UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

JAILON WILLIAM BRUNO OLIVEIRA DA SILVA

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DO CORTE MÁXIMO EM GRAFOS

RUSSAS

2024

JAILON WILLIAM BRUNO OLIVEIRA DA SILVA

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DO CORTE MÁXIMO EM GRAFOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Orientador: Prof. Dr. Pablo Luiz Braga Soares.

RUSSAS

2024

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

S58a Silva, Jailon William Bruno Oliveira da.
Aplicação de aprendizado de máquina supervisionado para o Problema do Corte Máximo em grafos / Jailon William Bruno Oliveira da Silva. – 2025.
71 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Russas, Curso de Ciência da Computação, Russas, 2025.
Orientação: Prof. Dr. Pablo Luiz Braga Soares.

1. Problema do Corte Máximo. 2. Aprendizado de Máquina. 3. Aprendizado supervisionado. 4. Problema NP-difícil. I. Título.

CDD 005

JAILON WILLIAM BRUNO OLIVEIRA DA SILVA

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DO CORTE MÁXIMO EM GRAFOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação do Campus Russas da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Ciência da Computação.

Aprovada em: 27/09/2024

BANCA EXAMINADORA

Prof. Dr. Pablo Luiz Braga Soares (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Tatiane Fernandes Figueiredo
Universidade Federal do Ceará (UFC)

Prof. Dr. Eurinardo Rodrigues Costa
Universidade Federal do Ceará (UFC)

À minha família, por sua capacidade de acreditar em mim e investir em mim. Mãe, seu cuidado e dedicação foi que deram, em alguns momentos, a esperança para seguir. Pai, sua presença significou segurança e certeza de que não estou sozinho nessa caminhada.

AGRADECIMENTOS

Gostaria de expressar minha profunda gratidão a todos que tornaram esta jornada de graduação possível.

Primeiramente, à minha mãe, Cristiane de Oliveira, ao meu pai, Cilton José, e ao meu irmão, Meneses Neto, por seu apoio incondicional e por sempre acreditarem em mim ao longo desta caminhada. Também, à minha avó, Francisca Rosélia de Oliveira Moura, cuja presença continua a ser uma fonte constante de inspiração e orientação, mesmo em sua ausência física.

À minha tia Rosélia de Oliveira e ao meu tio Josafá Inácio, por seus conselhos e apoio constante ao longo da minha vida. Suas orientações sempre foram uma fonte de força e confiança diante dos desafios.

Agradeço minha prima Letícia Yorrana, seu marido Pedro e à sua filha Lara Yorrana, pelos momentos especiais que sempre foram marcados por carinho e apoio incondicional. Aos momentos mais memoráveis da minha vida, sou igualmente grato à minha tia Lenice, ao meu padrinho Jorge, à minha madrinha Madalena e à minha prima Gabriele, cuja presença trouxe alegria e conforto em cada etapa importante da minha jornada.

Agradeço sinceramente ao Prof. Dr. Pablo Luiz pela orientação excepcional e pelo apoio contínuo ao longo desta gradução. Meus agradecimentos se estendem aos professores que integraram a banca examinadora, Profa. Dra. Tatiane Fernandes e Prof. Dr. Eurinardo Rodrigues, pelo tempo dedicado e pelas valiosas contribuições e sugestões enriquecedoras.

Finalmente, um reconhecimento especial vai para Francisco Arnaldo, que dividiu o teto comigo durante a faculdade, oferecendo sua amizade e apoio durante este período desafiador que compartilhamos. um agradecimento caloroso aos meus amigos e colegas da Universidade Federal do Ceará, especialmente Davi Monteiro, Carlos Vinicio, José Fábio e Francisco Keven. Vocês foram parceiros indispensáveis, compartilhando momentos e desafios que tornaram esta jornada ainda mais memorável.

"... Mas lembre-se de que esse seu medo significa que tem algo pelo que lutar, algo com que se importa tanto a ponto de não ser capaz de imaginar a vida sem isso."
(MAAS, S. J; Trono de Vidro: Reino de Cinzas, 2019, p. 647)

RESUMO

Este estudo é voltado para a criação de modelos supervisionados, juntamente com o ajuste de seus hiperparâmetros para abordar o Problema do Corte Máximo em Grafos. Devido a complexidade desse problema ser NP-difícil, mesmo para instâncias de tamanho intermediário, achar uma solução ótima é um desafio computacional. Entretanto, existem diversas aplicações desse problema em áreas como análise de redes sociais, agrupamento de dados, segmentação de imagens e design de Chips VLSI, o que continua a motivar pesquisadores. A literatura apresenta métodos inovadores, incluindo aqueles baseados em redes de ponteiros com aprendizado supervisionado e por reforço, bem como abordagens que utilizam Redes Neurais em Grafos. Esses métodos representam avanços promissores na resolução desse problema, fazendo uso de técnicas de Aprendizado de Máquina.

Palavras-chave: problema do corte máximo; aprendizado de máquina; aprendizado supervisionado; problema NP-difícil.

ABSTRACT

This study focuses on the creation of supervised models, coupled with the tuning of their hyperparameters, to address the Max-Cut Problem. Due to the NP-hard complexity of this problem, even for instances of intermediate size, finding an optimal solution poses a computational challenge. However, there are several applications of this problem in areas such as social network analysis, data clustering, image segmentation, and VLSI chip design, which continue to motivate researchers. The literature presents innovative methods, including those based on pointer networks with supervised and reinforcement learning, as well as approaches utilizing Neural Networks in Graphs. These methods represent promising advancements in solving this problem, thus providing potential solutions for its resolution.

Palavras-chave: max-cut problem; machine learning; supervised learning; NP-hard problem.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizado de Máquina
FN	Falsos Negativos
FP	Falsos Positivos
GNN	Redes Neurais em Grafos
kNN	k-Nearest-Neighbours
LSTM	Long Short Term Memory
PCA	Análise de Componentes Principais
RNN	Rede Neural Recorrente
SAT	Satisfatibilidade
VN	Verdadeiros Negativos
VP	Verdadeiros Positivos

SUMÁRIO

1 INTRODUÇÃO

Na área da Teoria dos Grafos, encontramos diversos problemas classificados como NP-completo ou NP-difícil, como o Problema do Isomorfismo de Subgrafos, o Problema do Caixeiro Viajante e o Problema de Cobertura dos Vértices (Garey; Johnson, 1979).

Entre esses problema, existe o Problema do Corte Máximo, sendo um problema de otimização combinatória, podendo ser definido como segue: em um grafo $G = (V, E)$, tal que V representa o conjunto de vértices e E o conjunto de arestas, sendo definido um subconjunto S contido em V, tem-se o conjunto de arestas que têm exatamente um vértice em S, denominado corte. Quando o grafo é ponderado, ou seja, contém pesos associados às suas arestas, o peso de um corte é determinado pela soma dos pesos das arestas contidas nesse corte. Assim, o Problema do Corte Máximo busca encontrar o subconjunto S de vértices que resulta no corte de maior peso em um grafo ponderado (Boros; Hammer, 1991).

Dada a relevância prática dos Problemas de Otimização Combinatória, surgiram diversos algoritmos para abordá-los, podendo ser categorizados como exatos ou aproximados. Além disso, é importante ressaltar que o espaço de soluções para problemas NP-completo ou NP-difícil influencia no tempo computacional para resolvê-los com os algoritmos conhecidos, visto que ele cresce de forma exponencial conforme o tamanho da entrada aumenta. Isso torna os métodos exatos inviáveis, visto que eles tendem a explorar boa parte ou todo o espaço de soluções. Por sua vez, os métodos aproximados tem como objetivo encontrar soluções que se aproximem o máximo possível da solução ótima, mas para isso, eles sacrificam a garantia de encontrar soluções ótimas (Blum; Roli, 2003).

O Aprendizado de Máquina (AM) é uma das áreas que vem crescendo em um ritmo notável. Nela existem diferentes algoritmos com várias formas de aplicações, além da contínua adaptação dos algoritmos para diversos propósitos (Faceli *et al.*, 2011). Entre as aplicações para essas técnicas, encontra-se: suporte na análise do sequenciamento genômico (Libbrecht; Noble, 2015), detecção de doenças cardiovasculares (Oliveira *et al.*, 2023) e predição de risco de evasão de alunos (Teodoro; Kappel, 2020). Além disso, o Aprendizado de Máquina (AM) também é utilizado para abordar problemas em grafos, como coloração de grafos (Goudet *et al.*, 2022), Problema do Caixeiro Viajante (Miki *et al.*, 2018) e Problema da Cobertura mínima de Vértices (Gianinazzi *et al.*, 2021).

Este trabalho concentra-se em investigar as capacidades do Aprendizado de Máquina no contexto do Problema de Otimização Combinatória do Corte Máximo. Por ser um problema

NP-difícil (Karp, 1972), é um desafio computacional lidar mesmo com instâncias de tamanho intermediário. Entretanto, ele possui uma amplas aplicações em domínios como a análise de redes sociais (Agrawal *et al.*, 2003), agrupamento de dados (Otterbach *et al.*, 2017), segmentação de imagens (Sousa *et al.*, 2013) e em projeto de Chips VLSI (*Very Large Scale Integrated*) (Liers *et al.*, 2011). O que incentiva pesquisadores de diversas áreas a investigar mais sobre o problema.

Na literatura, encontramos diferentes abordagens para o Problema do Corte Máximo, utilizando técnicas variadas de Aprendizado de Máquina. Um estudo conduzido por Gu e Yang (2018) propôs um algoritmo baseado em redes de ponteiros. Este modelo é treinado por meio de aprendizado supervisionado e se destaca pela sua capacidade de lidar com dados de forma sequencial, incorporando um mecanismo de atenção para identificar informações mais importantes.

O trabalho subsequente, proposto por Gu e Yang (2020) dois anos depois, também faz uso de redes de ponteiros, porém, adota uma abordagem que utiliza aprendizado supervisionado e por reforço no treinamento do modelo. Este estudo inclui uma comparação entre os dois métodos, demonstrando o desempenho de cada um. Por fim, o Yao *et al.* (2019) apresenta um método que utiliza Redes Neurais em Grafos. De maneira geral, essa arquitetura de rede neural é projetada para lidar com dados estruturados na forma de grafos, o que demonstrou bons resultados.

Com base nisso, esse trabalho se destaca em relação aos anteriores ao adotar uma abordagem central que envolve a utilização de instâncias de pequena escala do Problema do Corte Máximo no treinamento dos modelos de AM. Essa estratégia permite a resolução do problema em um tempo razoável, viabilizando o treinamento de um modelo supervisionado. O objetivo final é aplicar esse modelo posteriormente em instâncias de grande porte, proporcionando uma solução eficiente e escalável.

O restante deste trabalho está estruturado da seguinte forma: No Capítulo 2, são definidos os objetivos gerais e específicos a serem alcançados. No Capítulo 3, são abordados os conceitos e informações teóricas essenciais para compreender a pesquisa. O Capítulo 4 apresenta trabalhos relacionados que se assemelham à proposta deste trabalho, que visa aplicar técnicas de Aprendizado de Máquina no contexto do Problema do Corte Máximo. O Capítulo 5 descreve a metodologia empregada para atingir os objetivos apresentados no Capítulo 2. Por fím, o Capítulo 6 mostra os resultados obtido, enquanto o Capítulo 7 trás as conclusões dessa pesquisa e trabalhos futuros.

2 OBJETIVOS

Neste capítulo, será discutido o objetivo principal a ser alcançado com o desenvolvimento deste trabalho, juntamente com as objetivos específicos essenciais para obter os resultados desejados.

2.1 Objetivos gerais

Criação de modelos de aprendizado de máquina supervisionado para o Problema de Otimização Combinatória de Corte Máximo. A ideia principal é utilizar instâncias de tamanho pequeno do Problema do Corte máximo para ajustar o modelo supervisionado, visando a aplicação em instâncias de grande porte.

2.2 Objetivos específicos

- Adquirir e preparar conjuntos de dados pertinentes ao Problema do Corte Máximo;
- Realizar o pré-processamento nos dados das instâncias adquiridos para criar a base de dados de treinamento, descobrindo os atributos mais relevantes;
- Escolher algoritmos de Aprendizado de Máquina Supervisionado;
- Treinar e validar os modelos supervisionados usando a base de dados dividida em treinamento e teste, respectivamente;
- Avaliar o desempenho em termos de precisão na solução para o Problema do Corte Máximo;
- Aplicar o aprendizado adquirido em instâncias com tamanhos grande;
- Analisar e interpretar os resultados, identificando informações relevantes para a aplicação de algoritmos de aprendizado de máquina supervisionado para o Problema do Corte Máximo;

3 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os fundamentos para a compreensão do problema deste trabalho. Na Seção 3.1, são apresentados os conceitos introdutórios da Teoria dos Grafos. A Seção 3.2 descreve as classes de problemas, incluindo definições de Problemas de Decisão e Otimização, para proporcionar uma compreensão mais aprofundada. A Seção 3.3 define o Problema do Corte Máximo e explora suas aplicações na Subseção 3.3.1.

Em seguida, a Seção 3.4 introduz a teoria do Aprendizado de Máquina, incluindo a hierarquia de aprendizado e principais diferenças entre o Aprendizado Supervisionado e Não-Supervisionado. A Subseção 3.4.1 detalha o processo de treinamento no Aprendizado de Máquina Supervisionado, seguida da apresentação de algumas métricas de avaliação para modelos de classificação supervisionada na Subseção 3.4.1.1. As Subseções 3.4.1.2, 3.4.1.3, 3.4.2 3.4.1.4 abordam as técnicas de aprendizado de máquina que serão utilizados.

Por fim, a Subseção3.4.3 aborda técnicas de aprendizado não supervisionado. Mais especificamente, a Subseção 3.4.3.1 aprezenda a técnica de Análise de Principais Componentes, utilizado na parte de pre-processamento da base de dados dos modelos e para interpretação dos resultados, a Subseção 3.4.3.2 apresenta o Algoritmo *k-means*, que deu suporte na fase de interpretação dos resultado obtidos do modelo.

3.1 Conceitos introdutórios da Teoria dos Grafos

Um grafo simples é denotado como $G = (V, E)$, sendo composto por dois conjuntos distintos: V que representa os vértices e E que representa as arestas. Ambos os conjuntos possuem um número finito de elementos. Os elementos de E são subconjuntos de dois elementos de V, representando uma ligação entre dois vértices, ou seja, $E \subseteq \{\{u, v\} \mid u, v \in V\}$. De modo geral, dois vértices são considerados adjacentes (ou vizinhos) caso exista uma aresta entre eles (Trudeau, 1994).

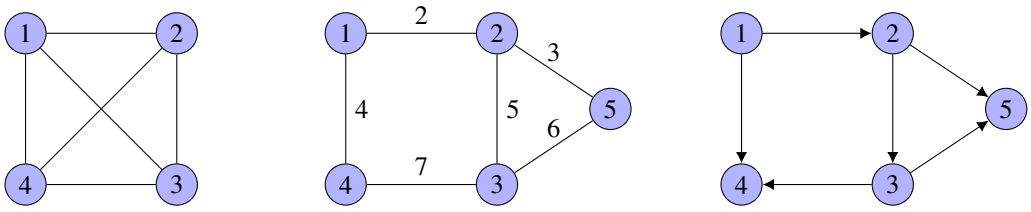
Em um grafo, cada aresta (u, v) pode ser dirigida (ou arco), sendo representado por um par ordenado, ou não dirigida, sendo representado por um par não ordenado denotado por conjuntos $\{u, v\}$ (Goodrich; Tamassia, 2009). No primeiro caso, para a aresta dirigida (u, v), temos que u (cauda) é dirigido para v (cabeça), indicando uma direção específica representada pelo par ordenado. No segundo caso, a ordem de representação da aresta não dirigida, seja $\{u, v\}$ ou $\{v, u\}$, é irrelevante, pois, em ambas, a ligação entre u e v é considerada (Gross jay yellen,

2013). Além disso, se um grafo possuir apenas arestas dirigidas, é chamado de digrafo. Por outro lado, se possuir apenas arestas não dirigidas, é chamado de grafo não direcionado (Goodrich; Tamassia, 2009).

Quando o grafo é não direcionado, o grau de um vértice v, que representa a quantidade de vértices adjacentes a ele, pode ser expressado como $d(v)$. Por sua vez, a expressão $d(G)$ refere-se à média dos graus de todos os vértices do grafo (Diestel, 2010). Além disso, existem outras características relacionadas às arestas, como direção e peso, que são associadas a grafos específicos.

A Figura 1 abaixo ilustra exemplos de dois grafos simples, sendo eles, não direcionado, não direcionado com arestas ponderadas e um digafo, respectivamente.

Figura 1 – Exemplos de grafos simples



(a) Grafo não direcionado

(b) Grafo não direcionado com arestas ponderadas

(c) Grafo Direcionado

Fonte: elaborada pelo autor.

O termo subgrafo é utilizado para referir-se a grafos contidos em outros. Em outras palavras, dado um grafo $G = (V, E)$ e um grafo $G' = (V', E')$, G' será um subgrafo de G se $V' \subseteq V$ e $E' \subseteq E$, podendo ser expressado como $G' \subseteq G$ (Diestel, 2010). A quantidade de vértices e arestas de um grafo pode ser expressa por $|V|$ e $|E|$, respectivamente. Desta forma, um grafo (ou subgrafo) simples é denso se o valor de $|E|$ estiver próximo de $|V|^2$, pois implica que quase todas as possíveis arestas já estão presentes. Caso contrário, se o grafo possuir poucas arestas, é classificado como esparso (Gross jay yellen, 2013). Note que a Figura 1 (a) é um exemplo de um grafo denso, pois ele já contém todas as possíveis arestas.

Além das características mencionadas, um grafo pode conter ciclos, que ocorrem quando há um caminho que retorna ao mesmo vértice de origem. Mais especificamente, um caminho é uma sequência de vértices e arestas, e um ciclo ocorre quando ele começa e termina no mesmo vértice. Além disso, quando um grafo é conexo, ou seja, há um caminho entre quaisquer

dois vértices nesse grafo, e não contém ciclos, é chamado de árvore (Goodrich; Tamassia, 2009).

O grau de um vértice pode ser utilizado para determinar sua importâncias em relação as coneções dentro do grafo, por meio de medidas de centralidade. A centralidade de grau de um vértice é determinada pela quantidade de arestas que os tocam diretamente, refletindo sua relevância em termos de conexões dentro do grafo. Já a centralidade de intermediação, está interessada na quantidade de caminhos mínimos que passam pelo vértice, ou seja, em intermediar conexões, medindo sua importância em relação ao fluxo de conxões dentro do grafo (Latora *et al.*, 2017).

Para avaliar o tamanho do grafo, isto é, o quão próximo estão seus vértices e arestas, é utilizado medidas como diâmetro, raio e excentricidade. Conforme Diestel (2017), o diâmetro de um grafo é a maior distância mínima entre quaisquer dois vértices, enquanto a excentricidade de um vértice é a maior distância mínima entre um vértice qualquer para todos os outros do grafo. Já o raio, é a menor excentricidade observada entre todos os vértices.

Por último, a fim de identificar formações de grupos locais, isto é, de vértices e arestas dentro do grafo, é utilizado coeficiente de agrupamento. Para tal, é utilizado a quantidade de triângulos, que é subgrafos completos de três vértices, que são formados em torno de um determinado vértice, em relação ao número total de triângulos que poderiam ser formados (Barabási, 2016).

3.2 Classes de Problemas

Um algoritmo é um conjunto de passos computacionais bem estruturados que transforma valores de entrada em valores de saída. Adicionalmente, pode ser visto como uma ferramenta para solucionar um problema computacional bem definido, onde o enunciado do problema define a relação desejada entre a entrada e saída, e o algoritmo que é a sequência de etapas computacionais, fornece o procedimento para alcança-la (Cormen *et al.*, 2012).

Os problemas podem ser classificados em vários tipos distintos, incluindo os Problemas de Decisão, que retornam uma resposta binária, ou seja, 0 ou 1, que indica sim ou não, e os Problemas de Otimização, que retornam a melhor solução dentro do espaço de possíveis soluções (Sipser, 2012).

Esses dois problemas possuem uma relação conveniente. Um problema de otimização pode ser abordado como um problema de decisão ao impor restrições para o valor a ser otimizado. Por exemplo, dado um inteiro k, um grafo conexo G com pesos inteiros nas arestas e um problema

de otimização qualquer, como o problema da árvore de cobertura mínima, que busca encontrar uma árvore contendo todos os vértices do grafo, com a menor soma dos pesos das arestas possível. Este grafo tem uma árvore de cobertura mínima com pesos menores que k? Como a resposta esperada é um valor binário, esse problema agora demonstra características de um problema de decisão (Goodrich; Tamassia, 2009).

A complexidade de um algoritmo pode ser classificada com base em duas classes fundamentais de problemas. A primeira sendo a classe P, que refere-se aos problemas de decisão que podem ser resolvidos por algum algoritmo no qual o número de passos é limitado por um polinômio fixo no comprimento da entrada. Já a segunda é a classe NP, que está relacionada com o tempo polinomial não determinístico, pois sua definição foi estabelecida em termos de máquina não determinística, ou seja, máquinas que tem mais de um movimento possível para uma determinada configuração (Cook, 2000).

Além disso, essa última classe é constituída por problemas de decisão que podem ser verificados em tempo polinomial (Arora sanjeev;barak, 2016). Desta forma, ao selecionar uma solução candidata e conduzir a verificação, o algoritmo deve ser capaz de fazê-lo em tempo polinomial, embora encontrar a solução em si não possa ser feita nesse tempo.

Já para os problemas de otimização, temos as classes PO e NPO, que referem-se, respectivamente, à extensão das classes P e NP. Desta forma, um problema NPO é um problema de otimização cujas versões de decisão estão em NP, enquanto um problema PO está em P, o que significa que é solucionável em tempo polinomial (Bazgan *et al.*, 2005).

Com base nisso, podemos definir duas outras classes. Dado um problema de decisão M, este será classificado como um problema NP-difícil se cada problema de decisão L em NP, for redutível a tempo polinomial a M. Em adição, se esse problema M for NP-difícil e também estiver dentro da classe NP, denotamos ele como NP-completo, sendo considerado um dos problemas mais difíceis dentro da classe NP (Goodrich; Tamassia, 2009). Adicionalmente, se um problema de otimização está na classe NP-difícil, não há um algoritmo de tempo polinomial capaz de encontrar a solução ótima, a menos que P seja igual a NP (Sipser, 2012).

Demonstrar que P $\subseteq$ NP é uma tarefa considerada trivial (Cook, 2000). No entanto, estabelecer a igualdade entre P e NP é uma tarefa mais difícil. Conforme Sipser (2012), esse é um dos maiores problemas não resolvidos na ciência da computação teórica e na matemática contemporânea. Se essas classes fossem iguais, qualquer problema polinomialmente verificável seria também decidível em tempo polinomial. Ademais, uma das estratégias de realizar esta

prova é demonstrar que algum problema NP-completo pode ser resolvido em tempo polinomial, pois implicaria diretamente que todo problema em NP tem uma solução em tempo polinomial, provando que P = NP (Arora sanjeev;barak, 2016).

Embora a classe NP-completo seja originalmente associada aos problemas de decisão, existem muitos desses problemas que são versões de decisão de problemas de otimização (Bazgan *et al.*, 2005). Como mencionado anteriormente, um problema de otimização pode ser transformado em um problema de decisão. Além disso, em contexto relevante à essa classe, se comprovar que um problema de decisão é difícil, então o problema de otimização correspondente também é difícil (Cormen *et al.*, 2012).

3.2.1 Problemas clássicos da literatura

Dentre os principais problemas da classe NP-completo, encontram-se o Problema da Satisfatibilidade (SAT), Cobertura de Vértices (*Vertex Cover Problem*) e o Problema do Caixeiro-Viajante (*The Traveling Salesman Problem*) (Goodrich; Tamassia, 2009). Além desses problemas, temos o Problema do Corte Máximo (*Max-Cut Problem*) abordado na Seção 3.3, classificado como NP-difícil (Karp, 1972).

Para exemplificar o Problema do Caixeiro-Viajante, considere o seguinte cenário: um vendedor deseja percorrer um conjunto de cidades uma única vez, buscando o caminho de menor custo e retornando no final à cidade de origem. Esse cenário caracteriza um problema de otimização. Em adição, é possível abordar esse problema utilizando a Teoria dos Grafos. Nesse contexto, um grafo simples ponderado $G = (V, E)$ é definido, seus vértices representam as cidades e as arestas indicam a existência de rotas entre elas. Por último, o peso associada a cada aresta corresponde ao custo da rota (Cormen *et al.*, 2012).

O Problema da Cobertura de Vértices busca determinar se um grafo simples $G = (V, E)$ possui uma cobertura de vértices de tamanho k, onde $k \in \mathbb{N}$ (Sipser, 2012). Em outras palavras, o objetivo é verificar se é possível escolher um conjunto de vértices V' de tamanho k de forma que, para toda aresta $\{u, v\} \in E$, os vértices u ou v pertençam ao conjunto $V' \subseteq V$.

O Problema SAT tem como objetivo determinar se uma fórmula booleana é satisfatível, ou seja, se existe uma atribuição de valores verdadeiros e falsos às variáveis de uma fórmula booleana, tal que esta fórmula seja avaliada como satisfatível. Para ser considerada no contexto do problema, as fórmulas devem estar na forma normal conjuntiva, isto é, a fórmula deve ser expressada como uma conjunção de cláusulas, onde cada cláusula é compostas por disjunções de

literais (Cook; Mitchell, 1996). Além disso, segundo Cook e Mitchell (1996), esse problema é o primeiro e um dos mais simples dentre os muitos que foram demonstrados ser da classe NP-completo.

3.3 O problema do Corte Máximo

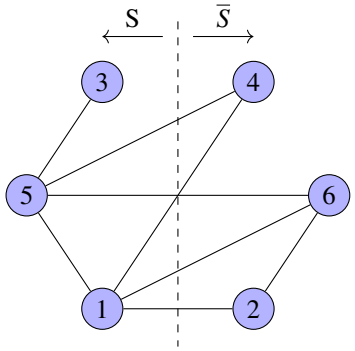
O Problema do Corte Máximo não é apenas de otimização, ele está inserido em uma área mais específica, sendo considerado um Problema de Otimização Combinatória. Conforme Miyazawa e Souza (2015), essa área engloba uma ampla gama de problemas voltados para a busca de soluções que otimizem a utilização dos recursos disponíveis. Em outras palavras, está além de encontrar uma solução viável, envolve a eficiente utilização dos recursos disponíveis, a otimização do tempo para a realização de ações e operações, a maximização de lucros e minimização de prejuízos. Ademais, outros problemas dessa natureza podem ser consultados nos trabalhos de Garey e Johnson (1979) e Gross Jay Yellen (2013).

Dado um grafo não direcionado $G = (V,\ E)$. Definimos como corte o ato de separar um conjunto de vértices em dois subconjuntos disjuntos, denotados S e $\overline{S}$, tal que $S \cup \overline{S} = V$ e $S \cap S = \emptyset$. O tamanho do corte é determinado pelo número de arestas que cruzam as partições, ou seja, que partem de um subconjunto e chegam no outro. Em outras palavras, as arestas de corte são as arestas uv, tal que $uv \in E$ e $u \in S$ enquanto $v \in \overline{S}$, ou alternativamente, $u \in \overline{S}$ e $v \in S$. Por outro lado, as arestas que conectam vértices pertencentes a um mesmo subconjunto, seja S ou $\overline{S}$, são conhecidas como arestas não cortadas (Sipser, 2012).

A Figura 2 ilustra um exemplo de corte em um grafo. Note que o conjunto dos vértices $S = \{1,3,5\}$ são todos os vértices a esquerda do corte (ilustrado por uma reta pontilhada) no grafo, enquanto o conjunto $\overline{S} = \{2,4,6\}$ são todos os vértices a direta. Já as arestas cortas são o conjunto $\{\{4,5\},\{5,6\},\{1,2\},\{1,4\},\{1,6\}\}$, enquanto o conjunto $\{\{3,5\},\{5,1\},\{2,6\}\}$ indica quais as arestas não cortadas.

O problema do Corte Máximo pode ser formulado usando um grafo não direcionado $G = (V,E)$, onde $|V| = n$ representa o número de vértices e $|E| = m$ indica o número de arestas ponderadas. A função $c_{ij} = c_{ji}$ retorna o peso de cada aresta $\{i,j\} \in E$, com $c_{ij} = 0$ para $\{i,j\} \notin E$ e $c_{ii} = 0$ para todo $1 \leq i \leq n$. Qualquer partição $(S, \overline{S} := V \backslash S)$ dos vértices de V define um corte em G, onde S ou $\overline{S}$ podem ser vazios. Assim, o problema consiste em encontrar uma combinação de $(S, \overline{S})$ de forma que a soma dos pesos das arestas de corte seja maximizada (Soares, 2018). Em adição, esse problema pode ter uma versão para grafos não ponderados,

Figura 2 – Exemplo de corte em um grafo

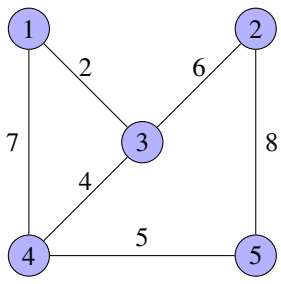


Fonte: Elaborada pelo autor.

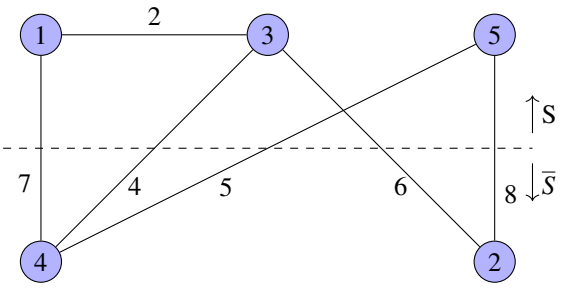
conhecida como *Simple Max-Cut Problem*. Para tal, é necessário restringir o peso de cada aresta do grafo para 1 (Garey; Johnson, 1979).

Um exemplo ilustrativo de uma instância do Problema do Corte Máximo é apresentado na Figura 3 abaixo. Na Figura 3 (a), é apresentado um grafo $G = (V,E)$, com $|V| = 5$ vértices e $|E| = 6$ arestas ponderadas. Já na Figura 3 (b), é apresentado a solução ótima para essa instância. Note que os vértices acima do corte são da partição $S = \{1,3,5\}$ e os que estão abaixo são da partição $\overline{S} = \{4,2\}$. As arestas de corte correspondem ao conjunto $\{\{1,4\},\{3,4\},\{4,5\},\{2,3\},\{2,5\}\}$, totalizando um valor de corte de 30. Por fim, a única aresta não cortada é $\{1,3\}$.

Figura 3 – Exemplo de uma instância do Problema do Corte Máximo



(a) Grafo não direcionado com arestas ponderadas



(b) Solução ótima do Corte Máximo

Fonte: Elaborada pelo autor.

3.3.1 Aplicações do problema de Corte Máximo

O *Data Clustering* é um problema importante na área de Aprendizado de Máquina não Supervisionado. Ele consiste na classificação não supervisionada de padrões, como observações ou vetores de características, em grupos (*clusters*). Esse desafio tem sido estudado por pesquisadores de diversas áreas, mostrando sua relevância na análise exploratória de dados. Além disso, ele possui aplicações práticas em segmentação de imagens, reconhecimento de objetos e recuperação de informação (Jain *et al.*, 1999).

De modo geral, esse problema envolve atribuir rótulos aos elementos de um conjunto de dados para agrupar os mais semelhantes entre si. Para representar a dissimilaridade entre eles, é definido uma medida de distância que é aplicada a cada par de elementos, de modo que dados mais distantes irão possuir rótulos diferente. Matematicamente, esse problema pode ser reduzido ao Problema do Corte Máximo. Para tal, cada vértice do grafo pode ser considerado um dos elementos desse conjunto de dados e o peso entre os vértices é a medida de distância. Assim, maximizar a soma de todos os pesos de vértices com rótulos diferentes representa um algoritmo de agrupação natural (Otterbach *et al.*, 2017).

Uma outra aplicação relevante do corte máximo é observado nas redes sociais, mais especificamente na análise do comportamento social das pessoas em relação a um determinado tópico em grupos de discussão. Nesses grupos, é comum que as pessoas tendem a responder com mais frequência a mensagens com as quais não concordam, especialmente em grupos de notícias. A partir desse comportamento, é possível construir um grafo no qual os vértices representam os autores das mensagens e as arestas indicam relações de resposta entre as mensagens. Portanto, ao determinar o corte máximo desse grafo, teremos dois grupos distintos: aqueles que apoiam e aqueles que se opõem a um determinado tópico (Agrawal *et al.*, 2003).

3.4 Aprendizado de Máquina

O AM é a área de estudo que possibilita que as ferramentas, sejam capazes de operar de forma a reduzir a necessidade de intervenção humana, agindo de maneira mais autônoma. Desta forma, os computadores têm a capacidade de aprender sem serem explicitamente programados. Em outras palavras, é a ciência que ensina computadores a aprender a partir de um conjuntos de dados (Géron, 2019).

Esse processo de aquisição de conhecimento é realizado através do aprendizado

indutivo, que generaliza padrões a partir de exemplos externamente adquiridos ao processo, partindo de conceitos específicos e os generalizando. Nesse método, um conceito é aprendido através de inferência indutiva aplicadas sobre os exemplos. Além disso, o aprendizado indutivo pode ser dividido em Aprendizado Supervisionado e Não-Supervisionado, sendo que o algoritmo de aprendizado também é conhecido como indutor (Monard; Baranauskas, 2003).

Dito isso, o AM está associado diretamente a criação e ao uso de modelos que são ensinados a partir de um conjunto de dados. Mais especificamente, um algoritmo de AM analisa esses dados buscando identificar padrões para especificar uma relação matemática (ou probabilística) existente entre variáveis distintas, sendo esta relação conhecida como modelo. Nos modelos de aprendizado de máquina supervisionado, os dados de treinamento vêm acompanhados de um rótulo, indicando a resposta correta para a aprendizagem, e nos modelos não supervisionados, não existem tais respostas (Grus, 2016).

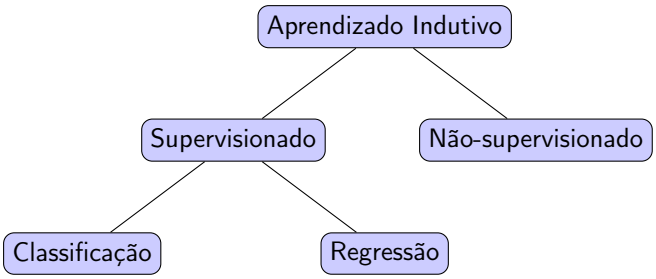
Os algoritmos de AM são organizados com base em duas tarefas principais. A primeira são as Tarefas Preditivas, onde a meta é encontrar uma função (também chamado de modelo) que atribui um rótulo ou classe a um conjunto de dados não visto anteriormente (Classificação) ou estima um valor numérico (Regressão), com base em um conjunto de dados previamente observado, juntamente com as respostas para o aprendizado. Já as tarefas de descrição não utilizam essas respostas, elas concentram-se na exploração ou na caracterização de um conjunto de dados. Assim, as tarefas Preditivas são associadas ao Aprendizado Supervisionado, enquanto as tarefas descritivas são associadas ao aprendizado Não-supervisionado (Faceli *et al.*, 2011).

A Hierarquia do Aprendizado, abordada nesta Seção, é ilustrada na Figura 4, englobando o Aprendizado Indutivo e seus dois ramos principais: Aprendizado Não-supervisionado e Aprendizado Supervisionado, o qual se divide em Classificação, que estima valores discretos, e Regressão, que estima valores contínuos.

3.4.1 Aprendizado de Máquina Supervisionado

Um conjunto de dados inclui objetos que representam diversos conceitos, tangíveis ou abstratos, como uma cadeira ou a avaliação de um atendimento ao cliente. Esses objetos são comumente referidos como registros ou exemplos e são representados por meio de um vetor de características, também conhecido como atributo. Cada atributo está associado a uma propriedade do registro. Para o treinamento do modelo supervisionado, cada registro é uma

Figura 4 – Hierarquia do Aprendizado



Fonte: Elaborada pelo autor.

instância do problema contendo atributos de entrada e saída, onde o atributo de saída pode ser estimados através dos atributos de entrada (Faceli *et al.*, 2011).

Desta forma, para esse treinamento é utilizado um conjunto de dados representados por registros $R = \{R_1, R_2, \ldots, R_n\}$, onde cada registro $R_i \in R$, é associado a um rótulo que representa a resposta esperada. Em termos mais específicos, cada registro é representado como uma tupla $R_i = (\vec{x}_i, y_i)$, onde $x_i \in \{\vec{x}_{i1}, \vec{x}_{i2}, \ldots, \vec{x}_{in}\}$, que é um vetor de valores que representam as características ou atributos do registro R_i, e y_i é a resposta esperada para essa instância. Assim, o objetivo do modelo é, a partir desses registros, aprender uma função $f(\vec{x}) = y$ que mapeia os valores de $\vec{x}$ para os valores de y, permitindo a predição de respostas para exemplos não vistos (Batista, 2003).

A Tabela 1 abaixo, exemplifica o conjunto de dados para realizar o treinamento. Ela segue o formato atributo-valor, composta por n registros e m atributos. A coluna Y representa a função $f(\vec{x}_i) = y_i$, que tenta realizar a predição com base nos atributos, onde cada valor nessa coluna pertence ao conjuntos de valores do atributo de saída.

Tabela 1 – Representação dos dados de treinamento

	X_1	X_2	$\cdots$	X_m	Y
R_1	x_{11}	x_{12}	$\cdots$	x_{1m}	y_1
R_2	x_{21}	x_{22}	$\cdots$	x_{2m}	y_2
$\vdots$	$\cdots$	$\cdots$	$\ddots$	$\cdots$	$\vdots$
R_n	x_{n1}	x_{n2}	$\cdots$	x_{nm}	y_n

Fonte: Adaptado de Batista (2003, p. 6)

Além disso, as linhas $(R_1, R_2, ..., R_n)$ representam diferentes tipos de registros, en-

quanto as colunas $(X_1, X_2, ..., X_m)$ indicam os atributos de entrada. A linha i corresponde ao i-ésimo registro $(i = 1, 2, ..., n)$ e a entrada x_{ij} representa o valor do j-ésimo atributo de X_j $(j = 1, 2, ..., m)$ do registro i. A tabela também ilustra como os registros são representados como tuplas $R_i = (x_{i1}, x_{i2}, ..., x_{im}, y_i) = (\vec{x}_i, y_i)$.

3.4.1.1 Métricas de avaliação para Modelos de Classificação

Com o intuito de realizar a validação e avaliação dos modelos de classificação, o conjunto de dados são separados em dois conjuntos disjuntos. O primeiro sendo o conjunto de treinamento, que contém o conjunto de registro juntamente com as respostas esperada para cada registro, utilizado para ensinar o modelo. O segundo é o conjunto de teste, utilizado para avaliar o desempenho do modelo (Nelli, 2018). Além disso, é importante ressaltar que esses conjuntos não devem compartilhar registros, pois seria similar a um estudante fazendo uma prova após ter estudado com as respostas, comprometendo a avaliação.

Conforme discutido anteriormente, o modelo de classificação atribui classes (ou rótulos) para novos registros. Desta forma, a avaliação do modelo treinado é realizada pela comparação das respostas previamente obtidas de forma externa, com as respostas fornecidas pelo modelo treinado.

Uma métrica amplamente utilizada para avaliar modelos de classificação é a Matriz de Confusão. Nela, cada linha representa uma classe real, enquanto cada coluna reflete uma classe prevista pelo modelo. A diagonal principal destaca os acertos, incluindo quando uma classe é corretamente identificada (verdadeiros positivos) e quando um registro não é atribuído a uma determinada classe corretamente (verdadeiros negativos). As demais células da matriz indicam erros, ou seja, quando uma classe é atribuída erroneamente a um registro (falso positivo) e quando um registro é classificado incorretamente (falso negativo) (Géron, 2019).

Conforme destacado por Raschka (2015), ao ter conhecimento dos Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN), diversas métricas de avaliação de modelos podem ser empregadas. Essas métricas incluem a *Accuracy*, a *Precision*, o *Recall* e a *F1-Score*, que são calculadas da seguinte maneira:

$$\text{Accuracy} = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.1}$$

A *Accuracy* avalia o desempenho do modelo através de uma média geral, calculando a proporção de uma classificação correta (VP e VN) em comparação com o total de classificação

(VP, VN, FP e FN).

$$\text{Precision} = \frac{VP}{VP+FP} \tag{3.2}$$

A *Precision* avalia o desempenho do modelo em classificar uma determinada classe. Para isso ele calcula a proporção dessa classe, quando foi classificado corretamente (VP), em relação ao total de classificação positivas (VP e FP).

$$\text{Recall} = \frac{VP}{VP+FN} \tag{3.3}$$

O *Recall* é conhecido por medir a sensibilidade do modelo, pois ele analisa a proporção das classes que foram classificadas corretamente (VP), em relação ao total de classificações que realmente pertencem a classe (VP e FN).

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.4}$$

O *F1-Scores* combina ambas as métricas *Precision* e *Recall*, sendo útil para avaliar classes desbalanceadas.

3.4.1.2 Aprendizagem baseada em instâncias: k-Nearest-Neighbours

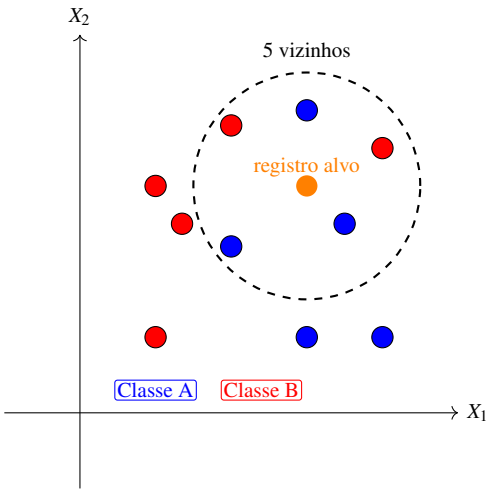
O k-Nearest-Neighbours (kNN) é um algoritmo de Aprendizado de Máquina Supervisionado utilizado em tarefas Preditivas, tendo aplicações em Reconhecimento de Padrões e Mineração de Dados (Xiong; Yao, 2021). Além disso, ele é um algoritmo de aprendizado preguiçoso (*lazy learner*), pois ao contrário de modelos tradicionais (conhecidos como paramétricos), ele não constrói explicitamente uma função que mapeia os atributos de entrada em um atributo de saída. Ele pertence a uma subcategoria de modelos não-paramétricos denotados de aprendizagem baseada em instâncias, que é conhecido por memorizar o conjunto de dados fornecido no treinamento do modelo (Raschka, 2015).

O kNN utiliza os registros da base de treinamento para avaliar novos registros, com o auxílio de uma função para medir a distância entre os registros de treinamento e o registro alvo. De modo geral, ele analisa os k vizinhos mais próximos que compõem a vizinhança desse registro a ser avaliado e determina sua classe através de uma votação majoritária entre os indivíduos dessa vizinhança, isto é, assumindo a classe do indivíduo que mais ocorre na mesma (Aggarwal, 2014).

A Figura 5 ilustra a classificação de um novo registro usando o método kNN. Note que k foi configurado como 5 e os cinco registros mais próximos consistem em três registros

classificados como classe A (azuis) e dois como classe B (vermelhos). Após a votação majoritária, o registro alvo é categorizado como classe A (azul).

Figura 5 – Exemplo de classificação do KNN com $k = 5$



Fonte: Elaborada pelo autor.

Dessa maneira, o desempenho do kNN está associado não apenas ao valor de k escolhido, mas também às funções utilizadas para calcular as medidas de distância e determinar os k registros mais próximos do registro alvo. De acordo com Williams e Li (2008), as medidas de distância, *Euclidean*, *Manhattan* e *Mahalanobis* podem ser utilizadas no kNN, sendo calculadas, entre um ponto $p = (p_1, ..., p_n)$ e outro ponto $q = (q_1, ..., q_n)$, da seguinte forma:

A distância *euclidiana* é calcula seguindo em uma linha reta.

$$EUD(p,\ q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.5}$$

A distância *Manhattan* é calculada seguindo um percurso em uma grade, como se percorre-se o quarteirão de uma cidade.

$$MN\ (p,\ q) = \sum_{i=1}^{n} |\ p_i - q_i\ | \tag{3.6}$$

A distância *Mahalanobis* é calculada com base nas correlações entre as variáveis. Neste caso, V representa uma matriz de covariância com atributos $A_1..Am$, sendo A_j o vetor de atributo j que está na base de treinamento.

$$MD(p,\ q) = \sqrt{(p_i - q_i)^T\ V^{-1}\ (p_i - q_i)} \tag{3.7}$$

3.4.1.3 Aprendizado Bayesiano: Naive Bayes Classifier

O classificador Naive Bayes, baseado no teorema de Bayes, é uma abordagem probabilística simples que utiliza probabilidades condicionais para determinar a probabilidade de um novo registro pertencer a uma determinada classe. Adicionalmente, ele é considerado ingênuo porque assume que cada atributo da base de dados são independente entre si, o que implica que a probabilidade para cada atributo também é independente (Parsian, 2015).

Conforme Rish *et al.* (2001), o Classificador Naive Bayes atribui a um novo registro (denotado por um vetor de características) a classe que tem a maior probabilidade de ser. O processo de treinamento se baseia na suposição de que os atributos são independentes entre si, contanto que a classe esteja previamente associada ao registro. Dessa forma, ao analisar todos os vetores de características pertencentes a uma classe C, é possível estimar a probabilidade de um registro pertencer a essa classe. De maneira mais específica, essa probabilidade é denotada pelo produto das probabilidades individuais dos atributos, sendo calculada da seguinte maneira:

$$P(X|C) = \prod_{i=1}^{n} P(X_i|C) \tag{3.8}$$

Observe que $X = \{X_1, ..., X_n\}$ representa o vetor de características e C representa uma classe. Além disso, a fórmula utiliza probabilidades condicionais, que são calculadas a partir de dois eventos. Segundo Sinai e Sinai (1992), a probabilidade condicional define a probabilidade de ocorrer um evento A dado que um evento B já ocorreu, pode ser denotada da seguinte forma:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \tag{3.9}$$

Além disso, conforme Faceli *et al.* (2011), essa probabilidade condicional pode ser derivada. Para isso, é utilizado a probabilidade a *priori* da classe, representada por $P(A)$, a probabilidade de observar vários elementos pertencentes à classe, denotado por $P(B|A)$, e a probabilidade de ocorrência desses elementos, representado por $P(B)$, resultando na seguinte equação:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.10}$$

3.4.1.4 Regressão Logística

A Regressão Logística, apesar do nome sugerir que é um modelo de regressão, trata-se de um modelo linear de classificação binária. Mais especificamente, é um modelo

probabilístico que estabelece uma relação linear entre os atributos de entrada e o atributo de saída, utilizando a função sigmoide para transformar a combinação linear das características em uma probabilidade de um evento ocorrer ou não (Raschka, 2015).

Segundo Aggarwal (2014), a regressão logística estima diretamente a distribuição de probabildiade $P(Y|X)$, onde Y é uma variável binária, como 0 ou 1. Formalmente, o modelo é definido como sendo a probabilidade de um evento ocorrer, isto é, $p(Y = 1|X)$, sendo expressado pela função sigmoide, assim como segue:

$$p(Y = 1|X) = g(\theta^T X) = \frac{1}{1 + e^{-\theta^T X}}, \tag{3.11}$$

onde a função sigmoide $g(z)$ é definida como:

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.12}$$

Aggarwal (2014) especifica que $\theta^T X$ representa o produto escalar entre o vetor de pesos θ e o vetor de características X de um registro qualquer, sendo expresso como:

$$\theta^T X = \theta_0 + \sum_{i=1}^{d} \theta_i X_i \tag{3.13}$$

Em geral, $p(Y = 1|X)$, ao ser combinado com a função logit, retorna a probabilidade em logaritmos das chances, permitindo que a relação seja representada linearmente (Bewick *et al.*, 2005). Conforme Raschka (2015), a função logit expressa essa relação linear entre os atributos e o logaritmo das chances de um evento ocorrer. A equação da função logit é definida como segue:

$$\text{logit}\,(p(y = 1|\mathbf{x})) = \sum_{i=0}^{n} w_i x_i = \mathbf{w}^T \mathbf{x} \tag{3.14}$$

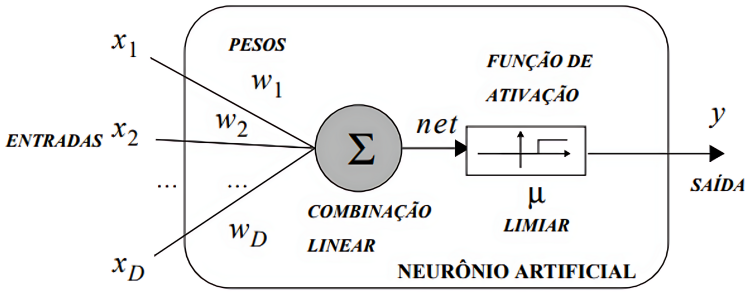
Dessa forma, $p(y = 1|x)$ é a probabilidade condicional de que um registro pertença à classe 1, dado o vetor de características x, onde $X = \{x_1, x_2, \ldots, x_n$. Os pesos $w_0, w_1, \ldots, w_n$ são ajustados durante o treinamento do modelo para maximizar a verossimilhança, que determina a influência de cada atributo no resultado final (Bewick *et al.*, 2005; Raschka, 2015).

3.4.2 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são modelos de AM inspirados no funcionamento do cérebro humano, o qual contém neurônios interconectados por meio de sinapses que realizam a troca de informações. O aprendizado ocorre através das alteração de força dessas conexões sinápticas em resposta a estímulos, ou seja, pela atualização do peso das conexões entre os neurônios. Adicionalmente, o neurônio é a unidade básica de processamento desse modelo, sendo organizado em diferentes quantidades e conexões, o que gera diversos tipos de arquiteturas de redes neurais (Aggarwal, 2014).

A Figura 6 ilustra o modelo simplificado de um neurônio artificial, baseado no trabalho de McCulloch e Pitts (1943). Conforme Rauber (2005), esse modelo busca simular os processos biológicos em uma célula nervosa. Para transmitir as informações, ele faz uso de entradas x_j (sinapses), onde cada uma possui um peso w_j, que representa sua importância. O neurônio processa essas entradas através de uma combinação linear, gerando um valor *net* que, com o uso de uma função Heaveside (função de escada), é comparado a um limiar μ. Desta forma, caso esse valor ultrapasse μ, o neurônio emite um sinal de saída $y = 1$, caso contrário, a saída $y = 0$. Adicionalmente, é possível utilizar outras funções de ativação, como a função sigmoidal e linear.

Figura 6 – Modelo de um neurônio de McCulloch e Pitts



Fonte: Adaptado de Rauber (2005, p. 6).

Note que, a combinação linear responsável por gerar o valor *net* pode ser expressada por um somatório com D elementos, assim como segue:

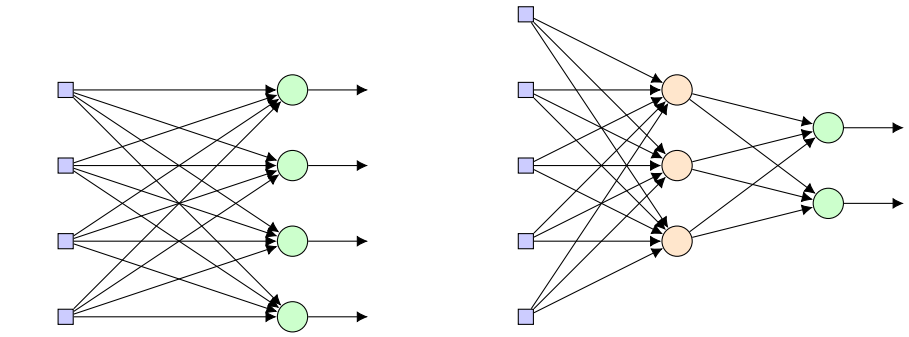
$$net = \sum_{i=1}^{D} x_i w_i = x_1 w_1 + x_2 w_2 + \ldots + x_D w_D \tag{3.15}$$

Consequentemente, uma Rede Neural Artificial consiste em neurônios artificiais organizados em camadas, seguindo um modelo semelhante ao neurônio proposto por McCulloch

e Pitts (1943) mencionado anteriormente. No entanto, é importante ressaltar que a maneira como essa rede neural aprende, as conexões utilizadas e o número de neurônios podem variar de acordo com a arquitetura específica da rede.

Em adição, existem duas arquiteturas principais: rede de camada única, que geralmente não é preferida devido às suas limitações de capacidade, e a de múltiplas camadas, que inclui camadas ocultas entre as camadas de entrada responsáveis por receber a informação e transmiti-la adiante, e a camada de saída, que retorna o resultado final (Kopiler *et al.*, 2019). A Figura 7 (a) ilustra uma arquitetura em camada única e Figura 7 (b) ilustra uma arquitetura em múltiplas camadas.

Figura 7 – Exemplo de arquiteturas



(a) Arquitetura de camada única (b) Arquitetura em múltiplas camadas

Fonte: Adaptado de Kopiler *et al.* (2019, p. 3).

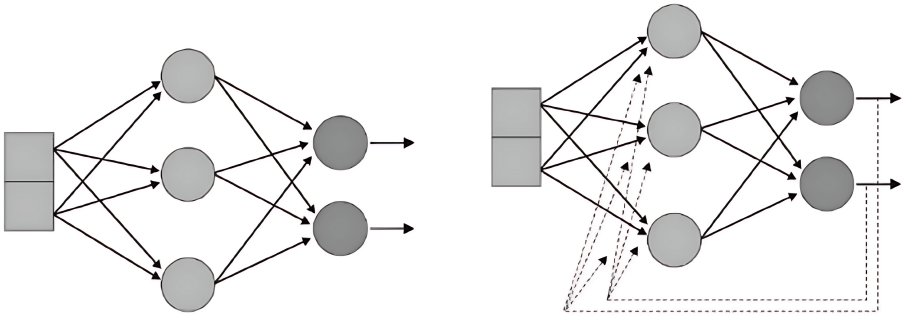
Além das diferentes arquiteturas, é importante ressaltar o processo de treinamento das RNA. Geralmente, o algoritmo de *backpropagation* assume o papel principal no treinamento de redes de múltiplas camadas (Kopiler *et al.*, 2019).

Esse algoritmo ajusta os pesos das conexões entre os neurônios, buscando minimizar o erro na saída da rede. No caso do aprendizado supervisionado, para cada exemplo apresentado a rede é calculado uma resposta correspondente, podendo determinar o nível de erro atual ao comparar com a saída desejada. Esse processo se repete até algum critério seja alcançado, como o nível de erro alcançar um valor aceitável (Neto, 1995). Desta forma, a ideia principal desse algoritmo consiste em propagar os erros das camadas de saída para corrigir imprecisões nas camadas anteriores, através desse processo de realimentação.

3.4.2.1 Rede Neural Feedforward e Recorrente

A informação em uma rede neural geralmente se propaga da camada de entrada até a camada de saída. Em uma rede de múltiplas camadas, esse processo ocorre de uma camada para a próxima. Adicionalmente, uma RNA possui conexões específicas, conhecidas como retropapagação (ou *feedback*), onde os neurônios recebem a saída de outro neurônio em seus terminais de entrada, seja da mesma camada, de camadas posteriores ou até mesmo a própria saída y. Redes com esse tipo de retropapagação são chamadas de Redes Neurais Recorrentes, sendo especialmente úteis para problemas que exigem o processamento de informações de forma sequencial. Por outro lado, redes sem esse tipo de conexão, que são mais comumente utilizadas, são denominadas de Redes Neurais Artificiais feedforward (Faceli *et al.*, 2011).

Figura 8 – Redes neurais *feedforward* e recorrente.



(a) Rede *feedforward* (b) Rede recorrente

Fonte: Faceli *et al.* (2011, p. 113).

Uma rede neural *feedforward* é composta por camadas discretas de neurônios, os quais estão conectados às camadas seguintes. A organização dessa camada geralmente segue o seguinte padrão: uma camada de entrada, que simplesmente transmite as informações adiante; uma ou mais camadas ocultas, formadas por neurônios que recebem as informações de saída dos neurônios da camada anterior, efetuando o processamento dos dados; e uma camada de saída, responsável por gerar a saída final (Grus, 2016). A Figura 8 (a) ilustra um exemplo de uma rede *feedforward* e a Figura 8 (b) ilustra uma rede recorrente.

3.4.3 Aprendizado Não Supervisionado

As tarefas realizadas pelos algoritmos de aprendizado não supervisionado são variadas e podem ser amplamente categorizadas em quatro principais grupos. A primeira dessas

tarefas é o agrupamento (clustering), onde o objetivo é dividir o conjunto de dados em grupos ou clusters de exemplos semelhantes. Essa técnica é amplamente utilizada na segmentação de mercado, onde consumidores com comportamentos similares são agrupados para otimizar estratégias de marketing, ou em bioinformática, para identificar padrões em dados genéticos (Faceli *et al.*, 2011).

Outra tarefa importante é a redução de dimensionalidade, que visa simplificar a estrutura dos dados ao reduzir o número de variáveis ou atributos, preservando ao máximo as informações relevantes. Essa abordagem é extremamente útil quando se trabalha com dados de alta dimensionalidade, como imagens ou grandes conjuntos de dados, onde há muitas variáveis redundantes ou irrelevantes (Monard; Baranauskas, 2003). A redução de dimensionalidade facilita a análise e a visualização dos dados, além de diminuir a complexidade computacional envolvida.

Além disso, o aprendizado não supervisionado é fundamental para a detecção de anomalias. Essa tarefa tem como foco identificar exemplos que se desviam significativamente do comportamento típico observado nos dados. Em áreas como segurança cibernética e prevenção de fraudes financeiras, essa técnica é utilizada para identificar transações suspeitas ou atividades fora do padrão, sem a necessidade de conhecer previamente o que caracteriza uma anomalia (Faceli *et al.*, 2011).

Por fim, a descoberta de associações é outra tarefa chave do aprendizado não supervisionado. Aqui, o algoritmo busca identificar padrões frequentes entre os atributos dos dados, como em sistemas de recomendação, onde se descobre que determinados produtos são frequentemente adquiridos em conjunto. Por exemplo, analisar cestas de compras revela associações úteis para sugerir produtos a clientes de plataformas de comércio eletrônico (Grus, 2016).

3.4.3.1 Análise de Componentes Principais (PCA)

A Análise de Componentes Principais (PCA) é uma técnica de aprendizado de máquina não supervisionado utilizada para a redução de dimensionalidade e extração de informações relevantes de um conjunto de dados complexos, sendo aplicada em diversos campos, como neurociência e computação gráfica (Shlens, 2014). Mais especificamente, transforma os dados originais em um novo espaço de menor dimensionalidade, preservando ao máximo a variabilidade, o que permite reduzir a complexidade dos dados enquanto mantém sua estrutura essencial (Jolliffe, 2002). Além disso, é um método linear e não-paramétrico, o que significa que

não faz suposições sobre a distribuição dos dados (Bishop; Nasrabadi, 2006).

O PCA projeta os dados em componentes principais, que são combinações lineares das variáveis originais. Cada componente principal captura a maior quantidade de variância possível, sendo o primeiro o mais informativo, seguido pelos demais, que são ortogonais entre si (Jolliffe; Cadima, 2016). Essa transformação permite eliminar dimensões menos relevantes, resultando em um conjunto de dados sintético mais simples, mas que preserva as características mais importantes.

De acordo com Shlens (2014), o cálculo do PCA começa com a centralização dos dados, subtraindo a média de cada variável. Em seguida, a matriz de covariância C é calculada para capturar as relações entre os atributos. A fórmula para calcular a matriz de covariância é dada por:

$$C = \frac{1}{n} XX^T \tag{3.16}$$

onde X é uma matriz de $n \times m$, onde cada linha m_i representa uma amostra e cada coluna n_i uma variável. A matriz X^T é a transposta de X, e a multiplicação XX^T resulta em uma matriz de covariância de $m \times m$, representando a covariância entre as variáveis. Esta matriz resume a correlação entre todas as variáveis no conjunto de dados.

De acordo com Jolliffe (2002), após calcular a matriz de covariância, o próximo passo no PCA é encontrar um vetor α_1 (autovetor) que maximiza a variância de uma combinação linear das variáveis, impondo a restrição de normalização $\alpha_1^T \alpha_1 = 1$. A função de maximização $\alpha_1^T \Sigma \alpha_1$, onde Σ é a matriz de covariância, leva à equação de autovalores $\Sigma\alpha_1 = \lambda_1 \alpha_1$, sendo λ_1 o autovalor correspondente, que indica a quantidade de variância explicada pela direção definida por α_1. Esse processo permite que o PCA encontre as direções que capturam a maior variabilidade nos dados, projetando-os em um novo espaço de menor dimensionalidade que preserva a maior parte da variabilidade original.

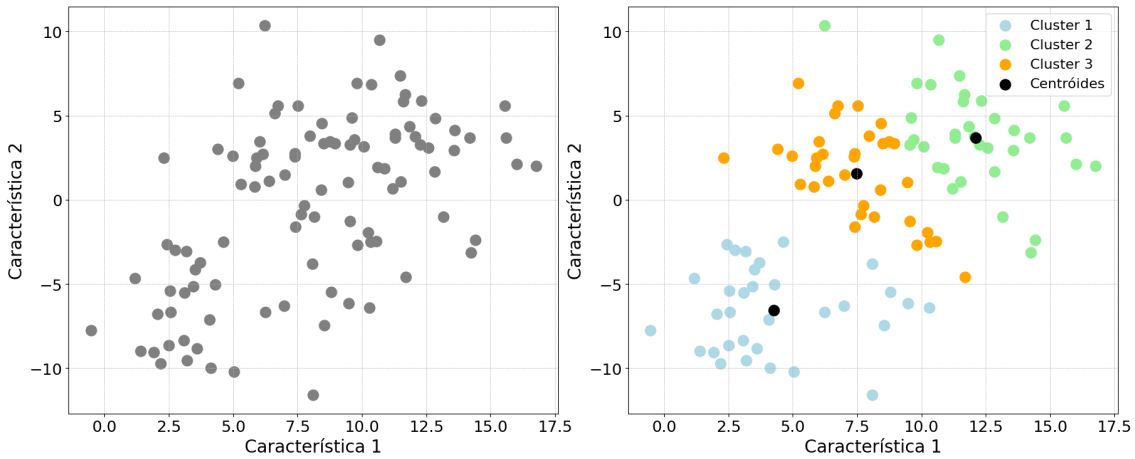
Por fim, após calcular os autovetores e autovalores, podemos projetar os dados no novo espaço de menor dimensionalidade, multiplicando a matriz de dados original pelos autovetores correspondentes aos maiores autovalores (Trendafilov; Gallo, 2021). Assim, a decomposição em autovetores e autovalores nos permite identificar os componentes principais que capturam a variabilidade mais significativa nos dados.

3.4.3.2 Algoritmo K-Means

O algoritmo kmeans é uma técnica de aprendizado de máquina não supervisionado utilizada para realizar a tarefa de agrupamento, isto é, seu objetivo é particionar um conjunto de dados em k grupos (*clusters*) de forma que os elementos dentro de cada *cluster* sejam mais semelhantes entre si do que em relação aos de outros *clusters*. Para isso, a técnica tenta minimizar o erro quadrático entre a média empírica (centróide) de um *cluster* e os pontos que pertencem a ele (Macqueen *et al.*, 1967).

A Figura 9 ilustra a comparação entre os dados antes e depois da aplicação do KMeans. Na Figura 9 (a), todos os pontos são exibidos na mesma cor, sem nenhum grupo definidos, representando o estado inicial dos dados. Já na Figura 9 (b), após aplicar o KMeans, os pontos são agrupados em grupos distintos, sendo representados com cores e símbolos diferentes, e os centróides são destacados na cor preta.

Figura 9 – Aplicação do método k-means para o particionamento dos dados.



Fonte: Elaborada pelo autor.

Conforme Bishop e Nasrabadi (2006), a função objetivo do algoritmo *k-means* é dada por:

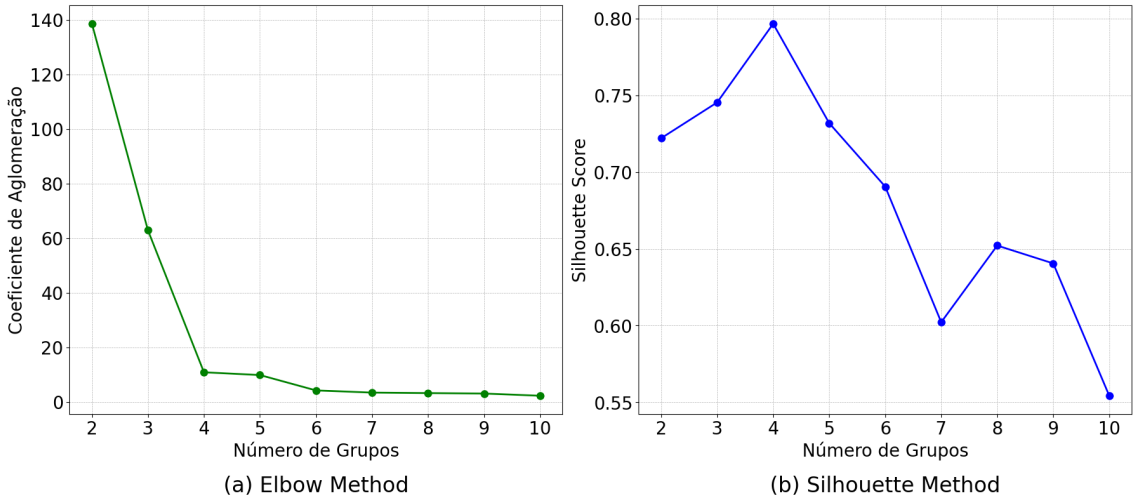
$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||x_n - \mu_k||^2 \tag{3.17}$$

onde J representa a soma das distâncias quadráticas entre cada ponto x_n e o centróide μ_k do *cluster* C_k, tal que a variável binária r_{nk} é 1 se o ponto x_n pertence ao *cluster* k, e 0 caso

contrário. O objetivo é minimizar J, garantindo que os pontos dentro de um *cluster* estejam o mais próximos possível de seu centróide. Cada iteração do *k-means* envolve duas etapas: primeiro, minimiza-se J em relação a r_{nk}, atribuindo cada ponto ao centróide mais próximo, e depois, minimiza-se J em relação a μ_k, recalculando os centróides como a média dos pontos atribuídos a cada *cluster*. Esse processo é repetido até a convergência, garantindo que cada ponto esteja otimamente alocado ao *cluster* mais próximo.

Durante o processo de execução do *k-means*, a escolha do número de grupos k é crucial e pode ser feita usando métodos como o *Elbow Method* (método do cotovelo) ou a Análise de Silhueta. O primeiro método baseia-se no coeficiente de aglomeração, um valor numérico que representa a fusão de vários pontos em um agrupamento. O procedimento envolve traçar o coeficiente no eixo y e o número de grupos no eixo x. Um achatamento acentuado do gráfico indica que os grupos combinados são muito diferentes, e o número ideal de divisões é encontrado no 'cotovelo' do gráfico (Ketchen; Shook, 1996). Já o Silhouette method (método de silhueta) avalia a qualidade das divisões com base no quão próximos os pontos estão dos seus próprios grupos em comparação com os vizinhos (Rousseeuw, 1987). A Figura 10 ilustra essas duas abordagens.

Figura 10 – Métodos *Elbow* e *Silhouette* para Determinação do Número Ótimo de Grupos.



(a) Elbow Method (b) Silhouette Method

Fonte: Elaborada pelo autor.

Note que a Figura 10 (a) mostra que, segundo o método do cotovelo, a partir de 4 grupos, a variação no coeficiente de aglomeração se torna mínima, indicando poucas mudanças significativas. Da mesma forma, a Figura 10 (b) confirma essa observação, já que o método da silhueta aponta que 4 grupos obtêm a maior pontuação de agrupamento, com valor de 0.8.

Embora o k-means seja amplamente utilizado por sua popularidade e facilidade de uso, ele possui limitações, especialmente na atribuição de centróides e no número de clusters, funcionando melhor com grupos compactos e esféricos, mas falhando com dados mais complexos (Ahmed *et al.*, 2020). Para mitigar essas questões, o k-means++, desenvolvido por Arthur e Vassilvitskii (2006), aprimora a inicialização ao selecionar centróides de forma ponderada, priorizando pontos mais distantes, o que aumenta a qualidade dos agrupamentos e reduz a probabilidade de configurações desfavoráveis.

4 TRABALHOS RELACIONADOS

Este capítulo explora trabalhos correlacionados à pesquisa, concentrando-se em aplicações de técnicas de Aprendizado de Máquina para o Problema do Corte Máximo, mesmo que alguns deles não se baseiem especificamente em aprendizado supervisionado. Em resumo, os três trabalhos relacionados empregam técnicas de aprendizado profundo, que, de forma simplificada, envolvem o uso de redes neurais com várias camadas ocultas.

Na Seção 4.1, é apresentado um algoritmo baseado em redes de ponteiros, sendo este modelo treinado por meio de aprendizagem supervisionada. Dessa forma, o modelo emprega técnicas para lidar com dados de forma sequencial e um mecanismo de atenção para destacar as informações mais relevantes. Já na Seção 4.2, também é utilizado o conceito de redes de ponteiros, porém, o modelo é treinado por meio de aprendizado supervisionado e aprendizado por reforço. Por fim, na Seção 4.3, é apresentado um método que faz uso de Redes Neurais em Grafos, uma arquitetura de rede neural projetada para lidar com dados representados na forma de grafos.

4.1 Um algoritmo de aprendizado profundo baseado em redes de ponteiros para o problema do corte máximo

Gu e Yang (2018) apresenta uma abordagem que utiliza um algoritmo de aprendizado profundo, especificamente uma Rede de Ponteiros (*Pointer Network*), para resolver o Problema do Corte Máximo. Essa técnica emprega o *framework* de aprendizado sequencial-para-sequencial (*sequence-to-sequence learning*) em conjunto com um mecanismo de atenção. Assim, o artigo propõe o uso dessa rede para construir um modelo que é treinado por meio de aprendizagem supervisionada.

A arquitetura da rede neural proposta pelo autor é uma variação do modelo Seq2seq, que faz uso das conexões ordenadas da Rede Neural Recorrente (RNN) para transmitir e reter informações durante o processo de predição. Nessa estrutura, a RNN desempenha um papel crucial no treinamento, analisando o atributo de entrada camada por camada, até a camada de saída, mantendo um estado interno que propaga as informações relevantes do passo anterior.

A Rede de Ponteiros incorpora um mecanismo de atenção modificado, integrado com um módulo chamado Seq2seq, para aprender a probabilidade condicional de uma saída, onde os valores correspondem às posições em uma sequência de entrada fornecida. Para tal, o

Seq2seq possui um codificador que transforma uma sequência de dados de tamanho variável em um vetor de tamanho fixo. Como discutido na Subseção 3.4.1, o processo de aprendizagem do modelo normalmente é realizado com base em vetores de tamanho fixo. Ademais, o Seq2seq também inclui um decodificador que realiza o processo inverso.

O mecanismo de atenção, que conecta o codificador ao decodificador, possibilita a este último acessar toda a sequência de estados do codificador, permitindo a extração de informações relevantes de um vetor de comprimento variável. Sua função principal é indicar à rede de decodificação quais partes da entrada são mais significativas, permitindo que o decodificador se concentre em identificar informações mais relevantes na sequência de entrada do codificador, aprimorando a resposta de saída.

Além disso, a entrada dessa rede de ponteiros corresponde a uma instância do Problema do Corte Máximo, onde o peso de cada aresta é um. Portanto, a entrada é representada por uma matriz de adjacência, e a saída consiste na partição de cada vértice. Adicionalmente, as instâncias criadas para o treinamento foram geradas utilizando um programa no MATLAB (*software* utilizado para computação numérica e análise de dados), que criou conjuntos de amostras de forma aleatória, com uma parte dedicada ao conjunto de teste.

Para avaliar o método proposto, o autor criou instâncias do Problema do Corte Máximo com 10, 20, 30, 40 e 50 vértices. O modelo foi submetido a cinco configurações distintas de treinamento, variando entre 1000 e 2000 períodos de treinamento, bem como entre 100 e 1000 amostras fornecidas para o treinamento. Os resultados de precisão para as instâncias com 10, 20, 30, 40 e 50 vértices variaram de 92.0% a 97.5%, de 88.3% a 94.2%, de 85.2% a 86.4% e de 68.7% a 81.4%, respectivamente.

Por último, Gu e Yang (2018) cita que os resultados dos experimentos evidenciam que o método proposto alcança uma solução aproximada satisfatória, ao mesmo tempo em que reduz consideravelmente o tempo necessário para obter essa solução, se comparado com algoritmos convencionais. Isso indica um potencial promissor desse método na área de problemas de otimização combinatória.

4.2 Algoritmo de aprendizado profundo para o problema do corte máximo baseado na estrutura de rede de ponteiros, com estratégias de aprendizado supervisionado e aprendizado por reforço.

Já nesse trabalho, o método proposto pelo Gu e Yang (2020) utiliza redes de ponteiros (*Pointer Network*) e aprendizagem profunda, empregando técnicas de aprendizado supervisionado e aprendizado por reforço. Em termos gerais, uma rede de ponteiros é uma rede neural com foco em aprendizado profundo sequencial-para-sequencial (*sequence-to-sequence*), o que significa que processa os dados de maneira sequencial, extraindo informações com o intuito de revelar alguma relação matemática ou probabilística oculta entre os dados de entrada e saída. Com isso em mente, o mecanismo de entrada e saída do modelo de redes de ponteiros foi projetado para estar em sintonia com as características do Problema do Corte Máximo. Posteriormente, o modelo é treinado utilizando técnicas de aprendizado supervisionado e aprendizado por reforço.

Conforme afirmado pelo Gu e Yang (2020), a Rede Neural Recorrente (RNN) é capaz de classificar eventos subsequentes usando informações de eventos anteriores, demonstrando eficácia ao lidar com dados sequenciais ou temporais. Para alcançar isso, ela processa as informações dos dados de forma contínua e cíclica, assegurando a persistência da informação. No entanto, ela não lida bem com entradas longas, sendo necessárias algumas melhorias, que foram apresentadas no artigo, sendo a mais eficaz delas o uso de um mecanismo de limitação. Para o algoritmo proposto, foi adotado o Long Short Term Memory (LSTM), uma variante da RNN, que faz uso de um mecanismo de controle de fluxo de informações chamado *gating*.

O método proposto também faz uso de um modelo codificador-decodificador, que é utilizado para que a sequência de entrada e a sequência de saída não precisem ter uma relação estrita ou o mesmo comprimento. Além desse modelo, existe um mecanismo de atenção utilizado pela rede para indicar quais dados são mais relevantes em uma sequência de entrada, evitando que a rede armazene mais informações do que o necessário. Desta forma, a rede requer menos neurônios para lidar com a quantidade de informação do que sem esse mecanismo.

Sobre a entrada e saída da rede, enquanto a entrada é uma sequência de comprimento T, a saída é uma sequência de índices relacionados à sequência de entrada. Por exemplo, se a entrada for uma sequência de números desordenados, a saída consistirá nos índices dos números na ordem correta de ordenação. Para o Problema do Corte Máximo, utilizando aprendizagem supervisionada, a rede de ponteiros recebe como entrada, de forma explícita, uma matriz de adjacência Q, onde o vetor $q_i \in \{q_1, ..., q_n\}$ representa as características do vértice $x_i \in \{x_1, ...x_n\}$,

ou seja, cada vetor contém os pesos das arestas entre dois vértices. Caso o peso seja igual a 0, isso indica a ausência de uma aresta. A saída será uma sequência de 0's e 1's que indica a partição de cada vértice.

A aprendizagem por reforço, ao contrário da aprendizagem supervisionada, pode ser descrita como um agente que interage em um ambiente e aprende continuamente conforme interage com o mesmo em busca de atingir um objetivo específico. Desta forma, na aprendizagem por reforço, ele vai construindo sua própria estratégia, não requisitando uma estratégia correta como na abordagem supervisionada. Dito isso, a entrada na rede para o aprendizado por reforço é semelhante à da aprendizagem supervisionada, diferindo apenas pela adição de um símbolo especial. A função desse símbolo é separar um vértice em uma partição. Já a saída, trata-se de uma sequência dos vértices acompanhada de um símbolo para dividir o conjunto de vértices de saída em dois conjuntos, sendo que este símbolo fica no meio da sequência.

Para adquirir as instâncias do problema que serão utilizadas para realizar o treinamento do modelo, o autor faz uso do Gerador de *Benchmark* para gerar instâncias aleatórias do Problema de Programação Quadrática {-1, 1}, os quais podem ser resolvidos em tempo polinomial. Em seguida, essas instâncias são convertidas em instâncias solucionadas do Problema do Corte Máximo. Para tal, eles associam as variáveis desse problema de programação quadrática, que podem assumir apenas valores -1 e 1, com os vértices particionados da instância do Problema do Corte Máximo.

Como a instância solucionada era originalmente um problema de programa quadrático {-1, 1}, suas características possuem propriedades específicas, as quais podem atrapalhar o treinamento do modelo. Em outros termos, isso pode afetar a capacidade do modelo de aprender as regras gerais para o corte máximo. Portanto, o gerador *Benchmark* foi utilizado na Biblioteca *Biq Mac*, que oferece uma coleção de instâncias do corte máximo. Desta forma, o artigo realizou experimentos com dois conjuntos de dados: o conjunto de dados Zhou e o conjunto de dados *Biq Mac Library*.

No experimento, foram utilizados diferentes tamanhos de instâncias para treinamento, variando de 10 a 200 vértices para o aprendizado supervisionado e de 10 a 300 para o aprendizado por reforço. Os resultados, indicam evidenciam uma tendência consistente na redução da precisão à medida que o número de vértices aumentava. Essa diminuição foi mais acentuada no caso do aprendizado por reforço em comparação ao aprendizado supervisionado.

Já para o segundo experimento, o conjunto de dados *Biq Mac Library*, assumiu o

papel de base de teste, enquanto o conjunto de dados Zhou foi utilizado para o treinamento. Desta forma, foram selecionados dez grupos de instâncias de 60, 80 e 100 vértices. Foi realizado apenas o teste para a aprendizagem por reforço, e para cada instância, houve uma média de 88.05%, 84.76% e 73.09%. Esses resultados demonstram que média é menor em relação ao teste passado, isso se deve a fato da *Biq Mac Library* conter instâncias com diferentes proporções e distribuições. Os grafos em si são mais complexos, representando melhor as características essenciais do Problema do Corte máximo.

No geral, os experimentos revelam que para instâncias abaixo de 50 vértices, a aprendizagem supervisionada e aprendizagem por reforço são precisos e consistentes. Entretanto, para instâncias acima de 50 vértices, a aprendizagem por reforço demonstrou-se mais capaz, visto que no teste para instâncias de 200 vértices, a aprendizagem supervisionada obteve 71,95% de precisão, enquanto para 300 vértices, a aprendizagem por reforço obteve 87.64% de precisão.

4.3 Desempenho experimental de redes neurais de grafos em instâncias aleatórias de corte máximo

O Yao *et al.* (2019) conduziu um estudo aplicando técnicas de Aprendizado de Máquina Não-supervisionado para a resolução do Problema do Corte Máximo. Mais especificamente, ele utilizou Redes Neurais em Grafos (GNN), que é uma arquitetura de rede neural especialmente projetada para lidar com dados que têm uma estrutura de grafo. A ideia dessa arquitetura, é propagar informações entre vértices e arestas, capturando suas relações complexas nestes dados interconectados.

O foco principal da pesquisa foi voltado para um tipo específico de grafo, chamado de grafos regulares aleatórios. Nesse grafo, cada vértice possui a mesma quantidade de grau e não segue uma estrutura predefinida. Segundo Yao *et al.* (2019), essa escolha se dá pelo fato de que as assíntotas do valor ótimo de corte máximo são bem conhecidas para esse tipo de distribuição, podendo ser utilizadas para medir o desempenho do método proposto. Além disso, mesmo que não houvesse uma solução explícita conhecida para comparar com a saída do algoritmo proposto, ainda é possível utilizar essas assíntotas.

Dito isso, o autor utilizou as assíntotas conhecidas para avaliar o desempenho da GNN em relação a outros dois métodos: otimização extrema, que é uma heurística de otimização local da literatura de física estatística, e algoritmo de relaxamento semidefinido de Goemans e Williamson (1995). Os resultados obtidos indicam que o desempenho da GNN é comparável

a esse último método. Entretanto, ambas as técnicas demonstraram um desempenho inferior quando comparadas à estratégia de otimização extrema. No geral, as técnicas de aprendizado de máquina não supervisionado conseguem se adaptar com sucesso para problemas de otimização difíceis em entradas aleatórias.

5 METODOLOGIA

A abordagem central consiste em utilizar instâncias de pequena escala do Problema do Corte Máximo para treinar o modelo supervisionado, com o intuito de aplicá-lo posteriormente em instâncias de grande porte. Dito isso, este capítulo apresenta a metodologia adotada para alcançar o objetivo deste estudo.

A Seção 5.1 detalha o procedimento de aquisição dos dados destinados ao treinamento e avaliação do modelo supervisionado. Esses dados serão adquiridos por meio de instâncias da *Biq Mac Library*, juntamente com suas respectivas respostas, utilizando um *solver* baseado em método exato. Já a Seção 5.2 detalha a etapa de pré-processamento dos dados, onde é realizada a criação da base de dados e o tratamento dos dados para serem utilizados no treinamento do modelo.

A Seção 5.3 dá foco em como será a estrutura do treinamento do modelo, especificando a abordagem escolhida para o treinamento, enquanto a Seção 5.5 discute sobre os ajustes dos hiperparâmetros para o refinamento do modelo. Por fim, as Seções 5.4 e 5.6 aborda como serão gerados os resultados e as ferramentas que serão utilizadas, respectivamente.

5.1 Coleta dos dados

Para conduzir essa pesquisa, foram selecionadas 330 instâncias disponíveis no trabalho da Wiegele (2007), *Biq Mac Library*, que é uma fonte reconhecida e respeitável na área do Problema do Corte Máximo. A biblioteca disponibiliza uma coleção diversificada e representativa de instâncias, o que proporciona uma base sólida para o estudo. Ela abrange instâncias de diferentes tamanhos, variando de 20 a 500 vértices, o que trás uma coleção de grafos com propriedades muito distintas, como número de triângulos, densidade, coeficiente de aglomeração, grau de centralidade e grau de intermediação dos vértices.

Para realizar a aquisição dos dados necessários para o treinamento do modelo supervisionado, é essencial não apenas as instâncias em si, mas também as respostas correspondentes, ou seja, as partições dos vértices. Portanto, essas instâncias foram submetidas ao *solver Biq Mac*, um algoritmo exato que se baseia no método *Branch & Bound* e utiliza Programação Semi-Definida (SDP) (Rendl *et al.*, 2010). Entretanto, ele não garante encontrar o valor ótimo para instâncias muito grandes.

5.2 Pré-processamento dos dados

O pré-processamento dos dados coletados é uma etapa crucial para preparar as instâncias para o treinamento do modelo. Nesta fase, diferentes passos foram realizados para garantir que os dados estivessem prontos e adequados para alimentar os algoritmos de aprendizado de máquina.

Devido à variação no número de vértices das instâncias, que pode oscilar entre 20 e 500, aplicar aprendizado de máquina diretamente sobre os atributos que indicam as arestas entre os vértices torna-se inviável. Isso ocorre porque a estrutura da base de dados escolhida assemelha-se a uma matriz de adjacência, cujo tamanho depende diretamente da quantidade de vértices de cada grafo. Por exemplo, um grafo com 20 vértices exigiria uma matriz com 20 colunas para representar as conexões, enquanto um grafo com 500 vértices precisaria de 500 colunas. Essa diferença na dimensionalidade dos dados dificulta diretamente o treinamento e predição do modelo, limitando o mesmo a instâncias que possuem o mesmo tamanho.

Para superar essa limitação e garantir uma representação compacta e eficiente das conexões entre os vértices, foi aplicada a técnica de Análise de Componentes Principais PCA. Essa abordagem permitiu a redução da dimensionalidade desses atributos para um número fixo de 20 componentes sintéticos, número este escolhido com base no menor grafo da base de dados. Desta forma, todos os grafos, independentemente do número de vértices, são representados de forma consistente em um espaço de 20 dimensões, possibilitando o treinamento e aplicação do modelo na predição de instâncias de diferentes tamanhos.

Além disso, os dados passaram por um processo de normalização, essencial para que os modelos de aprendizado de máquina tratem todos os atributos em uma escala uniforme. Isso evita que atributos com magnitudes significativamente diferentes influenciem de forma desproporcional o desempenho do modelo.

Por último, foram selecionadas outras características relevantes, como o grau do vértice, a média dos pesos das arestas conectadas, o menor e maior peso dessas arestas e o potencial total do vértice (soma dos pesos das arestas conectadas). Essas características foram escolhidas para oferecer uma representação robusta e abrangente das propriedades dos vértices e suas conexões no grafo. Portanto, além das colunas que representam as conexões dos vértices, existe colunas que especificam suas propriedades.

5.3 Treinamento do modelo supervisionado

Após o pré-processamento descrito na Subseção 5.2, as instâncias foram devidamente estruturadas para o treinamento dos modelos supervisionados. Cada registro da base de dados é representado por uma tupla $R_i = (\vec{x}_i, y_i)$, onde x_i contém as características extraídas dos vértices, conforme discutido anteriormente, e y_i indica a partição correspondente de cada vértice (resposta esperada). Cada amostra de treinamento corresponde a uma instância do Problema do Corte Máximo. Em adição, cada amostra é um grafo $G = (V, E)$, com $|V| = n$ vértices e $|E| = m$ arestas, representada por uma matriz de n registros. Para novas instâncias, o modelo supervisionado recebe a matriz e retorna a partição dos vértices.

Com a base de dados preparada, diferentes algoritmos de aprendizado de máquina foram aplicados para a construção dos modelos supervisionados, visando a exploração de abordagens distinta para realizar predições e solucionar o problema do corte máximo. Entre os algoritmos selecionados, tem-se o *Naive Bayes*, que explora a suposição de independência entre as variáveis, isto é, as características do vetor de características, permitindo a construção de modelos simples e eficazes.

Além disso, o algoritmo *k-Nearest Neighbors* (kNN) foi utilizado devido à sua capacidade de classificar instâncias não resolvidas com base nas instâncias já solucionadas mais próximas, isto é, considerando a similaridade entre elas. A Regressão Logística, é amplamente utilizada em problemas de classificação binária, possuindo bons resultados na maioria dos casos, foi outra abordagem empregada, sendo adaptada para predizer se um vértice pertence ao conjunto S ou não, levando em conta o contexto do problema do corte máximo.

Por fim, Redes Neurais Artificiais com arquitetura *feedforward* foram implementadas. Esse modelo processa as informações de entrada por meio de uma rede de neurônios interconectados, ajustando iterativamente os pesos dessas conexões com o objetivo de minimizar a taxa de erro na classificação de vértices. É importante destacar que todos os algoritmos foram treinados para mapear as características extraídas dos vértices às suas respectivas partições, buscando maximizar o valor do corte ao alinhar a predição das partições ao objetivo principal do problema, que é o somatório das arestas cortadas, isto é, que pertecem a dois vértices de partições distintas.

5.4 Geração de resultados

A geração dos resultados deste trabalho baseia-se na avaliação do desempenho dos modelos supervisionados desenvolvidos para a resolução do Problema do Corte Máximo. As métricas de avaliação *Precision*, *Accuracy*, *Recall* e *F1-Score*, descritas na Seção 3.4.1.1, foram utilizadas para medir a precisão do modelo ao predizer as partições dos vértices, comparando essas predições com as respostas corretas conhecidas. Entretanto, o foco principal está na avaliação da qualidade do corte obtido, isto é, o quão bem o modelo foi ao resolver uma instância nunca vista antes.

O valor de corte encontrado pelo modelo será comparado com o corte calculado pelo *Solver Biq Mac*, servindo como referência para avaliar a eficácia do modelo, bem como sua generalização e desempenho prático. O objetivo principal é determinar o quão bem o modelo consegue realizar a partição dos vértices.

Para uma análise mais aprofundada dos resultados, será realizada uma investigação exploratória das instâncias que tiveram um melhor valor de corte previsto pelo modelo, em relação as demais. O algoritmo de agrupamento k-means será aplicado com o objetivo de identificar padrões e características comuns entre os diferentes grupos de grafos. A determinação do número ideal de clusters será realizada por meio dos métodos do cotovelo e da silhueta, o que permitirá uma segmentação mais precisa dos grafos de acordo com suas propriedades estruturais e o desempenho do modelo.

Essa análise permitiu diferenciar as instâncias em que o modelo supervisionado obteve sucesso daquelas que tiveram um desempenho inferior. Além disso, foram extraídas características gerais dessas instâncias, sendo analisada a distribuição dos dados por meio de métricas estatísticas básicas, como média, desvio padrão e quartis. Para uma análise mais detalhada, utilizou-se o gráfico de estimativa de densidade do kernel (KDE), que permite estimar a função densidade de probabilidade de uma variável contínua com base em registros observados em uma base de dados, isto é, proporcionando uma visualização da distribuição dos dados em uma coluna (Lin *et al.*, 2020). Essa ferramenta revelou características específicas dos grafos que influenciam diretamente o sucesso do modelo na resolução do Problema do Corte Máximo.

5.5 Ajuste dos parâmetros

Embora a escolha apropriada das variáveis de entrada seja crucial no treinamento do modelo, o refinamento de seu desempenho é caracterizado pela seleção adequada dos hiperparâmetros (refinamento do algoritmo) (Seyedzadeh *et al.*, 2019). Para isso, foi necessário conduzir uma busca sistemática pelo conjunto ideal de hiperparâmetros que melhore o desempenho do modelo.

No ajuste dos hiperparâmetros no k-Nearest-Neighbors (kNN), a escolha apropriada do valor de k é crucial, como discutido na Subseção 3.4.1.2. Esse parâmetro determina o número de k vizinhos mais próximos envolvidos na predição por meio de uma votação majoritária. Por exemplo, quando o valor de k é muito baixo, o modelo pode ajustar-se excessivamente à base de dados de treinamento, afetando negativamente a capacidade de generalizar padrões. Adicionalmente, a escolha da função de distância também é essencial, pois ela determina os k vizinhos mais próximos do registro alvo a serem analisados e a ponderação dos vizinhos mais próximos devem ser ajustadas, testando se os registros mais próximos devem ter maior peso na decisão final do modelo.

Da mesma forma, as Redes Neurais Feedforward possui um conjunto de hiperparâmetros a serem ajustados, que incluem:

- Número de camadas e neurônios: determinar a quantidade de camadas na rede e o número de neurônios em cada camada;
- Taxa de Aprendizado: determina a taxa pela qual a rede irá ajustar seus pesos de conexões entre os neurônios, que busca minimizar o erro de saída;
- Função de Ativação: determinar qual função utilizar nos neurônios, como visto na Subseção 3.4.2;
- Algoritmo de Otimização: definir o algoritmo responsável por ajustar os pesos da rede com base nos gradientes de erro calculados pelo algoritmo *backpropagation*. Entre os algoritmos de otimização, tem-se o *Stochastic Gradient Descent* e o Adam;

Já para a Regressão Logística, foi analisado dois hiperparâmetros. O primeiro é o campo de regualização, que controla a taxa de regularização dos pesos, como discutido na Seção tal, sendo essencial para evitar o overfitting do modelo. O segundo é o algoritmo de otimização, possuindo dois algoritmos analisados: *lbfgs*, que é recomendado para grandes bases de dados e *liblinear*, que é mais adequada para problemas lineares.

Por último, o Naive Bayes é conhecido por sua abordagem ingênua, destacando-se

pela simplicidade e eficácia do modelo. Essa abordagem ingênua, de modo geral, é derivada da premissa de que as probabilidades condicionais dos atributos são independentes entre si, como visto na Subseção 3.4.1.3. Embora essa simplicidade seja justamente sua vantagem, ela também implica em poucos hiperparâmetros se comparado com modelos mais complexos, como kNN e Redes Neurais.

No caso do Naive Bayes com distribuição Gaussiana, o qual será utilizado, o hyperparâmetro que será ajusta é a suavização das estimativas de variância. Ajustar esse valor, dependendo do valor especificando, o modelo pode generalizar os conceitos aprendidos melhor.

5.6 Ferramentas utilizadas

Para a implementação deste trabalho, será utilizado o Python como linguagem de programação, fazendo uso da IDE *Visual Studio Code* com a extensão *Jupyter*. Essa extensão permite ao programador separar o código em células dentro do ambiente de desenvolvimento, possibilitando a execução de um trecho do código de forma individual. Adicionalmente, serão utilizadas as seguintes bibliotecas:

- Pandas: para realizar o pré-processamento dos dados, pois permite a criação e manipulação de bases de dados.
- *Scikit-learn (Sklearn)*: para implementação dos algoritmos de Aprendizado de Máquina, pré-processamento dos dados, ajustes dos hiperparâmetros e avaliação dos modelos.
- *NumPy*: para operações numéricas e manipulação de *arrays*.
- *Networkx*: utilizado para trabalhar com grafos no python, possuindo funções que calcula métricas como centralidade de grau e excentricidade.

6 RESULTADOS

Este capítulo apresenta os resultados obtidos ao longo deste trabalho. Na Seção 6.1, é discutido o primeiro experimento, que consistiu no treinamento do modelo com base na seleção aleatória das amostras. A Seção 6.2 apresenta uma busca exploratória nas instâncias que demonstraram melhor desempenho em comparação às demais, conduzindo uma análise suas propriedades para identificar as possíveis causas.

Na Seção 6.3, foram selecionadas apenas as instâncias com coeficiente de aglomeração positivo e distante de zero para o treinamento e avaliação do modelo. Além disso, são apresentados os resultados obtidos e esclarecidas as dúvidas levantadas no primeiro experimento. A Subseção 6.3.1 explora o comportamento do modelo, mostrando que uma baixa acurácia nem sempre resulta em um baixo valor de corte. Por fim, na 6.4, são apresentados os valores obtidos para os hiperparâmetros de cada modelo treinado e na Seção 6.5, é aplicado o modelo treinado para instâncias com mais de até 3000 vértices.

6.1 Treinamento do modelo com varias instâncias aleatórias

Ao realizar o treinamento do modelo supervisionado com as instâncias da *Biq Mac Library*, as bases de dados foram divididas em conjuntos de treinamento e teste. A estratégia adotada foi concatenar as instâncias selecionadas para o treinamento e, em seguida, utilizar o modelo para realizar a partição de instâncias não aprendidas.

Devido às diferenças nas propriedades dos grafos, como centralidade de grau e coeficiente de aglomeração, observou-se uma grande variação entre os conjuntos de dados. Isso resultou em correlações distintas entre as variáveis das bases de dados de diferentes instâncias, tornando o processo de generalização do modelo um desafio. Essas variações dificultaram a capacidade do modelo de identificar padrões consistentes e aplicáveis a grafos com características estruturais variadas.

Os resultados obtidos após o treinamento e teste indicaram que, das 330 instâncias testadas, o modelo não apresentou um desempenho satisfatório geral, com apenas 58 instâncias exibindo valores significativamente mais elevados do que as demais, ainda que relativamente baixos, conforme apresentado nas tabelas contidas no Apêndice A. Nessas tabelas, são listados os nomes das instâncias de teste, o valor de corte obtido através do *Solver Biq Mac*, bem como os valores obtidos pelos modelos de aprendizado de máquina Naive Bayes, kNN, Regressão

Logística e Rede Neural Artificial.

É importante destacar que o kNN foi o modelo que apresentou o melhor desempenho geral, especialmente na Tabela 8. Isso sugere que o aprendizado baseado em instâncias, que armazena as informações de cada amostra, foi eficaz em detectar padrões de instâncias que se assemelham aos grafos já aprendidos pelo modelo. No geral, os resultados obtidos se alinha com a expectativa de que a modelagem seria mais eficiente para um subconjunto de grafos que compartilham propriedades estruturais específicas, reforçando a importância de considerar a similaridade estrutural entre as instâncias nesse processo de treinamento.

Assim como em bases de dados tradicionais é importante remover valores extremos e realizar o pré-processamento dos dados para selecionar aqueles mais semelhantes, a fim de melhorar a qualidade do treinamento, esse mesmo princípio pode ser aplicado às instâncias do Problema do Corte Máximo. Para otimizar o aprendizado do modelo, é necessário selecionar apenas as instâncias que apresentam similaridades estruturais, aumentando, assim, o valor de corte obtido pelo modelo.

6.2 Análise das instâncias que tiveram um maior desempenho em relação as outras

A análise das instâncias focou nas características estruturais das 58 instâncias-alvo, avaliando suas estatísticas básicas e aplicando os métodos do cotovelo e da silhueta para determinar a quantidade ideal de grupos. O algoritmo k-means foi utilizado para realizar a separação, com o objetivo de identificar a característica chave que contribuiu para o bom desempenho do modelo nessas instâncias. A Tabela 2 apresenta as medidas gerais dessas características, incluindo a dispersão dos dados e os valores extremos. Para as métricas de centralidade e coeficiente de aglomeração, que são atributos dos vértices, foram registrados os valores mínimos, máximos e a média observada em cada instância, proporcionando uma visão da variação dessas características dentro de cada grafo.

Conforme evidenciado na Tabela 2, as instâncias-alvo estudadas variam em tamanho, de 20 a 125 vértices, com uma média de 85 vértices e um desvio padrão de 21. Em relação às arestas, a menor quantidade observada é de 210, enquanto a maior é de 7800, com uma média de 2500 arestas. No entanto, o desvio padrão de 1500 indica uma grande dispersão, sugerindo que alguns grafos são mais conectados do que outros. Essa diferença de conectividade pode ser confirmada ao analisar a densidade, que vai de 0.5 até 1.1, evidenciando tanto a presença de grafos completos quanto a ocorrência de pelo menos uma instância com laços em sua estrutura.

Tabela 2 – Estatísticas descritivas das propriedades do grafo

Características	mean	std	min	25%	50%	75%	max
Número de vértices	8.5×10^{1}	2.1×10^{1}	2.0×10^{1}	8.0×10^{1}	1.0×10^{2}	1.0×10^{2}	1.25×10^{2}
Número de arestas	2.5×10^{3}	1.5×10^{3}	2.1×10^{2}	1.6×10^{3}	2.5×10^{3}	2.5×10^{3}	7.8×10^{3}
Peso médio das aresta por vértices	6.6×10^{2}	1.3×10^{3}	3.0×10^{1}	4.0×10^{1}	5.0×10^{1}	4.9×10^{2}	6.1×10^{3}
Número de componentes conectados	1.0×10^{0}	0.0×10^{0}	1.0×10^{0}	1.0×10^{0}	1.0×10^{0}	1.0×10^{0}	1.0×10^{0}
Diâmetro	2.0×10^{0}	0.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}
Excentricidade média	1.9×10^{0}	1.8×10^{-1}	1.3×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}
Raio	1.8×10^{0}	3.7×10^{-1}	1.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}	2.0×10^{0}
Densidade	6.5×10^{-1}	2.2×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	9.0×10^{-1}	1.1×10^{0}
Média centralidade do grau	6.5×10^{-1}	2.2×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	9.0×10^{-1}	1.1×10^{0}
Máxima centralidade do grau	7.6×10^{-1}	1.8×10^{-1}	6.0×10^{-1}	6.3×10^{-1}	6.5×10^{-1}	9.6×10^{-1}	1.1×10^{0}
Mínima centralidade do grau	5.4×10^{-1}	2.6×10^{-1}	2.9×10^{-1}	3.5×10^{-1}	3.8×10^{-1}	8.2×10^{-1}	1.1×10^{0}
Média centralidade intermediação	1.4×10^{-2}	1.5×10^{-2}	5.1×10^{-3}	6.4×10^{-3}	8.6×10^{-3}	1.5×10^{-2}	8.2×10^{-2}
Máxima centralidade intermediação	5.2×10^{-2}	8.0×10^{-2}	7.4×10^{-3}	1.1×10^{-2}	2.0×10^{-2}	5.3×10^{-2}	4.2×10^{-1}
Mínima centralidade intermediação	1.8×10^{-3}	1.4×10^{-3}	0.0×10^{0}	2.1×10^{-4}	2.2×10^{-3}	2.9×10^{-3}	4.5×10^{-3}
Média coeficiente de aglomeração	4.4×10^{-1}	9.0×10^{-2}	2.4×10^{-1}	4.3×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	5.1×10^{-1}
Máximo coeficiente de aglomeração	4.8×10^{-1}	9.2×10^{-2}	2.7×10^{-1}	4.8×10^{-1}	5.2×10^{-1}	5.4×10^{-1}	5.8×10^{-1}
Mínimo coeficiente de aglomeração	3.9×10^{-1}	9.0×10^{-2}	2.1×10^{-1}	3.7×10^{-1}	4.3×10^{-1}	4.6×10^{-1}	4.8×10^{-1}
Número de triângulos	4.3×10^{4}	5.7×10^{4}	1.1×10^{3}	1.0×10^{4}	2.0×10^{4}	5.3×10^{4}	3.1×10^{5}
Transitividade	6.5×10^{-1}	2.1×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	5.0×10^{-1}	9.0×10^{-1}	9.9×10^{-1}

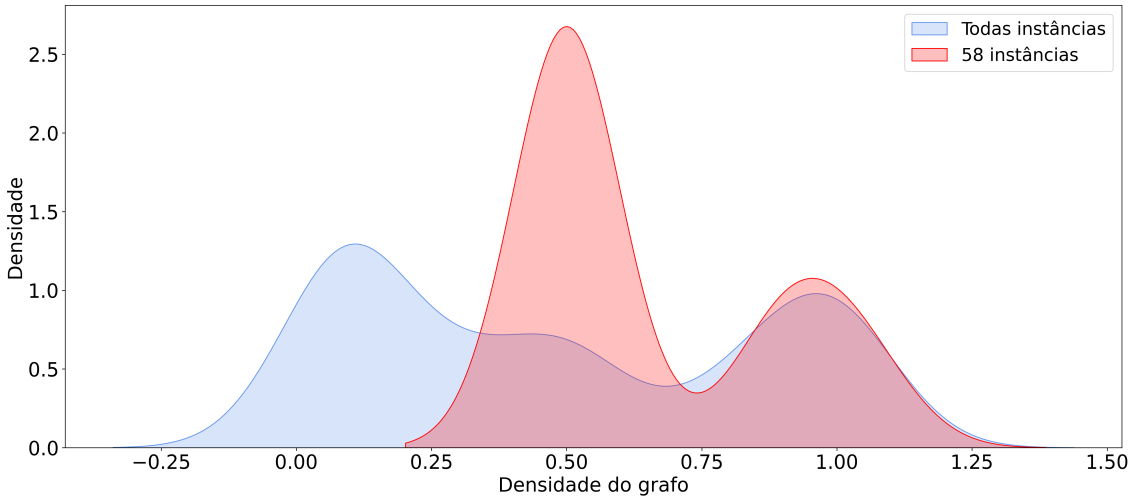
Fonte: elaborado pelo autor.

A Figura 11, demonstra a estimativa de densidade kernel (KDE) para a característica de densidade das instâncias-alvo em comparação com toda a coleção de instâncias, sendo que estas últimas variam de 0.01 até 1.1, com uma média de 0.49 e desvio padrão de 0.37. Dito isso, o modelo desenvolvido não lida bem com grafos esparsos, o que pode ser pelo uso do PCA nas colunas que representam as conexões e seus respectivos pesos. Em outras palavras, como os grafos são muito esparsos, apresentando poucas conexões entre os vértices, ou seja, contendo muitos valores 0, a matriz resultante possui baixa variabilidade. Isso dificulta a eficácia do PCA na redução de dimensionalidade, pois há pouca informação nas arestas com peso em comparação às arestas que não existem.

O peso médio das arestas por vértice também apresenta uma variação significativa, com valores que oscilam de 30 até 6100, possuindo uma disparidade entre o valor máximo, que é 200 vezes maior que o mínimo, sugerindo que alguns grafos contêm peso de arestas mais elevado, bem como mais conexões, indicando importantes diferenças estruturais. Outros aspectos importantes incluem a quantidade de componentes conexos e o diâmetro dos grafos, uma vez que todas as instâncias são conexas e compartilham o mesmo diâmetro, ou seja, sua maior distância mínima entre dois vértice em todo grafo é 2.

A excentricidade média tem seu valor próximo de 2, com um desvio padrão baixo de 0.16, indicando uma distribuição bastante homogênea dos vértices. Como a excentricidade mede a maior distância mínima entre um vértice e todos os outros, essa proximidade com o diâmetro

Figura 11 – KDE da característica de densidade.



Fonte: Elaborada pelo autor.

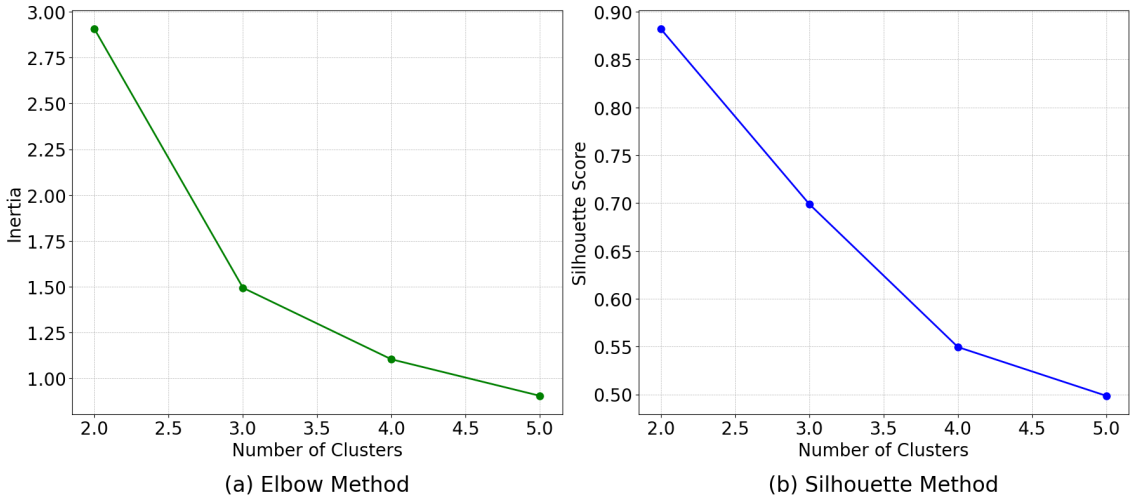
do grafo sugere que a maioria dos vértices está relativamente perto do centro. Isso evidencia uma distribuição equilibrada dos vértices, com distâncias similares ao centro do grafo, reforçando a ideia de uma estrutura compacta e bem conectada.

No que diz respeito à centralidade do grau, o valor médio é de 0.65, com as características de centralidade mínima e máxima variando ligeiramente entre 0.54 e 0.76. Isso indica que os grafos são equilibrados em termos de importância dos vértices, já que a centralidade do grau reflete o número de conexões diretas de cada vértice, medindo sua importância no grafo. Entretanto, as três características que dizem respeito a centralidade de intermediação possuem valores muito próximo de 0, o que confirma a ausência de "vértices-chave" responsáveis por intermediar muitas conexões entre outros vértices, sugerindo que o fluxo de conexões não depende fortemente de vértices específicos.

Como a centralidade de intermediação mostrou valores extremos e consistentes em todas as 58 instâncias, com um desvio padrão próximo de 0, ela pode ser um dos principais fatores que explicam o melhor desempenho do modelo nessas instâncias-alvo. Com isso em mente, foram utilizados os métodos do cotovelo e da silhueta para determinar a quantidade ideal de grupos, além da aplicação do k-means para particionar todas as 330 instâncias, levando em consideração apenas a centralidade de intermediação. A Figura 12 mostra que o método do cotovelo indica 2 grupos como ideal, enquanto o método da silhueta sugere 3 grupos.

Ao realizar a partição das instâncias com base exclusivamente na centralidade de intermediação, seja separando em 3 ou 2 grupos, observou-se que as 58 instâncias foram sempre

Figura 12 – Quantidade de grupos ideais com base na característica de centralidade de intermediação.



Fonte: Elaborada pelo autor.

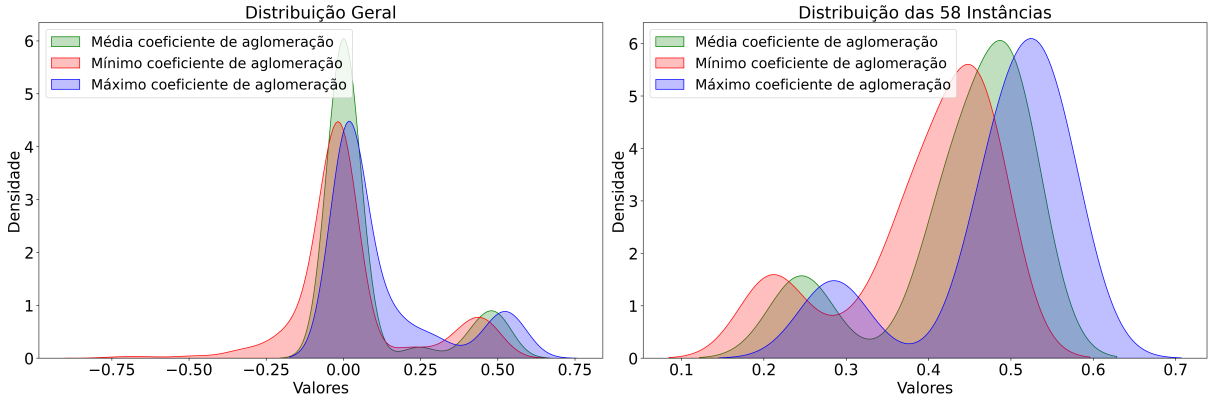
agrupadas no mesmo conjunto, juntamente com outras 10 instâncias. Esse fato sugere que a centralidade de intermediação é uma das características-chave para o bom desempenho do modelo. No entanto, a presença de outras instâncias nesse grupo, onde o modelo não teve um desempenho satisfatório, embora em pequena quantidade, indica que essa característica, por si só, não determina o sucesso do modelo. Isso sugere que outros fatores podem estar influenciando o desempenho e precisam ser considerados.

A característica que registra a média do coeficiente de aglomeração para cada vértice em todas as instâncias apresenta uma mediana de 0.5, com um desvio padrão muito baixo de 0.09. O grafo com a menor média possui um valor de 0.24, enquanto o grafo com a maior média tem 0.51. Adicionalmente, as medianas das características que registraram os valores mínimos e máximos do coeficiente de aglomeração são 0.46 e 0.54, respectivamente. No geral, esses dados sugerem uma quantidade moderada de agrupamentos locais.

A Figura 13 apresenta a estimativa de densidade kernel (KDE) para as colunas de média, máximo e mínimo do coeficiente de aglomeração, ilustrando a distribuição dos valores dentro do intervalo observado em cada uma dessas colunas. A Figura 13 (a) mostra a distribuição para todas as instâncias, enquanto a Figura 13 (b) foca no conjunto das 58 instâncias-alvo. Na Figura 13 (a), observa-se a presença de duas populações distintas: uma classe majoritária com valores muito próximos de zero, incluindo até valores negativos, e uma classe minoritária composta exclusivamente por valores positivos. A Figura 13 (b) demonstra precisamente essa classe minoritária, que corresponde às instâncias-alvo observadas, reforçando a separação entre

essas duas populações.

Figura 13 – KDE do coeficiente de aglomeração



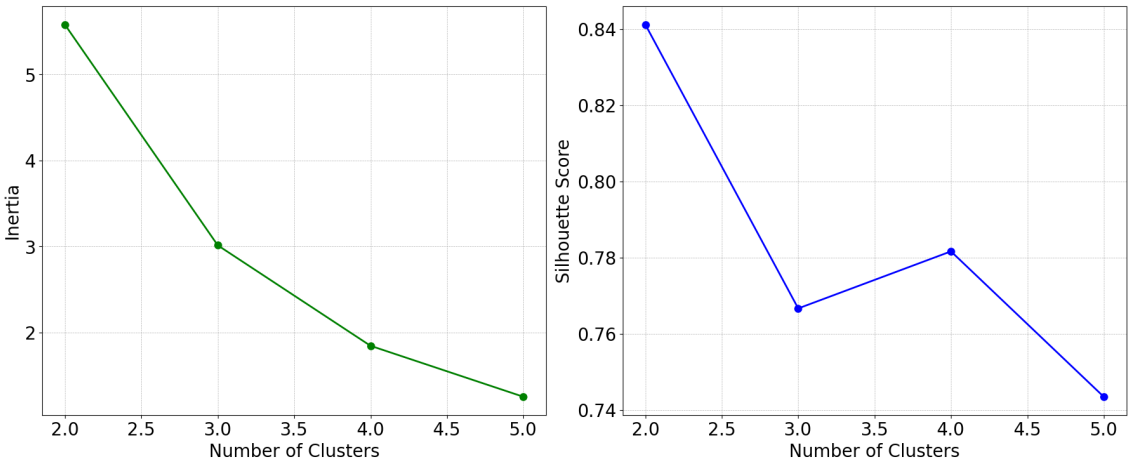
(a) Kernel Density Estimation (KDE) - Todas as Instâncias

(b) Kernel Density Estimation (KDE) - 58 Instâncias

Fonte: Elaborada pelo autor.

Já a Figura 14 comprova que o número ideal de grupos é 2, sendo indicado tanto pelo método do cotovelo quanto pelo método da silhueta. Além disso, ao aplicar o algoritmo k-means, todas as 58 instâncias foram corretamente separadas das demais, reforçando a ideia de que o coeficiente de aglomeração é uma característica chave para diferenciar as instâncias-alvo. Isso sugere que para que o modelo consiga ter um bom desempenho ao realizar o particionamento dos vértices para o Problema do Corte máximo, de novas instâncias, é necessário que elas possuem valores de coeficiente de aglomeração positivos e que não estejam muito próximo de zero.

Figura 14 – Quantidade de grupos ideais com base na característica de coeficiente de aglomeração.



(a) Elbow Method

(b) Silhouette Method

Fonte: Elaborada pelo autor.

6.3 Treinamento focado no coeficiente de aglomeração

A fim de confirmar a hipótese levantada na seção 6.1, de que a qualidade do treinamento do modelo seria maior ao utilizar instâncias de treinamento que compartilham características estruturais semelhantes, foi realizado um novo treinamento e teste focado na população minoritária apresentada na seção 6.2.

Com o agrupamento de instâncias baseadas no coeficiente de aglomeração, o modelo passou a se especializar em grafos que compartilham essa característica estrutural, especificamente as propriedades desse subconjunto de instâncias. Para testar essa abordagem, foi realizada uma validação cruzada, onde 57 instâncias foram utilizadas como conjunto de treinamento, enquanto uma servia como teste, até que todas as instâncias fossem avaliadas. Esse processo revelou que, na maioria dos casos, as instâncias alcançaram uma margem de 90% no valor de corte, se comparadas com o valor fornecido pelo solver. Esses resultados indicam que o modelo apresenta maior capacidade de generalização quando treinado em um conjunto de grafos com estruturas mais similares, comprovando a hipótese levantada anteriormente.

Os resultados detalhados deste experimento estão disponíveis nas Tabelas 3, 4 e 5, onde o valor de corte obtido pelos modelo supervisionados foi comparado com o valor fornecido pelo *solver Biq Mac*. O desempenho do modelo foi significativamente superior nas instâncias com coeficiente de aglomeração mais elevado, reforçando a importância de considerar essa propriedade específica dos grafos ao treinar o modelo.

6.3.1 Relação da Acurácia com valor de corte

Ao analisar métricas como *F1-Score*, *Recall*, *Acurácia* e *Precisão*, foi identificado uma relação importante. Embora o modelo apresente uma acurácia muito baixa para algumas instâncias do problema, o que também afeta negativamente as demais métricas de avaliação, isso não compromete o valor do corte, conforme demonstrado na Tabela 6. Nela foi listado alguns casos extremos, onde, apesar da baixa acurácia, o valor de corte se aproxima significativamente daquele fornecido pelo solver, o que faz com que essas métricas não sejem ideais para avaliar o modelo treinado.

Esse comportamento sugere que o modelo, em vez de se ajustar aos detalhes específicos de cada instância, está aprendendo uma lógica diferente, que é eficaz para resolver o Problema do Corte Máximo. Em outras palavras, embora o modelo não esteja selecionando

Tabela 3 – Resultados para o Treinamento especializado - Parte I

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
pw05_100.0	8190	7619.0	7018.0	7976.0	6998.0
pw05_100.1	8045	7149.0	6999.0	7834.0	6836.0
pw05_100.2	8039	7494.0	6934.0	7703.0	6654.0
pw05_100.3	8139	7568.0	6748.0	7730.0	7124.0
pw05_100.4	8125	6803.0	6998.0	7676.0	6658.0
pw05_100.5	8169	6488.0	7033.0	7827.0	7166.0
pw05_100.6	8217	6752.0	7223.0	7751.0	6924.0
pw05_100.7	8249	7590.0	7014.0	7754.0	6937.0
pw05_100.8	8199	7377.0	6893.0	7989.0	7168.0
pw09_100.0	13585	12331.0	12323.0	13184.0	11597.0
pw09_100.1	13417	12009.0	12588.0	12875.0	11659.0
pw09_100.2	13461	12520.0	12296.0	13274.0	12565.0
pw09_100.3	13656	12169.0	12217.0	13172.0	12428.0
pw09_100.4	13514	11773.0	12134.0	13250.0	12451.0
pw09_100.5	13574	11742.0	11883.0	13347.0	12044.0
pw09_100.6	13640	12440.0	12551.0	13352.0	12512.0
pw09_100.7	13501	12429.0	12172.0	13045.0	12524.0
pw09_100.8	13593	11743.0	12450.0	13021.0	11706.0
pw09_100.9	13658	13027.0	12555.0	13424.0	12351.0

Fonte: elaborada pelo autor.

Tabela 4 – Resultados para o Treinamento especializado - Parte II

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
gka1b	5744	4678.0	4330.0	5353.0	4652.0
gka2b	12451	9514.0	9757.0	12291.0	10320.0
gka3b	22115	16861.0	19459.0	21702.0	18761.0
gka4b	34857	29743.0	29856.0	33809.0	31412.0
gka5b	49942	45134.0	45281.0	47822.0	44976.0
gka6b	68189	58950.0	55934.0	67188.0	60696.0
gka7b	87428	65660.0	77946.0	84902.0	77778.0
gka8b	109969	95436.0	100895.0	108287.0	102557.0
gka9b	135757	97029.0	122643.0	130581.0	124552.0
gka10b	209946	118596.0	194859.0	206584.0	196701.0

Fonte: elaborada pelo autor.

exatamente os mesmos vértices que o *Solver Biq Mac* para o subconjunto S, algo que é necessário para melhorar as métricas de avaliação, como Acurácia e *F1-Score*, seu valor de corte indica que sua abordagem, de maneira geral, está alinhada com a solução esperada pelo solver.

6.4 Ajuste dos hiper-parâmetros

Para realizar o ajuste dos hiperparâmetros, foi utilizada a ferramenta *GridSearch*, que testa diversas combinações de parâmetros para encontrar a configuração que maximiza um

Tabela 5 – Resultados para o Treinamento especializado - Parte II

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
g05_60.1	532	492.0	464.0	504.0	460.0
g05_60.2	529	460.0	433.0	514.0	452.0
g05_60.3	538	474.0	450.0	518.0	440.0
g05_60.4	527	489.0	455.0	497.0	461.0
g05_60.5	533	481.0	470.0	526.0	433.0
g05_60.6	531	467.0	455.0	507.0	454.0
g05_60.7	535	483.0	422.0	515.0	427.0
g05_60.8	530	496.0	465.0	494.0	462.0
g05_60.9	533	511.0	421.0	519.0	477.0
g05_80.0	929	833.0	812.0	885.0	809.0
g05_80.1	941	875.0	764.0	884.0	801.0
g05_80.2	934	851.0	805.0	890.0	832.0
g05_80.3	923	854.0	801.0	891.0	810.0
g05_80.4	932	840.0	788.0	880.0	793.0
g05_80.5	926	856.0	789.0	882.0	751.0
g05_80.6	929	855.0	807.0	899.0	820.0
g05_80.7	929	889.0	769.0	894.0	837.0
g05_80.8	925	862.0	814.0	901.0	795.0
g05_80.9	923	831.0	813.0	894.0	777.0
g05_100.0	1430	1300.0	1236.0	1351.0	1226.0
g05_100.1	1425	1352.0	1289.0	1399.0	1277.0
g05_100.2	1432	1357.0	1245.0	1393.0	1275.0
g05_100.3	1424	1345.0	1233.0	1368.0	1253.0
g05_100.4	1440	1309.0	1270.0	1375.0	1286.0
g05_100.5	1436	1319.0	1276.0	1392.0	1288.0
g05_100.6	1434	1354.0	1239.0	1378.0	1282.0
g05_100.7	1431	1326.0	1257.0	1390.0	1242.0
g05_100.8	1432	1280.0	1281.0	1346.0	1293.0
g05_100.9	1430	1312.0	1265.0	1351.0	1234.0

Fonte: elaborada pelo autor.

valor de avaliação específico. Neste caso, o valor escolhido foi o corte máximo obtido pelo modelo, pois, como visto anteriormente, uma acurácia baixa não necessariamente resulta em um bom valor de corte. Dessa forma, os melhores hiperparâmetros obtidos para cada modelo foram:

1. Naive Bayes:
   - Taxa de suavização da variância: foram testada em 10 valores variando entre 10^{-9} e 10^{0}. O melhor valor foi de 0.001.
2. k-Nearest Neighbors (kNN):
   - Número de vizinhos mais próximos: avalia a quantidade de vizinhos que serão analisados para a predição. Os valores testados foram 5, 10 e 15, sendo 5 o número de vizinhos que obtive melhor resultado;
   - Métrica de distância: foram testadas as métricas Minkowski, Euclidean e Manhattan. A métrica com melhor desempenho foi Minkowski;

Tabela 6 – Relação da acurácia com valor de corte

Instâncias de Teste	Solver	Naive Bayes		kNN		Regressão Logística		Rede Neural	
		Acurácia	Corte	Acurácia	Corte	Acurácia	Corte	Acurácia	Corte
g05_80.8	925	0.31	862.0	0.38	814.0	0.31	901.0	0.4	795.0
g05_60.7	**535**	0.22	483.0	0.52	422.0	**0.22**	**515.0**	0.45	427.0
g05_60.6	531	0.3	467.0	0.43	455.0	**0.27**	**507.0**	0.45	454.0
g05_60.8	530	0.33	496.0	0.52	465.0	0.35	494.0	0.53	462.0
g05_100.1	1425	0.37	1352.0	0.35	1289.0	0.36	1399.0	0.47	1277.0
g05_100.7	**1431**	0.24	1326.0	0.45	1257.0	**0.19**	**1390.0**	0.44	1242.0
g05_100.8	1432	0.3	1280.0	0.36	1281.0	0.29	1346.0	0.36	1293.0
g05_100.9	**1430**	**0.23**	**1312.0**	0.34	1265.0	**0.18**	**1351.0**	0.38	1234.0
pw05_100.0	**8190**	0.29	7619.0	0.42	7018.0	**0.22**	**7976.0**	0.39	6998.0
pw09_100.2	**13461**	**0.23**	**12520.0**	0.47	12296.0	**0.15**	**13274.0**	0.33	12565.0
pw09_100.9	**13658**	**0.22**	**13027.0**	0.42	12555.0	**0.14**	**13424.0**	0.46	12351.0
pw05_100.8	**8199**	0.21	7377.0	0.38	6893.0	**0.14**	**7989**.0	0.29	7168.0
gka2b	12451	0.23	9514.0	0.5	9757.0	**0.07**	**12291.0**	0.43	10320.0
gka3b	**22115**	0.25	16861.0	0.42	19459.0	**0.18**	**21702.0**	0.5	18761.0
gka4b	34857	0.32	29743.0	0.48	29856.0	0.28	33809.0	0.48	31412.0
gka5b	49942	0.3	45134.0	0.33	45281.0	0.27	47822.0	0.52	44976.0

Fonte: elaborada pelo autor.

- Ponderação dos vizinhos: esse campo define se o peso dos vizinhos será uniforme ou baseado na distância, isto é, onde os vizinhos mais próximos têm maior impacto. O melhor desempenho foi para uniforme.

3. Regressão Logística:
   - Parâmetro de regularização (C): o melhor resultado foi 0.046415888336127774, sendo que foram testados 10 valores entre 10^{-4} e 10^{4};
   - Algoritmo de otimização: entre *lbfgs* e *liblinear*, o que performou melhor foi *lbfgs*.
4. Rede Neural Artificial (RNA):
   - Função de ativação: entre tangente hiperbólica e ReLU, a que performou melhor foi ReLU;
   - Quantidade de camadas ocultas e neurônios: duas camadas ocultas com 100 neurônios cada tiveram os melhores resultados. Além dessa, também foram testadas uma camada com 50 neurônios, uma camada com 100 neurônios, além de duas camadas com 50 neurônios cada;
   - Taxa de aprendizado: o tipo de taxa de aprendizado testado foi constante, onde o taxa permaneceria a mesma com o passar das épocas, e adaptativa, que iria se adaptando conforme o desempenho do modelo, sendo que esta última retornou melhores resultados;
   - Algoritmo de otimização: entre *Stochastic Gradient Descent* e *Adam*, o *Stochastic*

Gradient Descent retornou melhores resultados.

6.5 Aplicação do modelo treinado em instâncias de grande porte

O conjunto de instâncias G, gerado por Helmberg e Rendl (2000), contém grafos de grande porte, com tamanhos variando de 800 a 3000 vértices. Neste contexto, foram aplicados os modelos desenvolvidos ao longo deste trabalho para obter o valor de corte nessas instâncias, mas com uma limitação, isto é, devido à grande quantidade de vértices e arestas, não foi possível extrair características dos grafos, como coeficiente de aglomeração e medidas de centralidade. Esse detalhe importante impediu que o modelo fosse treinado com instâncias ideais para cada tipo de grafo existente no conjunto G.

Outro fator importante a ser informado é que, devido à dificuldade de extrair as características dos grafos, não foi possível identificar aqueles que possuem uma média de coeficiente de aglomeração com valores positivos e distante de zero, além da média de centralidade de intermediação próxima de zero ou uma mínima centralidade de grau superior a 0.53. Como visto na Seção 2, essas características são fundamentais para o desempenho dos modelos desenvolvidos, tornando o processo de ajuste e validação dos mesmos inviável para as instâncias em questão.

Para o treinamento, foram utilizadas todas as instâncias da *Biq Mac Library*. O algoritmo com melhor desempenho foi o kNN, apresentando resultados e comportamentos semelhantes aos gerados na Seção 6.1, onde o treinamento foi realizado com instâncias aleatórias. No geral, essa situação se repete neste experimento também. Assim como no experimento anterior, o modelo conseguiu um desempenho melhor em algumas instâncias em específico, sugerindo que há grafos na base de treinamento com estruturas semelhantes ou padrões semelhantes com a instância-alvo. Os resultados podem ser conferidos na Tabela 7.

Tabela 7 – Resultados para o Conjunto G

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
G1	11609	119	8842	169	561
G2	11612	134	8606	179	619
G3	11614	168	8763	121	477
G4	11638	165	8669	165	353
G5	11624	233	8969	122	407
G6	2176	-22	289	-25	-72
G7	2006	-30	311	-30	-35
G8	2004	-22	56	-19	19
G9	2047	-41	340	-41	-4
G10	1998	-13	105	-17	-10
G11	564	0	64	0	-8
G12	556	-4	-6	0	2
G13	582	-6	36	-6	-8
G14	3054	0	1866	24	660
G15	3044	0	2194	0	715
G16	3043	0	2027	0	691
G17	3042	0	1928	0	740
G18	988	-8	-70	-62	19
G19	906	-11	-27	-29	15
G20	940	6	37	-33	-5
G21	929	-59	-73	-78	-2
G22	13338	83	9596	61	148
G23	13287	113	9696	52	125
G24	13306	56	9654	38	164
G25	13288	79	9680	71	390
G26	13264	71	9674	16	219
G27	3304	-41	-23	-39	-21
G28	3253	-8	-449	-8	-6
G29	3352	-8	-164	-12	2
G30	3363	-12	317	-12	-20
G31	3269	-15	-601	-15	-11
G32	1386	-2	-70	0	6
G33	1368	-4	-100	-4	12
G34	1376	-10	-102	-10	-2
G35	7651	0	5529	0	925
G36	7641	166	5408	0	1379
G37	7660	336	5630	0	1503
G38	7646	0	5352	0	884
G39	2381	-48	17	-121	-56
G40	2373	-6	5	-40	2
G41	2386	-25	-89	-52	37
G42	2453	3	65	-74	-48
G43	6654	51	4599	64	122
G44	6649	84	4839	67	129
G45	6642	49	4607	57	142
G46	6643	52	4500	32	218
G47	6641	113	4526	44	178
G48	6000	8	1912	4	28
G49	6000	16	2082	8	38
G50	5880	12	1706	4	36
G51	3838	0	2712	0	739
G52	3841	0	2662	0	906
G53	3838	0	2627	0	730
G54	3837	0	2541	0	847

Fonte: elaborada pelo autor.

7 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foram exploradas aplicações de algoritmos de aprendizado de máquina supervisionado para a criação de modelos treinados a partir de instâncias de tamanhos pequenos e intermediários, com o objetivo de solucionar o Problema do Corte Máximo. Esse problema consiste em encontrar um subconjunto S de vértices, de tal forma que a soma das arestas com exatamente um vértice em S seja a maior possível. O modelo foi treinado com instâncias previamente solucionadas pelo *Solver Biq Mac* e utilizado para realizar a partição dos vértices de instâncias não observadas.

A análise revelou uma relação importante entre as propriedades estruturais dos grafos e o desempenho dos modelos supervisionados tradicionais na solução desse problema. Mais especificamente, as instâncias com maior coeficiente de aglomeração e centralidade de grau, além da centralidade de intermediação próxima de zero, da coleção de instâncias da *Biq Mac Library*, demonstraram um desempenho superior em termos de valor de corte obtido pelo modelo. Isso sugere que essas características estruturais são importantes para um treinamento eficiente dos modelos.

O uso de uma base de dados de treinamento composta por instâncias selecionadas de forma aleatória evidenciou que, assim como em bases de dados tradicionais, é necessário que os dados não apresentem grandes disparidades em suas características estruturais. Neste caso, a estrutura dos grafos, como coeficiente de aglomeração e grau de centralidade desempenha esse papel. Portanto, utilizar o coeficiente de aglomeração como critério para agrupar as instâncias, selecionando apenas aquelas cujos valores são positivos e distantes de zero, foi um fator determinante no treinamento do modelo, gerando bons resultados, estando próximos aos fornecidos pelo *Solver Biq Mac*.

Um ponto importante foi a baixa acurácia observada em algumas instâncias. Embora isso impacte negativamente as métricas tradicionais de avaliação de modelos preditivos, não comprometeu o valor de corte, sugerindo que o modelo conseguiu aprender uma lógica eficiente para o particionamento dos vértices. Por fim, os resultados obtidos ao treinar o modelo com instâncias menores e aplicá-lo nas instâncias do conjunto G indicam que, dependendo das amostras de treinamento, há potencial para que o modelo alcance melhores desempenhos no futuro.

7.1 Trabalhos futuros

Para trabalhos futuros, espera-se expandir o conjunto de instâncias utilizadas para o treinamento e teste, abrangendo grafos de diferentes tamanhos e complexidades, além daqueles observados na *Biq Mac Library*. Esse agrupamento de novas instâncias visa melhorar a qualidade do treinamento dos modelos de aprendizado de máquina, trazendo grafos com estruturas mais variadas. O objetivo é particionar a coleção de instâncias com base em múltiplas características estruturais, como coeficiente de aglomeração, centralidade de grau e conectividade, com especial atenção para grafos com coeficientes de aglomeração positivos.

Além disso, pretende-se explorar outras formas de representar os grafos nas bases de dados. Uma abordagem promissora é o uso de *graph embeddings* gerados por redes neurais, que capturam as estruturas dos grafos com base nas características tanto dos vértices quanto das arestas. Além disso, é interessante explorar outros algoritmos de aprendizado de máquina, como Random Forest e Máquina de Vetores de Suporte (SVM) para avaliar o comportamento desses modelos em relação aqueles observados nesse trabalho.

Por fim, busca-se desenvolver um método para selecionar amostras de treinamento ideais, visando otimizar o desempenho dos modelos de aprendizado supervisionado na solução do Problema do Corte Máximo.

REFERÊNCIAS

AGGARWAL, C. C. **Data Classification: Algorithms and Applications**. 1. ed. [*S. l.*]: Chapman and Hall/CRC, 2014. (Chapman and Hall/CRC Data Mining and Knowledge Discovery Series). ISBN 978-1-4665-8675-8,978-1-4665-8674-1.

AGRAWAL, R.; RAJAGOPALAN, S.; SRIKANT, R.; XU, Y. Mining newsgroups using networks arising from social behavior. In: **Proceedings of the 12th international conference on World Wide Web**. [*S. l.*: *s. n.*], 2003. p. 529–535.

AHMED, M.; SERAJ, R.; ISLAM, S. M. S. The k-means algorithm: A comprehensive survey and performance evaluation. **Electronics**, MDPI, v. 9, n. 8, p. 1295, 2020.

ARORA SANJEEV;BARAK, B. **Computational complexity A Modern Approach**. 4th printing. ed. [*S. l.*]: Cambridge University Press, 2016. ISBN 9780521424264,9780511530753,0521424267.

ARTHUR, D.; VASSILVITSKII, S. **k-means++: The advantages of careful seeding**. [*S. l.*], 2006.

BARABáSI, A.-L. **Network Science**. [*S. l.*]: Cambridge University Press, 2016.

BATISTA, G. E. d. A. P. **Pré-processamento de dados em aprendizado de máquina supervisionado**. Tese (Doutorado) – Universidade de São Paulo, 2003.

BAZGAN, C.; ESCOFFIER, B.; PASCHOS, V. T. Completeness in standard and differential approximation classes: Poly-(d) apx-and (d) ptas-completeness. **Theoretical Computer Science**, Elsevier, v. 339, n. 2-3, p. 272–292, 2005.

BEWICK, V.; CHEEK, L.; BALL, J. Statistics review 14: Logistic regression. **Critical care**, Springer, v. 9, p. 1–7, 2005.

BISHOP, C. M.; NASRABADI, N. M. **Pattern recognition and machine learning**. [*S. l.*]: Springer, 2006. v. 4.

BLUM, C.; ROLI, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 35, n. 3, p. 268–308, 2003.

BOROS, E.; HAMMER, P. L. The max-cut problem and quadratic 0–1 optimization; polyhedral aspects, relaxations and bounds. **Annals of Operations Research**, Springer, v. 33, n. 3, p. 151–180, 1991.

COOK, S. The p versus np problem. **Clay Mathematics Institute**, v. 2, 2000.

COOK, S. A.; MITCHELL, D. G. Finding hard instances of the satisfiability problem: A survey. **Satisfiability Problem: Theory and Applications**, v. 35, p. 1–17, 1996.

CORMEN, T. H.; RIVEST, R. L.; STEIN, C.; LEISERSON, C. E. **Algoritmos - Teoria e Prática - 3ª Ed.** [*S. l.*]: Elsevier, 2012. ISBN 9788535236996.

DIESTEL, R. **Graph Theory**. 4th. ed. [*S. l.*]: Springer, 2010. (Graduate Texts in Mathematics, 173). ISBN 3642142788,9783642142789.

DIESTEL, R. **Graph Theory**. [*S. l.*]: Springer, 2017.

FACELI, K.; LORENA, A. C.; GAMA, J.; ALMEIDA, T. A. d.; CARVALHO, A. C. P. d. L. F. d. Inteligência artificial: uma abordagem de aprendizado de máquina. Rio de Janeiro: LTC, p. 378, 2011.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. First edition. [*S. l.*]: W. H. Freeman, 1979. (Series of Books in the Mathematical Sciences). ISBN 0716710455,9780716710455.

GÉRON, A. **Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow**. Second. Sebastopol, CA: O'Reilly Media, Inc., 2019. ISBN 9781492032649.

GIANINAZZI, L.; FRIES, M.; DRYDEN, N.; BEN-NUN, T.; BESTA, M.; HOEFLER, T. Learning combinatorial node labeling algorithms. **arXiv preprint arXiv:2106.03594**, 2021.

GOEMANS, M. X.; WILLIAMSON, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 42, n. 6, p. 1115–1145, 1995.

GOODRICH, M. T.; TAMASSIA, R. **Projeto de algoritmos: fundamentos, análise e exemplos da internet**. [*S. l.*]: Bookman Editora, 2009.

GOUDET, O.; GRELIER, C.; HAO, J.-K. A deep learning guided memetic framework for graph coloring problems. **Knowledge-Based Systems**, Elsevier, v. 258, p. 109986, 2022.

GROSS JAY YELLEN, P. Z. J. L. **Handbook of Graph Theory, Second Edition**. 2. ed. [*S. l.*]: Chapman and Hall/CRC, 2013. (Discrete Mathematics and Its Applications). ISBN 1439880182,9781439880180.

GRUS, J. **Data Science do Zero**. 1. ed. Rio de Janeiro: Alta Books, 2016. 336 p. ISBN 978-85-508-0387-6.

GU, S.; YANG, Y. A pointer network based deep learning algorithm for the max-cut problem. In: SPRINGER. **Neural Information Processing: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13-16, 2018, Proceedings, Part I 25**. [*S. l.*], 2018. p. 238–248.

GU, S.; YANG, Y. A deep learning algorithm for the max-cut problem based on pointer network structure with supervised learning and reinforcement learning strategies. **Mathematics**, MDPI, v. 8, n. 2, p. 298, 2020.

HELMBERG, C.; RENDL, F. A spectral bundle method for semidefinite programming. **SIAM Journal on Optimization**, SIAM, v. 10, n. 3, p. 673–696, 2000.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. **ACM computing surveys (CSUR)**, Acm New York, NY, USA, v. 31, n. 3, p. 264–323, 1999.

JOLLIFFE, I. T. **Principal component analysis for special types of data**. [*S. l.*]: Springer, 2002.

JOLLIFFE, I. T.; CADIMA, J. Principal component analysis: a review and recent developments. **Philosophical transactions of the royal society A: Mathematical, Physical and Engineering Sciences**, the Royal Society publishing, v. 374, n. 2065, p. 20150202, 2016.

KARP, R. M. **Reducibility among combinatorial problems. Complexity of Computer Computations (The IBM Research Symposia Series)**. [*S. l.*]: Springer, New York, 1972.

KETCHEN, D. J.; SHOOK, C. L. The application of cluster analysis in strategic management research: an analysis and critique. **Strategic management journal**, Wiley Online Library, v. 17, n. 6, p. 441–458, 1996.

KOPILER, A. A.; SILVA, V.; OLIVEIRA, L. A. A. d.; LINDEN, R.; SILVA, L.; FONSECA, B. L. d. C. Redes neurais artificiais e suas aplicações no setor elétrico. **Revista de Engenharias da Faculdade Salesiana**, v. 1, n. 9, p. 27–33, 2019.

LATORA, V.; NICOSIA, V.; RUSSO, G. **Complex Networks: Principles, Methods and Applications**. Cambridge: Cambridge University Press, 2017.

LIBBRECHT, M. W.; NOBLE, W. S. Machine learning applications in genetics and genomics. **Nature Reviews Genetics**, Nature Publishing Group UK London, v. 16, n. 6, p. 321–332, 2015.

LIERS, F.; NIEBERG, T.; PARDELLA, G. Via minimization in vlsi chip design-application of a planar max-cut algorithm. Universität zu Köln, 2011.

LIN, M.; XU, W.; LIN, Z.; CHEN, R. Determine owa operator weights using kernel density estimation. **Economic research-Ekonomska istraživanja**, Taylor and Francis Group, v. 33, n. 1, p. 1441–1464, 2020.

MACQUEEN, J. *et al.* Some methods for classification and analysis of multivariate observations. In: OAKLAND, CA, USA. **Proceedings of the fifth Berkeley symposium on mathematical statistics and probability**. [*S. l.*], 1967. v. 1, n. 14, p. 281–297.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, p. 115–133, 1943.

MIKI, S.; YAMAMOTO, D.; EBARA, H. Applying deep learning and reinforcement learning to traveling salesman problem. In: IEEE. **2018 international conference on computing, electronics & communications engineering (ICCECE)**. [*S. l.*], 2018. p. 65–70.

MIYAZAWA, F. K.; SOUZA, C. C. de. Introdução à otimização combinatória. **Sociedade Brasileira de Computação**, 2015.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

NELLI, F. **Python Data Analytics: With Pandas, NumPy, and Matplotlib**. 2. ed. [*S. l.*]: Apress, 2018. ISBN 1484239121, 978-1484239124.

NETO, I. A. de Q. Aplicação do método de região de confiança ao algoritmo backpropagation. 1995.

OLIVEIRA, M. C.; FERREIRA, L. V. B.; BARREIROS, M. de O. Classificação de doenças cardiovasculares utilizando aprendizado de máquina. **Multidebates**, v. 7, n. 1, p. 38–44, 2023.

OTTERBACH, J. S.; MANENTI, R.; ALIDOUST, N.; BESTWICK, A.; BLOCK, M.; BLOOM, B.; CALDWELL, S.; DIDIER, N.; FRIED, E. S.; HONG, S. *et al.* Unsupervised machine learning on a hybrid quantum computer. **arXiv preprint arXiv:1712.05771**, 2017.

PARSIAN, M. **Data Algorithms: Recipes for Scaling Up with Hadoop and Spark**. [*S. l.*]: O'Reilly Media, 2015. ISBN 1491906189,9781491906187.

RASCHKA, S. **Python Machine Learning**. Birmingham - Mumbai: Packt Publishing, 2015. ISBN 978-1-78355-513-0.

RAUBER, T. W. Redes neurais artificiais. **Universidade Federal do Espírito Santo**, v. 29, 2005.

RENDL, F.; RINALDI, G.; WIEGELE, A. Solving Max-Cut to optimality by intersecting semidefinite and polyhedral relaxations. **Math. Programming**, v. 121, n. 2, p. 307, 2010.

RISH, I. *et al.* An empirical study of the naive bayes classifier. In: **IJCAI 2001 workshop on empirical methods in artificial intelligence**. [*S. l.*: *s. n.*], 2001. v. 3, n. 22, p. 41–46.

ROUSSEEUW, P. J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. **Journal of computational and applied mathematics**, Elsevier, v. 20, p. 53–65, 1987.

SEYEDZADEH, S.; RAHIMIAN, F. P.; RASTOGI, P.; GLESK, I. Tuning machine learning models for prediction of building energy loads. **Sustainable Cities and Society**, Elsevier, v. 47, p. 101484, 2019.

SHLENS, J. A tutorial on principal component analysis. **arXiv preprint arXiv:1404.1100**, 2014.

SINAI, Y. G.; SINAI, Y. G. Conditional probability and independence. **Probability Theory: An Introductory Course**, Springer, p. 43–53, 1992.

SIPSER, M. **Introduction to the Theory of Computation**. 3. ed. [*S. l.*]: Course Technology, 2012. ISBN 113318779X,9781133187790.

SOARES, P. L. B. Problemas quadráticos binários: abordagem teórica e computacional. 2018.

SOUSA, S. de; HAXHIMUSA, Y.; KROPATSCH, W. G. Estimation of distribution algorithm for the max-cut problem. In: SPRINGER. **Graph-Based Representations in Pattern Recognition: 9th IAPR-TC-15 International Workshop, GbRPR 2013, Vienna, Austria, May 15-17, 2013. Proceedings 9**. [*S. l.*], 2013. p. 244–253.

TEODORO, L. de A.; KAPPEL, M. A. A. Aplicação de técnicas de aprendizado de máquina para predição de risco de evasão escolar em instituições públicas de ensino superior no brasil. **Revista Brasileira de Informática na Educação**, v. 28, p. 838–863, 2020.

TRENDAFILOV, N.; GALLO, M. **Multivariate data analysis on matrix manifolds**. [*S. l.*]: Springer, 2021.

TRUDEAU, R. J. **Introduction to Graph Theory**. 2. ed. [*S. l.*]: Dover Publications, 1994. (Dover Books on Mathematics). ISBN 0486678709,9780486678702.

WIEGELE, A. Biq mac library—a collection of max-cut and quadratic 0-1 programming instances of medium size. **Preprint**, v. 51, 2007.

WILLIAMS, J.; LI, Y. Comparative study of distance functions for nearest neighbors. In: . [*S. l.*: *s. n.*], 2008. p. 79–84. ISBN 978-90-481-3659-9.

XIONG, L.; YAO, Y. Study on an adaptive thermal comfort model with k-nearest-neighbors (knn) algorithm. **Building and Environment**, Elsevier, v. 202, p. 108026, 2021.

YAO, W.; BANDEIRA, A. S.; VILLAR, S. Experimental performance of graph neural networks on random instances of max-cut. In: SPIE. **Wavelets and Sparsity XVIII**. [*S. l.*], 2019. v. 11138, p. 242–251.

APÊNDICE A – RESULTADOS PARA O TREINAMENTO COM AMOSTRAS ALEATÓRIAS

Tabela 8 – Resultados para o treinamento com amostras aleatórias - Parte I

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
gka1b	5744	2511	4080	4611	4728
gka2b	12451	5370	9927	8052	11265
gka3b	22115	6855	17099	9023	18824
gka4b	34857	6613	30888	15636	22967
gka5b	49942	10829	45506	12950	31168
gka6b	68189	15627	61157	18069	43732
gka7b	87428	14477	79095	21019	51731
gka8b	109969	8125	100913	31543	43661
gka9b	135757	14150	125546	39740	53472
gka10b	209946	23623	193989	44435	63274

Fonte: elaborada pelo autor.

Tabela 9 – Resultados para o treinamento com amostras aleatórias - Parte II

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
pw05_100.0	8190	2478	5868	1156	799
pw05_100.1	8045	1792	6181	474	1086
pw05_100.2	8039	2303	5967	952	1074
pw05_100.3	8139	1873	6106	558	1044
pw05_100.4	8125	1588	6196	701	766
pw05_100.5	8169	1803	6442	642	1768
pw05_100.6	8217	2063	6322	470	1920
pw05_100.7	8249	1666	5719	806	1370
pw05_100.8	8199	2110	6485	973	1325
pw09_100.0	13585	2669	11330	938	933
pw09_100.1	13417	2985	10658	1296	2376
pw09_100.2	13461	3502	11339	454	1418
pw09_100.3	13656	3977	10973	2271	2430
pw09_100.4	13514	1723	11108	845	876
pw09_100.5	13574	2157	10735	864	3247
pw09_100.6	13640	4244	11505	1402	1991
pw09_100.7	13501	4926	10972	1366	526
pw09_100.8	13593	1365	10562	901	1859
pw09_100.9	13658	3502	9695	1778	2423

Fonte: elaborada pelo autor.

Tabela 10 – Resultados para o treinamento com amostras aleatórias - Parte III

Instância de Teste	Solver	Naive Bayes	kNN	Regressão Logística	RNA
g05_100.0	1430	293	1048	96	233
g05_100.1	1425	353	1138	138	97
g05_100.2	1432	255	1237	0	268
g05_100.3	1424	308	1164	149	50
g05_100.4	1440	345	898	53	170
g05_100.5	1436	335	1043	0	174
g05_100.6	1434	349	1125	134	239
g05_100.7	1431	270	1180	52	199
g05_100.8	1432	192	1085	144	271
g05_100.9	1430	260	1135	125	149
g05_60.1	532	150	433	64	105
g05_60.2	529	94	430	30	209
g05_60.3	538	146	432	29	51
g05_60.4	527	125	399	27	108
g05_60.5	533	214	387	77	104
g05_60.6	531	227	420	24	30
g05_60.7	535	106	428	0	107
g05_60.8	530	248	422	112	52
g05_60.9	533	166	388	117	81
g05_80.0	929	210	744	37	73
g05_80.1	941	235	702	116	109
g05_80.2	934	206	716	105	170
g05_80.3	923	249	772	76	223
g05_80.4	932	206	740	67	112
g05_80.5	926	172	710	76	86
g05_80.6	929	231	766	72	35
g05_80.7	929	144	793	79	215
g05_80.8	925	205	676	71	221
g05_80.9	923	105	724	34	182

Fonte: elaborada pelo autor.