UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS RUSSAS

CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS VINICIO DANTAS ARAÚJO

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DA DIVERSIDADE MÁXIMA EM GRAFOS

RUSSAS

2025

CARLOS VINICIO DANTAS ARAÚJO

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DA DIVERSIDADE MÁXIMA 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

2025

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)

A688a Araújo, Carlos Vinicio Dantas.
Aplicação de Aprendizado de Máquina Supervisionado para o Problema da Diversidade Máxima em Grafos / Carlos Vinicio Dantas Araújo. – 2025.
72 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 da Diversidade Máxima. 2. Aprendizado de Máquina. 3. Aprendizado Supervisionado. I. Título.

CDD 005

CARLOS VINICIO DANTAS ARAÚJO

APLICAÇÃO DE APRENDIZADO DE MÁQUINA SUPERVISIONADO PARA O PROBLEMA DA DIVERSIDADE MÁXIMA 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: 06/03/2025

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. Cenez Araújo de Rezende
Universidade Federal do Ceará (UFC)

Dedico este trabalho a minha mãe, Veralucia Dantas, ao meu pai, Carlos Gonzaga, ao meu irmão, Carlos Victor e a minha irmã, Carla Victória.

AGRADECIMENTOS

À minha mãe, Veralucia Dantas, que esteve comigo, me ouvindo e apoiando em cada momento da minha vida; ao meu pai, Carlos Gonzaga, que sempre foi exemplo e me mostrou o caminho a seguir; e por fim, ao meu irmão, Carlos Victor, que a vida inteira vem puxando minha orelha e me dando conselhos. Além disso, não consigo agradecer o suficiente por tudo que fez por mim desde que entrei na faculdade. Obrigado, meu irmão.

Ao meu grande amor, Virna Diolino, que apareceu na minha vida sem nenhuma explicação, e agora, com menos explicação ainda, não consigo me ver sem você ao meu lado para o restante de nossas vidas. Amo você mais do que pode imaginar ❤.

Aos meus amigos, em especial Pedro Epifânio, que esteve comigo desde o ensino médio e me acompanhou de perto na vida universitária, me ajudando nos maus momentos e me fazendo rir nos bons. Sempre vou ser grato por tudo que faz por mim, meu amigo; William Bruno, que esteve comigo em todos os sufocos que passamos nesta graduação. Desde o primeiro momento em que tivemos nossa primeira reunião da monitoria, eu sabia que você iria se tornar um amigo para a vida toda; Marcello Ricardo, que sempre escutou e riu dos meus problemas, tornando tudo mais leve. As histórias das barbaridades da sua vida também me rendem boas risadas; e por último, mas não menos importante, Henrique Martins, que esteve ao meu lado desde a primeira semana da faculdade, ainda durante a pandemia, e me deu força para enfrentar aquele começo tão assustador. Se não fosse por cada um de vocês, eu não estaria aqui hoje. Muito obrigado!

Ao Prof. Dr. Pablo Soares, por além de me guiar neste trabalho de conclusão de curso, ser uma figura que me motiva e apoia desde o meu início na faculdade.

Aos professores participantes da banca examinadora, Prof. Dr. Cenez de Rezende e Profa. Dra. Tatiane Figueiredo, pelo tempo dedicado e pelas valiosas colaborações e sugestões.

“Percebi como agimos sob a influência do medo e da desconfiança, em vez de nos deixarmos guiar pelo amor que tantas vezes reprimimos."
(A. G. Roemmers: O retorno do jovem príncipe, 2011, p. 50)

RESUMO

O Problema da Diversidade Máxima em Grafos, classificado como NP-difícil, busca selecionar subconjuntos de vértices que maximizem a soma das distâncias entre eles, representando um desafio computacional significativo para instâncias de grande porte. Este trabalho propõe a aplicação de técnicas de aprendizado de máquina supervisionado para otimizar a alocação de vértices em grafos, utilizando rotulagens geradas a partir de algoritmos exatos e metaheurísticas para criar uma base de dados de treinamento. A metodologia inclui a modelagem de grafos, a extração de características relevantes e o treinamento de modelos supervisionados. Os resultados indicam que os modelos supervisionados alcançaram uma proximidade de 91% a 95% em relação aos resultados obtidos pela metaheurística, além de demonstrar boa capacidade de generalização. Conclui-se que o uso de aprendizado supervisionado pode ser uma estratégia eficiente para resolver o Problema da Diversidade Máxima, especialmente quando combinado com metaheurísticas para gerar rótulos em instâncias de grande porte a serem usadas no treinamento, tornando os modelos aplicáveis em contextos complexos e de maior escala.

Palavras-chave: problema da diversidade máxima; aprendizado de máquina; aprendizado supervisionado.

ABSTRACT

The Maximum Diversity Problem in Graphs, classified as NP-hard, seeks to select subsets of vertices that maximize the sum of the distances between them, representing a significant computational challenge for large-scale instances. This work proposes the application of supervised machine learning techniques to optimize vertex allocation in graphs, using labels generated from exact algorithms and metaheuristics to create a training dataset. The methodology includes graph modeling, extraction of relevant features, and training of supervised models. The results indicate that the supervised models achieved a proximity of 91% to 95% compared to the results obtained by the metaheuristic, while also demonstrating good generalization capacity. It is concluded that supervised learning can be an efficient strategy for solving the Maximum Diversity Problem, especially when combined with metaheuristics to generate labels for large-scale instances to be used in training, making the models applicable in complex and large-scale contexts.

Keywords: maximum diversity problem; machine learning; supervised learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ANN	Artificial Neural Networks
DT	Decision Trees
FN	False Negatives
FNN	Feed-forward Neural Network
FP	False Positives
KNN	K-Nearest Neighbors
LR	Logistic Regression
MDP	Maximum Diversity Problem
ML	Machine Learning
PCA	Principal Component Analysis
ReLU	Rectified Linear Unit
RF	Random Forest
SL	Supervised Learning
TANH	Hyperbolic Tangent
TN	True Negatives
TP	True Positives
TSP	Travelling Salesman Problem

SUMÁRIO

1 INTRODUÇÃO

Na área da Teoria dos Grafos, diversos problemas combinatórios são classificados como NP-difíceis, ou seja, problemas cuja solução exata é inviável em grandes instâncias devido ao crescimento exponencial do tempo computacional necessário (Garey; Johnson, 1979). Entre esses problemas, se destaca o Problema da Diversidade Máxima, do inglês Maximum Diversity Problem (MDP), que busca selecionar subconjuntos de vértices em um grafo de modo a maximizar a soma das distâncias entre eles (Gonzalez *et al.*, 2020).

O MDP é formalmente definido no contexto de grafos da seguinte forma: dado um grafo $G = (V, E)$, onde V representa o conjunto de vértices e E o conjunto de arestas, o objetivo é selecionar um subconjunto $S \subseteq V$ que maximize a soma das distâncias entre os vértices selecionados. Esse problema, por ser NP-difícil, torna inviável o uso de algoritmos exatos para instâncias de grande escala, uma vez que o tempo computacional aumenta exponencialmente com o tamanho do grafo. Isto impõe a necessidade de abordagens aproximadas, como heurísticas e metaheurísticas, para resolver o problema em tempo aceitável, embora com alguma perda de precisão (Blum; Roli, 2003).

A resolução de forma eficiente do MDP tem implicações práticas significativas. Por exemplo, em bioinformática, a escolha de subconjuntos diversos de dados pode acelerar a descoberta científica e melhorar a qualidade dos resultados experimentais (Porter *et al.*, 1975). Mais recentemente, o MDP teve aplicação na otimização de cadeias de suprimento (Parreno *et al.*, 2024). Dessa forma, encontrar soluções para o MDP não é apenas um desafio teórico, mas também um fator importante para o avanço de várias áreas de pesquisa e desenvolvimento operacional.

Neste contexto, o Aprendizado de Máquina, do inglês Machine Learning (ML) surge como uma ferramenta promissora para resolver o MDP de maneira eficiente. Técnicas de ML, tanto supervisionadas quanto não supervisionadas, se destacam pela capacidade de identificar padrões complexos em grandes volumes de dados e gerar soluções aproximadas em tempo computacional reduzido (Faceli *et al.*, 2011).

Diferente de outras abordagens presentes na literatura, este estudo inova ao aplicar técnicas de aprendizado de máquina especificamente ao MDP. Embora o aprendizado de máquina tenha sido utilizado para resolver outros problemas combinatórios, como o Problema do Caixeiro Viajante (TSP) (Shi; Zhang, 2022) e o Problema do Corte Máximo (Gu; Yang, 2020), sua aplicação ao MDP é uma área ainda pouco explorada. Este trabalho busca preencher essa lacuna,

investigando como diferentes modelos de aprendizado supervisionado podem ser utilizados para resolver instâncias do MDP, com o objetivo de melhorar a eficiência computacional e a qualidade das soluções, especialmente em grafos de grande escala.

O restante deste trabalho está organizado da seguinte maneira: na Seção 2, são apresentados os objetivos gerais e específicos a serem realizados. Na Seção 3, são discutidos os conceitos da Teoria dos Grafos e as técnicas de Aprendizado de Máquina aplicadas. A Seção 4 apresenta uma análise de estudos relevantes sobre problemas NP-Difíceis e Aprendizado de Máquina. A Seção 5 detalha a metodologia adotada, enquanto a Seção 6 apresenta os resultados obtidos e a análise do desempenho dos modelos. Por fim, a Seção 7 discute as conclusões e sugere possíveis trabalhos futuros, como o aprimoramento dos modelos e a aplicação em contextos mais complexos.

2 OBJETIVOS

Nesta seção, será discutido o principal objetivo a ser alcançado com este trabalho, junto aos objetivos específicos que auxiliam no processo para a obtenção dos resultados desejados.

2.1 Objetivos Gerais

O objetivo geral é aplicar modelos de aprendizado de máquina ao Problema da Diversidade Máxima em grafos, utilizando instâncias de pequeno porte para ajustar os modelos supervisionados, visando sua aplicação em instâncias de grande porte.

2.2 Objetivos Específicos

- Adquirir conjuntos de dados pertinentes ao Problema da Diversidade Máxima;
- Conseguir soluções ótimas e/ou aproximadas de instâncias para montagem da base de dados;
- Extrair e selecionar características dos grafos relevantes para a predição das partições que maximizam a diversidade;
- Escolher algoritmos de Aprendizado de Máquina Supervisionado e Não Supervisionado;
- Aplicar e ajustar os modelos de aprendizado supervisionado em instâncias de pequeno e médio porte do Problema da Diversidade Máxima.
- Treinar e validar os modelos supervisionados usando a base de dados dividida em treinamento e teste, respectivamente;
- Avaliar os modelos em termos de precisão na solução para o Problema da Diversidade Máxima;
- Aplicar os modelos treinados em instâncias novas de grande porte.

3 FUNDAMENTAÇÃO TEÓRICA

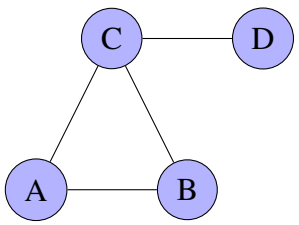
Nesta seção, serão apresentados os conceitos fundamentais que embasam este trabalho, organizados em três grandes áreas. Primeiramente, discutiremos os conceitos essenciais da Teoria dos Grafos na Seção 3.1, abordando a representação e propriedades dos grafos. Em seguida, será apresentada a formulação do Problema da Diversidade Máxima na Seção 3.2, que é o foco central deste trabalho. Por fim, serão discutidos os principais métodos de aprendizado de máquina na Seção 3.3, aplicados para a resolução do problema proposto, abordando os algoritmos supervisionados escolhidos, além de explicar suas características, vantagens e limitações em contextos de grafos.

3.1 Conceitos Necessários de Teoria dos Grafos

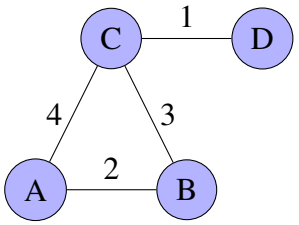
Um grafo simples é representado por $G = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas que conectam pares de vértices. O número de vértices $|V|$ em um grafo representa a quantidade total de pontos de conexão, enquanto o número de arestas $|E|$ indica as ligações entre esses pontos (Diestel, 2017). O grau de um vértice v, denotado por $d(v)$, é o número de arestas que incidem sobre ele (Shrimali; Singh, 2020). Em grafos ponderados, cada aresta $\{u, v\} \in E$ pode ter um peso $w(\{u, v\})$ associado, que representa uma medida de custo ou distância (Diestel, 2017).

Em grafos simples, não há laços (arestas que conectam um vértice a ele mesmo) nem arestas múltiplas entre os mesmos pares de vértices. A Figura 1 ilustra um exemplo de grafo simples, onde as arestas conectam pares de vértices sem direção definida e podem ser ponderadas, com os valores associados às arestas indicando o peso $w(\{u, v\})$.

Figura 1 – Exemplo de um grafo simples não direcionado e um dígrafo simples ponderado



(a) Grafo Simples Não Direcionado

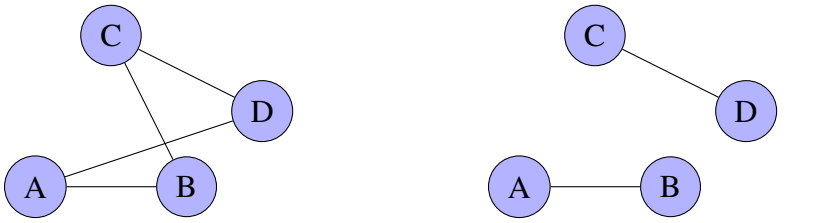


(b) Dígrafo Simples Não Direcionado e Ponderado

Fonte: elaborada pelo autor.

A conectividade de um grafo é o conceito que define a existência de caminhos entre os vértices. Um subgrafo $G' = (V', E')$ é definido por subconjuntos $V' \subseteq V$ e $E' \subseteq E$, onde G' mantém a estrutura do grafo original G. Um grafo é considerado conexo se houver um caminho entre quaisquer dois vértices. Caso contrário, ele pode ser decomposto em componentes conexas, que são subgrafos maximamente conexos (Diestel, 2017). A Figura 2 ilustra a diferença entre um grafo conexo e um grafo desconexo, que pode ser dividido em componentes conexas.

Figura 2 – Exemplos de conectividade de grafos



(a) Grafo Conexo

(b) Grafo Desconexo com duas componentes

Fonte: elaborada pelo autor.

3.2 O Problema da Diversidade Máxima

O Problema da Diversidade Máxima (MDP) é uma formulação combinatória que busca selecionar um subconjunto de vértices de modo a maximizar a diversidade entre eles. Dado um grafo $G = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas, o objetivo do MDP é encontrar um subconjunto $S \subseteq V$ de tamanho k que maximize a soma das diferenças entre os vértices em S. Essas diferenças são geralmente medidas pela quantidade de arestas que conectam dois vértices, ou, em grafos ponderados, pelos pesos das arestas, de modo a maximizar a separação entre os vértices selecionados (Gonzalez *et al.*, 2020; Duarte *et al.*, 2015; Carrabs *et al.*, 2022).

Formalmente, o problema pode ser expresso da seguinte forma:

$$\max_{S \subseteq V, |S|=k} \sum_{v_i, v_j \in S} d(v_i, v_j) \tag{3.1}$$

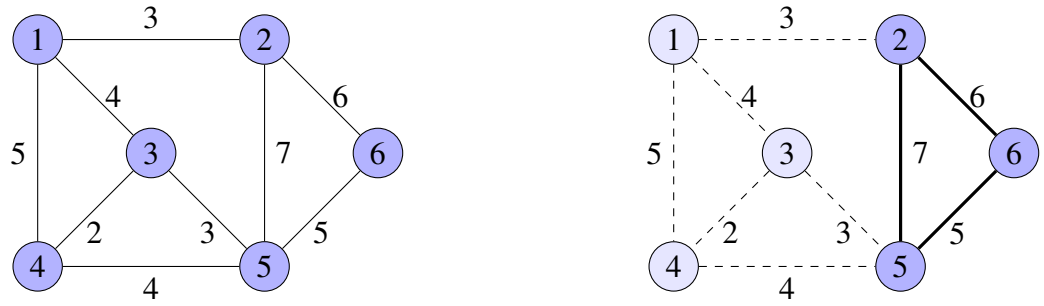
onde $d(v_i, v_j)$ representa a medida de diferença entre os vértices v_i e v_j. O objetivo é maximizar a soma dessas medidas entre os vértices selecionados, caracterizando o MDP como um problema NP-difícil, o que significa que não há um algoritmo eficiente conhecido para

resolvê-lo exatamente em tempo polinomial, especialmente em grafos de grande escala (Lopez *et al.*, 2019). Por isso, técnicas heurísticas e metaheurísticas são usadas com frequência para encontrar soluções aproximadas.

Na literatura, é comum encontrar instâncias do MDP em que o valor de k é previamente definido como uma proporção do número total de vértices n. Por exemplo, para grafos com $n = 100$ vértices, k pode ser definido como $n/10$, ou seja, a seleção seria feita para um subconjunto de 10 vértices. Essa definição de k pode variar conforme o problema e a estrutura do grafo, mas a ideia principal é balancear a quantidade de vértices selecionados e a maximização da diversidade.

Para ilustrar o processo de maximização da diversidade, a Figura 3 apresenta um grafo com seis vértices e arestas ponderadas, onde o objetivo é selecionar um subconjunto S de vértices que maximize a soma das distâncias entre eles.

Figura 3 – Demonstração da seleção do conjunto com diversidade máxima



(a) Grafo original com vértices e arestas ponderadas

(b) Subconjunto $S = \{2,5,6\}$ maximizando a diversidade

Fonte: elaborada pelo autor.

Na Figura 3 (a), temos o grafo original, onde cada aresta tem um peso associado, representando a distância entre dois vértices. Já na Figura 3 (b), o subconjunto $S = \{2,5,6\}$ foi selecionado por maximizar a soma das distâncias entre os vértices, totalizando $7+5+6 = 18$. Esse subconjunto foi escolhido porque as arestas que conectam os vértices 2, 5 e 6 apresentam os maiores pesos do grafo, o que indica que a distância entre esses vértices é grande, maximizando assim a diversidade no conjunto.

A escolha de apenas três vértices ocorreu de forma arbitrária, porém, um fator a ser comentado é que, ao incluir mais vértices, a soma das distâncias entre eles tende a diminuir ou a redundância aumenta, pois as conexões entre os novos vértices adicionados provavelmente serão

mais próximas (com arestas de menor peso). Para maximizar a diversidade, é optado por um subconjunto onde as distâncias entre os vértices sejam as maiores possíveis, sem sacrificar a separação entre eles. No caso deste grafo, a melhor solução é alcançada com três vértices.

Outra forma de modelar o MDP é utilizando uma formulação quadrática binária. Nesse caso, é definida uma variável binária x_i para cada vértice v_i no grafo, onde $x_i = 1$ indica que o vértice v_i foi selecionado para o subconjunto S, e $x_i = 0$ caso contrário. A função objetivo e as restrições do problema podem ser expressas como:

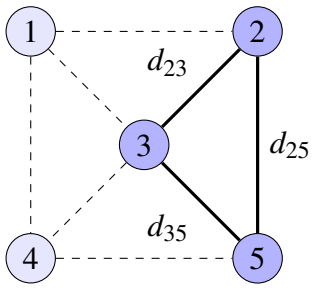
$$\max \quad \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{3.2}$$

$$s.a \quad \sum_{i=1}^{n} x_i = k, \tag{3.3}$$

$$x_i \in \{0,1\}, \quad 1 \leq i \leq n \tag{3.4}$$

A Figura 4 ilustra a seleção de vértices baseada na formulação apresentada. Nela, os vértices selecionados ($x_i = 1$) estão conectados por arestas com pesos d_{ij}. Especificamente, os vértices v_2, v_3 e v_5 foram escolhidos, formando as arestas d_{23}, d_{25} e d_{35}, cuja soma das distâncias compõe a função objetivo. A restrição $\sum_{i=1}^{n} x_i = k$ assegura que exatamente k vértices sejam selecionados.

Figura 4 – Exemplo de seleção de vértices $x_2 = 1, x_3 = 1, x_5 = 1$ com pesos d_{ij}



Fonte: elaborada pelo autor.

3.2.1 Aplicações do Problema da Diversidade Máxima

O MDP tem sido utilizado em várias áreas ao longo do tempo, o que evidencia a sua aplicabilidade em problemas práticos de diferentes campos:

- **Locação de Instalações**: aplicações iniciais de Erkut e Neuman (1991) exploraram o MDP para dispersar instalações de modo a evitar concentração excessiva em locais vulneráveis, visando maior eficiência operacional e redução de riscos.
- **Tratamentos Médicos**: Kuo *et al.* (1993) propuseram heurísticas para otimizar a seleção de tratamentos médicos, maximizando a diversidade de abordagens terapêuticas e aumentando as chances de sucesso no tratamento de doenças complexas.
- **Genética e Biotecnologia**: desde os trabalhos de Porter *et al.* (1975), o MDP tem sido utilizado para maximizar a diversidade genética em programas de melhoramento de plantas e animais, assegurando uma maior variabilidade genética nas populações selecionadas.
- **Design de Produtos**: no contexto de desenvolvimento de novos produtos, Ghyczy (1985), Besanko *et al.* (1990) utilizaram o MDP para maximizar a diversidade de portfólios de produtos, atendendo a múltiplos segmentos de mercado com uma gama diversificada de opções.

Em trabalhos mais recentes, o MDP teve aplicações na otimização de cadeias de suprimento, garantindo a diversidade na seleção de fornecedores e recursos de forma estratégica para aumento de produtividade operacional (Parreno *et al.*, 2024).

3.3 Aprendizado de Máquina

O Aprendizado de Máquina (ML) é um campo da inteligência artificial que permite a modelos aprenderem com dados, sem a necessidade de instruções programadas especificamente para cada tarefa. O objetivo do ML é desenvolver algoritmos que possam identificar padrões em dados e, com base neles, tomar decisões ou fazer previsões sobre novas informações. Essa abordagem permite que os modelos melhorem continuamente seu desempenho à medida que são expostos a mais dados (Faceli *et al.*, 2011).

Conforme a definição de Mitchell (1997): "Dizemos que um programa de computador aprende a partir da experiência E com respeito a uma classe de tarefas T e uma medida de desempenho P, se o desempenho em T, medido por P, melhora com a experiência E". Essa definição é o resumo do processo fundamental do aprendizado de máquina: a melhoria contínua na execução de uma tarefa com base em exemplos passados, utilizando dados como fonte de experiência.

O ML é guiado pela capacidade de representar dados de maneira eficaz. Esses dados podem ser numéricos, textuais, visuais ou até temporais, e precisam ser organizados em atributos

que descrevam as amostras de forma relevante para o problema em questão. A qualidade dos dados e sua representação influenciam diretamente a eficácia dos modelos (Domingos, 2012). Além disso, uma seleção adequada de características é essencial, pois características irrelevantes ou redundantes tendem a prejudicar a capacidade do modelo de generalizar e encontrar padrões úteis nos dados (Guyon; Elisseeff, 2003).

A generalização é uma das propriedades mais importantes em um modelo de aprendizado de máquina. Um modelo que generaliza bem é capaz de fazer previsões precisas em novos dados que não foram utilizados durante o treinamento. Quando um modelo é ajustado de forma exagerada aos dados de treinamento, ocorre o sobreajuste, onde o modelo se torna altamente sensível aos detalhes específicos desses dados, comprometendo seu desempenho em dados desconhecidos (Hastie *et al.*, 2009).

3.3.1 Aprendizado Supervisionado

A principal característica do Aprendizado Supervisionado, do inglês Supervised Learning (SL) é a utilização de um conjunto de dados rotulados para treinar um modelo, que deve aprender a mapear as entradas (características) para as saídas (rótulos) com base em exemplos fornecidos durante o treinamento. Essa abordagem é supervisionada no sentido de que o modelo recebe *feedback* direto sobre os resultados corretos para cada exemplo (Bishop, 2006).

De forma técnica, o objetivo central do SL é encontrar uma função $f : X \rightarrow Y$ que mapeie um conjunto de entradas X para suas respectivas saídas Y. Isso é feito através da minimização de uma função de perda, que quantifica o erro entre a saída prevista pelo modelo e o valor real esperado. A função de perda, $L(f(x), y)$, é utilizada para guiar o processo de otimização do modelo, ajustando seus parâmetros para minimizar o erro total sobre o conjunto de dados (Murphy, 2012).

Uma das principais características do SL é a divisão do processo em três etapas fundamentais: treinamento, validação e teste.

- Treinamento: nesta fase, o modelo é ajustado com base em um conjunto de dados rotulado, onde o objetivo é minimizar a função de perda. Isso permite ao modelo aprender a relação entre as características de entrada e os rótulos de saída.
- Validação: após o treinamento inicial, o modelo é validado utilizando um subconjunto separado dos dados, com o objetivo de ajustar hiperparâmetros e prevenir o sobreajuste. A validação ajuda a garantir que o modelo não se ajuste excessivamente aos dados de

treinamento, mantendo sua capacidade de generalizar para novos dados (Hastie *et al.*, 2009).

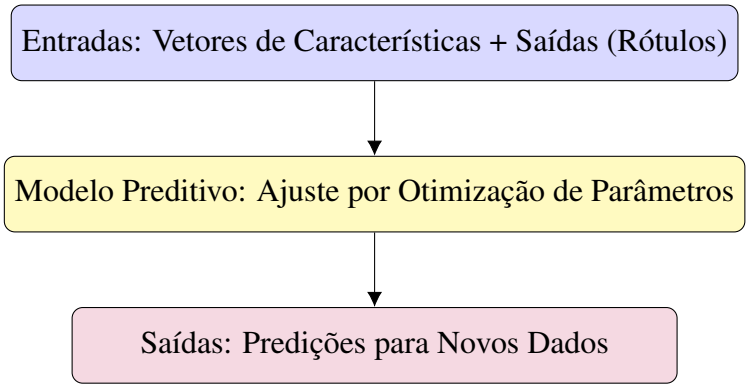
- Teste: a fase final consiste na avaliação do modelo em dados que ele nunca viu durante o treinamento. Isso fornece uma estimativa da sua capacidade de generalização para novas situações e valida seu desempenho em um cenário real (Russell; Norvig, 2010).

Outro ponto importante no aprendizado supervisionado é a sua dependência de dados rotulados. A obtenção de grandes volumes de dados rotulados de alta qualidade pode ser desafiadora em muitos domínios. Essa dependência direta de dados previamente rotulados diferencia o aprendizado supervisionado de outras abordagens, como o aprendizado não supervisionado, onde o modelo tenta descobrir padrões por conta própria (Goodfellow *et al.*, 2016).

3.3.2 Modelos de Aprendizado Supervisionados

Os modelos de aprendizado supervisionado seguem uma estrutura padronizada que pode ser generalizada em três componentes principais: entradas, modelo preditivo e saídas, como obervado na Figura 5 abaixo.

Figura 5 – Fluxo de um Modelo Supervisionado



Fonte: elaborada pelo autor.

As entradas são os vetores de características que descrevem os dados observados e são usados para treinar o modelo. O modelo preditivo é uma função matemática que processa as entradas para produzir uma predição, geralmente ajustada a partir de um conjunto de treinamento por meio de otimização de hiperparâmetros (Goodfellow *et al.*, 2016).

Por fim, as saídas são as variáveis dependentes que o modelo supervisionado tenta prever. Dependendo da natureza do problema, as saídas podem ser rótulos de classe (em proble-

mas de classificação) ou valores contínuos (em problemas de regressão). Após o treinamento, o modelo supervisionado deve ser capaz de generalizar e fazer predições precisas quando exposto a novos dados (Murphy, 2012).

3.3.2.1 Regressão Logística

A Regressão Logística, do inglês Logistic Regression (LR) é uma ferramenta estatística projetada para modelar a relação entre variáveis descritivas e um resultado binário, permitindo a estimativa da probabilidade de ocorrência de um determinado evento com base em um conjunto de características observadas (Hosmer *et al.*, 2013). Diferente de modelos que prevêem valores contínuos, como altura ou peso, a Regressão Logística é voltada para situações onde o objetivo é prever categorias distintas, como "sim" ou "não", "positivo" ou "negativo".

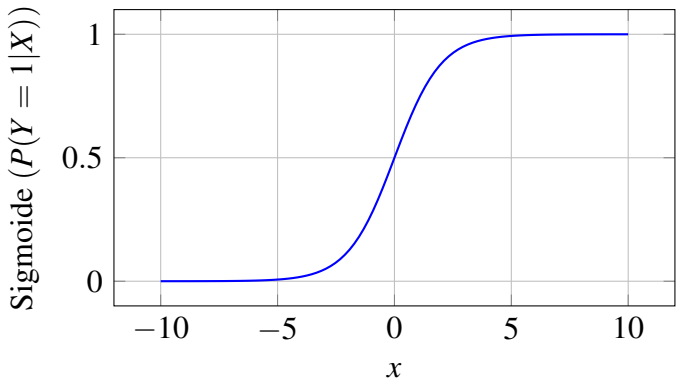
De acordo com Cox (1958), a regressão logística modela a probabilidade de um evento ocorrer usando a função sigmoide, definida pela seguinte equação:

$$P(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_p x_p)}} \tag{3.5}$$

onde $P(Y=1|X)$ é a probabilidade de o evento $Y=1$ ocorrer, dado o conjunto de variáveis independentes $X=(x_1,x_2,\ldots,x_p)$, β_0 representa a razão de chances básica do evento ocorrer quando todas as variáveis preditoras são iguais a zero, $\beta_1,\beta_2,\ldots,\beta_p$ são os coeficientes associados às variáveis independentes, e e é a base do logaritmo natural. A função sigmoide, como ilustrado na Figura 6, produz uma saída contínua entre 0 e 1, o que a torna ideal para modelar probabilidades. Sua forma em "S" suaviza a transição entre os extremos, de modo que, conforme as variáveis independentes mudam, a probabilidade de $Y=1$ aumenta ou diminui gradualmente.

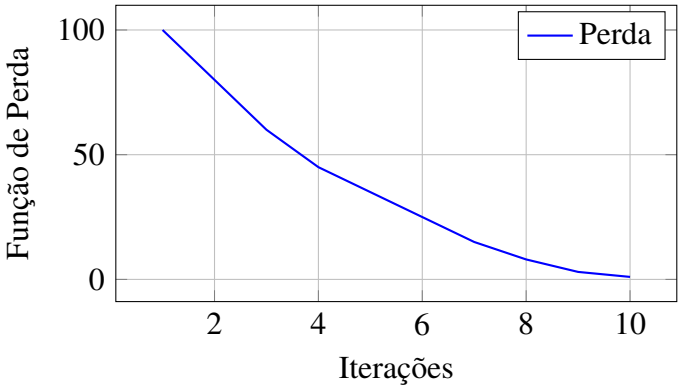
Além disso, os coeficientes $\beta_0,\beta_1,\ldots,\beta_p$ são estimados por meio do método da máxima verossimilhança, que encontra os valores que maximizam a probabilidade de observar os dados de acordo com o modelo proposto. A Figura 7 mostra o processo de convergência dos coeficientes da Regressão Logística durante o treinamento, conforme a função de perda é minimizada.

Figura 6 – Curva da função sigmoide.



Fonte: elaborada pelo autor.

Figura 7 – Convergência dos coeficientes da Regressão Logística durante o treinamento.



Fonte: elaborada pelo autor.

Uma das principais vantagens da regressão logística é a interpretação dos coeficientes como razões de chances, permitindo avaliar a influência de cada variável na probabilidade do evento. Essa interpretação pode ser feita a partir do logaritmo das razões de chances, conforme definido a seguir:

$$\log\left(\frac{P(Y=1|X)}{1-P(Y=1|X)}\right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p \tag{3.6}$$

Este modelo assume que existe uma relação linear entre o logaritmo das chances de sucesso e as variáveis preditoras, dessa forma, a regressão logística também pode ser generalizada para o caso de múltiplas classes, permitindo a modelagem de variáveis dependentes com mais de duas categorias (Agresti, 2015).

A regressão logística possui aplicações em diversas áreas sociais. No setor de

saúde, Shah *et al.* (2015) utilizaram a regressão logística para prever a ocorrência de doenças cardíacas com base em fatores de risco como idade, gênero e histórico familiar, resultando em um modelo preditivo robusto. No marketing, ela é usada para prever a probabilidade de um cliente realizar uma compra com base em características de comportamento (Gurumurthy; Khare, 2016). Em outra aplicação também feita na área de marketing, Lemon e Verhoef (2016) aplicaram a regressão logística para prever a probabilidade de cancelamento de assinaturas por clientes, ajudando a otimizar estratégias de retenção.

3.3.2.2 K-Vizinho mais Próximo

O K-Vizinho mais Próximo, do inglês K-Nearest Neighbors (KNN) é um algoritmo de aprendizado supervisionado que realiza a classificação e a regressão de novos dados com base em sua proximidade a amostras já conhecidas no espaço dos atributos. A classificação é feita considerando os k vizinhos mais próximos da nova amostra, atribuindo a ela a classe mais comum entre esses vizinhos no caso de classificação, ou a média das classes no caso de regressão (Altman, 1992). Uma característica essencial do KNN é que ele é um método não paramétrico, o que significa que não assume uma distribuição específica dos dados.

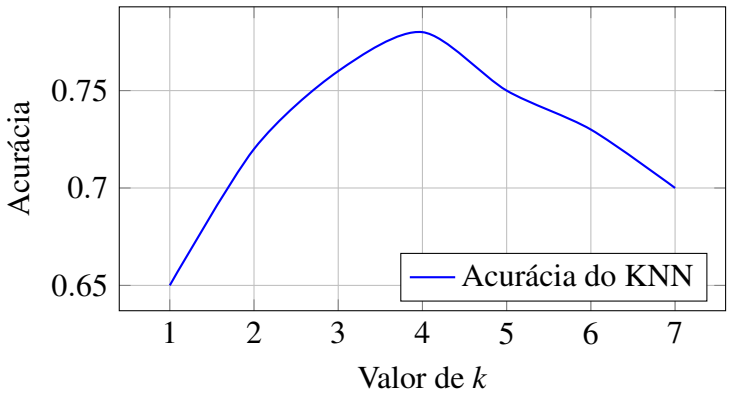
De acordo com Cover e Hart (1967), a proximidade entre as amostras é facilmente medida pela distância Euclidiana, definida pela seguinte equação:

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2} \tag{3.7}$$

onde p e q são dois pontos no espaço n-dimensional e p_i e q_i são as coordenadas desses pontos. A métrica de distância pode ser adaptada para diferentes tipos de dados, sendo a Euclidiana a mais comum em dados contínuos, mas outras métricas, como distância de Manhattan ou de Minkowski, também podem ser utilizadas, dependendo da natureza do problema.

A escolha do valor de k é um dos pontos mais importantes para o desempenho do KNN. Valores pequenos de k podem tornar o modelo sensível ao ruído, enquanto valores grandes podem diluir a influência dos vizinhos mais próximos e introduzir viés. Como mostrado na Figura 8, o valor de k impacta diretamente a acurácia do modelo, podendo variar de acordo com o conjunto de dados utilizado.

Figura 8 – Sensibilidade do KNN ao valor de k



Fonte: elaborada pelo autor.

De acordo com Wang *et al.* (2021), o valor ótimo de k pode ser encontrado através de métodos como a validação cruzada que atua dividindo o conjunto de dados em subconjuntos de treinamento e teste, permitindo avaliar o desempenho do modelo para diferentes valores de k e, assim, escolher aquele que minimiza o erro de predição.

O KNN é especialmente útil em situações onde o conjunto de características é bem compreendido e as tomadas de decisão não são lineares. Entretanto, ele pode se tornar computacionalmente caro à medida que o número de amostras no conjunto de treinamento aumenta, uma vez que, para cada nova predição, é necessário calcular a distância de todas as amostras de treinamento para a nova amostra (Peterson, 2009).

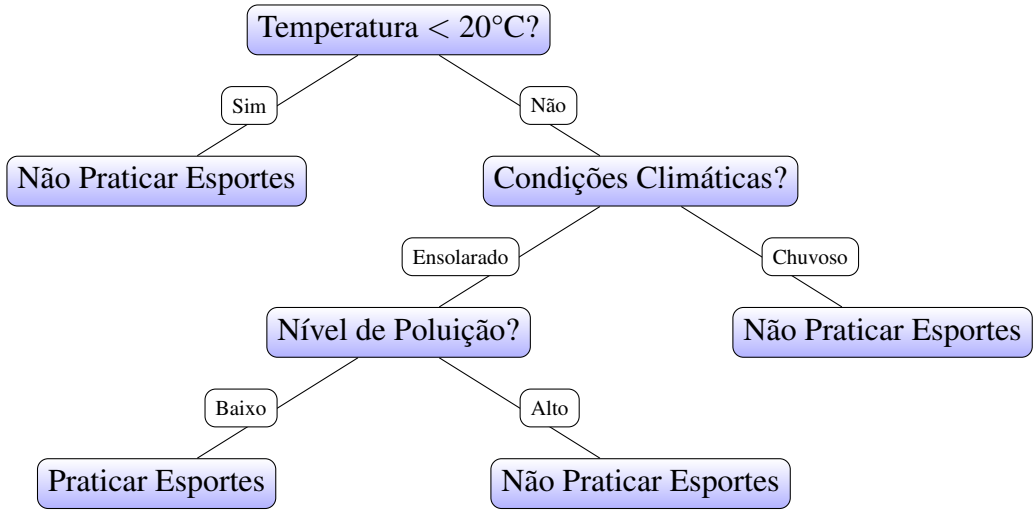
As aplicações do KNN variam para diversos campos. Em visão computacional, por exemplo, Boiman *et al.* (2008) usaram o KNN para classificação de imagens, alcançando uma alta precisão ao comparar diretamente as distâncias entre características de imagens. Na área de recomendação de filmes, Bell e Koren (2007) aplicaram o KNN para prever a avaliação de filmes baseada nas avaliações dos vizinhos mais próximos de usuários com gostos semelhantes. Já no campo da biologia computacional, Liaw e Wiener (2017) usaram o KNN de forma versátil para prever a função de genes com base em perfis de expressão gênica.

3.3.2.3 Árvores de Decisão

As Árvores de Decisão, do inglês Decision Trees (DT) são modelos de aprendizado supervisionado que funcionam através de uma estrutura hierárquica, na qual os dados são divididos recursivamente em subconjuntos menores de acordo com condições de decisão predefinidas. Cada nó interno da árvore corresponde a uma condição de teste em uma característica, os ramos

indicam os resultados desses testes, e os nós folha representam as predições finais (Quinlan, 1986). A Figura 9 mostra um exemplo de árvore de decisão que auxilia na determinação de se uma pessoa deve praticar esportes ao ar livre, considerando três fatores: temperatura, condições climáticas e nível de poluição.

Figura 9 – Árvore de decisão para determinar a prática de esportes ao ar livre com base em temperatura, condições climáticas e nível de poluição.



Fonte: elaborado pelo autor.

O modelo começa avaliando a temperatura: se ela estiver abaixo de 20°C, a pessoa não deve praticar esportes. Caso contrário, a árvore considera as condições climáticas e, se o tempo estiver ensolarado, avalia o nível de poluição para tomar a decisão final.

De acordo com Breiman *et al.* (1984), a construção de uma árvore de decisão envolve a seleção da melhor característica para dividir os dados em cada nó, com base em critérios como o índice Gini, que mede a impureza de um nó. O índice Gini é definido como:

$$Gini(D) = 1 - \sum_{i=1}^{c} p_i^2 \tag{3.8}$$

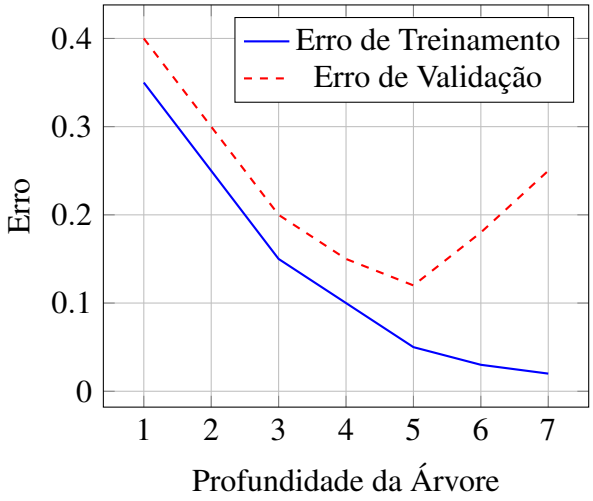
onde p_i é a proporção de observações da classe i em um conjunto de dados D, e c é o número total de classes. Outro critério frequentemente utilizado é a redução de entropia, que maximiza a quantidade de informação obtida em cada divisão, sendo calculada pela seguinte equação:

$$H(t) = -\sum_{i=1}^{c} p_i \log_2 p_i \tag{3.9}$$

onde p_i representa a proporção de elementos da classe i em um nó t.

As árvores de decisão apresentam diversas vantagens, como a facilidade de interpretação e visualização, além de serem capazes de identificar facilmente relações não lineares entre as variáveis. No entanto, um dos desafios desses modelos é o risco de sobreajuste, que ocorre quando o modelo consegue realizar previsões precisas para os dados de treinamento, mas falha em generalizar para novos dados. Em árvores de decisão, esse problema tende a surgir quando a profundidade da árvore aumenta excessivamente, levando a uma complexidade elevada (Quinlan, 1993). A Figura 10 demonstra esse fenômeno, mostrando a relação entre a profundidade da árvore e os erros de treinamento e validação.

Figura 10 – Gráfico de erro de treinamento e validação em função da profundidade da árvore



Fonte: elaborada pelo autor.

Note que, na figura, o erro de treinamento diminui continuamente com o aumento da profundidade da árvore, enquanto o erro de validação começa a aumentar após certo ponto, indicando que a árvore está se ajustando de forma excessiva aos dados de treinamento e perdendo a capacidade de generalizar para novos exemplos.

Para diminuir o problema de sobreajuste das árvores individuais, foi introduzida a técnica de Floresta Randômica, do inglês Random Forest (RF), onde múltiplas árvores de decisão são construídas a partir de diferentes amostras do conjunto de dados. Além disso, em cada nó de uma árvore, um subconjunto aleatório das características é escolhido para determinar a melhor divisão, aumentando a diversidade das árvores e reduzindo a correlação entre elas. Por fim, a RF combina os resultados dessas árvores para fazer uma predição final, que, no caso da classificação, é a votação da maioria, e, no caso da regressão, é a média das predições individuais (Ho, 1995).

De acordo com Breiman (2001), a predição final de uma RF para uma amostra x pode ser expressa como:

$$\hat{f}(x) = \frac{1}{T} \sum_{t=1}^{T} f_t(x) \tag{3.10}$$

onde T é o número total de árvores na floresta, e $f_t(x)$ é a predição da t-ésima árvore.

As Florestas Randômicas são aplicadas com sucesso em diversas áreas. Na biologia, são utilizadas para a seleção de genes e predição de resultados clínicos (Díaz-uriarte; Andrés, 2006). Em economia, RFs foram empregadas para prever tendências de mercado e o comportamento de consumidores (Varian, 2014). Além disso, no campo de detecção de fraudes, esse modelo é amplamente utilizado para identificar atividades suspeitas em transações financeiras (Ngai *et al.*, 2011).

3.3.3 Análise de Componentes Principais

A Análise de Componentes Principais, do inglês Principal Component Analysis (PCA) é uma técnica estatística de redução de dimensionalidade usada para simplificar conjuntos de dados com muitas variáveis. O principal objetivo da PCA é permitir a redução de dimensionalidade dos dados sem perder uma quantidade significativa de informação, transformando um conjunto de variáveis observadas que podem ser correlacionadas entre si em um novo conjunto de variáveis não correlacionadas (Jolliffe; Cadima, 2016).

De acordo com Shlens (2014), a redução de dimensionalidade feita pela PCA ocorre por meio da decomposição dos dados em direções de variância máxima, chamadas de componentes principais. O processo começa com a normalização dos dados, garantindo que todas as variáveis estejam na mesma escala. Em, seguida, é feito o cálculo da matriz de convariância, usada para encontrar as correlações entre as variáveis, bem como a determinação dos autovalores e autovetores dessa matriz, para identificar as principais direções dos dados. Mais especificamente, dado um conjunto de dados X com n observações e p variáveis, a primeira etapa é normalizar as variáveis para que cada uma tenha média zero e variância única para cada variável:

$$z_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j} \tag{3.11}$$

onde $\bar{x}_j$ representa a média da variável x_j e s_j o seu desvio padrão. Em seguida, é

calculada a matriz de covariância dos dados normalizados:

$$C = \frac{1}{n-1} Z^T Z \tag{3.12}$$

onde Z é a matriz dos dados normalizados. Os autovalores λ_i e autovetores e_i dessa matriz são determinados para identificar as direções e magnitudes das componentes principais:

$$C_{e_i} = \lambda_i e_i \tag{3.13}$$

Os autovalores são ordenados em ordem decrescente, com os primeiros k componentes sendo selecionados para representar uma parte significativa da variância total:

$$\text{Proporção da variância} = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \tag{3.14}$$

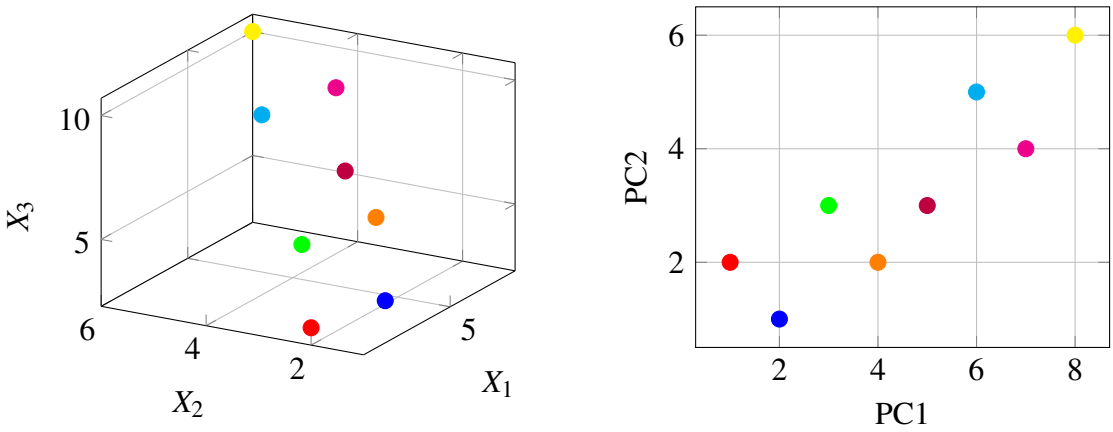
Por fim, os dados são traçados nos novos eixos formados pelos componentes principais, ocasionando em uma representação com a dimensionalidade reduzida:

$$T = ZE \tag{3.15}$$

onde T é a matriz transformada e E é a matriz dos autovetores. Por exemplo, considerando um conjunto de dados original com três variáveis (X_1, X_2, X_3), onde é aplicada a técnica de PCA para projetar esses dados em um novo espaço bidimensional utilizando os dois primeiros componentes principais (PC_1, PC_2). A Figura 11 demonstra como essa projeção funciona.

Na Figura 11 (a), os dados são visualizados em seu espaço tridimensional original, enquanto na Figura 11 (b), os mesmos dados são projetados no plano 2D formado pelos dois primeiros componentes principais, dessa forma, a projeção torna a estrutura dos dados mais simples e reduzida, enquanto mantém a maior parte da informação relevante, permitindo uma visualização mais clara dos dados.

Figura 11 – Comparação entre dados originais e projetados após PCA



(a) Dados originais 3D

(b) Dados projetados 2D

Fonte: elaborada pelo autor.

A PCA possui aplicações em diferentes áreas. No campo da verificação de assinaturas on-line, um estudo combinou PCA e Análise de Componentes Menores (MCA) para melhorar a precisão da verificação, resultando em uma taxa de erro de aproximadamente 5% (Li *et al.*, 2004). Na detecção de ruído em rótulos, a PCA é usada para identificar discrepâncias entre rótulos observados e reais, melhorando a qualidade dos dados e a performance dos algoritmos de aprendizado de máquina (Salekshahrezaee *et al.*, 2021). Na bioinformática, a PCA ajuda a extrair assinaturas de nível de via em dados proteogenômicos, facilitando a identificação de padrões e a interpretação dos resultados (Liu *et al.*, 2019).

3.3.4 SMOTE e RandomUnderSampler

A desproporção entre as classes é um problema comum em problemas de classificação, especialmente quando a classe minoritária tem significativamente menos exemplos do que a classe majoritária. Esse desbalanceamento pode prejudicar o desempenho dos modelos de aprendizado de máquina, já que eles tendem a se concentrar mais na classe majoritária e, consequentemente, a desconsiderar as características da classe minoritária. O desbalanceamento também pode levar a métricas de avaliação distorcidas, como a acurácia, que pode ser muito alta devido à predominância da classe majoritária, mas não reflete corretamente a capacidade do modelo de classificar adequadamente a classe minoritária. Para mitigar esse problema, técnicas de balanceamento de classes são comumente empregadas para gerar um conjunto de dados mais equilibrado e, assim, melhorar a precisão do modelo. Dentre as técnicas mais eficazes, se desta-

cam o **SMOTE** (Synthetic Minority Over-sampling Technique) e o **RandomUnderSampler**, altamente utilizados para lidar com cenários de desbalanceamento (Chawla *et al.*, 2002; He *et al.*, 2008).

O método **SMOTE**, proposto por Chawla et al. (2002), é uma técnica de aumento de amostras que visa gerar exemplos sintéticos da classe minoritária. Em vez de simplesmente replicar as instâncias existentes da classe minoritária, o SMOTE cria novas instâncias interpolando aleatoriamente entre as instâncias da classe minoritária. O processo de geração de novos exemplos ocorre em uma área definida pela distância entre as instâncias da classe minoritária. Para cada instância da classe minoritária, o SMOTE seleciona um ou mais vizinhos mais próximos e gera novos exemplos ao longo da linha entre os vizinhos selecionados e a instância original. Esse aumento de exemplos sintéticos ajuda o modelo a aprender melhor a distribuição da classe minoritária e melhora a capacidade do modelo de classificar essas instâncias (Chawla *et al.*, 2002). A vantagem do SMOTE sobre outras técnicas de aumento de amostras é que ele cria instâncias com variações, ao invés de duplicar instâncias existentes, o que melhora a generalização do modelo.

O algoritmo SMOTE pode ser descrito da seguinte forma: para cada exemplo da classe minoritária, o número k de vizinhos mais próximos é calculado usando uma medida de distância (como a distância Euclidiana). Um novo exemplo é então gerado como uma combinação linear entre o exemplo original e um dos seus vizinhos mais próximos. Esse processo pode ser repetido N vezes para gerar N exemplos sintéticos para cada instância minoritária. A técnica pode ser ajustada variando o número de vizinhos considerados e a quantidade de exemplos sintéticos gerados.

Por outro lado, o **RandomUnderSampler** é uma técnica que busca balancear as classes, mas em vez de aumentar a classe minoritária, ela reduz o número de instâncias da classe majoritária. Esse método faz uma amostragem aleatória dos exemplos da classe majoritária e elimina uma parte deles até que as classes estejam equilibradas. Embora essa abordagem seja eficaz para reduzir o desbalanceamento, ela tem o potencial de eliminar informações valiosas da classe majoritária, especialmente se o número de exemplos for muito reduzido. A principal desvantagem do RandomUnderSampler é que ele pode levar a uma perda significativa de dados, o que pode afetar a capacidade do modelo de aprender as características importantes da classe majoritária (He *et al.*, 2008). Além disso, a eliminação de dados pode reduzir a capacidade do modelo de capturar a variabilidade da classe majoritária, prejudicando sua capacidade de

generalização.

No entanto, o **RandomUnderSampler** também tem suas aplicações, especialmente em cenários onde o número de exemplos da classe majoritária é muito alto e a remoção de uma parte deles pode resultar em uma melhor representação das classes. O uso dessa técnica deve ser ponderado cuidadosamente, levando em consideração a quantidade de dados disponíveis e a complexidade do problema em questão (He *et al.*, 2008).

3.3.5 Redes Neurais Artificiais

As Redes Neurais Artificiais, do inglês Artificial Neural Networks (ANN) são modelos de aprendizado inspirados no funcionamento do cérebro humano. Elas consistem em um conjunto de unidades de processamento chamadas neurônios, organizadas em camadas interconectadas, que permitem ao modelo aprender padrões complexos a partir dos dados. Cada conexão entre os neurônios possui um peso associado, que é ajustado durante o treinamento para minimizar o erro entre as previsões da rede e os valores reais (Goodfellow *et al.*, 2016).

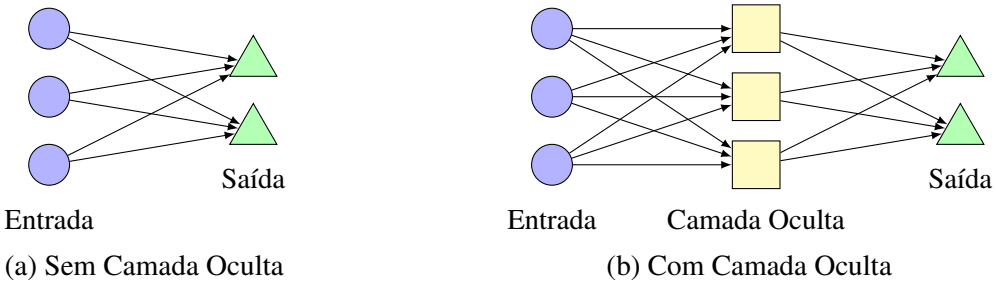
De acordo com Rosenblatt (1958), uma rede neural é composta por uma camada de entrada, uma ou mais camadas ocultas, e uma camada de saída. A camada de entrada recebe os dados brutos e os processa através de várias transformações nas camadas intermediárias, chamadas de camadas ocultas, até gerar uma predição final na camada de saída. A função que transforma os sinais de entrada nos sinais de saída é chamada de função de ativação, sendo a função de Unidade Linear Retificada, do inglês Rectified Linear Unit (ReLU) uma das mais utilizadas, definida pela seguinte equação:

$$f(x) = \max(0, x) \tag{3.16}$$

onde x é a entrada do neurônio. A função ReLU é utilizada devido à sua eficiência computacional, permitindo a criação de modelos profundos sem o problema de desaparecimento do gradiente, comum em funções de ativação sigmoides.

A Figura 12 ilustra duas arquiteturas de ANN. A Figura 12 (a) mostra uma rede neural simples sem camada oculta, onde os dados da camada de entrada são diretamente conectados à camada de saída. Já a Figura 12 (b) apresenta uma rede neural com uma camada oculta, onde a transformação dos dados ocorre entre a entrada e a saída.

Figura 12 – Arquitetura de Redes Neurais Simples: com e sem camada oculta



(a) Sem Camada Oculta (b) Com Camada Oculta

Fonte: elaborada pelo autor.

O processo de aprendizado de uma rede neural envolve a retropropagação do erro (backpropagation), que ajusta os pesos das conexões entre os neurônios para minimizar a função de perda. A função de perda mais comumente utilizada para problemas de classificação é a entropia cruzada, dada por:

$$L = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{3.17}$$

onde y_i é o valor real da classe para a amostra i, $\hat{y}_i$ é a probabilidade prevista pela rede para essa classe, e n é o número total de amostras.

A retropropagação utiliza o algoritmo de descida de gradiente, que ajusta os pesos de acordo com a derivada da função de perda em relação a cada peso, permitindo que o modelo aprenda a partir dos erros e melhore suas predições (Rumelhart *et al.*, 1986).

Redes neurais têm sido aplicadas com sucesso em diversas áreas. Por exemplo, em visão computacional, Krizhevsky *et al.* (2012) utilizaram uma rede neural convolucional para vencer a competição ImageNet, alcançando resultados significativamente melhores do que os métodos tradicionais da época. Em outro exemplo, no campo da medicina, Esteva *et al.* (2017) desenvolveram uma rede neural capaz de diagnosticar câncer de pele com precisão comparável à de dermatologistas. No setor financeiro, redes neurais são amplamente usadas para previsão de mercado e detecção de fraudes em transações financeiras (Heaton *et al.*, 2017).

3.3.5.1 Rede Neural Feed-forward

A Rede Neural Feed-forward, do inglês Feed-forward Neural Network (FNN) é a forma mais básica e direta de uma rede neural artificial, onde as informações fluem em uma

única direção: da camada de entrada, através das camadas ocultas, até a camada de saída, sem ciclos ou realimentação. Este tipo de rede é usado tanto para tarefas de classificação quanto de regressão (Rosenblatt, 1958).

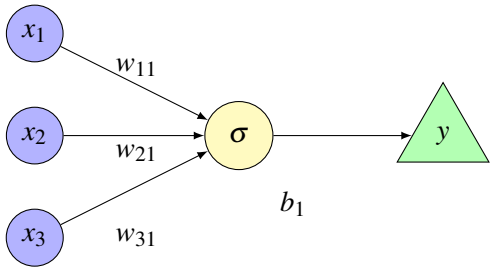
Cada camada de uma FNN é composta por neurônios, que são conectados aos neurônios da camada seguinte. A saída de cada neurônio é calculada como uma combinação linear ponderada das entradas, seguida por uma função de ativação não linear, tipicamente uma função Sigmoide, ReLU ou Tangente Hiperbólica, do inglês Hyperbolic Tangent (TANH). A saída do neurônio i na camada l, por exemplo, é dada por:

$$a_i^{(l)} = \sigma\left(\sum_{j=1}^{n} w_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)}\right) \tag{3.18}$$

onde $a_i^{(l)}$ é a ativação do neurônio i na camada l, $w_{ij}^{(l)}$ são os pesos conectando o neurônio j da camada anterior à camada i na camada atual, $a_j^{(l-1)}$ é a ativação do neurônio j na camada anterior, $b_i^{(l)}$ é o bias associado ao neurônio i, e σ é a função de ativação não linear (Haykin, 1999).

A Figura 13 demonstra o cálculo da ativação de um neurônio em uma FNN. Nela, são mostradas três entradas x_1, x_2, x_3, que são multiplicadas pelos respectivos pesos w_{11}, w_{21}, w_{31}, e somadas a um valor de bias b_1. O resultado da soma é então passado pela função de ativação σ, gerando a saída y do neurônio. Este processo é repetido para cada neurônio em cada camada da rede.

Figura 13 – Cálculo da Ativação de um Neurônio em uma FNN



Fonte: elaborada pelo autor.

Uma característica importante das FNNs é a sua habilidade de aproximar qualquer função contínua, desde que haja neurônios suficientes nas camadas ocultas. Esse resultado é formalizado pelo teorema da aproximação universal, que afirma que uma FNN com uma única camada oculta e um número suficientemente grande de neurônios pode aproximar qualquer

função contínua em um intervalo compacto (Cybenko, 1989). No entanto, em alguns casos, redes com múltiplas camadas ocultas (também chamadas de redes profundas) são escolhidas, pois permitem modelar relações mais complexas entre os dados.

A simplicidade das FNNs as torna aplicáveis em diversos domínios. Em reconhecimento de padrões, por exemplo, LeCun *et al.* (1998) aplicaram redes feedforward para o reconhecimento de dígitos manuscritos no conjunto de dados MNIST, um marco importante no desenvolvimento de redes neurais para visão computacional. Na área de finanças, Trippi e Turban (1992) usaram FNNs para prever o mercado de ações, mostrando a capacidade dessas redes de capturar padrões não lineares nos dados financeiros. Além disso, no campo da engenharia, redes feedforward são utilizadas para prever falhas em sistemas e otimizar processos industriais (Samarasinghe, 2006).

3.3.6 Métricas de Avaliação

As métricas de avaliação possuem um papel fundamental na análise de modelos de aprendizado supervisionado, permitindo mensurar a performance e determinar o quanto um modelo é adequado a um determinado conjunto de dados. A escolha da métrica apropriada depende majoritariamente da natureza do problema, seja ele de classificação ou regressão. Entre as métricas mais comuns estão a acurácia, precisão, revocação, F1-score, e o erro quadrático médio, com cada uma delas oferecendo uma perspectiva diferente sobre o desempenho do modelo (Powers, 2011).

No contexto de classificação, especialmente quando as classes estão balanceadas, uma das métricas mais amplamente utilizadas é a acurácia, que mede a proporção de predições corretas em relação ao total de amostras:

$$\text{Acurácia} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.19}$$

onde Verdadeiros Positivos, do inglês True Positives (TP) e Verdadeiros Negativos, do inglês True Negatives (TN) são as predições corretas, e Falsos Positivos, do inglês False Positives (FP) e Falsos Negativos, do inglês False Negatives (FN) representam as predições incorretas. No entanto, em problemas onde há desbalanceamento entre as classes, a acurácia pode agir como um valor enganoso, levando em conta que um modelo pode atingir alta acurácia apenas prevendo a classe majoritária. Para lidar com esses casos, são utilizadas outras métricas

como precisão e revocação.

Outra abordagem que atua na avaliação de modelos é a busca em grade, uma técnica utilizada para encontrar a combinação ideal de hiperparâmetros de um modelo. O processo consiste em definir um conjunto de valores possíveis para cada hiperparâmetro e testar exaustivamente todas as combinações desses valores. O desempenho de cada combinação é avaliado utilizando uma métrica de desempenho específica, como acurácia ou F1-score, e a melhor configuração de hiperparâmetros é selecionada com base nesses resultados (Bergstra *et al.*, 2012).

Para garantir que o ajuste de hiperparâmetros generalize bem para novos dados, a validação cruzada é utilizada em conjunto com a busca em grade. Um método comum de validação cruzada é o *k-fold*, que divide o conjunto de dados em k subconjuntos (*folds*). O modelo é treinado k vezes, usando $k-1$ *folds* para treinamento e o *fold* restante para validação. Ao final, a média dos desempenhos nas k execuções é calculada, trazendo uma estimativa mais confiável do desempenho do modelo em dados não vistos, além de minimizar a chance de sobreajuste ou sub-ajuste (Hastie *et al.*, 2009).

4 TRABALHOS RELACIONADOS

Esta seção explora os principais trabalhos relacionados à pesquisa, com foco em aplicações de técnicas de aprendizado de máquina para resolver problemas de otimização combinatória, como o Problema do Caixeiro Viajante, do inglês Travelling Salesman Problem (TSP) na Seção 4.1, o Problema do Corte Máximo na Seção 4.2 e o Problema da Diversidade Máxima na Seção 4.3. Cada seção discute em detalhes os métodos aplicados, as abordagens adotadas, e os resultados obtidos nos respectivos trabalhos.

4.1 Os métodos de redes neurais para resolver o Problema do Caixeiro Viajante

Shi e Zhang (2022) fazem uma revisão abrangente sobre o uso de redes neurais para lidar com o Problema do Caixeiro Viajante (TSP), um desafio clássico de otimização combinatória, conhecido por sua complexidade NP-difícil. O TSP exige que um viajante visite um conjunto de cidades exatamente uma vez, retornando ao ponto de partida pelo menor caminho possível. Tradicionalmente, a solução exata do TSP para grandes instâncias tem sido inviável devido ao crescimento exponencial do espaço de busca, o que motivou a exploração de métodos aproximados, incluindo o uso de redes neurais.

Os autores focam em três abordagens principais de redes neurais:

- **Redes Neurais de Hopfield**: Esta abordagem utiliza uma rede neural recorrente, onde cada cidade é representada por um neurônio, e as conexões entre os neurônios são ponderadas de acordo com as distâncias entre as cidades. O objetivo é minimizar uma função de energia, que corresponde à soma das distâncias percorridas na solução proposta pelo modelo. Por ser uma das primeiras aplicações de redes neurais ao TSP, a rede de Hopfield apresenta desafios significativos, como a dificuldade de evitar mínimos locais e a necessidade de ajustes precisos nos parâmetros para garantir convergência. Além disso, sua escalabilidade é limitada, a tornando menos eficaz para grandes instâncias do problema.
- **Redes Neurais em Grafos (GNNs)**: As GNNs oferecem uma abordagem mais moderna e eficiente para o TSP, permitindo que as redes aprendam diretamente a partir da estrutura do grafo que representa o problema. As GNNs são projetadas para capturar as interações entre vértices (cidades) e arestas (rotas), processando o grafo de forma global. Isso permite ao modelo identificar padrões e relações complexas entre os elementos do grafo, resultando em soluções de alta qualidade, mesmo para instâncias maiores. A adaptabilidade das GNNs

a diferentes tipos de grafos e sua capacidade de generalização são vantagens importantes em relação às redes de Hopfield.

- **Redes Neurais com Aprendizado por Reforço**: Nesta abordagem, as redes neurais são combinadas com aprendizado por reforço para melhorar de forma iterativa as soluções propostas. Especificamente, as Redes de Ponteiros, um tipo de rede neural recorrente, são utilizadas para prever a sequência de cidades a serem visitadas. O modelo é treinado usando uma política de reforço, onde as decisões são refinadas com base no *feedback* das soluções geradas, buscando maximizar a recompensa (ou minimizar a distância total percorrida). Esta técnica se mostra particularmente eficaz na adaptação a diferentes instâncias do TSP e na superação de limitações de métodos tradicionais, como a busca por mínimos locais.

Os resultados apresentados pelos autores indicam que as redes neurais, especialmente quando combinadas com técnicas de aprendizado por reforço, podem superar métodos heurísticos tradicionais, como algoritmos genéticos e colônia de formigas, tanto em precisão quanto em eficiência. As redes neurais demonstram grande potencial para melhorar a qualidade das soluções aproximadas do TSP. O estudo sugere que futuras pesquisas devem explorar a integração dessas técnicas com outras abordagens metaheurísticas para maximizar seu potencial.

4.2 Um 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

Gu e Yang (2020) apresentam um avanço significativo no uso de redes neurais para resolver o Problema do Corte Máximo, um problema de otimização combinatória que busca maximizar a soma dos pesos das arestas que cortam a partição de um grafo em dois subconjuntos. O Corte Máximo é um problema NP-difícil, e sua resolução exata é inviável para grafos grandes, o que torna a busca por soluções aproximadas uma necessidade.

O trabalho propõe uma abordagem baseada em Redes de Ponteiros para capturar a estrutura do problema de forma eficiente. As Redes de Ponteiros que foram originalmente desenvolvidas para problemas de roteamento e tradução de linguagem natural, são redes neurais recorrentes que aprendem a mapear sequências de entrada para sequências de saída. No contexto do Corte Máximo, essas redes são adaptadas para processar grafos, identificando cortes que maximizam a soma dos pesos das arestas cortadas.

A abordagem é dividida em duas fases principais:

- **Aprendizado Supervisionado**: Inicialmente, o modelo é treinado com dados rotulados, onde as soluções ótimas ou próximas do ótimo são conhecidas. Esse treinamento supervisionado permite que a rede aprenda a identificar características e padrões nos grafos que levam a cortes eficientes. O modelo é configurado para processar a matriz de adjacência do grafo, onde os pesos das arestas determinam a probabilidade de um corte ocorrer em uma determinada posição.
- **Aprendizado por Reforço**: Após o treinamento supervisionado, o modelo é refinado utilizando aprendizado por reforço, uma técnica que não requer exemplos rotulados. Aqui, o modelo interage com o ambiente de solução, recebendo recompensas por melhorar as soluções de corte ou penalidades por soluções subótimas. Essa fase permite que o modelo melhore continuamente suas previsões, se ajustando de forma dinâmica às características dos grafos, fator importante para evitar mínimos locais e melhorar a qualidade das soluções.

Os resultados experimentais mostram que o modelo proposto supera os métodos clássicos de otimização, como algoritmos de relaxamento semidefinido, especialmente em termos de tempo de computação e qualidade das soluções para instâncias grandes do Corte Máximo. O uso de Redes de Ponteiros combinado com aprendizado supervisionado e por reforço demonstra uma forte capacidade de adaptação a diferentes estruturas de grafos, o que é um diferencial importante frente a outras abordagens. Os autores destacam que essa metodologia pode ser estendida a outros problemas de otimização combinatória, oferecendo uma plataforma versátil para soluções aproximadas em contextos complexos.

4.3 Rede de Hopfield Competitiva Combinada com Estimativa de Distribuição para Problemas de Diversidade Máxima

Wang *et al.* (2009) propõem uma abordagem híbrida que combina Redes Neurais de Hopfield Competitivas (DCHNN) com o Algoritmo de Estimativa de Distribuição (EDA) para resolver o Problema da Diversidade Máxima (MDP). A metodologia proposta pelos autores integra as capacidades exploratórias e de otimização das redes de Hopfield com as técnicas de estimativa probabilística do EDA da seguinte forma:

- **Redes Neurais de Hopfield Competitivas (DCHNN)**: As DCHNNs são utilizadas para gerar soluções iniciais para o MDP. Essas redes, baseadas na dinâmica competitiva entre os neurônios, são capazes de selecionar subconjuntos promissores de elementos. No entanto, uma limitação crítica dessas redes é a tendência a ficarem presas em mínimos locais, o que

pode impedir a identificação de soluções globais ótimas.

- **Algoritmo de Estimativa de Distribuição (EDA)**: Para superar as limitações das DCHNNs, o EDA é aplicado após a geração das soluções iniciais. O EDA constrói um modelo probabilístico baseado nas soluções geradas, utilizando-o para amostrar novas soluções que são então reavaliadas. Esse processo iterativo permite mais exploração do espaço de soluções, ajudando o modelo a escapar de mínimos locais e a convergir para soluções de maior qualidade.

Os experimentos realizados pelos autores envolvem 120 problemas de *benchmark* com diferentes tamanhos, variando de 100 a 5000 elementos. Os resultados demonstram que o DCHNN-EDA supera outras variantes de DCHNN e métodos metaheurísticos tradicionais, como GRASP e Busca Tabu, em termos de qualidade das soluções e tempo de computação. A abordagem proposta se mostra eficaz para instâncias de grande escala, onde a capacidade de exploração do EDA combinada com a busca inicial das DCHNNs resulta em soluções de alta qualidade para o MDP. Os autores concluem que a combinação de EDA com DCHNN é uma estratégia poderosa para resolver o MDP e sugerem que essa abordagem pode ser aplicada a outros problemas de otimização combinatória complexos.

5 METODOLOGIA

Esta seção apresenta a metodologia adotada para alcançar o objetivo deste estudo, que consiste na aplicação de métodos de aprendizado supervisionado em instâncias do Problema da Diversidade Máxima.

A Seção 5.1 detalha o procedimento de aquisição dos dados para o treinamento e a avaliação dos modelos supervisionados, obtidos a partir da MDPLIB e resolvidos por métodos exatos. A Seção 5.2 descreve o processo de pré-processamento dos dados, que envolve a organização das instâncias e a extração das características dos grafos. Em seguida, a Seção 5.3 explica a estrutura de treinamento dos modelos supervisionados, incluindo as abordagens utilizadas e as justificativas para as escolhas feitas. A Seção 5.4 explica como será feito o tratamento de instâncias com diferentes dimensões. A Seção 5.5 discute o ajuste dos parâmetros dos modelos para melhorar sua performance. A Seção 5.7 aborda como os resultados serão gerados. A Seção 5.6 mostra os métodos utilizados para geração de novas instâncias. Por fim, a Seção 5.8 apresenta as ferramentas e bibliotecas utilizadas no desenvolvimento do trabalho.

5.1 Coleta de Dados

Para este estudo, foram selecionadas 315 instâncias da MDPLIB, uma biblioteca reconhecida para o problema da diversidade máxima. As instâncias foram extraídas dos trabalhos de (Glover *et al.*, 1998), (Silva *et al.*, 2004), (Martí *et al.*, 2010) e (Duarte; Martí, 2007) e abrangem as seguintes estruturas, onde n representa a quantidade total de vértices e m a quantidade de vértices que compõem o conjunto com a máxima diversidade:

SOM (Silva, Ochi e Martins): Este conjunto de dados consiste em 70 matrizes com números aleatórios entre 0 e 9, gerados a partir de uma distribuição uniforme discreta.

- SOM-a (Martí *et al.*, 2010): 50 instâncias com $n \in \{25, 50, 100, 125, 150\}$ e $m \in \{2, 7, 5, 15, 10, 30, 12, 37, 15, 45\}$.
- SOM-b (Silva *et al.*, 2004): 20 instâncias com $n \in \{100, 200, 300, 400, 500\}$ e $m \in \{10, 20, 30, 40, 50, 100, 150, 200\}$.

GKD (Glover, Kuo e Dhir): Este conjunto de dados consiste em 145 matrizes cujos valores foram calculados como distâncias euclidianas de pontos gerados aleatoriamente com coordenadas no intervalo de 0 a 10.

- GKD-a (Glover *et al.*, 1998): 75 instâncias com $n \in \{10, 15, 30\}$ e $m \in \{2, 3, 4, 6, 8, 9, 12,$

$18, 24\}$.

- GKD-b (Martí *et al.*, 2010): 50 instâncias com $n \in \{25, 50, 100, 125, 150\}$ e $m \in \{2, 7, 5, 15, 10, 30, 12, 37, 15, 45\}$.
- GKD-c (Duarte; Martí, 2007): 20 instâncias com $n = 500$ e $m = 50$.

MDG (Martí, Duarte e Gallego): Este conjunto de dados consiste em 100 matrizes com números reais selecionados aleatoriamente entre 0 e 10 de uma distribuição uniforme.

- MDG-a (Duarte; Martí, 2007): 40 instâncias com $n \in \{500, 2000\}$ e $m \in \{50, 200\}$.
- MDG-b (Duarte; Martí, 2007), (Gallego *et al.*, 2009): 40 instâncias com $n \in \{500, 2000\}$ e $m \in \{50, 200\}$.
- MDG-c: 20 instâncias com $n = 3000$ e $m \in \{300, 400, 500, 600\}$, as maiores instâncias presentes na MDPLIB, similares às utilizadas em (Palubeckis, 2007).

O método exato proposto por Soares (2018) foi aplicado às instâncias para calcular os valores ótimos de alocação dos vértices, com o objetivo de maximizar a diversidade nos conjuntos. Este método utiliza técnicas de t-linearização e suavização hiperbólica, permitindo resolver problemas quadráticos binários de forma eficiente e exata. As soluções obtidas foram usadas como rótulos para o treinamento dos modelos supervisionados. Contudo, é importante notar que esse método pode não garantir os valores ótimos para instâncias de grande escala.

Além do método exato, foi utilizada a metaheurística Têmpera Simulada, do inglês Simulated Annealing (SA), conforme descrito por Kirkpatrick *et al.* (1983), para gerar soluções ótimas aproximadas para instâncias maiores. O algoritmo SA se baseia em uma analogia com o processo de recozimento, onde, a cada iteração, um movimento aleatório é gerado (uma troca entre um vértice selecionado e um não selecionado). Se o movimento for aprimorado, ele é automaticamente executado; caso contrário, pode ser aceito com uma certa probabilidade, utilizando uma distribuição de Boltzmann controlada por um parâmetro de temperatura. O algoritmo começa com uma temperatura inicial igual ao maior valor de distância, que vai sendo gradualmente reduzida ao longo das iterações. Esse método foi especialmente útil para instâncias maiores do problema, proporcionando soluções de alta qualidade.

5.2 Preparação dos Dados e Criação dos Dataframes

Após a coleta das instâncias de grafos, o próximo passo foi organizar os dados em estruturas adequadas para a análise e o treinamento dos modelos. Para isso, foram criados dataframes que organizaram as características dos grafos de maneira estruturada, facilitando o

processo de aprendizado.

As instâncias foram modeladas utilizando diferentes representações dos grafos, incluindo:

- **Matriz de Adjacência**: Uma matriz A, onde A_{ij} indica a presença (ou ausência) de uma aresta entre os vértices v_i e v_j. Em grafos ponderados, A_{ij} armazena o peso da aresta correspondente.
- **Lista de Adjacência**: Uma lista de listas, onde cada vértice v_i contém uma lista dos vértices adjacentes a ele, juntamente com as arestas que os conectam.

Após a modelagem, cada grafo foi processado para extrair as seguintes características:

- **Grau (Ponderado)**: Reflete o número total de arestas conectadas a um vértice, levando em consideração o peso das arestas. Essa medida fornece uma ideia da conectividade global de cada vértice dentro do grafo.
- **Peso Médio das Arestas**: Representa o peso médio das arestas em todo o grafo, oferecendo uma visão geral da intensidade média das conexões.
- **Desvio Padrão dos Pesos**: Mede a dispersão dos pesos das arestas em relação à média, indicando a variabilidade nas forças das conexões entre os vértices.
- **Peso Máximo das Arestas**: Reflete o maior peso entre todas as arestas do grafo, destacando a conexão mais forte entre dois vértices.
- **Peso Mínimo das Arestas**: Indica o menor peso entre as arestas do grafo, capturando as conexões mais fracas.
- **Soma dos Graus dos Vizinhos**: Soma dos graus dos vizinhos de cada vértice, ou seja, o número de conexões de cada vizinho, fornecendo uma medida da conectividade local do vértice.

Além dessas, várias outras características foram extraídas, como **centralidade de grau**, **centralidade de intermediação**, **coeficiente de aglomeração**, **coeficiente de proximidade** e **densidade local**. Contudo, devido à alta complexidade de extração dessas características em grafos não esparsos (como os do problema da Diversidade Máxima, que são completos e conexos), decidiu-se por simplificar as características extraídas para garantir um equilíbrio entre desempenho e viabilidade computacional. Embora algumas dessas características adicionais possam contribuir para melhorias nos resultados, elas não justificariam o aumento significativo no tempo de processamento. O objetivo deste trabalho é treinar modelos que possam atuar de

maneira comparativa às heurísticas e metaheurísticas da literatura, justificando a escolha das características extraídas de forma mais simples.

Finalmente, foi adicionada uma coluna denominada “Conjuntos” em cada dataframe, que contém uma variável binária x_i para cada vértice v_i. Essa variável indica se o vértice pertence ao conjunto de máxima diversidade, sendo definida da seguinte forma:

$$x_i = \begin{cases} 1, & \text{se } v_i \text{ pertence ao conjunto de máxima diversidade;} \\ 0, & \text{caso contrário.} \end{cases}$$

5.3 Treinamento dos Modelos Supervisionados

O treinamento dos modelos supervisionados foi realizado utilizando os *dataframes* preparados e as instâncias agrupadas, com o objetivo de prever a alocação dos vértices em conjuntos que maximizem a diversidade. Diversos modelos supervisionados foram aplicados, cada um com abordagens complementares e capacidades de generalização distintas, incluindo:

- **Redes Neurais Artificiais (Feed-Forward)**.
- **Árvores de Decisão e Random Forest**.
- **Regressão Logística**.
- **K-ésimo Vizinho mais Próximo (KNN)**.

A escolha dos modelos de aprendizado de máquina foi fundamentada no fato de que, após a seleção das características, os dados apresentaram uma leve curvatura linear. As características extraídas, como Grau (Ponderado), Peso Médio das Arestas, Desvio Padrão dos Pesos, Peso Máximo das Arestas, Peso Mínimo das Arestas, Soma dos Graus dos Vizinhos, introduziram uma leve separabilidade linear nos dados. Isso justifica a escolha de modelos eficazes em dados com esse perfil.

- Regressão Logística: Como o problema é binário e as características extraídas induziram uma leve curvatura linear, a Regressão Logística se mostrou uma escolha natural. Ela é eficaz em problemas binários e consegue capturar as relações entre os vértices com boa precisão.
- Redes Neurais Feed-forward: Quando a linearidade induzida não é suficiente, as Redes Neurais Feed-forward são uma boa escolha, pois lidam melhor com não linearidades nos dados, sendo eficazes em casos onde modelos lineares falham.

- Árvores de Decisão e Random Forest: Ambas as técnicas são boas para dados com interações complexas entre variáveis. Como as características extraídas podem interagir de maneira não linear, as Árvores de Decisão e Random Forest são adequadas para modelar essas relações, dividindo os dados em regiões mais homogêneas.
- K-ésimo Vizinho mais Próximo (KNN): O KNN foi escolhido por sua capacidade de classificar vértices com base na proximidade, o que facilita a previsão de quais vértices pertencem ao conjunto de máxima diversidade com base em sua similaridade com outros vértices.

Adicionalmente, como as instâncias do problema da Diversidade Máxima possuem uma quantidade fixa de vértices definidos para o conjunto com máxima diversidade, mas o tamanho desse conjunto é uma escolha arbitrária (geralmente definida como uma margem de 5% a 40% do tamanho do grafo), a acurácia dos modelos não pode garantir por si só que o número correto de vértices será alocado. Os modelos podem alocar mais ou menos vértices do que o esperado.

Para lidar com essa limitação, foi implementado um mecanismo de ranqueamento de probabilidades com base na atuação dos modelos. Essa abordagem faz uso das distribuições de probabilidade geradas por cada modelo, organizando-as em um ranking de acordo com a probabilidade associada a cada vértice. A autonomia do modelo na previsão não é anulada, mas ajustes são feitos considerando a saída do modelo. Caso o modelo preveja uma quantidade de vértices diferente da necessária (por exemplo, prevê 18 vértices quando o ideal são 20), o ranking é usado para adicionar os vértices ausentes, começando pelos que têm as maiores probabilidades. Caso o modelo preveja mais vértices do que o esperado (por exemplo, prevê 22 vértices ao invés de 20), uma poda é realizada, removendo os vértices com as menores probabilidades.

Esse processo garante que o número de vértices no conjunto de máxima diversidade seja ajustado para o tamanho correto, independentemente da previsão inicial do modelo. Importante destacar que todos os modelos de predição já geram uma distribuição de probabilidade por natureza, sendo apenas armazenada e organizada para executar os ajustes.

5.4 Tratamento de Instâncias com Diferentes Dimensões

Uma das dificuldades ao aplicar modelos de aprendizado de máquina em grafos é a incompatibilidade de dimensões entre as instâncias de treino e teste. Isso ocorre quando as instâncias de treino são representadas por matrizes de adjacência $n \times n$, enquanto uma nova

instância pode ter um tamanho diferente. Para lidar com essa discrepância, foram aplicadas duas abordagens principais:

- **Interpolação de Matrizes de Adjacência**: Neste método, as colunas faltantes nas instâncias de grafo são preenchidas com zeros, uma vez que, em uma matriz de adjacência, o valor zero indica a ausência de arestas. Embora simples, essa abordagem pode ser menos eficiente em termos de memória e processamento, especialmente quando as instâncias de grafo possuem tamanhos significativamente diferentes.
- **Redução de Dimensionalidade com PCA**: O método de Análise de Componentes Principais (PCA) foi aplicado tanto ao conjunto de treinamento quanto ao de teste. O PCA reduz as instâncias para um número fixo de componentes principais, mantendo as informações mais relevantes e garantindo a compatibilidade entre os dados. Testes com diferentes quantidades de componentes (1, 10, 15, 25, 50, 75, 100, 150, 250 e 500 componentes) foram realizados. O único caso em que houve perda de resultados foi quando a redução foi feita para 1 componente, mas para 10 a 500 componentes os resultados foram equivalentes, com a diferença principal sendo o aumento do custo computacional e de memória à medida que o número de componentes aumentava. Com isso, foi decidido utilizar 10 componentes, pois esse número apresentou um bom equilíbrio entre desempenho e viabilidade de processamento.

Entre as duas abordagens, o PCA foi escolhido como a solução mais eficiente, pois, além de ser mais prática, apresentou um desempenho superior e um custo de memória significativamente menor em comparação com a interpolação de matrizes.

5.5 Ajuste de Hiperparâmetros

Os hiperparâmetros dos modelos supervisionados foram ajustados para otimizar o desempenho e garantir que cada modelo estivesse adequado à tarefa específica de maximização da diversidade em grafos. O ajuste foi realizado por meio de técnicas de busca em grade combinadas com validação cruzada. Os principais hiperparâmetros ajustados incluem:

- **Redes Neurais**: ajuste do número de camadas, quantidade de neurônios por camada, taxa de aprendizado, função de ativação (ReLU ou Sigmoid), e algoritmo de otimização (Adam ou SGD).
- **Árvores de Decisão e Random Forest**: ajuste da profundidade máxima das árvores, número mínimo de amostras por folha, número de árvores na floresta e critério de divisão

(Gini ou entropia).

- **Regressão Logística**: ajuste do parâmetro de regularização C, que controla a penalização por complexidade do modelo.
- **kNN**: ajuste do valor de k (número de vizinhos) e da métrica de distância para determinar os vizinhos mais próximos.

Além disso, para melhorar a precisão dos modelos e lidar com o desbalanceio das classes, foram aplicadas técnicas de **balanceamento de classes**. O método **SMOTE** (Synthetic Minority Over-sampling Technique) foi utilizado para aumentar a classe minoritária (classe 1, correspondente ao conjunto de máxima diversidade), enquanto o **RandomUnderSampler** foi empregado para reduzir a classe majoritária (classe 0). Essas técnicas ajudaram a mitigar o impacto negativo do desbalanceamento, permitindo que os modelos aprendessem de maneira mais eficiente.

5.6 Geração de Novas Instâncias Para Teste

A escolha dos métodos de geração de instâncias para este estudo teve como objetivo garantir que os modelos de aprendizado de máquina fossem testados em um conjunto variado de grafos, de modo a avaliar sua capacidade de generalização. Para isso, foram selecionados métodos distintos da literatura, a fim de fornecer instâncias com características de distribuição de pesos diferentes.

Para a realização dos novos testes, diferentes métodos foram utilizados para gerar instâncias de grafos com características variadas. Cada um desses métodos foi projetado para simular diferentes tipos de distribuições de pesos nas arestas do grafo, possibilitando a análise de como os modelos de aprendizado de máquina se comportam diante dessas variações. Abaixo, segue a descrição de como cada um desses métodos foi implementado e o que eles buscam representar em termos de estrutura do grafo.

- 1. Geração de Pesos de Hamming: Gera pesos baseados na distância de Hamming entre vetores binários. Para cada vértice, é gerado um vetor binário aleatório. A distância de Hamming entre dois vetores binários é dada pela soma das posições onde os valores dos elementos são diferentes.
- 2. Geração de Pesos Geométricos: Gera pesos baseados em distâncias geométricas entre vértices em um espaço bidimensional. A abordagem é similar à geração de pesos euclidianos, mas se foca em distâncias entre vértices que estão dispostos em um plano

com coordenadas aleatórias entre 0 e 100. Esse método é útil para simular problemas em que as distâncias espaciais ou geométricas entre os vértices são críticas, como em redes de sensores ou redes espaciais, além de utilizar escalar de distribuição diferentes para ver como os modelos se comportam.

- 3. Geração de Pesos de Preferência: Gera pesos baseados em um modelo de preferência. Para cada par de vértices i e j, é gerado um número aleatório entre 0 e 1, representando a "preferência" entre esses vértices. Esse método de criação de instâncias é útil para representar cenários como sistemas de recomendação, onde a interação entre os vértices reflete preferências ou afinidades.

Cada método foi escolhido com base na necessidade de simular diferentes tipos de estrutura de grafo em diferentes escalas das instâncias que foram utilizadas para treinar os modelos, para verificar e garantir que os modelos generalizam bem para novos casos.

5.7 Análise dos Resultados

Após o ajuste dos hiperparâmetros, os modelos foram aplicados ao conjunto de teste para gerar as previsões finais. A avaliação dos resultados foi baseada na capacidade dos modelos de classificar corretamente os vértices nos conjuntos que maximizam a diversidade. As predições foram comparadas com os valores ótimos e aproximados, calculados pelo método exato de Soares (2018) e pela metaheurística Têmpera Simulada (Simulated Annealing - SA).

- **Predições de Alocação e Valor da Diversidade Máxima**: Os modelos supervisionados foram avaliados pela precisão das alocações de vértices, comparando as soluções obtidas com as soluções ótimas calculadas pelos métodos exatos e metaheurísticos.

A métrica principal utilizada para avaliar a performance dos modelos foi a proximidade das soluções obtidas em relação à solução aproximada da ótima, medida em termos de percentagem de acerto no valor de diversidade máxima. As soluções geradas pela metaheurística SA foram comparadas com as soluções ótimas obtidas pelo algoritmo exato de Soares (2018), sendo ambas usadas como referência para avaliar os modelos de aprendizado de máquina.

5.8 Ferramentas e Implementação

O desenvolvimento e implementação dos métodos descritos foram realizados utilizando a linguagem de programação *Python*. As principais bibliotecas e ferramentas utilizadas

incluem:

- **Pandas e NumPy**: utilizadas para manipulação de dados e operações numéricas, oferecendo estruturas como *dataframes* e matrizes, além de permitir cálculos eficientes e manipulação de grandes volumes de dados. Essas bibliotecas foram escolhidas pela sua capacidade de lidar com dados de forma estruturada e pela otimização no processamento de operações vetorizadas, fundamentais para o trabalho com grandes conjuntos de dados.
- **Scikit-Learn**: implementação de algoritmos de aprendizado supervisionado e não supervisionado, com ferramentas para pré-processamento, ajuste de hiperparâmetros e validação de modelos. Foi escolhida por ser uma das bibliotecas mais completas e de fácil utilização para Aprendizado de Máquina, permitindo experimentação rápida e avaliação de diferentes modelos.
- **PyTorch**: ferramenta utilizada para construção e treinamento de redes neurais. A escolha do PyTorch se deve à sua flexibilidade e eficiência no desenvolvimento de redes neurais, permitindo a construção de modelos dinâmicos e facilitando o ajuste e a depuração de redes complexas.
- **Matplotlib e Seaborn**: ferramentas de visualização para criação de gráficos, com Seaborn oferecendo gráficos estatísticos mais delicados. Essas bibliotecas foram escolhidas pela sua ampla gama de recursos gráficos e pela facilidade em gerar visualizações informativas, fundamentais para a análise de dados e comunicação de resultados.
- **NetworkX**: biblioteca para modelagem, extração de características e análise de grafos. Foi escolhida por ser uma das mais eficientes para manipulação de grafos, oferecendo funções específicas para análise estrutural e extração de métricas importantes para o problema abordado.
- **Jupyter Notebooks**: ambiente interativo para desenvolvimento, experimentação e documentação dos modelos, integrando código e visualizações. O Jupyter foi escolhido por sua capacidade de combinar desenvolvimento e documentação de maneira prática e interativa, facilitando a visualização de resultados em tempo real.

6 RESULTADOS

Esta seção apresenta os resultados obtidos a partir da aplicação de modelos de aprendizado de máquina ao problema da diversidade máxima em grafos. O objetivo principal foi avaliar a capacidade dos modelos em alocar corretamente os vértices em um conjunto de máxima diversidade e, com isso, alcançar um valor de diversidade máxima próximo ao valor ótimo. A seguir, são apresentados os resultados dos testes realizados, as métricas utilizadas para avaliação, e uma análise crítica sobre o desempenho dos modelos.

6.1 Treinamento com Instâncias Aleatórias

Os modelos supervisionados de aprendizado foram treinados com as instâncias da MDPLib, sendo divididos em treinamento e teste. O conjunto de dados inicial foi composto por 255 instâncias de grafos com tamanhos variando de 10 a 500 vértices, e os testes realizados visaram avaliar o desempenho dos modelos em diferentes cenários de treinamento.

Três testes arbitrários foram realizados utilizando diferentes quantidades de instâncias para o treinamento, com a avaliação sendo feita nas instâncias restantes. Os testes realizados foram:

- Teste 1: Treinamento com 100 instâncias e avaliação nas 155 restantes.
- Teste 2: Treinamento com 150 instâncias e avaliação nas 105 restantes.
- Teste 3: Treinamento com 200 instâncias e avaliação nas 55 restantes.

Pode ser observado nas Tabelas 1, 2 e 3 abaixo que, embora a acurácia dos modelos tenha ficado na faixa de 0.45 a 0.60, os valores de diversidade máxima alcançados pelos modelos foram consideravelmente mais promissores. A acurácia, por si só, não foi um indicador perfeito da qualidade das previsões, evidenciando que ela não deve ser a principal métrica para avaliar o desempenho no contexto do problema da diversidade máxima. Em vez disso, a capacidade dos modelos em alocar corretamente os vértices e atingir a máxima diversidade foi uma métrica mais relevante.

Tabela 1 – Treinamento com 100 Instâncias

Instâncias de Teste	Valor Objetivo (Metaheurística)	Árvores de Decisão		Redes Neurais		Floresta Randômica		Regressão Logística		KNN	
		Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor
SOM-a-2-(n25 e m2)	9	0.45	6	0.40	5	0.50	6	0.45	6	0.52	6
SOM-a-5-(n25 e m2)	9	0.60	7	0.60	8	0.60	8	0.60	8	0.62	7
SOM-a-7-(n25 e m7)	150	0.50	105	0.60	90	0.62	93	0.60	90	0.62	93
SOM-a-9-(n25 e m7)	147	0.50	88	0.58	85	0.60	88	0.62	91	0.62	92
SOM-a-25-(n100 e m10)	337	0.52	202	0.62	216	0.63	212	0.65	219	0.62	210
SOM-a-27-(n100 e m30)	2477	0.54	1730	0.63	1550	0.64	1587	0.65	1600	0.64	1580
SOM-a-32-(n125 e m12)	472	0.51	241	0.62	292	0.61	288	0.63	297	0.63	297
SOM-a-37-(n125 e m37)	3648	0.53	1932	0.63	2290	0.64	2334	0.65	2377	0.62	2268
SOM-a-42-(n150 e m15)	719	0.52	374	0.62	446	0.61	440	0.64	460	0.62	446
SOM-a-46-(n150 e m45)	5379	0.58	3120	0.64	3443	0.63	3389	0.64	3453	0.62	3332
SOM-b-6-(n200 e m40)	4432	0.53	2350	0.63	2791	0.62	2747	0.64	2857	0.62	2747
SOM-b-8-(n200 e m60)	16161	0.55	8884	0.62	10015	0.61	9860	0.63	10134	0.62	10034
SOM-b-9-(n300 e m30)	2676	0.50	1338	0.58	1553	0.60	1606	0.61	1635	0.62	1650
SOM-b-12-(n300 e m120)	35745	0.53	18946	0.60	21447	0.61	21756	0.63	22461	0.62	22161
SOM-b-14-(n400 e m80)	16824	0.54	9070	0.62	10442	0.63	10674	0.64	10876	0.62	10442
SOM-b-15-(n400 e m120)	36106	0.55	19858	0.63	22786	0.64	23190	0.65	23469	0.62	22342
SOM-b-16-(n400 e m160)	62217	0.56	34842	0.63	39197	0.62	38574	0.64	39859	0.62	38573
SOM-b-17-(n500 e m50)	7074	0.54	3819	0.61	4315	0.62	4384	0.64	4510	0.62	4386
SOM-b-18-(n500 e m100)	26062	0.56	14594	0.63	16459	0.62	16175	0.64	16740	0.62	16175
SOM-b-19-(n500 e m150)	56308	0.55	31069	0.61	34368	0.62	34910	0.64	35957	0.62	34910

Fonte: elaborada pelo autor.

A análise dos resultados revela uma tendência clara: à medida que mais instâncias eram incluídas no treinamento, a proximidade dos valores previstos em relação à solução ótima aumentava significativamente. Por exemplo, no Teste 1, com 100 instâncias de treinamento, a proximidade com a solução ótima variou entre 70% e 75%. No Teste 2, com 150 instâncias, essa proximidade subiu para 82% a 87%, e no Teste 3, com 200 instâncias, o desempenho dos modelos melhorou ainda mais, alcançando uma proximidade de 93% a 97%. Em alguns casos, os modelos conseguiram atingir até 100% da solução ótima, demonstrando que, com a adição de mais dados, o modelo se torna progressivamente mais capaz de identificar soluções ótimas ou muito próximas delas.

Tabela 2 – Treinamento com 150 Instâncias

Instâncias de Teste	Valor Objetivo (Metaheurística)	Árvores de Decisão		Redes Neurais		Floresta Randômica		Regressão Logística		KNN	
		Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor
SOM-a-2-(n25 e m2)	7	0.4536	5	0.3945	6	0.4936	5	0.5205	6	0.5205	5
SOM-a-5-(n25 e m2)	7	0.6045	6	0.6110	5	0.6048	6	0.6264	6	0.6264	5
SOM-a-7-(n25 e m7)	129	0.5016	108	0.5958	109	0.6100	106	0.6271	110	0.6271	111
SOM-a-9-(n25 e m7)	125	0.5082	107	0.5870	107	0.6052	104	0.6258	109	0.6258	105
SOM-a-25-(n100 e m10)	294	0.5279	248	0.6111	256	0.6375	251	0.6279	255	0.6279	246
SOM-a-27-(n100 e m30)	2032	0.5388	1746	0.6202	1699	0.6425	1698	0.6281	1770	0.6281	1745
SOM-a-32-(n125 e m12)	406	0.5102	353	0.6232	346	0.6100	347	0.6243	364	0.6243	351
SOM-a-37-(n125 e m37)	3128	0.5242	2597	0.6395	2727	0.6507	2731	0.6090	2713	0.6090	2742
SOM-a-42-(n150 e m15)	625	0.5228	543	0.6211	531	0.6212	536	0.6124	553	0.6124	537
SOM-a-46-(n150 e m45)	4723	0.5761	3988	0.6440	4101	0.6297	3996	0.6085	4033	0.6085	4066
SOM-b-6-(n200 e m40)	3838	0.5251	3328	0.6300	3317	0.6311	3148	0.6242	3379	0.6242	3204
SOM-b-8-(n200 e m60)	13854	0.5476	12182	0.6218	12067	0.6034	11966	0.6313	12354	0.6313	11983
SOM-b-9-(n300 e m30)	2288	0.5059	1876	0.5786	2000	0.5881	1893	0.6252	1945	0.6252	1881
SOM-b-12-(n300 e m120)	30950	0.5258	26583	0.6116	26788	0.6161	26538	0.6310	26915	0.6310	25385
SOM-b-14-(n400 e m80)	14325	0.5397	12388	0.6192	12314	0.6222	12029	0.6231	12539	0.6231	11845
SOM-b-15-(n400 e m120)	30916	0.5411	26266	0.6345	26336	0.6467	26167	0.6280	26841	0.6280	26754
SOM-b-16-(n400 e m160)	51418	0.5499	44585	0.6390	44308	0.6221	45143	0.6078	44825	0.6078	44065
SOM-b-17-(n500 e m50)	6198	0.5359	5256	0.5983	5154	0.6131	5439	0.6126	5542	0.6126	5178
SOM-b-18-(n500 e m100)	22660	0.5496	19564	0.6394	19739	0.6185	18800	0.6202	20378	0.6202	19095
SOM-b-19-(n500 e m150)	48948	0.5485	43033	0.6158	42175	0.6134	42343	0.6315	43767	0.6315	41782

Fonte: elaborada pelo autor.

Esse aumento contínuo na proximidade com a solução ótima esperada reforça a ideia de que os modelos não sofreram viés de aprendizado. Em vez disso, o aumento no número de instâncias no conjunto de treinamento parece ter fortalecido a capacidade dos modelos em aprender e generalizar melhor sobre as instâncias de teste. Em particular, a tendência de melhoria ao longo dos testes sugere que, ao contrário de ocorrer um sobreajuste (overfitting), os modelos se beneficiaram com mais dados, atingindo níveis mais altos de precisão.

Porém, um ponto importante a ser observado é que, em certos casos, o modelo obteve bons resultados mesmo quando houve diferenças na distribuição dos vértices. Por exemplo, em alguns grafos, a solução ótima poderia ser alcançada ao selecionar vértices específicos como 1 e 5 ou 2 e 8, ainda que os vértices fossem distribuídos de forma distinta. Isso indica que, embora o modelo possa ter "errado"na escolha exata dos vértices, ele ainda foi capaz de identificar o conjunto com a máxima diversidade corretamente, o que é uma evidência positiva de sua robustez.

Tabela 3 – Treinamento com 200 Instâncias

Instâncias de Teste	Valor Objetivo (Metaheurística)	Árvores de Decisão		Redes Neurais		Floresta Randômica		Regressão Logística		KNN	
		Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor	Acurácia	Valor
SOM-a-2-(n25 e m2)	6	0.4536	5	0.3945	5	0.4936	5	0.5205	5	0.5205	5
SOM-a-5-(n25 e m2)	6	0.6045	5	0.6110	5	0.6048	5	0.6264	5	0.6264	5
SOM-a-7-(n25 e m7)	114	0.5016	101	0.5958	99	0.6100	102	0.6271	108	0.6271	103
SOM-a-9-(n25 e m7)	111	0.5082	99	0.5870	101	0.6052	97	0.6258	105	0.6258	99
SOM-a-25-(n100 e m10)	258	0.5279	234	0.6111	226	0.6375	231	0.6279	255	0.6279	228
SOM-a-27-(n100 e m30)	1861	0.5388	1669	0.6202	1648	0.6425	1640	0.6281	1811	0.6281	1639
SOM-a-32-(n125 e m12)	366	0.5102	317	0.6232	319	0.6100	318	0.6243	359	0.6243	320
SOM-a-37-(n125 e m37)	2834	0.5242	2502	0.6395	2462	0.6507	2511	0.6090	2734	0.6090	2522
SOM-a-42-(n150 e m15)	548	0.5228	499	0.6211	504	0.6212	491	0.6124	542	0.6124	485
SOM-a-46-(n150 e m45)	4269	0.5761	3856	0.6440	3746	0.6297	3845	0.6085	4116	0.6085	3779
SOM-b-6-(n200 e m40)	3396	0.5251	2938	0.6300	3008	0.6311	2995	0.6242	3207	0.6242	3055
SOM-b-8-(n200 e m60)	12580	0.5476	10973	0.6218	11556	0.6034	10854	0.6313	11894	0.6313	11105
SOM-b-9-(n300 e m30)	2101	0.5059	1861	0.5786	1915	0.5881	1910	0.6252	1987	0.6252	1847
SOM-b-12-(n300 e m120)	26738	0.5258	24480	0.6116	23762	0.6161	23978	0.6310	25742	0.6310	24036
SOM-b-14-(n400 e m80)	12728	0.5397	11262	0.6192	11298	0.6222	11379	0.6231	12467	0.6231	11362
SOM-b-15-(n400 e m120)	26923	0.5411	23368	0.6345	24678	0.6467	23421	0.6280	25364	0.6280	24513
SOM-b-16-(n400 e m160)	44420	0.5499	38484	0.6390	39209	0.6221	39743	0.6078	41798	0.6078	38234
SOM-b-17-(n500 e m50)	5463	0.5359	4729	0.5983	4709	0.6131	4755	0.6126	5381	0.6126	4942
SOM-b-18-(n500 e m100)	20426	0.5496	17944	0.6394	18155	0.6185	18043	0.6202	19339	0.6202	18674
SOM-b-19-(n500 e m150)	42976	0.5485	37192	0.6158	39155	0.6134	38035	0.6315	41101	0.6315	37234

Fonte: elaborada pelo autor.

6.2 Análise Estatística das Instâncias

Antes de continuar com a aplicação dos modelos nas instâncias maiores, foi realizada uma análise estatística descritiva das 255 instâncias utilizadas até esse ponto. Esta análise ajuda a entender como os dados estão organizados e se há padrões nas características dos grafos.

Tabela 4 – Estatísticas descritivas das Instâncias

Características	mean	std	min	25%	50%	75%	max
Quantidade de Vértices	1.96×10^2	1.50×10^2	0.0	6.2×10^1	1.63×10^2	3.23×10^2	4.99×10^2
Grau (Ponderado)	5.75×10^4	1.01×10^5	7.5×10^1	2.38×10^3	5.72×10^3	1.81×10^4	2.75×10^5
Peso Médio das Arestas	1.29×10^2	1.99×10^2	2.76	4.87	1.22×10^1	1.61×10^2	5.51×10^2
Desvio Padrão dos Pesos	6.68×10^1	1.16×10^2	1.56	2.76	2.90	2.39×10^1	3.10×10^2
Peso Máximo das Arestas	2.46×10^2	3.98×10^2	7.0	9.96	1.81×10^1	2.09×10^2	1.00×10^3
Peso Mínimo das Arestas	1.12×10^1	2.83×10^1	0.0	0.0	6.40	5.53	1.73×10^2
Soma dos Graus dos Vizinhos	2.81×10^7	5.06×10^7	2.49×10^3	1.12×10^6	1.25×10^6	3.21×10^6	1.25×10^8

Fonte: Elaborado pelo autor.

Como pode ser observado na Tabela 4, as características extraídas das 255 instâncias da MDPLib apresentam uma variedade considerável, com médias e desvios padrão que indicam uma ampla gama de estruturas de grafos. Embora os grafos da MDPLib, em sua forma original, possuam uma distribuição uniforme e constante nas suas conexões e pesos, a extração e escolha das características (como o Grau (Ponderado), Peso Médio das Arestas, Desvio Padrão dos Pesos, Peso Máximo das Arestas, Peso Mínimo das Arestas, Soma dos Graus dos Vizinhos) permitiu

induzir uma leve linearidade nos dados. Isso foi suficiente para que os modelos, especialmente a regressão logística, pudessem atuar de forma eficiente.

Ao observar a tabela, vemos que, apesar de uma grande variação no número de vértices (com uma média de 196 vértices e um desvio padrão de 150) e na conectividade das instâncias (grau ponderado médio de 57.500, com um desvio padrão de 101.000), a estrutura geral dos grafos não é excessivamente complexa. Esse tipo de estrutura favoreceu a adaptação dos modelos, uma vez que as características extraídas permitiram identificar padrões de conectividade e peso com um nível de simplicidade suficiente para os modelos de aprendizado, evitando o risco de sobreajuste.

A regressão logística, em particular, se destacou nesse cenário. Embora o problema da diversidade máxima em grafos envolva relações complexas entre os vértices e suas conexões, a linearidade induzida pelas características extraídas fez com que a regressão logística fosse eficaz na modelagem da probabilidade de um vértice pertencer ao conjunto de máxima diversidade. A regressão logística é um modelo particularmente eficaz para problemas de classificação binária com fronteiras de decisão lineares, e a relação entre as características extraídas e a alocação de vértices foi suficientemente linear para permitir que o modelo gerasse boas previsões.

A simplicidade das características extraídas não só favoreceu o bom desempenho da regressão logística, mas também contribuiu para o desempenho dos modelos em diferentes instâncias de treinamento. A análise dos resultados mostrou que a proximidade das soluções em relação ao valor ótimo aumentava à medida que mais instâncias eram incluídas no treinamento, refletindo a capacidade da regressão logística de se adaptar de forma eficaz a dados adicionais sem sofrer com viés de aprendizado.

6.3 Ajustes de Hiperparâmetros

Os seguintes ajustes de hiperparâmetros foram aplicados aos modelos de aprendizado de máquina, utilizando a biblioteca GridSearchCV do pacote scikit-learn, que realiza uma busca automática dos melhores parâmetros por meio de validação cruzada. Os melhores parâmetros encontrados para cada modelo foram os seguintes:

1. Floresta Randômica
    - **pesos das classes**: Foi utilizado o parâmetro *class_weight='balanced'* para lidar com o desbalanceamento entre as classes.
    - **critério**: O critério de divisão dos nós foi ajustado para *criterion='gini'*, pois se

mostrou mais eficiente em termos de tempo computacional do que o critério *entropy*.

- **número de estimadores**: O número de árvores foi ajustado para 200, o que resultou em bom desempenho equilibrado, sem comprometer o tempo de execução.
- **profundidade máxima**: Foi mantido o valor *None*, permitindo que as árvores crescessem até sua profundidade máxima.
- **número mínimo de amostras por folha**: O valor foi ajustado para 1, permitindo modelar as divisões de forma mais detalhada.

2. Regressão Logística
    - **pesos das classes**: O parâmetro *class_weight='balanced'* foi utilizado para lidar com o desbalanceamento entre as classes.
    - **regularização**: O parâmetro de regularização *C* foi ajustado para 1, com base nos testes que indicaram ser o valor ideal para balancear viés e variância.
    - **número máximo de iterações**: O número de iterações foi configurado para 1000, garantindo que o modelo convergisse adequadamente.
3. K-Nearest Neighbors (KNN)
    - **número de vizinhos**: O número de vizinhos foi ajustado para 3, proporcionando um bom equilíbrio entre precisão e complexidade computacional.
    - **métrica de distância**: A métrica de distância foi ajustada para *manhattan*, pois apresentou maior robustez frente a outliers presentes nas instâncias dos dados.
4. Árvore de Decisão
    - **pesos das classes**: O parâmetro *class_weight='balanced'* foi utilizado para lidar com o desbalanceamento entre as classes.
    - **critério**: O critério de divisão foi ajustado para *criterion='entropy'*, proporcionando uma árvore mais equilibrada e explicativa.
    - **profundidade máxima**: Foi mantido o valor *None*, permitindo que a árvore crescesse até sua profundidade máxima.
    - **número mínimo de amostras por folha**: O valor foi ajustado para 1, permitindo maior detalhamento nas divisões da árvore.
5. Redes Neurais Feed-Forward
    - **número de camadas ocultas**: Foram testadas redes com 1 a 3 camadas ocultas, contendo entre 32 e 128 neurônios por camada, para equilibrar a complexidade do modelo e o tempo de treinamento.

- **função de ativação**: A função *ReLU* (Rectified Linear Unit) foi escolhida devido à sua capacidade de promover uma convergência mais rápida e bom desempenho em tarefas de classificação.
- **taxa de aprendizado**: A taxa foi ajustada para 0.01, evitando tanto o subajuste quanto o sobreajuste, permitindo uma convergência eficiente.
- **otimizador**: Foi utilizado o otimizador *Adam*, que combina as vantagens de outros métodos como o AdaGrad e o RMSProp, e se mostrou altamente eficaz.
- **número de épocas**: O número de épocas foi configurado para 2500, com a possibilidade de aumento para 5000, caso o modelo não convergisse adequadamente.
- **regularização dropout**: Uma taxa de dropout entre 0.2 e 0.5 foi aplicada para evitar o sobreajuste e melhorar a capacidade de generalização.

6.4 Testes com Instâncias Grandes da MDPLib (2000 e 3000 Vértices)

Após os testes realizados com as instâncias menores, os modelos foram aplicados nas instâncias grandes da MDPLib, que incluem 40 instâncias de tamanho 2000 vértices e 20 instâncias de tamanho 3000 vértices. Essas instâncias maiores foram selecionadas para avaliar o desempenho dos modelos em cenários de maior complexidade, visando verificar se a capacidade de generalização dos modelos seria mantida à medida que o tamanho das instâncias aumentasse.

Os modelos foram treinados com as 255 instâncias anteriores, que variavam de 10 a 500 vértices. Surpreendentemente, os resultados obtidos nas instâncias maiores foram ainda mais promissores, com a proximidade da solução ótima alcançando um valor mínimo de 91%, conforme mostrado nas tabelas 5 e 6.

Tabela 5 – Resultados para Instâncias de Tamanho 2000

Instância de Teste	$\|V\|$	$\|S\|$	**Valor Objetivo (Metaheurística)**	**Valor Alcançado (Modelos)**	**Proximidade em Porcentagem**
MDG-a_21_n2000_m200	2000	200	113162	103067.95	91.08
MDG-a_22_n2000_m200	2000	200	112962	103156.90	91.32
MDG-a_23_n2000_m200	2000	200	113080	104746.00	92.63
MDG-a_24_n2000_m200	2000	200	113001	104299.92	92.30
MDG-a_25_n2000_m200	2000	200	113265	103580.84	91.45
MDG-a_26_n2000_m200	2000	200	113112	104900.07	92.74
MDG-a_27_n2000_m200	2000	200	113046	104137.98	92.12
MDG-a_28_n2000_m200	2000	200	113218	104069.99	91.92
MDG-a_29_n2000_m200	2000	200	113066	103692.83	91.71
MDG-a_30_n2000_m200	2000	200	113149	104798.60	92.62
MDG-a_31_n2000_m200	2000	200	113006	105005.18	92.92
MDG-a_32_n2000_m200	2000	200	113117	104814.21	92.66
MDG-a_33_n2000_m200	2000	200	113096	103562.01	91.57
MDG-a_34_n2000_m200	2000	200	113183	102973.89	90.98
MDG-a_35_n2000_m200	2000	200	113080	103242.04	91.30
MDG-a_36_n2000_m200	2000	200	113173	104005.99	91.90
MDG-a_37_n2000_m200	2000	200	113045	104566.63	92.50
MDG-a_38_n2000_m200	2000	200	113205	103605.22	91.52
MDG-a_39_n2000_m200	2000	200	113058	103504.60	91.55
MDG-a_40_n2000_m200	2000	200	113115	104190.23	92.11

Os resultados indicam que os modelos, quando adequadamente treinados, conseguem lidar de forma eficaz com instâncias de tamanhos maiores, com um desempenho excelente nas instâncias de 2000 e 3000 vértices. A proximidade com a solução ótima, que foi consistentemente acima de 91%, reforça a capacidade de generalização mesmo em problemas mais complexos e de maior escala.

A análise das instâncias de tamanho 3000, por exemplo, revela que, à medida que a complexidade do grafo aumenta, os modelos continuam a apresentar alta proximidade com os valores ótimos, alcançando até 95.66% de proximidade nas instâncias com 600 vértices de conjunto. Tais resultados demonstram que o treinamento realizado com dados de tamanho moderado (como os de 2000 vértices) pode ser generalizado para problemas com instâncias significativamente maiores, garantindo um bom desempenho nas predições mesmo com grafos de grande porte.

Tabela 6 – Resultados para Instâncias de Tamanho 3000

Instância de Teste	$\|V\|$	$\|S\|$	Valor Objetivo (Metaheurística)	Valor Alcançado (Modelos)	Proximidade em Porcentagem
MDG-c_1_n3000_m300	3000	300	24704176	22498093.08	91.07
MDG-c_2_n3000_m300	3000	300	24701619	22478473.29	91.00
MDG-c_3_n3000_m300	3000	300	24698742	22522782.83	91.19
MDG-c_4_n3000_m300	3000	300	24693815	22486187.94	91.06
MDG-c_5_n3000_m300	3000	300	24678847	22462686.54	91.02
MDG-c_6_n3000_m400	3000	400	43135740	39956635.96	92.63
MDG-c_7_n3000_m400	3000	400	43165190	39953699.86	92.56
MDG-c_8_n3000_m400	3000	400	43133072	39915344.83	92.54
MDG-c_9_n3000_m400	3000	400	43121905	39930884.03	92.60
MDG-c_10_n3000_m400	3000	400	43147020	39919622.90	92.52
MDG-c_11_n3000_m500	3000	500	66625758	62608224.79	93.97
MDG-c_12_n3000_m500	3000	500	66593447	62411378.53	93.72
MDG-c_13_n3000_m500	3000	500	66604075	62441320.31	93.75
MDG-c_14_n3000_m500	3000	500	66595684	62366858.07	93.65
MDG-c_15_n3000_m500	3000	500	66642407	62363964.47	93.58
MDG-c_16_n3000_m600	3000	600	95136120	90874021.82	95.52
MDG-c_17_n3000_m600	3000	600	95096043	90873778.69	95.56
MDG-c_18_n3000_m600	3000	600	95045765	90692668.96	95.42
MDG-c_19_n3000_m600	3000	600	95132938	90928062.14	95.58
MDG-c_20_n3000_m600	3000	600	95070981	90944900.42	95.66

6.5 Testes com Novas Instâncias

Novos métodos de geração de instâncias, como distâncias geométricas, método de Hamming e cálculo de preferência foram aplicados para avaliar a generalização dos modelos em grafos com características diferentes das instâncias originais. Para cada método, foram geradas 500 novas instâncias, com tamanhos variando de 10 a 500 vértices.

O método de distâncias geométricas gerou instâncias com distribuições que se mostraram desafiadoras para os modelos em alguns casos, especialmente quando os pesos das arestas eram representados por valores reais com várias casas decimais, configurando um ruído nos dados. Mesmo assim, os modelos conseguiram alcançar uma boa aproximação das soluções ótimas. Isso pode ser evidenciado pelos resultados obtidos para as novas instâncias, conforme apresentado na Tabela 7, que mostra a proximidade das soluções alcançadas pelos modelos em relação aos valores ótimos calculados pelas metaheurísticas.

Tabela 7 – Resultados para Novas Instâncias com o Método de Distâncias Geométricas

Instância de Teste	$\|V\|$	$\|S\|$	**Valor Objetivo (Metaheurística)**	**Valor Alcançado (Modelos)**	**Proximidade em Porcentagem**
geométricos_1_n10_m2	10	2	11.76191	10.67302	91.07
geométricos_2_n10_m2	10	2	11.99626	10.87512	91.00
geométricos_12_n15_m2	15	2	10.62354	9.66012	91.02
geométricos_25_n15_m4	15	4	49.75058	45.68616	91.19
geométricos_27_n15_m6	15	6	104.37467	95.23514	91.06
geométricos_35_n25_m2	25	2	11.66389	10.62443	91.02
geométricos_46_n25_m7	25	7	161.67912	149.75896	92.63
geométricos_58_n50_m12	50	12	476.83372001	440.48745	92.56
geométricos_70_n100_m10	100	10	351.27977	325.62511	92.54
geométricos_85_n100_m40	100	40	5315.61806	4932.47286	92.60
geométricos_97_n125_m37	125	37	4761.84777	4398.16045	92.52
geométricos_101_n150_m15	150	15	842.13384	786.06517	93.97
geométricos_105_n150_m15	150	15	839.29166001	776.23432	91.02
geométricos_117_n200_m20	200	20	1423.3287	1324.47435	93.72
geométricos_122_n200_m40	200	40	5522.2959003	5145.52345	93.75
geométricos_131_n200_m80	200	80	21202.19622005	19870.02012	93.65
geométricos_138_n300_m30	300	30	3466.92491006	3177.67798	93.58
geométricos_147_n300_m90	300	90	28191.85019001	25859.40957	95.52
geométricos_152_n300_m120	300	120	47633.29081002	43617.59161	91.02
geométricos_158_n400_m40	300	40	6008.6131005	5473.92591	95.56
geométricos_164_n400_m80	400	80	23540.7334005	21634.92991	91.02
geométricos_166_n400_m120	400	120	50646.48608999	48384.46014	95.42
geométricos_174_n400_m160	400	160	86101.3574007	79717.22674	91.02
geométricos_180_n500_m50	500	50	9292.87728	8450.11580	91.02
geométricos_185_n500_m100	500	100	35155.85274	32099.26715	91.02
geométricos_188_n500_m150	500	150	75453.44698002	71945.77702	95.58
geométricos_190_n500_m150	500	150	77700.33107999	70740.07503	91.02
geométricos_195_n500_m200	500	200	131597.5093512	122597.58706	95.66

Observe que, nos testes realizados com as instâncias criadas pelo método de distâncias geométricas, o desempenho dos modelos foi altamente satisfatório, com uma proximidade média das soluções em relação ao valor ótimo variando de 91% a 95%. As instâncias com tamanhos maiores (como as de 400 vértices ou mais) apresentaram um desempenho ainda melhor. Notavelmente, os modelos de aprendizado de máquina, especialmente a regressão logística, se mostraram eficazes mesmo diante das distorções causadas pelos pesos representados por valores reais, os quais poderiam ser considerados ruídos nos dados.

Para o método de preferências de diversidade, os resultados apresentados na Tabela 8 mostram que o desempenho dos modelos foi igualmente bom, com uma proximidade das soluções em relação ao valor ótimo variando de 89% a 92%. As instâncias com tamanhos maiores também apresentaram desempenho superior, refletindo a capacidade dos modelos em generalizar bem mesmo para instâncias com características mais complexas. O fato de os modelos terem

superado as distorções causadas pelos pesos reais reforça a observação de que os modelos foram treinados com sucesso, alcançando resultados muito próximos aos valores ótimos.

Tabela 8 – Resultados para Novas Instâncias com o Método de Preferências de Diversidade

Instância de Teste	$\|V\|$	$\|S\|$	**Valor Objetivo (Metaheurística)**	**Valor Alcançado (Modelos)**	**Proximidade em Porcentagem**
preferência_1_n10_m2	10	2	0.9990044502513654	0.896042341000354	89.60
preferência_2_n10_m2	10	2	0.9973742544049912	0.897010217175941	89.90
preferência_12_n15_m2	15	2	0.9951176263950358	0.896061692879725	89.95
preferência_25_n15_m4	15	4	5.074016937296909	4.576883424407438	90.20
preferência_27_n15_m6	15	6	11.599700917185796	10.731391499121953	90.10
preferência_35_n25_m2	25	2	0.9987941635450188	0.897053672461039	89.80
preferência_46_n25_m7	25	7	16.225029198082076	14.905837515803076	91.10
preferência_58_n50_m12	50	12	45.32294419931172	40.80469392286849	91.00
preferência_70_n100_m10	100	10	35.3553313132769	32.90656327313442	90.90
preferência_85_n100_m40	100	40	459.703833334976	417.9686709194444	91.00
preferência_97_n125_m37	125	37	402.7414104145026	365.32471301995734	90.95
preferência_101_n150_m15	150	15	78.34338371276168	71.80606090832862	91.80
preferência_105_n150_m15	150	15	76.40490860040632	69.30985374224656	90.70
preferência_117_n200_m20	200	20	135.41031929507227	123.99882113745338	91.70
preferência_122_n200_m40	200	40	486.8916756333932	445.4544290590341	91.80
preferência_131_n200_m80	200	80	1771.9477099832154	1620.9014827368507	91.60
preferência_138_n300_m30	300	30	292.9398334384042	267.5357981586093	91.50
preferência_147_n300_m90	300	90	2264.658646852521	2112.623294878586	92.00
preferência_152_n300_m120	300	120	3881.351558219324	3533.830984796063	90.60
preferência_158_n400_m40	300	40	499.5236416609262	455.0562403775305	92.10
preferência_164_n400_m80	400	80	1841.0756449571284	1685.7643121775734	90.50
preferência_166_n400_m120	400	120	3965.746772475766	3664.509990089259	92.00
preferência_174_n400_m160	400	160	6913.856715890424	6262.532581448218	90.70
preferência_180_n500_m50	500	50	771.779271468533	698.2385104172768	90.80
preferência_185_n500_m100	500	100	2834.944001686276	2577.675019033839	90.90
preferência_188_n500_m150	500	150	6159.00736974976	5672.472586899079	92.20
preferência_190_n500_m150	500	150	6126.867571730437	5553.404939147302	90.80
preferência_195_n500_m200	500	200	10689.0665164261	9820.279141381871	92.30

Finalmente, o método de Hamming, cujos resultados estão apresentados na Tabela 9, também demonstrou a eficiência dos modelos. Embora o desempenho tenha sido um pouco inferior nas instâncias com pesos reais, as instâncias com pesos inteiros apresentaram proximidade de solução ótima de 100%, com valores alcançados idênticos aos valores ótimos, como mostrado nas instâncias de 10 vértices. Em geral, as instâncias maiores também apresentaram desempenho superior, com proximidade das soluções variando entre 90% e 100%. O modelo mostrou-se particularmente robusto para instâncias menores e de tamanhos intermediários, atingindo índices de proximidade excepcionais, o que reflete sua capacidade de generalizar bem para grafos com características variadas.

Tabela 9 – Resultados para Novas Instâncias com o Método de Hamming

Instância de Teste	$\|V\|$	$\|S\|$	Valor Objetivo (Metaheurística)	Valor Alcançado (Modelos)	Proximidade em Porcentagem
hamming_1_n10_m2	10	2	8	8	100.00
hamming_2_n10_m2	10	2	7	6	85.71
hamming_12_n15_m2	15	2	12	11	91.67
hamming_25_n15_m4	15	4	56	55	98.21
hamming_27_n15_m6	15	6	128	120	93.75
hamming_35_n25_m2	25	2	21	21	100.00
hamming_46_n25_m7	25	7	296	268	90.54
hamming_58_n50_m12	50	12	1767	1732	98.02
hamming_70_n100_m10	100	10	2428	2330	95.99
hamming_85_n100_m40	100	40	39788	38000	95.45
hamming_97_n125_m37	125	37	42530	40913	96.20
hamming_101_n150_m15	150	15	8268	8084	97.77
hamming_105_n150_m15	150	15	8260	7874	95.32
hamming_117_n200_m20	200	20	19689	19222	97.62
hamming_122_n200_m40	200	40	79487	78772	99.23
hamming_131_n200_m80	200	80	319135	308783	96.77
hamming_138_n300_m30	300	30	66855	64132	95.93
hamming_147_n300_m90	300	90	605996	593270	97.90
hamming_152_n300_m120	300	120	1078157	1024667	95.03
hamming_158_n400_m40	300	40	158750	156400	98.51
hamming_164_n400_m80	400	80	638122	607767	95.23
hamming_166_n400_m120	400	120	1437380	1406518	97.86
hamming_174_n400_m160	400	160	2556847	2428886	95.01
hamming_180_n500_m50	500	50	310609	301921	97.20
hamming_185_n500_m100	500	100	1246945	1199380	96.22
hamming_188_n500_m150	500	150	2808636	2762662	98.36
hamming_190_n500_m150	500	150	2808649	2677666	95.36
hamming_195_n500_m200	500	200	4995093	4995093	100.00

Portanto, é possível concluir que os modelos não só são capazes de generalizar bem para novas instâncias com diferentes características, como também mantêm sua eficiência mesmo diante de instâncias com características mais desafiadoras.

6.6 Testes com Novas Instâncias Grandes

Foram geradas 150 novas instâncias grandes com base nos três métodos originais das MDPLIB. Os resultados foram extremamente positivos para a maioria das instâncias testadas, especialmente para aquelas com até 5000 vértices. Contudo, para instâncias com tamanhos superiores, como aquelas com 7000 vértices ou mais, observou-se uma leve queda na proximidade entre as soluções obtidas pelos modelos e os valores ótimos calculados pela metaheurística.

A Tabela 10 apresenta os resultados obtidos para essas instâncias. Nela, é possível

notar que os modelos se saíram muito bem para instâncias com 1000 a 5000 vértices, com a proximidade variando entre 99% e 100%. No entanto, as instâncias com tamanhos maiores de 7000 vértices, embora ainda apresentando um desempenho satisfatório, mostraram um declínio gradual na proximidade das soluções.

Tabela 10 – Resultados para Novas Instâncias Grandes Geradas com os Métodos Originais

Instância de Teste	$\|V\|$	$\|S\|$	**Valor Objetivo (Metaheurística)**	**Valor Alcançado (Modelos)**	**Proximidade em Porcentagem**
G_1_n1000_m100	1000	100	37420.90882	37000.95936	99.02
G_5_n1000_m150	1000	150	83333.821270001	82076.75998	98.66
G_9_n1000_m200	1000	200	141499.776819997	140922.70747	99.65
G_13_n2000_m200	2000	200	152434.403049996	151775.67949	99.43
G_15_n2000_m200	2000	200	152985.03035	152010.95689	99.54
G_19_n2000_m400	2000	400	575548.52648005	569753.93619	98.75
G_24_n2000_m600	2000	**600**	1237564.0273500006	**1233469.27758**	99.88
G_29_n3000_m300	3000	300	339820.46619000006	331537.65126	97.55
G_33_n3000_m400	3000	400	586365.56827	570732.46469	97.12
G_37_n3000_m600	3000	600	1292960.97558992	1279987.88459	99.20
G_43_n5000_m200	5000	200	158428.559379996	156976.17129	98.74
G_46_n5000_m300	5000	300	347297.48591005	344679.42629	98.88
G_50_n5000_m500	5000	500	943959.31622996	935261.46735	99.14
G_53_n5000_m700	5000	700	1802476.464089987	1783627.72248	99.26
G_59_n5000_m1500	5000	**1500**	7797362.1048712	**7797362.10487**	100.00
G_65_n7000_m900	7000	900	2999433.30829024	2810863.42442	93.92
G_68_n7000_m1000	7000	1000	3670035.13716	3456759.01039	94.12
G_72_n7000_m2000	7000	**2000**	13865101.615429984	**13058328.36138**	94.15
G_74_n7000_m3000	7000	**3000**	29750179.743870065	**28283685.17979**	95.03
G_79_n9000_m1500	9000	1500	8183522.707300003	7549900.44633	92.25
G_85_n9000_m2500	9000	2500	21735280.989	20308472.55274	93.42
G_88_n9000_m3000	9000	3000	30621887.96537	28610471.12969	93.66
G_92_n10000_m1000	10000	1000	3775278.1465999	3445928.21251	91.19
G_93_n10000_m1000	10000	1000	3783815.3677399	3449320.30661	91.22
G_95_n10000_m2000	10000	2000	14344692.0934998	13232349.12638	92.36
G_98_n10000_m3000	10000	3000	31008618.597505	28854789.69212	93.12
G_102_n10000_m4000	10000	4000	53444729.3306901	50597661.38376	94.55

Para lidar com o declílio nas coluções encontradas, foram incluídas 30 instâncias de tamanho 1000 na base de treinamento, além das 255 instâncias com tamanho de 10 a 500 vértices. O objetivo foi verificar se o declínio na proximidade estava relacionado à perda de capacidade dos modelos em compreender a estrutura das instâncias maiores. De fato, essa abordagem se mostrou eficaz, já que os resultados melhoraram significativamente após a inclusão dessas instâncias, como observado na Tabela 11.

Tabela 11 – Resultados para Novas Instâncias Grandes Geradas com os Métodos Originais

Instância de Teste	$\|V\|$	$\|S\|$	**Valor Objetivo (Metaheurística)**	**Valor Alcançado (Modelos)**	**Proximidade em Porcentagem**
G_1_n1000_m100	1000	100	37420.90882	37000.95936	99.02
G_5_n1000_m150	1000	150	83333.821270001	82076.75998	99.02
G_9_n1000_m200	1000	200	141499.776819997	140922.70747	99.65
G_13_n2000_m200	2000	200	152434.403049996	151775.67949	99.43
G_15_n2000_m200	2000	200	152985.03035	152010.95689	99.54
G_19_n2000_m400	2000	400	575548.52648005	569753.93619	99.00
G_24_n2000_m600	2000	600	1237564.0273500006	1233469.27758	99.88
G_29_n3000_m300	3000	300	339820.46619000006	331537.65126	99.03
G_33_n3000_m400	3000	400	586365.56827	570732.46469	99.04
G_37_n3000_m600	3000	600	1292960.97558992	1279987.88459	99.30
G_43_n5000_m200	5000	200	158428.559379996	156976.17129	99.10
G_46_n5000_m300	5000	300	347297.48591005	344679.42629	99.10
G_50_n5000_m500	5000	500	943959.31622996	935261.46735	99.14
G_53_n5000_m700	5000	700	1802476.464089987	1783627.72248	99.26
G_59_n5000_m1500	5000	1500	7797362.1048712	**7797362.1048712**	100.00
G_65_n7000_m900	7000	900	2999433.30829024	2810863.42442	99.02
G_68_n7000_m1000	7000	1000	3670035.13716	3456759.01039	99.04
G_72_n7000_m2000	7000	2000	13865101.615429984	13058328.36138	99.06
G_74_n7000_m3000	7000	3000	29750179.743870065	**30410633.73418398**	102.22
G_79_n9000_m1500	9000	1500	8183522.707300003	7549900.44633	99.08
G_85_n9000_m2500	9000	2500	21735280.989	20308472.55274	99.09
G_88_n9000_m3000	9000	3000	30621887.96537	**31574228.681093004**	103.11
G_92_n10000_m1000	10000	1000	3775278.1465999	3445928.21251	99.15
G_93_n10000_m1000	10000	1000	3783815.3677399	3449320.30661	99.15
G_95_n10000_m2000	10000	2000	14344692.0934998	13232349.12638	99.20
G_98_n10000_m3000	10000	3000	31008618.597505	**31141955.65747427**	100.43
G_102_n10000_m4000	10000	4000	53444729.3306901	50597661.38376	99.40

Esse resultado indica que é viável treinar modelos com instâncias menores e aplicá-los em instâncias maiores, desde que haja uma proporção bem definida entre os tamanhos das bases de treinamento e teste. No caso observado, a base de treinamento representou 10% do tamanho da base de teste, o que permitiu preservar a qualidade das soluções obtidas. O ajuste na base de treinamento, incluindo instâncias de tamanho 1000, resultou em um aumento considerável no desempenho, especialmente para as instâncias maiores, reforçando a ideia de que, ao seguir uma proporção adequada, é possível aplicar modelos treinados com instâncias menores de forma eficiente em instâncias maiores. Dessa forma, a experiência mostrou que, para preservar o máximo de qualidade de solução, é recomendável seguir uma proporção entre os tamanhos das bases de treinamento e teste.

7 CONCLUSÕES E TRABALHOS FUTUROS

O presente estudo investigou a aplicação de métodos de aprendizado de máquina no Problema da Diversidade Máxima (MDP), um problema NP-difícil que exige soluções aproximadas em razão do elevado tempo computacional necessário para instâncias de grande escala. Através da utilização de modelos supervisionados, como Redes Neurais Feed-forward, Árvores de Decisão, Florestas Aleatórias, Regressão Logística e KNN, foi possível avaliar a capacidade desses modelos em classificar corretamente os vértices nos conjuntos que maximizam a diversidade, alcançando, em muitos casos, soluções próximas aos valores ótimos obtidos por métodos exatos e metaheurísticos.

Os resultados demonstraram que os modelos de aprendizado de máquina foram eficazes, especialmente quando treinados com um número maior de instâncias, o que resultou em uma melhoria contínua no desempenho. Com o aumento das instâncias no treinamento, a proximidade das soluções em relação à solução ótima se manteve consistentemente superior a 90%, em especial para grafos com 1000 ou mais vértices. Embora a acurácia não tenha sido a principal métrica de avaliação, a análise do valor de diversidade máxima demonstrou que os modelos foram capazes de alocar corretamente os vértices e, assim, maximizar a diversidade.

Apesar dos resultados promissores, diversos aprimoramentos podem ser implementados. Trabalhos futuros devem explorar a aplicação de Redes Neurais Convolucionais (CNNs) e Redes Neurais Recorrentes (RNNs), que, devido às suas capacidades de lidar com estruturas mais complexas, podem melhorar ainda mais a modelagem das instâncias de grafos, especialmente em cenários com grafos de maior escala e topologias mais variadas. Além disso, técnicas de aprendizado supervisionado combinadas com otimização podem ser investigadas para aprimorar ainda mais a qualidade das soluções para o MDP, considerando diferentes critérios de alocação.

Outro avanço relevante seria a aplicação de Graph Neural Networks (GNNs), que são especialmente projetadas para lidar com a estrutura dos grafos, permitindo uma extração mais eficiente e precisa das características dos vértices. As GNNs têm o potencial de melhorar a generalização dos modelos, principalmente em grafos de grande porte, e podem ser exploradas para superar limitações encontradas nos modelos tradicionais.

Adicionalmente, a adaptação dinâmica dos parâmetros de aprendizado, como a taxa de aprendizado e a regularização, bem como a exploração de novas abordagens de aprendizado por reforço, podem ser áreas promissoras para melhorar a performance dos modelos, permitindo uma adaptação mais eficiente aos diferentes tipos de grafos e suas características. A inclusão de

novas instâncias de treinamento com grafos de diferentes distribuições de pesos e topologias deve ser considerada, com a intenção de testar a robustez dos modelos em cenários mais complexos e desafiadores.

Por fim, é esperado que a combinação de técnicas de aprendizado de máquina com métodos de otimização possa fornecer soluções ainda mais eficientes para o problema da diversidade máxima, contribuindo para o avanço tanto teórico quanto prático na resolução de problemas combinatórios de grande escala.

REFERÊNCIAS

AGRESTI, A. **Foundations of Linear and Generalized Linear Models**. [*S. l.*]: John Wiley & Sons, 2015.

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

BELL, R. M.; KOREN, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: **Proceedings of the 7th IEEE International Conference on Data Mining**. [*S. l.*]: IEEE, 2007. p. 43–52.

BERGSTRA, J.; BARDENET, R.; BENGIO, Y.; KÉGL, B. Random search for hyper-parameter optimization. **Journal of Machine Learning Research**, v. 13, n. Feb, p. 281–305, 2012.

BESANKO, D.; PERRY, M.; SPADY, R. The economics of product differentiation. **International Journal of Industrial Organization**, v. 8, p. 431–447, 1990.

BISHOP, C. M. **Pattern Recognition and Machine Learning**. [*S. l.*]: Springer, 2006.

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.

BOIMAN, O.; SHECHTMAN, E.; IRANI, M. In defense of nearest-neighbor based image classification. In: **2008 IEEE Conference on Computer Vision and Pattern Recognition**. [*S. l.*]: IEEE, 2008. p. 1–8.

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

BREIMAN, L.; FRIEDMAN, J. H.; OLSHEN, R. A.; STONE, C. J. **Classification and Regression Trees**. [*S. l.*]: Wadsworth and Brooks/Cole, 1984.

CARRABS, F.; CERULLI, R.; TOTH, P. New approaches for solving the maximum diversity problem in large graphs. **Journal of Heuristics**, v. 28, n. 3, p. 231–256, 2022.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. Smote: Synthetic minority over-sampling technique. **Journal of Artificial Intelligence Research**, AI Access Foundation, v. 16, p. 321–357, 2002.

COVER, T. M.; HART, P. E. Nearest neighbor pattern classification. **IEEE Transactions on Information Theory**, IEEE, v. 13, n. 1, p. 21–27, 1967.

COX, D. R. The regression analysis of binary sequences. **Journal of the Royal Statistical Society: Series B (Methodological)**, Wiley Online Library, v. 20, n. 2, p. 215–242, 1958.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals and Systems**, Springer, v. 2, n. 4, p. 303–314, 1989.

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

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

DUARTE, A.; MARTÍ, R. Tabu search and grasp for the maximum diversity problem. **European Journal of Operational Research**, Elsevier, v. 178, n. 1, p. 71–84, 2007.

DUARTE, A.; MARTí, R.; SáNCHEZ-ORO, J.; GLOVER, F. Metaheuristics for the maximum diversity problem. **European Journal of Operational Research**, v. 240, n. 1, p. 24–36, 2015.

DíAZ-URIARTE, R.; ANDRéS, S. Alvarez de. Gene selection and classification of microarray data using random forest. **BMC Bioinformatics**, Springer, v. 7, n. 1, p. 3, 2006.

ERKUT, E.; NEUMAN, S. The discrete p-dispersion problem. **European Journal of Operational Research**, v. 46, p. 48–60, 1991.

ESTEVA, A.; KUPREL, B.; NOVOA, R. A.; KO, J.; SWETTER, S. M.; BLAU, H. M.; THRUN, S. Dermatologist-level classification of skin cancer with deep neural networks. **Nature**, Nature Publishing Group, v. 542, n. 7639, p. 115–118, 2017.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. d. L. F. d. **Inteligência artificial: uma abordagem de aprendizado de máquina**. [*S. l.*]: LTC, 2011.

GALLEGO, M.; DUARTE, A.; LAGUNA, M.; MARTÍ, R. Hybrid heuristics for the maximum diversity problem. **Computational Optimization and Applications**, Springer, v. 44, n. 3, p. 411–426, 2009.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. [*S. l.*]: W.H. Freeman and Company, 1979.

GHYCZY, T. V. Product diversity and proliferation as a new mode of competing in the motor car. **International Journal of Vehicle Design**, v. 6, p. 423–425, 1985.

GLOVER, F.; KUO, C.; DHIR, K. Heuristic algorithms for the maximum diversity problem. **Journal of Information and Optimization Sciences**, v. 19, n. 1, p. 109–132, 1998.

GONZALEZ, J. A.; HERNáNDEZ-PéREZ, H.; LOZANO, M. A review on maximum diversity problems. **European Journal of Operational Research**, v. 287, n. 2, p. 401–420, 2020.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S. l.*]: MIT Press, 2016.

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**, v. 8, n. 2, 2020. ISSN 2227-7390. Disponível em: https://www.mdpi.com/2227-7390/8/2/298.

GURUMURTHY, S.; KHARE, A. Predicting consumer preferences using logistic regression. **International Journal of Business Analytics**, IGI Global, v. 3, n. 1, p. 1–14, 2016.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. **Journal of Machine Learning Research**, v. 3, p. 1157–1182, 2003.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. 2nd. ed. [*S. l.*]: Springer, 2009.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. Upper Saddle River, NJ: Prentice Hall PTR, 1999.

HE, H.; GARCIA, E. A.; CHEN, S. Learning from imbalanced data. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 21, n. 9, p. 1263–1284, 2008.

HEATON, J. B.; POLSON, N. G.; WITTE, J. H. Deep learning for finance: Deep portfolios. **Applied Stochastic Models in Business and Industry**, v. 33, n. 1, p. 3–12, 2017.

HO, T. K. Random decision forests. In: IEEE. **Proceedings of the 3rd International Conference on Document Analysis and Recognition**. [*S. l.*], 1995. v. 1, p. 278–282.

HOSMER, D. W.; LEMESHOW, S.; STURDIVANT, R. X. **Applied Logistic Regression**. 3rd. ed. [*S. l.*]: Wiley, 2013.

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, v. 374, n. 2065, p. 20150202, 2016.

KIRKPATRICK, S.; JR, C. D. G.; VECCHI, M. P. Optimization by simulated annealing. **science**, American association for the advancement of science, v. 220, n. 4598, p. 671–680, 1983.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: **Advances in Neural Information Processing Systems**. [*S. l.*: *s. n.*], 2012. v. 25, p. 1097–1105.

KUO, C. C.; GLOVER, F.; DHIR, K. S. Analyzing and modeling the maximum diversity problem by zero-one programming. **Decision Sciences**, v. 24, n. 6, p. 1171–1185, 1993.

LECUN, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, v. 86, n. 11, p. 2278–2324, 1998.

LEMON, K. N.; VERHOEF, P. C. Understanding customer experience throughout the customer journey. **Journal of Marketing**, v. 80, n. 6, p. 69–96, 2016.

LI, B.; WANG, K.; ZHANG, D. On-line signature verification based on pca (principal component analysis) and mca (minor component analysis). In: SPRINGER. **International Conference on Biometric Authentication**. [*S. l.*], 2004. p. 540–546.

LIAW, A.; WIENER, M. Classification and regression by randomforest. **R News**, R Foundation for Statistical Computing, v. 2, n. 3, p. 18–22, 2017.

LIU, W.; PAYNE, S. H.; MA, S.; FENYö, D. Extracting pathway-level signatures from proteogenomic data in breast cancer using independent component analysis. **Molecular & Cellular Proteomics**, American Society for Biochemistry and Molecular Biology, 2019.

LOPEZ, F.; MONTERO, J.; GALLEGO, M. An efficient grasp algorithm for solving maximum diversity problems. **Computers & Operations Research**, v. 101, p. 201–211, 2019.

MARTÍ, R.; GALLEGO, M.; DUARTE, A. A branch and bound algorithm for the maximum diversity problem. **European Journal of Operational Research**, Elsevier, v. 200, n. 1, p. 36–44, 2010.

MITCHELL, T. M. **Machine Learning**. New York, NY: McGraw Hill, 1997.

MURPHY, K. P. **Machine Learning: A Probabilistic Perspective**. [*S. l.*]: MIT Press, 2012.

NGAI, E. W. T.; HU, Y. H.; WONG, Y. H.; CHEN, Y.; SUN, X. The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. **Decision Support Systems**, Elsevier, v. 50, n. 3, p. 559–569, 2011.

PALUBECKIS, G. Iterated tabu search for the maximum diversity problem. **Applied Mathematics and Computation**, Elsevier, v. 189, n. 1, p. 371–383, 2007.

PARRENO, F.; MORTES, J.; MARTí, R. Measuring diversity: A review and an empirical analysis. **European Journal of Operational Research**, 2024.

PETERSON, L. E. K-nearest neighbor. **Scholarpedia**, v. 4, n. 2, p. 1883, 2009.

PORTER, R. *et al.* **The Genetics of Livestock Improvement**. [*S. l.*]: Academic Press, 1975.

POWERS, D. M. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. **Journal of Machine Learning Technologies**, v. 2, n. 1, p. 37–63, 2011.

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

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. [*S. l.*]: Morgan Kaufmann, 1993.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, American Psychological Association, v. 65, n. 6, p. 386–408, 1958.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **Nature**, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [*S. l.*]: Pearson Education, 2010.

SALEKSHAHREZAEE, Z.; LEEVY, J. L.; KHOSHGOFTAAR, T. M. A reconstruction error-based framework for label noise detection. **Journal of Big Data**, Springer, v. 8, n. 1, p. 1–24, 2021.

SAMARASINGHE, S. **Neural Networks for Applied Sciences and Engineering: From Fundamentals to Complex Pattern Recognition**. Boca Raton, FL: CRC Press, 2006.

SHAH, R. V.; KANTOR, E. D.; WILSON, P. W. F.; HOFFMANN, U. Predicting heart disease risk using machine learning. **Journal of the American Medical Association**, American Medical Association, v. 314, n. 18, p. 1875–1876, 2015.

SHI, Y.; ZHANG, Y. The neural network methods for solving traveling salesman problem. **Procedia Computer Science**, v. 199, p. 681–686, 2022. ISSN 1877-0509. The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19.

SHLENS, J. **A Tutorial on Principal Component Analysis**. 2014.

SHRIMALI, L.; SINGH, A. Analysis of large-scale networks using network analysis techniques. **Journal of Network Science**, Springer, 2020.

SILVA, G.; OCHI, L.; MARTINS, S. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In: **Experimental and Efficient Algorithms**. [*S. l.*]: Springer, 2004. p. 498–512.

SOARES, P. L. B. **Métodos exatos para problemas quadráticos binários**. Tese (Doutorado) – Universidade Federal de Ceará, 2018.

TRIPPI, R. R.; TURBAN, E. **Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance**. New York, NY: McGraw-Hill, 1992.

VARIAN, H. R. Big data: New tricks for econometrics. **Journal of Economic Perspectives**, American Economic Association, v. 28, n. 2, p. 3–28, 2014.

WANG, J.; ZHOU, Y.; YIN, J.; ZHANG, Y. Competitive hopfield network combined with estimation of distribution for maximum diversity problems. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 39, n. 4, p. 1048–1066, 2009.

WANG, Q.; WANG, S.; WEI, B.; CHEN, W.; ZHANG, Y. Weighted k-nn classification method of bearings fault diagnosis with multi-dimensional sensitive features. **IEEE Access**, v. 9, p. 45428–45440, 2021.