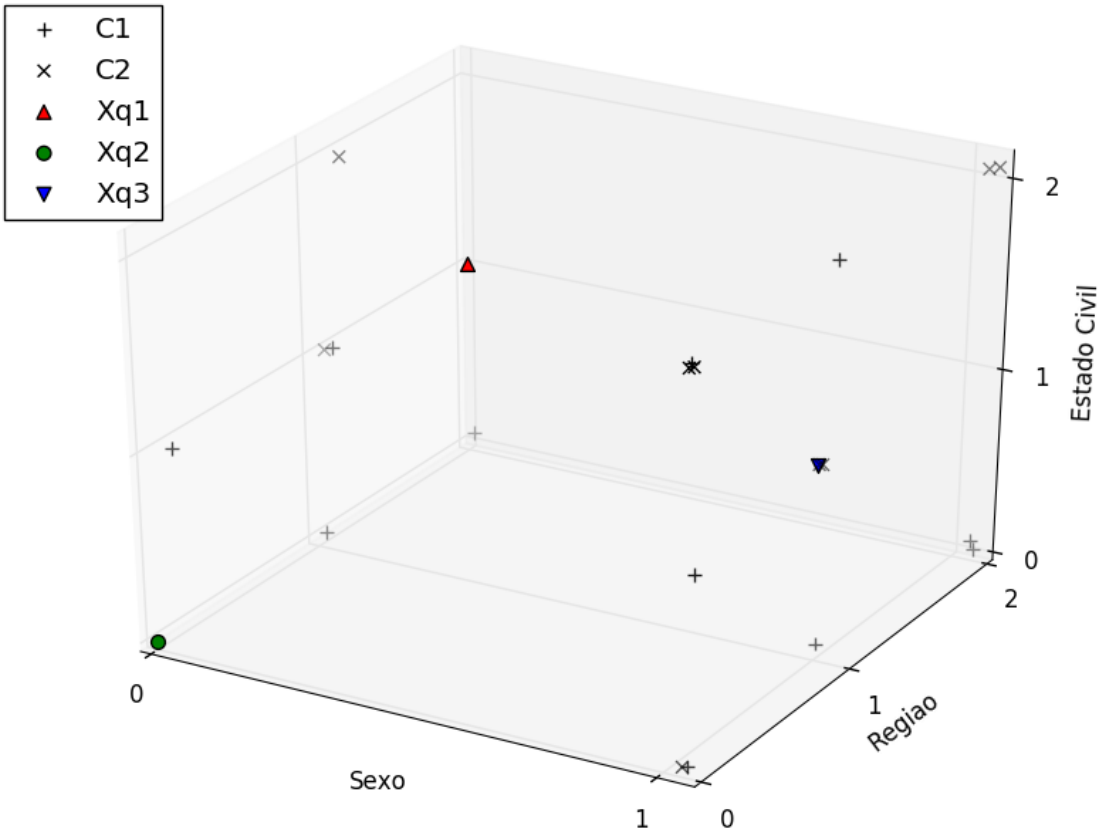

Figura 10 – Conjunto de exemplos de treinamento e instâncias a serem classificadas

B.1 $k = 3$

Para a instância x_{q1} as k instâncias mais próximas dentre os exemplos de treinamentos são x_7, x_{14} e x_{18}, cujas distâncias euclidianas são dadas por:

$$d(x_{q1}, x_7) = \sqrt{(2-2)^2 + (0-1)^2 + (2-2)^2} = 1 \tag{B.1}$$

$$d(x_{q1}, x_{14}) = d(x_{q1}, x_{18}) = \sqrt{(2-1)^2 + (0-0)^2 + (2-2)^2} = 1 \tag{B.2}$$

As classes dessas k instâncias mais próximas são $f(x_7) = C2$, $f(x_{14}) = C2$ e $f(x_{18}) = C2$. A classificação para a instância x_{q1} é dada por:

$$\hat{f}(x_{q1}) = \underset{v \in \{C1,C2\}}{\operatorname{argmax}} \sum_{i=\{7,14,18\}} \delta(v, f(x_i)) \tag{B.3}$$

$$\hat{f}(x_{q1}) = C2 \tag{B.4}$$

Para a instância x_{q2}, existem 4 instâncias que apresentam a menor distância, logo a escolha das k instâncias mais próximas se torna arbitrária. As instâncias escolhidas dentre os exemplos de treinamentos são x_2, x_8 e x_{20}, cujas distâncias euclidianas são dadas por:

$$d(x_{q2}, x_2) = \sqrt{(0-0)^2 + (0-0)^2 + (0-1)^2} = 1 \tag{B.5}$$

$$d(x_{q2}, x_8) = \sqrt{(0-0)^2 + (0-1)^2 + (0-0)^2} = 1 \tag{B.6}$$

$$d(x_{q2}, x_{20}) = \sqrt{(0-1)^2 + (0-0)^2 + (0-1)^2} = 1 \tag{B.7}$$

As classes dessas k instâncias mais próximas são $f(x_2) = C1$, $f(x_8) = C2$ e $f(x_{20}) = C1$. A classificação para a instância x_{q2} é dada por:

$$\hat{f}(x_{q2}) = \underset{v \in \{C1,C2\}}{\operatorname{argmax}} \sum_{i=\{2,8,20\}} \delta(v, f(x_i)) \tag{B.8}$$

$$\hat{f}(x_{q2}) = C1 \tag{B.9}$$

Por fim, para a instância x_{q3} existem duas instâncias nos exemplos que são sobrepostas, logo a distância para esses pontos é 0. Outras 3 instâncias estão equidistantes a uma distância 1, logo uma dessas deve ser escolhida arbitrariamente. As k instâncias mais próximas escolhidas dentre os exemplos de treinamentos são x_5, x_{13} e x_{19}, cujas distâncias euclidianas são dadas por:

$$d(x_{q3}, x_5) = d(x_{q3}, x_{13}) = \sqrt{(1-1)^2 + (1-1)^2 + (1-1)^2} = 0 \tag{B.10}$$

$$d(x_{q3}, x_{19}) = \sqrt{(1-0)^2 + (1-1)^2 + (1-1)^2} = 1 \tag{B.11}$$

As classes dessas k instâncias mais próximas são $f(x_5) = C2$, $f(x_{13}) = C2$ e $f(x_{19}) = C1$. A classificação para a instância x_{q3} é dada por:

$$\hat{f}(x_{q3}) = \underset{v \in \{C1,C2\}}{\operatorname{argmax}} \sum_{i=\{5,13,19\}} \delta(v, f(x_i)) \tag{B.12}$$

$$\hat{f}(x_{q3}) = C2 \tag{B.13}$$

B.2 $k = 5$

Para a classificação das instâncias utilizando $k = 5$, pode-se manter as instâncias mais próximas obtidas na seção B.1, sendo necessário encontrar outras 2 mais próximas utilizando a mesma técnica.

Para a instância x_{q1} as k instâncias mais próximas dentre os exemplos de treinamentos, com suas respectivas classes, são $x_7 = C2$, $x_{14} = C2$, $x_{15} = C1$, $x_{17} = C1$ e $x_{18} = C2$.

Para a instância x_{q2} as k instâncias mais próximas dentre os exemplos de treinamentos, com suas respectivas classes, são $x_2 = C1$, $x_8 = C2$, $x_9 = C1$, $x_{10} = C2$ e $x_{20} = C1$.

Por fim, para a instância x_{q3} as k instâncias mais próximas dentre os exemplos de treinamentos, com suas respectivas classes, são $x_5 = C2$, $x_{13} = C2$, $x_{15} = C1$, $x_{16} = C1$ e $x_{19} = C1$.

Como resultado, o Algoritmo 2 retorna as seguintes classificações:

$$\hat{f}(x_{q1}) = C2 \tag{B.14}$$

$$\hat{f}(x_{q2}) = C1 \tag{B.15}$$

$$\hat{f}(x_{q3}) = C1 \tag{B.16}$$

APÊNDICE C – Exemplo operador *Crossover*

A partir dos progenitores $parent1$ e $parent2$, descritos como:

$$parent1 = \{[PRODUTO6, PRODUTO4], [PRODUTO1, PRODUTO3], [PRODUTO2, PRODUTO5]\} \tag{C.1}$$

$$parent2 = \{[PRODUTO5, PRODUTO6], [PRODUTO3, PRODUTO4], [PRODUTO1, PRODUTO2]\} \tag{C.2}$$

e considerando $n_0 = 3$ e $n_1 = 5$, ao aplicar o operador *crossover* de dois pontos se obtém os seguintes descendentes:

$$offspring1 = \{[PRODUTO6, PRODUTO4], [PRODUTO1, PRODUTO4], [PRODUTO1, PRODUTO5]\} \tag{C.3}$$

$$offspring2 = \{[PRODUTO5, PRODUTO6], [PRODUTO3, PRODUTO3], [PRODUTO2, PRODUTO2]\} \tag{C.4}$$

Percebe-se que ambos os descendentes gerados apresentam repetição de classes, logo deve-se substituir as classes repetidas pelas não utilizadas. No descendente $offspring1$, por exemplo, no segundo *cluster* a classe $PRODUTO4$ é repetida, logo deve ser substituída pela classe $PRODUTO2$, que é a primeira classe, em ordem crescente, não utilizada. Essa mesma regra é aplicada nos demais casos de repetição, de modo que se obtém:

$$offspring1 = \{[PRODUTO6, PRODUTO4], [PRODUTO1, PRODUTO2], [PRODUTO3, PRODUTO5]\} \tag{C.5}$$

$$offspring2 = \{[PRODUTO5, PRODUTO6], [PRODUTO3, PRODUTO1], [PRODUTO2, PRODUTO4]\} \tag{C.6}$$

Anexos

ANEXO A – Tabela representando agrupamento obtido com Algoritmo Genético

Tabela 26 – Agrupamento obtido com o Algoritmo Genético

Cluster	Classe1	Classe2	Classe3	Classe4	Classe5	Classe6	Classe7
1	RELOGIOS	COFRES	AMPLIFICADORES	DIVERSOS PESSOAL	ACESSORIOS	ALMOFADAS	CANECAS
2	PORTA RETRATOS	MORINGAS E GARRAFAS	PAPELARIA	COSMETICOS	PORTA BIJOUX	CARTOES	DIVERSOS COZINHA
3	ELETRONICOS	ALBUNS	MOCHILAS	VESTUARIO	ORGANIZADORES	BOLSAS	TACAS
4	ELETRODOMESTICOS	DIVERSOS ESCRITORIO	COPOS TERMICOS	DIVERSOS CASA	FRONHAS	CANECOS E TULIPAS	PORTA PIJAMAS
5	BIJOUX	FOTOGRAFIA	JOGOS E BRINCADEIRAS	COPOS	COPOS COM CANUDO	BANDEJAS LAPTOP	LUMINARIAS