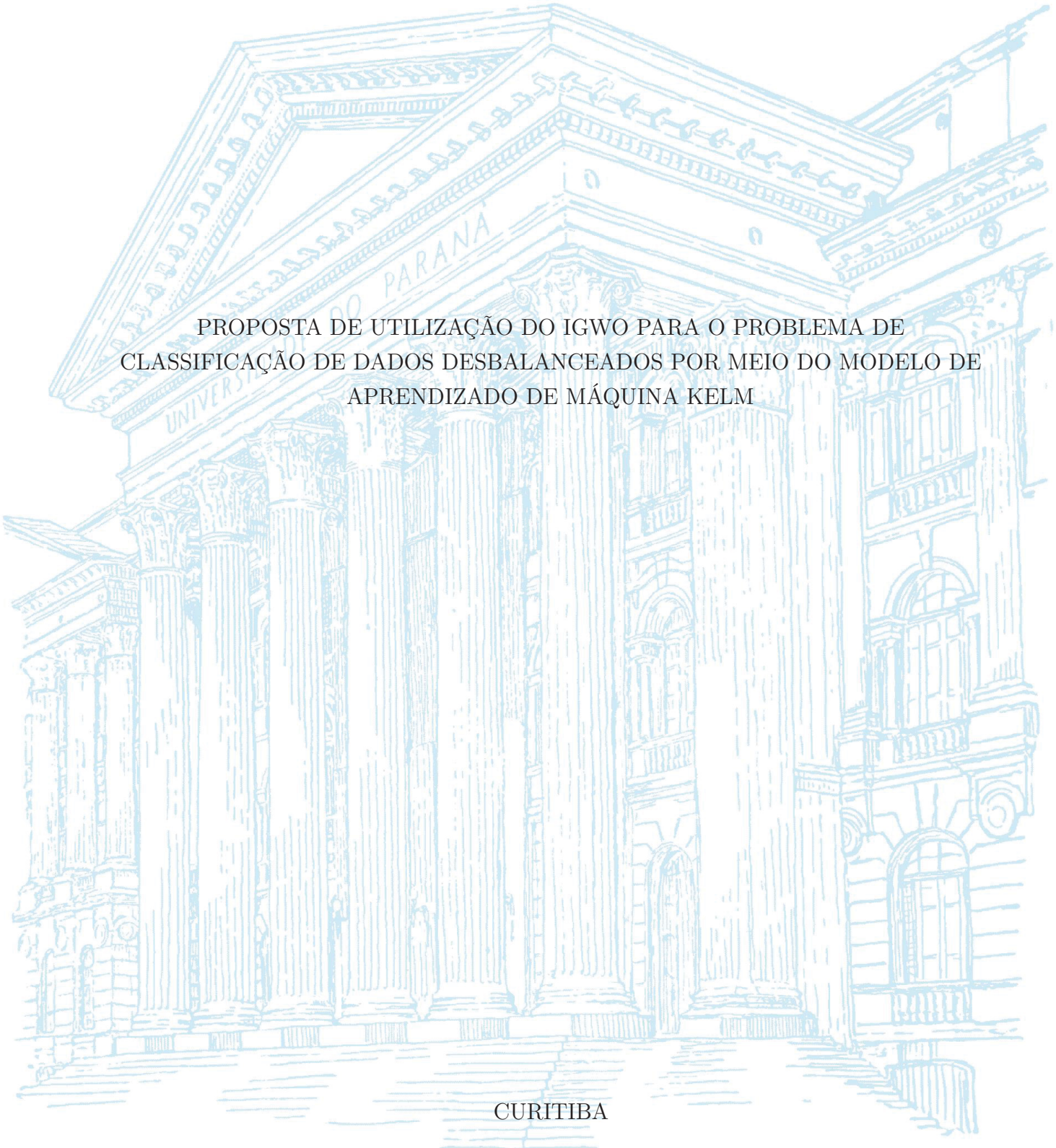
UNIVERSIDADE FEDERAL DO PARANÁ

THAYSE DUARTE DE OLIVEIRA



PROPOSTA DE UTILIZAÇÃO DO IGWO PARA O PROBLEMA DE CLASSIFICAÇÃO DE DADOS DESBALANCEADOS POR MEIO DO MODELO DE APRENDIZADO DE MÁQUINA KELM

CURITIBA

2023

THAYSE DUARTE DE OLIVEIRA

PROPOSTA DE UTILIZAÇÃO DO IGWO PARA O PROBLEMA DE CLASSIFICAÇÃO DE DADOS DESBALANCEADOS POR MEIO DO MODELO DE APRENDIZADO DE MÁQUINA KELM

Dissertação apresentada ao Curso de Pós-Graduação em Métodos Numéricos em Engenharia, Área de Concentração em Programação Matemática, do Departamento de Matemática, Setor de Ciências Exatas e do Departamento de Construção Civil, Setor de Tecnologia, Universidade Federal do Paraná, como requisito para a obtenção do título de Mestre em Ciências.

Orientador: Prof. Dr. Gustavo Valentim Loch

CURITIBA

2023

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Oliveira, Thayse Duarte de
Proposta de utilização do IGWO para o problema de classificação de dados desbalanceados por meio do modelo de aprendizado de máquina KELM / Thayse Duarte de Oliveira. – Curitiba, 2023.
1 recurso on-line : PDF.

Dissertação (Mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Métodos Numéricos em Engenharia.

Orientador: Gustavo Valentim Loch

1. Aprendizado de Máquinas. 2. Redes neurais (Computação). 3. Estimativa de parâmetros. 4. Kernel, Computação gráfica de (Sistema de computador). 5. Otimização matemática. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Métodos Numéricos em Engenharia. III. Loch, Gustavo Valentim. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

ATA Nº370

ATA DE SESSÃO PÚBLICA DE DEFESA DE MESTRADO PARA A OBTENÇÃO DO GRAU DE MESTRA EM MÉTODOS NUMÉRICOS EM ENGENHARIA

No dia vinte e oito de agosto de dois mil e vinte e tres às 15:00 horas, na sala https://teams.microsoft.com/l/meetup-join/19%3ameeting_OGVmYTU5NjUtOTQ5Zi00ODUxLWFlNjAtYjRmMzdiMWYwYTFi%40thread.v2/0?context=%7b%22Tid%22%3a%22c37b37a3-e9e2-42f9-bc67-4b9b738e1df0%22%2c%22Oid%22%3a%22be43c2d0-aa93-4ff2-a163-8fd66e5ea611%22%7d, Teams, foram instaladas as atividades pertinentes ao rito de defesa de dissertação da mestranda **THAYSE DUARTE DE OLIVEIRA**, intitulada**: PROPOSTA DE UTILIZAÇÃO DO IGWO PARA O PROBLEMA DE CLASSIFICAÇÃO DE DADOS DESBALANCEADOS POR MEIO DO MODELO DE APRENDIZADO DE MÁQUINA KELM**, sob orientação do Prof. Dr. GUSTAVO VALENTIM LOCH. A Banca Examinadora, designada pelo Colegiado do Programa de Pós-Graduação MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná, foi constituída pelos seguintes Membros: GUSTAVO VALENTIM LOCH (UNIVERSIDADE FEDERAL DO PARANÁ), TALITA MARIANA PINHO SCHIMIDT (null), LUIZ CARLOS MATIOLI (UNIVERSIDADE FEDERAL DO PARANÁ). A presidência iniciou os ritos definidos pelo Colegiado do Programa e, após exarados os pareceres dos membros do comitê examinador e da respectiva contra argumentação, ocorreu a leitura do parecer final da banca examinadora, que decidiu pela APROVAÇÃO. Este resultado deverá ser homologado pelo Colegiado do programa, mediante o atendimento de todas as indicações e correções solicitadas pela banca dentro dos prazos regimentais definidos pelo programa. A outorga de título de mestra está condicionada ao atendimento de todos os requisitos e prazos determinados no regimento do Programa de Pós-Graduação. Nada mais havendo a tratar a presidência deu por encerrada a sessão, da qual eu, GUSTAVO VALENTIM LOCH, lavrei a presente ata, que vai assinada por mim e pelos demais membros da Comissão Examinadora.

Curitiba, 28 de Agosto de 2023.

Assinatura Eletrônica
04/09/2023 17:24:56.0
GUSTAVO VALENTIM LOCH
Presidente da Banca Examinadora

Assinatura Eletrônica
05/09/2023 11:53:40.0
TALITA MARIANA PINHO SCHIMIDT
Avaliador Externo (null)

Assinatura Eletrônica
01/09/2023 22:18:22.0
LUIZ CARLOS MATIOLI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da dissertação de Mestrado de **THAYSE DUARTE DE OLIVEIRA** intitulada: **PROPOSTA DE UTILIZAÇÃO DO IGWO PARA O PROBLEMA DE CLASSIFICAÇÃO DE DADOS DESBALANCEADOS POR MEIO DO MODELO DE APRENDIZADO DE MÁQUINA KELM**, sob orientação do Prof. Dr. GUSTAVO VALENTIM LOCH, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestra está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 28 de Agosto de 2023.

Assinatura Eletrônica
04/09/2023 17:24:56.0
GUSTAVO VALENTIM LOCH
Presidente da Banca Examinadora

Assinatura Eletrônica
05/09/2023 11:53:40.0
TALITA MARIANA PINHO SCHIMIDT
Avaliador Externo (null)

Assinatura Eletrônica
01/09/2023 22:18:22.0
LUIZ CARLOS MATIOLI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Dedico este trabalho aos meus pais,
que me ensinaram desde a infância que educação
é um princípio que transforma vidas
e que eu sou capaz de aprender qualquer coisa.

AGRADECIMENTOS

Quero agradecer aos meus familiares cujo incentivo, apoio emocional e financeiro foram fundamentais durante todo este longo caminho. Também gostaria de expressar minha profunda gratidão aos meus amigos, que pacientemente me ouviram falar sobre matemática e suas aplicações no mundo real. A todos que me ajudaram de alguma forma, oferecendo valiosas explicações e correções, meu sincero agradecimento. Além disso, não poderia deixar de mencionar meu orientador, o Dr. Gustavo Valentim Loch, por ter depositado sua confiança em mim ao longo desta jornada. Por fim, quero agradecer à Coordenação de Aperfeiçoamento Pessoal de Nível Superior (CAPES) por todo o apoio concedido na realização deste trabalho.

Eu tenho uma porção
De coisas grandes pra conquistar
E eu não posso ficar aí parado
Raul Seixas

RESUMO

O presente estudo contribui na resolução do problema de dados desbalanceados, onde o desequilíbrio na distribuição das classes pode prejudicar a precisão da classificação. Propôs-se abordagens de aprendizado de máquina com o intuito de adaptar modelos matemáticos de forma a capturar padrões nas características relevantes da classe minoritária, visando melhorias significativas no desempenho geral da classificação. Os experimentos computacionais do algoritmo IGWO-KELM demonstraram que o modelo foi eficaz, rápido e robusto em diversos cenários, alcançando altas taxas de AUC em conjuntos de dados desbalanceados. Comparado com outros classificadores como o Naive Bayes, Nearest Neighbors e Incremental Gene Expression Programming Classifier o IGWO-KELM apresentou vantagem em cerca de 76% dos casos avaliados, mesmo em bases com taxa de desbalanceamento inferior a 6%. Além disso, o método ainda apresenta oportunidades de melhoria por meio dos estudos de diferentes algoritmos de busca de hiperparâmetros, técnicas de reamostragem e outras abordagens complementares. Em conclusão, o IGWO-KELM oferece uma solução eficiente e confiável para melhorar o desempenho do classificador em termos de AUC e tempo de processamento em cenários de dados desbalanceados.

Palavras-chaves: Aprendizado de Máquina. Redes Neurais. KELM. Hiperparâmetros. GWO. IGWO. Dados desbalanceados.

ABSTRACT

This study contributes to solving the problem of imbalanced data, where the imbalance in the distribution of classes can impair classification accuracy. Machine learning approaches were proposed to adapt mathematical models to capture patterns in the relevant features of the minority class, aiming for significant improvements in overall classification performance. Computational experiments with the IGWO-KELM algorithm demonstrated that the model was effective, fast, and robust in various scenarios, achieving high AUC rates on imbalanced datasets. Compared to other classifiers such as Naive Bayes, Nearest Neighbors, and Incremental Gene Expression Programming Classifier, IGWO-KELM showed an advantage in approximately 76% of the evaluated cases, even in datasets with an imbalance rate below 6%. Furthermore, the method still presents opportunities for improvement through the study of different hyperparameter search algorithms, resampling techniques, and other complementary approaches. In conclusion, IGWO-KELM offers an efficient and reliable solution to enhance classifier performance in terms of AUC and processing time in imbalanced data scenarios.

Key-words: Machine Learning. Neural Networks. KELM. Hyperparameters. GWO. IGWO. Unbalanced Data.

LISTA DE ILUSTRAÇÕES

FIGURA 1 – Classificação 21
FIGURA 2 – Clusterização 23
FIGURA 3 – Eventos Raros 24
FIGURA 4 – Sobreamostragem 25
FIGURA 5 – Subamostragem 26
FIGURA 6 – Rede Neural Artificial 28

FIGURA 7 – Hierarquia dos Lobos 41
FIGURA 8 – Posição dos Lobos 42
FIGURA 9 – IGWO 43
FIGURA 10 – Fluxograma IGWO 45

LISTA DE TABELAS

TABELA 1 – Principais diferenças entre GWO e IGWO 43

TABELA 2 – Faixas de segmentação 47
TABELA 3 – Variáveis abalone19 47
TABELA 4 – Variáveis ecoli-0-vs-1 48
TABELA 5 – Tabela intervalo [0,79% - 3,03%] 49
TABELA 6 – Tabela intervalo [3,17% - 6,78%] 50
TABELA 7 – Tabela intervalo [7,14% - 10,27%] 51
TABELA 8 – Tabela intervalo [10,36% - 65,45%] 52
TABELA 9 – Resumo por intervalo de taxas 52
TABELA 10 – Resumo por intervalo de volume 53
TABELA 11 – Resultados intervalo [0,79% - 3,03%] 54
TABELA 12 – Resultados intervalo [3,17% - 6,78%] 55
TABELA 13 – Resultados intervalo [7,14% - 10,27%] 56
TABELA 14 – Resultados intervalo [10,36% - 65,45%] 57
TABELA 15 – Comparação por classificador 59
TABELA 16 – Comparação por classificador 60
TABELA 17 – Comparação individual 61
TABELA 18 – Comparação remostragem 63
TABELA 19 – Comparação remostragem 64
TABELA 20 – Comparação individual 65

SUMÁRIO

1 INTRODUÇÃO 13
1.1 OBJETIVO 16
1.1.1 Objetivo Geral 17
1.1.2 Objetivos Específicos 17
1.2 ESTRUTURA DO TRABALHO 17

2 FUNDAMENTAÇÃO TEÓRICA 18
2.1 APRENDIZADO DE MÁQUINA 18
2.1.1 Aprendizado Supervisionado 20
2.1.2 Aprendizado Não Supervisionado 22
2.1.3 Dados desbalanceados 23
2.1.4 Redes Neurais Artificiais 27
2.1.5 Extreme Learning Machine (ELM) 29
2.1.6 Kernel Extreme Learning Machine (KELM) 30
2.2 META-HEURÍSTICA 31
2.2.1 Grey Wolf Optimizer (GWO) 32
2.2.2 Improved Grey Wolf Optimizer (IGWO) 32
2.3 MÉTRICAS DE DESEMPENHO 33

3 METODOLOGIA PROPOSTA 36
3.1 DESCRIÇÃO DO PROBLEMA 36
3.2 TRABALHOS CORRELATOS 36
3.3 CLASSIFICADOR 38
3.4 OTIMIZADOR DE HIPERPÂMETROS 40

4 EXPERIMENTOS COMPUTACIONAIS 47
4.1 CONJUNTO DE DADOS 48
4.2 RESULTADOS 52
4.3 COMPARAÇÃO ENTRE CLASSIFICADORES 58
4.4 COMPARAÇÃO ENTRE ALGORITMOS DE REAMOSTRAGEM . . . 61

5 CONCLUSÃO 66

REFERÊNCIAS 68

1 INTRODUÇÃO

Com o avanço da tecnologia nos últimos anos, a Internet das Coisas ou *Internet of Things* (IoT), permite a conexão entre aparelhos inteligentes com acesso à nuvem e, consequentemente, produz uma grande quantidade de informações em tempo real. Estes dados são valiosos, pois possibilitam o estudo do comportamento dos usuários destes dispositivos e a extração de informações relevantes (ZANELLA et al., 2014).

Setores de mercado financeiro, hospitais, instituições de ensino, centros de pesquisa, empresas de energia, telecomunicações, transporte e fornecimento de serviços no geral, utilizam dados coletados para gerenciar risco de perdas e tomar decisões eficientes automaticamente (HABEEB et al., 2019).

O termo *big data*, se refere a quantidade de informações criadas, capturadas, copiadas e consumidas em grande escala dentro de um intervalo de tempo pequeno, como por exemplo, quantidade de curtidas em redes sociais. Inteligência artificial e processamento paralelo são exemplos de abordagens utilizadas para contornar o desafio de processar, analisar e armazenar de maneira eficiente dados gerados por meio da IoT (MARR, 2015).

De acordo com Alpaydin (2010), o aprendizado de máquina é um tipo de inteligência artificial que possibilita que computadores aprendam por meio de dados e experiência, com o propósito de melhorar sua performance ao longo do tempo. Este conjunto de algoritmos e modelos, baseados em técnicas matemáticas e estatísticas, permite que sistemas de computadores façam análises complexas, como encontrar padrões de maneira automática (JORDAN; MITCHELL, 2015).

Existem problemas que podem ser resolvidos por meio de inteligência artificial em diversas áreas. Reconhecimento de imagens e sons, utilizado para detectar objetos, rostos, vozes e músicas (GOODFELLOW; BENGIO; COURVILLE, 2016) é uma aplicação. Outros exemplos são processamento de linguagem natural, análise de sentimentos, chatbots, tradutores de idiomas (JURAFSKY; MARTIN, 2019), sistemas de recomendações personalizados de produtos, filmes e vídeos (RICCI; ROKACH; SHAPIRA, 2011), detecção de atividades e transações suspeitas ou fraudulentas (HODGE; AUSTIN, 2004), exames toxicológicos e diagnóstico de doenças (TOPOL, 2019).

No campo da aprendizagem de máquina, os modelos são divididos entre algoritmos supervisionados e não supervisionados. Essa distinção é essencial para orientar a escolha do método adequado de acordo com a natureza dos dados e os objetivos da análise (BISHOP, 2006).

O aprendizado supervisionado, é a classe de algoritmos que utiliza uma variável resposta, que indica a ocorrência de um evento que representa o resultado ou a medida de

interesse. Exemplos de eventos utilizados como variável resposta são pagamento de uma fatura de cartão de crédito (HASTIE; TIBSHIRANI; FRIEDMAN, 2009), cancelamento de serviços de assinatura (BUREZ; VAN DEN POEL, 2009), preço de ações (LEIPPOLD; WANG; ZHOU, 2022), previsão do tempo (HANOON et al., 2021) e recomendações de tratamentos (TOPOL, 2019).

Durante o processo de treinamento, os modelos mapeiam o comportamento do conjunto de dados por meio do histórico, utilizando os padrões encontrados para calcular as chances de ocorrência do evento em novas observações (ALPAYDIN, 2010). Alguns dos principais modelos que compõem esta classe são árvores de decisão, vizinho mais próximo, máquinas de vetores de suporte, redes neurais artificiais (JORDAN; MITCHELL, 2015), regressão linear, polinomial e logística (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Outra classe de algoritmos denominada aprendizado não supervisionado, tem como característica dados de treinamento que não possuem variável resposta predefinida, onde o modelo deve encontrar padrões e estruturas por conta própria (BISHOP, 2006). Algumas aplicações são algoritmos de clusterização como k-médias, agrupamento hierárquico e espacial, cujo objetivo é encontrar a relação dentro do conjunto em análise (JAIN; DUBES, 1988).

Outras metodologias são análise de componentes principais e redução de dimensionalidade, utilizadas para selecionar variáveis de uma base de dados sem perder informações importantes (JOLLIFFE, 2011), detecção de anomalias (CHANDOLA; BANERJEE; KUMAR, 2009), *autoencoders* (GOODFELLOW; BENGIO; COURVILLE, 2016), mineração de dados e sistemas de recomendações (AGRAWAL; IMIELI'NSKI; SWAMI, 1993).

A principal característica do problema de dados desbalanceados é a ocorrência de uma classe com frequência muito maior do que a outra (HE; GARCIA, E., 2009). Portanto, o processo de aprendizado deve considerar que qualquer modelo pode apresentar métricas de avaliação altas quando classifica todas as observações como pertencentes a classe majoritária (FERNÁNDEZ; LÓPEZ et al., 2013).

Na prática, eventos considerados raros ocorrem em múltiplas aplicações, por exemplo, classificação de spam, defeitos em linha de produção, previsão de eventos climáticos extremos ou comentários ofensivos online. Classificar as observações da classe minoritária de maneira errada pode causar impactos negativos, como perdas financeiras ou diagnósticos incorretos (HAIXIANG et al., 2017).

Algoritmos de reamostragem como subamostragem, sobreamostragem, SMOTE (CHAWLA et al., 2002) e classificação como *AdaBoost*, *gradient boosting* (SUN; KAMEL et al., 2007), máquinas de vetores de suporte e redes neurais (SUN; WONG; KAMEL, 2009) são abordagens comumente citadas na literatura para resolver esse tipo de problema.

No entanto, não há uma técnica específica que se aplique de forma satisfatória em

todos os cenários (JEDRZEJOWICZ; JEDRZEJOWICZ, 2021). Desta forma, o objetivo desta pesquisa é propor uma abordagem eficaz e rápida de forma generalizada para problemas de dados desbalanceados, utilizando uma adaptação com kernel da rede neural *Extreme Learning Machine* (ELM), introduzida por Huang, Zhu e Siew (2006). O algoritmo se destaca por otimizar o tempo de processamento em relação a redes neurais comuns no cenário de *big data*, pois não treina o modelo de maneira iterativa (TANG; DENG; HUANG, 2015).

O ELM apresentou resultados superiores em relação aos algoritmos de redes neurais tradicionais em aplicações na área da saúde (KARPAGACHELVI; ARTHANARI; SIVAKUMAR, 2010), classificação de rostos (MOHAMMED et al., 2011), segmentação de imagens (PAN et al., 2012) e reconhecimento de ação humana (MINHAS et al., 2010). Huang, Zhou e Ding (2012) também apresentaram uma adaptação do ELM, o *Kernel Extreme Learning Machine* (KELM), propondo o acréscimo de funções não lineares para o mapeamento dos modelos.

O KELM apresenta vantagens por possuir poucos paramêtros variáveis, ser computacionalmente eficiente, de fácil implementação e evitar ótimos locais (LU et al., 2017). Os paramêtros variáveis, também chamados de hiperparâmetros, são constantes de entrada do modelo definidas antes da realização do treinamento (GOODFELLOW; BENGIO; COURVILLE, 2016), não possuem relação com o treinamento e podem ser definidos de forma arbitrária.

A quantidade de camadas ocultas dentro de uma rede neural é um exemplo de hiperparâmetro. No algoritmo de floresta aleatória, o número de árvores também é um paramêtro variável (GERON, 2019). Para a implementação do KELM, a constante de penalidade C e o γ da função kernel gaussiana, são dois hiperparâmetros que quando definidos de maneira aleatória, podem apresentar instabilidade durante o treinamento do modelo (LU et al., 2017).

O *grid search*, citado com frequência na literatura, é uma técnica de otimização de hiperparâmetros para ajuste e treinamento de modelos. A metologia consiste em realizar uma busca exaustiva em um espaço predefinido, testando cada combinação possível. Apesar de eficaz, a estratégia pode se tornar computacionalmente custosa à medida que o espaço de busca aumenta (GERON, 2019).

A configuração de hiperparâmetros pode ter seu critério de seleção definido por algoritmos de meta-heurística. Tais técnicas foram desenvolvidas para encontrar soluções satisfatórias em um período de tempo menor do que uma solução exata em dados complexos ou de larga escala (TALBI, 2009). Aplicações práticas são os problemas de transporte (RIZQI; LESTARI; SAPUTRO, 2018), da mochila (ABDEL-BASSET et al., 2021) e de agendamento (TAMSSAOUET; DAUZÈRE-PÉRÈS; YUGMA, 2018).

Métodos de meta-heurística como *Particle Swarm Optimization* (PSO) (LU et al., 2017), *Genetic Algorithm* (GA) (AVCI; DOGANTEKIN, 2016) e *Grey Wolf Optimization* (GWO) (WANG; ZHANG et al., 2017) apresentaram bons resultados quando aplicados como critério de seleção de hiperparâmetros para o KELM.

O GWO, meta-heurística proposta por Mirjalili, Mirjalili e Lewis (2014), tem sido utilizado e adaptado para diversas classes de problemas, por apresentar vantagens como a implementação simples e convergência rápida. O método possui como inspiração a organização hierárquica dos lobos e seu comportamento durante a caça. Todavia, o algoritmo original possui a fragilidade de cair fácilmente em ótimos locais. Adaptações foram propostas buscando aproximar os resultados da convergência global (FARIS et al., 2018).

Algumas contribuições são o *Multi-Objective Grey Wolf Optimizer* (MOGWO), para problemas de otimização multi-objetivo (MIRJALILI; SAREMI et al., 2016), *Hybrid Grey Wolf Optimization* (HGWO), uma versão híbrida entre o GWO e o *Differential Evolution* (DE) (ZHU et al., 2015) e o *Representative-based Grey Wolf Optimizer* (R-GWO) para problemas de fluxo de potência e otimização de projetos (BANAIEDEZFOULI et al., 2021).

O IGWO, proposto por Cai et al. (2019), se diferencia por realizar dois tipos de busca, dentro do intervalo das melhores soluções e de maneira aleatória, como método de diversificação para evitar ótimos locais.

Este estudo, baseado no artigo de Cai et al. (2019), propõe a implementação do algoritmo IGWO-KELM aplicado ao problema de dados desbalanceados (JEDRZEJOWICZ; JEDRZEJOWICZ, 2021). Experimentos computacionais foram realizados para validação do modelo e comparações entre diferentes soluções.

Foram utilizadas 99 bases de dados desbalanceados disponibilizados por Fernández, García, Luengo et al. (2010), no repositório online *the KEEL dataset repository* (ALCALÁ-FDEZ et al., 2011). Mais informações sobre estes conjuntos de dados podem ser obtidas no trabalho de Alcalá-Fdez et al. (2011).

Considerando as pesquisas com a proposta de resolver problemas de dados desbalanceados, foi identificado como oportunidade testar uma abordagem de solução utilizando o modelo IGWO-KELM. O objetivo deste trabalho é trazer ganhos em tempo computacional, processamento e performance comparado com outros algoritmos de classificação para este problema.

1.1 OBJETIVO

Nesta subseção serão apresentados o objetivo geral da dissertação bem como os objetivos específicos, que visam explicitar os processos realizados no trabalho.

1.1.1 Objetivo Geral

O objetivo principal deste trabalho é avaliar a capacidade de classificação do algoritmo IGWO-KELM aplicados ao problema de dados não balanceados. Para tal, é necessário otimizar os hiperpârametros da rede neural KELM por meio da meta-heurística IGWO.

1.1.2 Objetivos Específicos

Para obtenção do objetivo geral da pesquisa, a implementação computacional e adaptação do modelo IGWO-KELM para problemas de dados desbalanceados pode ser resumida em quatro objetivos específicos:

1. Estimar os hiperparâmetros: constante de penalidade C e γ da função kernel gaussiana por meio da meta-heurística IGWO;
2. Aplicar C e o γ estimados pela IGWO ao KELM e treinar o modelo;
3. Avaliar a performance do modelo;
4. Comparar os resultados com outros modelos de aprendizado de máquina.

1.2 ESTRUTURA DO TRABALHO

Esta dissertação está organizada em cinco capítulos. O primeiro, introdução, apresenta as temáticas gerais analisadas, especificando os modelos utilizados, o problema de pesquisa abordado e os objetivos geral e específicos. O segundo capítulo fornece a fundamentação teórica necessária para a apresentação do modelo, com o objetivo de esclarecer e revisar as metodologias utilizadas para resolver problemas de dados desbalanceados. O terceiro, apresenta os conceitos técnicos e as fórmulas do modelo. O quarto capítulo apresenta os resultados da implementação. O último capítulo inclui as conclusões e possibilidades de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

O objetivo deste capítulo é fornecer uma introdução aos conceitos fundamentais e noções gerais necessárias para compreender a teoria e a aplicação do modelo IGWO-KELM. Como principais referências bibliográficas Geron (2019), Goodfellow, Bengio e Courville (2016), Bishop (2006), James et al. (2013), Buda, Maki e Mazurowski (2018) e Cai et al. (2019), reconhecidas e estabelecidas no campo da inteligência artificial, fornecem bases teóricas sólidas e exemplos práticos para a compreensão do assunto.

2.1 APRENDIZADO DE MÁQUINA

O aprendizado de máquina é definido na literatura como área de estudo dedicada a desenvolver modelos capazes de aprender com dados e experiências (MURPHY, 2012). Trata-se de uma subárea da inteligência artificial que busca automatizar a construção de métodos analíticos por meio de algoritmos que identificam padrões e tomam decisões sem intervenção humana (RASCHKA; MIRJALILI, 2017).

De acordo Bishop (2006), um dos objetivos principais do aprendizado de máquina é criar uma generalização a partir de dados, convertendo a experiência em conhecimento. Este processo envolve a identificação de comportamentos que podem ser utilizados para fazer previsões ou classificações em novas observações.

Além disso, Murphy (2012) ressalta que a generalização é um dos desafios fundamentais da área, pois envolve encontrar um equilíbrio entre a capacidade do modelo de se ajustar aos dados disponíveis e a capacidade de extrair informações úteis que possam ser aplicados em novos dados.

A abordagem padronizada para o tratamento inicial dos dados, especialmente em modelos supervisionados, consiste em dividir as instâncias da base em dois conjuntos: treinamento e teste. O conjunto de treinamento é utilizado para construir e ajustar o modelo, enquanto o conjunto de teste é utilizado para avaliar o desempenho em observações desconhecidas (CHOLLET, 2018).

Por outro lado, em modelos não supervisionados, geralmente não há uma divisão predefinida dos dados em conjuntos de treinamento e teste, uma vez que não há uma variável de resultado específica que o modelo esteja tentando prever. Em vez disso, o objetivo muitas vezes é descobrir uma estrutura dentro dos próprios dados, sem nenhum alvo em particular em mente (MURPHY, 2012).

No entanto, um modelo só é útil se os dados forem confiáveis. O desempenho e a precisão dos algoritmos dependem diretamente da qualidade e da representatividade da

informação utilizada no treinamento (MURPHY, 2012). Se as variáveis forem imprecisas, incompletas, enviesadas ou de baixa qualidade, o modelo pode aprender padrões incorretos ou inadequados, fazendo previsões e classificações errôneas em novas observações (GOODFELLOW; BENGIO; COURVILLE, 2016).

Portanto, realizar análises e pré-processamento adequado como considerar a ética e a privacidade dos dados, verificar a integridade das variáveis, remover ruídos e anomalias, tratar observações faltantes e equilibrar as distribuições das classes é essencial para garantir a capacidade de generalização do modelo (CHOLLET, 2018).

Algumas das possibilidades mais comuns que podem prejudicar a performance de um modelo são o sobreajuste, que ocorre quando um modelo se ajusta tão bem ao conjunto de treinamento que não consegue ter uma fórmula generalizada para outros conjuntos (CHOLLET, 2018). No subajuste, o modelo é tão simples que não consegue encontrar padrões nos dados, falhando em fazer previsões e classificações corretas no conjunto de teste e de treino (BISHOP, 2006).

Outros fatores que podem afetar a qualidade dos resultados incluem a escolha inadequada de algoritmos, sem levar em consideração o tipo de dados que serão analisados (MURPHY, 2012). Variáveis irrelevantes ou redundantes também podem comprometer a acurácia dos resultados (BISHOP, 2006).

Algumas aplicações de aprendizado de máquina incluem a detecção de anomalias por meio de algoritmos que identificam padrões incomuns nos dados, por exemplo transações bancárias suspeitas, sinalizando a necessidade de investigação adicional (DUA; DU, 2016).

Outras aplicações que podem ser citadas são reconhecimento de imagens, que pode ser utilizado para reconhecimento facial (PAN et al., 2012). Detecção de objetos e cenas, aplicados em diversas áreas da indústria, tais como finanças, saúde, transporte e varejo (LECUN; BENGIO; HINTON, 2015). Recomendações personalizadas em lojas online, plataformas de mídias, serviços de *streaming* são baseados em preferências e comportamento do usuário (ADOMAVICIUS; TUZHILIN, 2005).

Algoritmos de processamento de linguagem natural, técnicas que aprendem padrões em dados textuais, como *chatbots*, tradução de idiomas e análise de sentimentos, podem ser úteis em setores como atendimento ao cliente, marketing e mídias sociais (JURAFSKY; MARTIN, 2019). Outro exemplo é a manutenção preditiva, que tem como objetivo reduzir o tempo de inatividade e aumentar a eficiência em setores de manufatura e transporte (SERRADILLA et al., 2022).

Diversas classes de algoritmos compõem a chamada inteligência artificial, técnicas e métodos utilizados para ensinar sistemas de computadores a tomar decisões de maneira automática (BISHOP, 2006). Por meio da construção de modelos que identificam padrões, é possível fazer previsões, classificações e encontrar informações valiosas em múltiplas

aplicações (RASCHKA; MIRJALILI, 2017).

2.1.1 Aprendizado Supervisionado

O aprendizado supervisionado é uma classe de métodos de aprendizado de máquina. Os modelos supervisionados são treinados para ajustar uma função que mapeia entradas para saídas com base em rotulados associados a caracteristicas da base de dados. O conjunto de treinamento inclui saídas corretas para cada entrada, permitindo saídas precisas ajustadas iterativamente. O objetivo final é generalizar o aprendizado para novas observações. Algoritmos de classificação, regressão e previsão fazem parte desta segmentação. (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

Algoritmos de previsão possuem aplicações em planejamento de produção, gerenciamento de estoques, vendas futuras de produtos e serviços, definição de estratégias de marketing (WANG, 2020). Preços de ativos financeiros como ações, moedas e *commodities* podem ter decisões de investimento facilitadas por meio destes modelos (SHAHVAROUGHI FARAHANI; RAZAVI HAJIAGHA, 2021).

Técnicas de análise de séries temporais como ARIMA (*AutoRegressive Integrated Moving Average*), suavização exponencial que atribui pesos decrescentes aos valores passados, e o SARIMA (*Seasonal AutoRegressive Integrated Moving Average*), que considera a sazonalidade nos dados, são exemplos de modelos utilizados para realizar previsões (HYNDMAN; ATHANASOPOULOS, 2018).

Modelos como regressão linear, que ajusta uma equação linear aos dados, modelando a relação entre a variável dependente e uma ou mais variáveis independentes, são utilizados para realizar previsões. A regressão polinomial, uma extensão da regressão linear, utiliza um polinômio de grau n para ajustar a relação entre os dados (JAMES et al., 2013).

Regressão de árvore de decisão, também muito conhecida, é um modelo não paramétrico que particiona recursivamente os dados em subconjuntos com base nos valores das variáveis independentes. Regressão de floresta aleatória combina vários modelos de árvore de decisão para melhorar a precisão e robustez das previsões. Outros exemplos são regressão de vetores de suporte, regressão de rede neural e regressão de *gradient boosting* (JAMES et al., 2013).

Estes modelos são aplicados em problemas de previsão de preços, como valores de casas baseados em quantidade de quartos, banheiros, tamanho do lote e localização e estimar o preço de um carro usado com base em seu ano de fabricação, quilometragem e tipo de combustível (CHATTERJEE; HADI, 1998).

Também é possível estimar o tempo necessário para executar uma tarefa, a probabilidade de chuva em uma região com o histórico meteorológico e avaliar a taxa de

retenção de clientes de uma empresa baseada em informações comportamentais como tempo de permanência na loja e as compras realizadas anteriormente (RAWLINGS; PANTULA; DICKEY, 2001).

Técnicas utilizadas para atribuir rótulos ou categorias a instâncias com base em seus atributos, são denominados algoritmos de classificação. O processo de modelagem envolve a divisão dos dados em conjuntos de treinamento e teste. No treinamento, o algoritmo aprende os padrões nos dados e ajusta seus parâmetros internos. A exposição do modelo a exemplos rotulados, categoriza as características apresentadas as suas respectivas classes (HASTIE; TIBSHIRANI; FRIEDMAN, 2009). O conjunto de teste é utilizado para avaliar o desempenho do modelo.

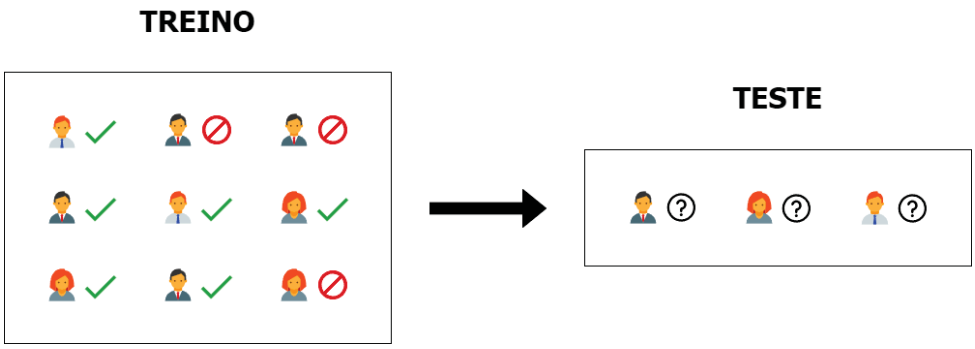
O problema de variável binária, é um problema de classificação comum. Neste caso, a variável resposta pode assumir apenas dois valores distintos. Uma aplicação clássica é filtrar e detectar e-mails de spam. O modelo usa recursos relevantes dos e-mails, como palavras-chave, características do remetente e do assunto para aprender a distinguir entre spam e não spam (GRAHAM, 2002).

Problemas multiclasse são aqueles em que o objetivo é classificar instâncias em mais de duas classes distintas. Um exemplo de aplicação é o reconhecimento de dígitos manuscritos. O objetivo é identificar automaticamente dígitos escritos à mão utilizando pré-processamento de imagem para extrair características relevantes, como bordas e pontos de interesse (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dividir adequadamente os dados em amostras de treino e teste, pode evitar o sobreajuste do modelo. O conjunto de teste é utilizado para avaliar o desempenho do classificador, contendo observações não vistas durante o treinamento que são comparadas aos rótulos verdadeiros. A proporção entre os conjuntos pode variar de acordo com o tamanho e a disponibilidade dos dados (JAIN; MURTY; FLYNN, 1999).

Na FIGURA 1 abaixo, a representação de como funciona a divisão de um conjunto de treino e teste para um modelo de classificação é apresentada para melhor entendimento.

FIGURA 1 – Classificação



Fonte: A autora (2022)

Exemplos de classificadores citados com frequência na literatura são árvores de decisão (BREIMAN et al., 1984) e florestas aleatórias (BREIMAN, 2001), algoritmos que constroem uma estrutura de árvore dividindo recursivamente o conjunto de dados em subconjuntos menores com base em determinados critérios, onde cada nó interno representa um teste em uma característica dos dados, e cada ramo representa o resultado do teste (MITCHELL, 1997).

Outros classificadores populares são máquinas de vetores de suporte (SVM), que separam os dados por meio de hiperplanos maximizando a separação entre as classes, de maneira linear e com kernel (CORTES; VAPNIK, 1995). Redes neurais artificiais são inspirados na estrutura e funcionamento do cérebro humano, e possuem variações como redes neurais multicamadas (MLP) e redes neurais convolucionais (CNN) (BISHOP, 1995).

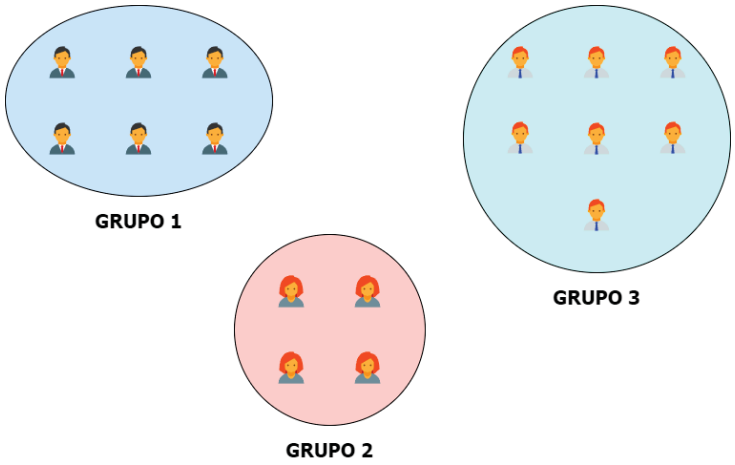
A escolha adequada do algoritmos de classificação pode ser orientada pela análise exploratória dos dados, dependendo de fatores como a natureza das informações, quantidade de características, presença de ruído ou anomalias, necessidade de interpretabilidade e a disponibilidade de recursos computacionais. É importante destacar que não existe uma resposta única, e a seleção deve ser baseada nas características específicas do problema e dos dados envolvidos (HASTIE; TIBSHIRANI; FRIEDMAN, 2009).

2.1.2 Aprendizado Não Supervisionado

O aprendizado não supervisionado é uma classe de algoritmos que tem como objetivo descobrir estruturas e padrões em um conjunto de dados sem a necessidade de rotulagem ou supervisão externa. Tais problemas apresentam um conjunto de entrada sem uma variável resposta e os modelos buscam extrair informações úteis mapeando similaridades entre os dados (HINTON; SEJNOWSKI, 1999).

Na literatura, os algoritmos de agrupamento conhecidos como clusterização, são uma das principais técnicas de aprendizado não supervisionado. O método separa subconjuntos com características semelhantes com base em critérios predefinidos. Um exemplo é o algoritmo k-médias, que utiliza as médias das variáveis para determinar a separação dos grupos (JAIN; MURTY; FLYNN, 1999). A FIGURA 2 seguir, é um exemplo de representação visual de clusterização.

FIGURA 2 – Clusterização



Fonte: A autora (2022)

Outros casos incluem modelos de redução de dimensionalidade, empregados para simplificar dados sem perder informações essenciais sem diminuir a precisão do modelo, reduzindo o tempo de treinamento. Aplicações práticas são processamento de sinais de áudio e vídeo de alta dimensão, redução de pixels em imagens e análise de sequência de DNA (HINTON; SALAKHUTDINOV, 2006).

O *deep learning* também chamado de aprendizado profundo, da classe de modelos de aprendizado não supervisionado, são redes neurais modificadas para aprender representações complexas de dados. Estas redes são compostas por várias camadas de neurônios capazes de capturar informações em diferentes níveis de abstração, permitindo a extração de características não lineares do conjunto analisado. O uso de redes profundas é especialmente eficaz em tarefas de visão computacional e processamento de linguagem natural (GOODFELLOW; BENGIO; COURVILLE, 2016).

2.1.3 Dados desbalanceados

Uma das principais preocupações ao desenvolver um modelo de aprendizado de máquina é a sua capacidade de generalização para novos dados. Em casos onde o conjunto de dados envolve eventos raros, ou seja, quando uma classe é muito maior que a outra, essa tarefa pode se tornar um desafio (JEDRZEJOWICZ; JEDRZEJOWICZ, 2021).

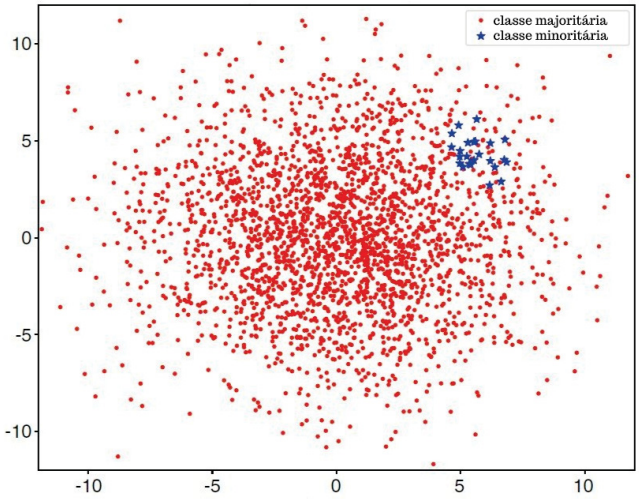
O problema de dados desbalanceados apresenta como principal dificuldade a referência para avaliar a qualidade do modelo. Como a classe minoritária é rara em relação à classe majoritária, métricas tradicionais de avaliação podem ser enganosas (FERNÁNDEZ; GARCÍA; GALAR et al., 2018).

Modelar eventos raros refere-se a identificar ou prever eventos que ocorrem com baixa frequência em relação ao restante dos dados. Estes eventos de classe minoritária podem ser críticos, impactantes ou indicativos de situações específicas. Na área de finanças,

riscos extremos como a ocorrência de uma crise financeira, é um desses casos. Na saúde, modelos são utilizados para prevenir doenças. Em engenharia, é possível mitigar falhas em equipamentos (CHAWLA et al., 2002).

A FIGURA 3 abaixo destaca a disparidade entre os conjuntos, o que demonstra que nem sempre é possível encontrar uma solução que represente ambas as classes de maneira equilibrada.

FIGURA 3 – Eventos Raros



Fonte: (FERNÁNDEZ; GARCÍA; GALAR et al., 2018)

Esse desequilíbrio pode prejudicar a performance dos modelos levando a resultados enganosos, gerando classificações incorretas. Quando o conjunto de dados apresenta uma classe majoritária super representada, o modelo pode se tornar tendencioso em direção a essa classe, diminuindo a taxa de acerto na classificação da classe minoritária. Alguns algoritmos possuem a lógica voltada para priorizar a classe dominante, então torna-se essencial considerar métodos que levem em conta as classes independentemente do tamanho (ZHANG; MA, 2012).

Classificar eventos raros incorretamente pode causar prejuízos significativos. Isso pode ser especialmente problemático em casos onde a classe minoritária é a de interesse principal, como em diagnósticos médicos ou detecção de fraudes financeiras, onde a taxa de ocorrência da classe minoritária pode ser menor que 1% (NANNI; FANTOZZI; LAZZARINI, 2015b). Além disso, tais decisões podem trazer consequências negativas para os indivíduos envolvidos, como tratamentos médicos inadequados ou reprovação em propostas de crédito (HE; GARCIA, E. A., 2009).

Algumas das principais técnicas utilizadas para abordar esse problema são mineração de dados, reamostragem e criação de amostras sintéticas (CHAWLA et al., 2002), e também métodos de classificação, como algoritmos com modificações sensíveis ao custo,

modelos de *ensemble*, métodos evolucionários e meta-aprendizado (HE; GARCIA, E. A., 2009).

A mineração envolve a reorganização do conjunto de dados a serem treinados, de forma que as classes tenham proporções diferentes do conjunto original. A aplicação de modelos de classificação após a mineração pode levar a resultados mais precisos e confiáveis (BUDA; MAKI; MAZUROWSKI, 2018).

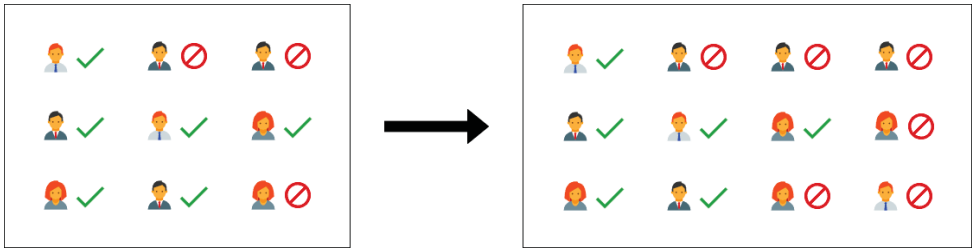
A reamostragem modifica a distribuição de classes por meio da duplicação ou eliminação de instâncias, a fim de atingir um equilíbrio entre as classes (BUDA; MAKI; MAZUROWSKI, 2018). Os métodos mais comuns de reamostragem incluem subamostragem aleatória, sobreamostragem aleatória e SMOTE (CHAWLA et al., 2002).

Na literatura, o SMOTE aparece com frequência como abordagem para tratar dados desbalanceados. O método consiste em criar observações sintéticas por meio de interpolação utilizando os dados vizinhos para balancear a base. Este método foi criado para contornar o problema que a técnica de sobreamostragem pode trazer ao reforçar o comportamento já conhecido ao replicar amostras aleatórias da classe negativa (BUDA; MAKI; MAZUROWSKI, 2018).

A sobreamostragem é uma técnica onde a classe minoritária é ampliada por meio da replicação de exemplos existentes. O objetivo é criar um conjunto no qual as classes possuam a mesma representatividade durante o treinamento de um modelo (NANNI; FANTOZZI; LAZZARINI, 2015b). Algumas técnicas de sobreamostragem são replicação aleatória, clusterização, *Adaptive Synthetic Sampling* (ADASYN) (HE; BAI et al., 2008), entre outros. Cada método possui suas particularidades e vantagens, e a escolha do método mais adequado depende das características específicas do conjunto de dados em questão (BUDA; MAKI; MAZUROWSKI, 2018).

A FIGURA 4 abaixo, refere-se a um conjunto de dados em que uma sobreamostragem foi aplicada para aumentar o número de exemplos da classe minoritária, a fim de equilibrar a distribuição das classes.

FIGURA 4 – Sobreamostragem



Fonte: A autora (2022)

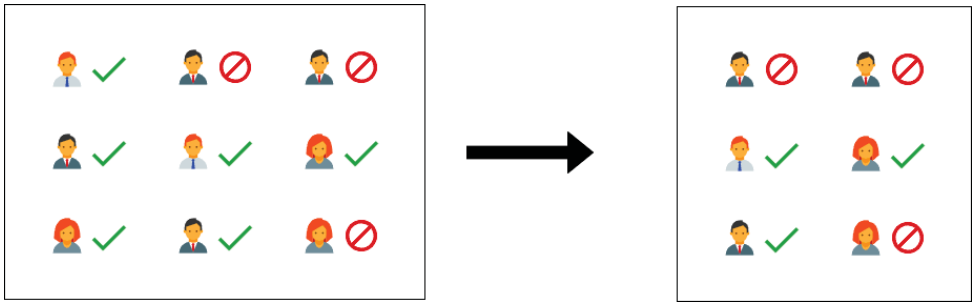
A subamostragem é utilizada para remover amostras, com a intenção de equilibrar a proporção entre as classes. Diferentes critérios podem ser aplicados como a remoção

aleatória de instâncias da classe majoritária, a eliminação de observações com baixa confiança e a exclusão com base na distância ou clusterização das amostras (BUDA; MAKI; MAZUROWSKI, 2018).

A escolha da técnica a ser utilizada depende das características específicas do conjunto de dados e do problema em questão. No entanto, é importante ressaltar que a subamostragem pode levar à perda de informações relevantes (HAIXIANG et al., 2017).

A FIGURA 5 abaixo, serve como uma representação visual do processo, demonstrando como a remoção de observações da classe majoritária permite que o modelo considere ambas as classes de maneira igualitária, contribuindo para uma análise mais justa e precisa dos dados.

FIGURA 5 – Subamostragem



Fonte: A autora (2022)

É importante ressaltar que a seleção da abordagem depende do contexto específico do problema e das características do conjunto de dados. Não existe uma solução única e ideal, e a combinação de diferentes técnicas pode ser necessária para lidar efetivamente com dados desbalanceados. Para contornar estes desafios, algoritmos de classificação podem empregar adaptações, tais como o uso de pesos nas classes, técnicas de reamostragem e algoritmos específicos para o problema em questão (CHAWLA et al., 2002).

Algoritmos com modificações sensíveis ao custo são técnicas que alteram a função objetivo, ajustando o peso de classificar errado as amostras de uma classe específica (PRATI; BATISTA; SILVA, 2015). Modelos compostos por múltiplos classificadores são denominados *ensembles*. Eles combinam a saída de cada classificador gerando decisões finais mais precisas, o que aumenta a robustez e a capacidade de generalização do modelo (FERNÁNDEZ; GARCÍA; GALAR et al., 2018).

Métodos evolucionários, como algoritmos genéticos e programação evolutiva, são técnicas de otimização utilizadas para simular a evolução biológica na busca da melhor solução para um problema. Estes métodos podem ser aplicados na combinação de parâmetros para um modelo de classificação, visando melhorar a precisão nas classes minoritárias em problemas de dados desbalanceados (EIBEN; SMITH, 2015).

O aprendizado meta são algoritmos que aprendem a aprender, ou seja, aprendem a

selecionar, adaptar e combinar modelos e técnicas de aprendizado de máquina de maneira eficiente para lidar com novas tarefas ou domínios. Isso pode incluir escolher a melhor técnica de amostragem ou algoritmo de classificação para um conjunto de dados específico (HE; GARCIA, E. A., 2009).

Exemplos de métodos que possuem funções que podem ter seus parâmetros ajustados são matriz de confusão custo-sensível (SUN; KAMEL et al., 2007), árvores de decisão, redes neurais, regressão logística, *naive bayes* e máquinas de vetores de suporte (ELKAN, 2001). No caso dos *ensembles*, alguns classificadores utilizados para resolver o problema de dados desbalanceados (ZHOU, 2012) citados na literatura são *BalanceCascade* (LIU; TING; ZHOU, 2009) e o *RUSBoost* (SEIFFERT et al., 2010).

Algoritmos populares no aprendizado meta incluem o método de gradiente descendente (ANDRYCHOWICZ et al., 2016), Model-Agnostic Meta-Learning (MAML) (FINN; ABBEEL; LEVINE, 2017), utilizado para acelerar a aprendizagem do classificador, e o *Almost No Inner Loop* (ANIL), usado para otimizar hiperparâmetros com poucas repetições com o intuito de reduzir o tempo de processamento (RAGHU et al., 2019).

Embora as técnicas de classificação não solucionem automaticamente o problema de dados desbalanceados, elas podem ser ajustadas para enfrentar tal situação. As metodologias mencionadas anteriormente auxiliam na melhoria do desempenho dos modelos ao dar mais ênfase às classes minoritárias e reduzir o impacto do desbalanceamento na performance (HAN; KAMBER; PEI, 2012).

2.1.4 Redes Neurais Artificiais

Uma rede neural é um modelo computacional inspirado na estrutura e funcionamento do cérebro humano. É composto por um conjunto de funções de processamento simples, chamadas neurônios artificiais. Os neurônios artificiais são organizados em camadas, geralmente divididas entre camada de entrada, camadas intermediárias e camada de saída, que é o resultado do modelo (BISHOP, 1995).

Cada neurônio recebe entradas da camada anterior e gera uma saída que é enviada para a camada seguinte ou para a saída da rede. Durante o processo de treinamento da rede neural, as conexões entre os neurônios são ajustadas de forma iterativa para minimizar uma função de perda que mede o erro entre as saídas e os valores reais correspondentes. As camadas são responsáveis por processar e transformar as informações que são recebidas pela rede (RUMELHART; HINTON; WILLIAMS, 1986).

A camada de entrada é a primeira fase do modelo. Tem a responsabilidade de receber o conjunto de treinamento e enviar para a próxima camada. Em outras palavras, a camada de entrada representa a interface entre os dados de entrada e a rede neural, permitindo que as informações sejam inseridas nas camadas subsequentes para o

processamento e aprendizado. Cada neurônio na camada de entrada corresponde a uma variável do conjunto de dados (NIELSEN, 2015).

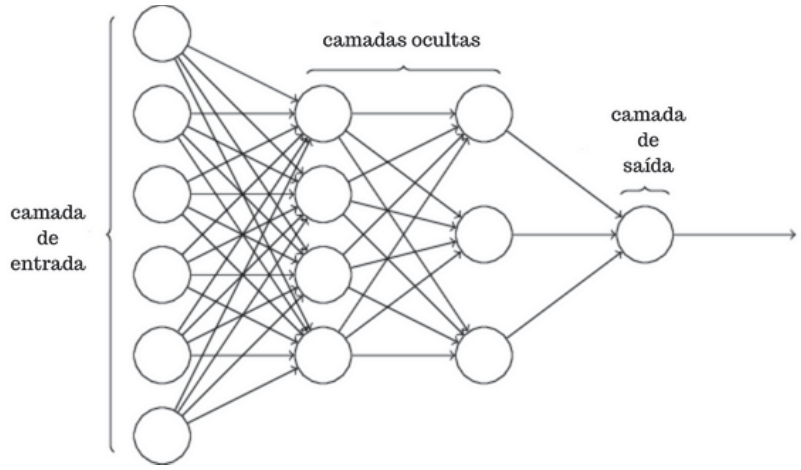
As camadas ocultas ou camadas intermediárias, estão entre a camada de entrada e a camada de saída. São responsáveis por extrair características importantes dos dados de entrada e transformá-las em representações mais compactas e significativas, calculando cada neurônio por meio de uma soma ponderada dos outros neurônios conectados a ele, utilizando os resultados para o cálculo das entradas da próxima camada (ESHTAY; FARIS; OBEID, 2019).

A camada de saída é a última camada da rede neural responsável por gerar os resultados do modelo. O número de neurônios na camada de saída é determinado pelo tipo de problema que está sendo resolvido. Para problemas de classificação, por exemplo, cada neurônio na camada de saída pode corresponder a uma classe diferente, enquanto em problemas de regressão, a camada de saída pode conter apenas um neurônio que representa o valor da variável de saída (BISHOP, 1995).

A arquitetura de uma rede neural pode ser personalizada, variando o número de camadas ocultas, o número de neurônios em cada camada e a função de ativação utilizada em cada neurônio. Isso permite que a rede neural seja adaptada para diferentes tipos de problemas de aprendizado de máquina (GOODFELLOW; BENGIO; COURVILLE, 2016).

A FIGURA 6 abaixo, representa uma rede neural simplificada, mas existem abordagens com diferentes arquiteturas, tamanhos e configurações de camadas, dependendo do problema em questão. A imagem serve como uma forma intuitiva de transmitir a estrutura e o funcionamento básico do modelo.

FIGURA 6 – Rede Neural Artificial



Fonte: (GOODFELLOW; BENGIO; COURVILLE, 2016)

Hornik, Stinchcombe e White (1989) provou que redes neurais com uma única camada oculta, podem aproximar qualquer função contínua. Este resultado implica que uma rede neural, com a quantidade de camadas adequada e a relação determinística entre

os dados de treinamento e a variável resposta, pode encontrar uma solução ótima para o modelo.

No entanto, tais métodos apresentam fragilidades, principalmente quando não há dados representativos e suficientes (LECUN; BENGIO; HINTON, 2015). Funções de perda baseadas em métodos de gradiente podem cair em ótimos locais e/ou apresentar um tempo computacional muito alto para realizar o treinamento, comparado com outros métodos (HUANG; ZHU; SIEW, 2006).

O avanço da tecnologia facilitou e popularizou o uso de redes neurais profundas ou *deep learning*, uma adaptação com várias camadas intermediárias de processamento. Cada camada é responsável por detectar e representar diferentes níveis de complexidade das características dos dados, ampliando aspectos importantes para discriminação e suprimindo variações irrelevantes (GOODFELLOW; BENGIO; COURVILLE, 2016).

As redes neurais são amplamente utilizadas em tarefas de classificação, regressão e processamento de imagem, fala e linguagem natural. Elas se mostraram particularmente eficazes em problemas complexos que exigem o processamento de grandes quantidades de dados, como reconhecimento de imagens e processamento de linguagem natural (BISHOP, 2006).

2.1.5 Extreme Learning Machine (ELM)

O *Extreme Learning Machine* (ELM) proposta por Huang, Zhu e Siew (2006), tem como objetivo diminuir o tempo de processamento e a aumentar eficiência dos resultados dos modelos de rede neural, sem a necessidade de ajuste manual dos parâmetros. Baseado em uma ideia simples, os pesos da camada de entrada são gerados aleatoriamente e a camada de saída é calculada diretamente por meio de uma solução analítica. A estrutura do ELM apresenta semelhanças com outras arquiteturas de redes neurais. Possuindo camada de entrada, uma ou mais camadas ocultas e camada de saída.

O processo de treinamento envolve a atribuição aleatória de pesos das conexões entre a camada de entrada e as camadas ocultas, o cálculo das saídas da camada oculta usando uma função de ativação não linear, o cálculo dos pesos das conexões entre as camadas ocultas e, finalmente, o cálculo da camada de saída usando uma fórmula analítica simples (HUANG; ZHOU; DING, 2012).

As expressões matemáticas relevantes para o entendimento do modelo serão apresentadas no próximo capítulo, enquanto neste, os conceitos serão explicados com o intuito de proporcionar a compreensão da metodologia aplicada. O treinamento do ELM pode ser dividido em duas etapas. Na primeira, a matriz de entrada é multiplicada por uma matriz de pesos aleatórios que conecta a camada de entrada à camada oculta. Em seguida, a função de ativação é aplicada ao resultado da multiplicação para produzir a

saída da camada oculta (TANG; DENG; HUANG, 2015).

Na segunda etapa, os pesos da camada de saída são determinados por meio da matriz de Moore-Penrose, generalização da matriz inversa que permite encontrar uma solução para sistemas de equações lineares singulares. O resultado contém as saídas da camada anterior para cada exemplo de treinamento. O teste do modelo é realizado por meio de um conjunto de validação, e os parâmetros podem ser ajustados, se necessário. Uma vez que os pesos da camada de saída são determinados, a rede neural está pronta para ser usada para fazer previsões em novos dados (HUANG; ZHU; SIEW, 2006).

O ELM tem várias vantagens em relação ao treinamento tradicional de redes neurais, incluindo a capacidade de processar grandes conjuntos de dados, evitar o sobreajuste e diminuir o tempo de treinamento. Além disso, algumas aplicações bem sucedidas em problemas de classificação e regressão, incluem reconhecimento de imagens, análise de dados biológicos e previsão financeira (HUANG; ZHOU; DING, 2012).

2.1.6 Kernel Extreme Learning Machine (KELM)

O *Kernel Extreme Learning Machine* (KELM) (HUANG; ZHOU; DING, 2012), é uma adaptação proposta para aumentar a precisão do ELM em problemas de classificação não-lineares. Estes casos possuem dados ou padrões que não podem ser separados por um limite de decisão linear em um espaço de características.

Em outras palavras, não é possível traçar uma linha reta, um plano ou um hiperplano para dividir claramente as classes ou agrupamentos de dados. Portanto, a separação requer uma fronteira de decisão curva ou complexa. Essa não linearidade pode surgir devido à complexidade dos dados e relação entre as características (CORTES; VAPNIK, 1995).

O truque do kernel é uma abordagem utilizada para lidar com problemas de separação não linear. A ideia central é aproveitar as propriedades matemáticas de funções que utilizam os produtos internos calculados para encontrar hiperplanos de separação ótimos em espaços de maior dimensionalidade. Alguns exemplos comuns incluem o kernel linear, polinomial, gaussiano e sigmoidal. Cada um desses kernels tem suas próprias propriedades e é adequado para diferentes tipos de problemas (BISHOP, 2006).

O KELM, assim como o ELM é rápido, eficiente e adequado para grande quantidade de dados. O truque do kernel permite que o ELM seja aplicado aos dados sem a necessidade de computar novos recursos. Para mapear dados que não são linearmente separáveis no espaço de entrada original, o KELM projeta o conjunto de treinamento em um espaço de alta dimensão sem utilizar diretamente as funções de ativação lineares ou não-lineares nas camadas ocultas (CAI et al., 2019).

A seleção dos pesos do KELM é feita aleatoriamente, e em seguida, a matriz de

produtos internos entre a entrada e os pesos é calculada usando um kernel não linear. Uma aproximação da matriz inversa de produtos internos é calculada por meio da matriz de Moore-Penrose. O KELM também usa a técnica de regularização para controlar a magnitude dos pesos dos vetores de suporte, permitindo ajustar o modelo para evitar o sobreajuste (LU et al., 2017).

2.2 META-HEURÍSTICA

Uma meta-heurísica é um algoritmo genérico que pode ser aplicado a uma ampla variedade de problemas de otimização, sem precisar conhecer detalhes específicos do problema em questão. A utilização desses métodos pode trazer diversos benefícios, como a possibilidade de encontrar soluções de alta qualidade em um tempo razoável, a capacidade de lidar com problemas de grande escala com muitas restrições e a flexibilidade para lidar com problemas que não possuem uma solução algorítmica exata (TALBI, 2009).

De acordo com Talbi (2009), as meta-heurísticas podem ser divididas em três categorias principais: as baseadas em população, as baseadas em traços e as baseadas em modelos. Cada uma dessas categorias possui um conjunto de algoritmos específicos, que podem ser combinados ou adaptados para atender às necessidades de um problema específico. Alguns exemplos de meta-heurísticas são busca local, algoritmos genéticos, busca tabu, recozimento simulado, entre outros.

Existem diversas aplicações de meta-heurísticas em multiplas áreas como otimização de rotas de veículos em logística e transporte, processos industriais como produção de energia, controle de qualidade e a definição de carteiras de investimentos em finanças com a intenção de maximizar o retorno (EIBEN; SMITH, 2015).

Por meio destes métodos, é possível encontrar configurações ótimas ou uma boa aproximação de maneira automatizada para hiperparâmetros. Um hiperparâmetro é um parâmetro configurável externamente ao algoritmo de aprendizado de máquina, que influencia o comportamento e o desempenho do modelo (ALPAYDIN, 2010).

Os hiperparâmetros, determinados antes do treinamento do modelo, são responsáveis por ajustar a complexidade, controlar a regularização e definir outros aspectos do algoritmo. Alguns exemplos são número de camadas ocultas, neurônios em cada camada ou a função de ativação de uma rede neural (ESHTAY; FARIS; OBEID, 2019).

Uma meta-heurística citada com frequência para otimização de hiperparâmetros é a busca aleatória, que define valores de maneira arbitrária para cada hiperparâmetro. Outros exemplos são o algoritmo genético, que utiliza a seleção natural e a reprodução para gerar novas soluções, colônia de formigas, enxame de partículas e busca em grade. Cada uma dessas técnicas possui vantagens e desvantagens e pode ser mais adequada para problemas específicos (FEURER; HUTTER, 2019).

2.2.1 Grey Wolf Optimizer (GWO)

O *Grey Wolf Optimizer* (GWO) proposto por Mirjalili, Mirjalili e Lewis (2014), é uma meta-heurística de busca que tem como objetivo encontrar soluções ótimas para problemas de otimização não-lineares. O algoritmo foi inspirado na hierarquia e no comportamento de caça dos lobos cinzentos. O GWO tem mostrado resultados promissores em diversos problemas de engenharia e de aprendizado de máquina.

Cada lobo representa um vetor solução para os hiperparâmetros. Como passo inicial, a população é definida de maneira aleatória, e o processo de busca é dividido em três fases principais: busca do lobo alfa, busca do lobo beta e busca do lobo delta (MIRJALILI; SAREMI et al., 2016).

Durante a fase de busca do lobo alfa, a posição do lobo é atualizada com base na melhor solução de hiperparâmetros contido no conjunto da população de lobos atual. A posição dos lobos beta e delta, são definidas como a segunda melhor e a terceira melhor solução, respectivamente (MIRJALILI; MIRJALILI; LEWIS, 2014).

O GWO original utiliza três operadores: busca aleatória, atualização do líder alfa e atualização dos outros lobos. Desde a proposta inicial, variações foram desenvolvidas para melhorar o desempenho e a eficiência do algoritmo (CAI et al., 2019).

Alguns exemplos incluem o Grey Wolf Optimizer with Levy Flight (GWO-LF) (HEIDARI; PAHLAVANI, 2017), proposto para melhorar a capacidade de exploração, *Adaptive Grey Wolf Optimizer* (AGWO) (MEIDANI et al., 2022), mais flexível e adaptável a diferentes problemas e o *Grey Wolf Optimizer on Differential Evolution and Elimination Mechanism* (GWO-DEEM), que utiliza um mecanismo de eliminação como critério para ampliar o desempenho e a precisão da solução (WANG; LI, 2019).

No GWO, cada lobo representa uma possível solução para o problema de otimização. O algoritmo considera as dimensões do espaço de solução e movimenta os lobos em direção ao alfa. No entanto, o alfa pode não ser a melhor solução possível, pois a hierarquia pode convergir para um ótimo local.

2.2.2 Improved Grey Wolf Optimizer (IGWO)

O *Improved Grey Wolf Optimizer* (IGWO) (CAI et al., 2019) é uma variação do GWO que propõe uma nova hierarquia para evitar ótimos locais na seleção de hiperparâmetros.

A proposta é manter o alfa como a melhor solução conhecida, mas beta é definido como um conjunto de lobos que realizam uma busca aleatória ao redor do alfa. Ômega contém as piores soluções, atualizadas aleatoriamente no espaço global de solução, para garantir a capacidade de busca global. Os lobos restantes são o conjunto delta, atualizado por meio da posição de alfa (CAI et al., 2019).

As principais diferenças entre GWO e IGWO estão em sua hierarquia e como os lobos são atualizados durante o processo de otimização. O GWO utiliza uma hierarquia fixa, enquanto o IGWO introduz uma hierarquia dinâmica para melhorar o desempenho da busca global. O IGWO tem como objetivo evitar que o conjunto de soluções fique preso em ótimos locais, aprimorando o processo geral de otimização (CAI et al., 2019).

Neste caso, a busca é utilizada para aumentar precisão de classificação da rede neural KELM. O mecanismo de aprendizado utiliza o reforço para adaptar o tamanho do passo e a posição inicial dos lobos, melhorando a eficiência do algoritmo (CAI et al., 2019).

Resultados experimentais mostraram que o IGWO-KELM pode ser aplicado em diferentes conjuntos de dados, alcançando métricas maiores do que outras abordagens, incluindo o KELM padrão e outras variantes do ELM (CAI et al., 2019).

2.3 MÉTRICAS DE DESEMPENHO

As métricas de desempenho permitem avaliar a performance do modelo em relação ao conjunto de dados, o que determina se a solução é adequada. Tais indicadores fornecem medidas para analisar a qualidade das previsões, comparar diferentes modelos, identificar problemas como a presença de viés ou sobreajuste (GERON, 2019).

A escolha da métrica apropriada depende do problema abordado e do tipo de dados envolvidos. Se houver classes desbalanceadas, a acurácia e a precisão podem não refletir a realidade dos dados do conjunto minoritário (NANNI; FANTOZZI; LAZZARINI, 2015b).

A matriz de confusão, utilizada para avaliar o desempenho de modelos de classificação em tarefas de aprendizado supervisionado, é uma tabela composta por quatro células que indicam o número de amostras classificadas corretamente, verdadeiros positivos (VP) e verdadeiros negativos (VN) e o número de amostras classificadas incorretamente, falsos positivos (FP) e falsos negativos (FN). Indicadores como a acurácia, precisão, sensibilidade e o score F1 são calculadas a partir dos valores da matriz de confusão (BOWLES, 2015).

		Previsão do Modelo	
Variável Resposta		Verdadeiro Positivo	Verdadeiro Negativo
		Falso Positivo	Falso Negativo

A acurácia mede a precisão geral de um modelo de classificação. Ela representa a proporção de predições corretas em relação ao total de predições realizadas. É uma métrica simples e intuitiva, mas pode ser enganosa em casos de desbalanceamento de classes pois mede a taxa de acerto do modelo em relação a todas as classes.

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.1}$$

A precisão é uma métrica que calcula a proporção de predições corretamente classificadas como positivas em relação a todas as predições positivas feitas pelo modelo. É relevante em situações onde o custo de um falso positivo é significativo, ou seja, quando é mais prejudicial classificar erroneamente uma instância como positiva quando, na verdade, ela é negativa.

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{2.2}$$

O *recall*, também conhecido como sensibilidade, é uma métrica que mede a proporção de predições corretas entre todos os verdadeiros positivos existentes na base de dados. Ele representa a habilidade do modelo em identificar corretamente os verdadeiros positivos e evitar falsos negativos. O *recall* é especialmente útil quando o objetivo é minimizar os falsos negativos.

$$\textit{Recall} = \frac{VP}{VP + FN} \tag{2.3}$$

O *F1-score* é uma medida que combina tanto a precisão quanto o *recall* em uma única métrica. Ele representa uma média harmônica entre as duas métricas e fornece um valor balanceado. O *F1-score* é útil quando é importante considerar tanto os falsos positivos quanto os falsos negativos, buscando um equilíbrio entre ambas as métricas.

$$\textit{F1-score} = \frac{2VP}{2VP + FP + FN} \tag{2.4}$$

Em resumo, a acurácia mede a precisão geral do modelo, a precisão e o *recall* medem a capacidade do modelo em evitar falsos positivos e falsos negativos e o *F1-score* fornece um equilíbrio entre a precisão e o *recall*. Essas métricas são amplamente utilizadas para avaliar o desempenho de modelos de classificação em problemas de aprendizado de máquina, e maximizar as métricas é objetivo da otimização de hiperparâmetros (D., 2018).

Quando se aborda dados raros, é necessário avaliar a capacidade do modelo de classificar corretamente tanto as instâncias da classe minoritária quanto as da classe majoritária, o que torna a acurácia uma métrica não confiável. Neste cenário, a taxa de acerto pode ter um bom desempenho simplesmente prevendo a classe majoritária em todos os casos (NANNI; FANTOZZI; LAZZARINI, 2015b).

A métrica *Receiver Operationg Characteristic* (ROC) é uma ferramenta gráfica utilizada para avaliar a performance de um classificador binário. Ela é calculada utilizando a taxa de verdadeiro positivo acumulada no eixo X, contra a taxa de falso negativo acumulada no eixo Y. A área abaixo da curva ROC, também conhecida como AUC é uma métrica amplamente utilizada em problemas com dados desbalanceados devido à sua

capacidade de avaliar o desempenho de um modelo independentemente da distribuição das classes (PEDREGOSA et al., 2011).

A AUC avalia a capacidade do modelo de classificar corretamente observações positivas e negativas, independente da proporção entre as classes, fornecendo uma medida abrangente do poder discriminativo do modelo. Além disso, não depende de ponto de corte para classificar novas instâncias, o que a torna mais robusta do que outras métricas, sendo adequada para avaliar cenários com desequilíbrio de classes (BUDA; MAKI; MAZUROWSKI, 2018).

3 METODOLOGIA PROPOSTA

Neste capítulo, serão apresentados os principais conceitos, definições e expressões matemáticas utilizados para realizar a implementação do modelo baseado no artigo "Evolving an optimal kernel extreme learning machine by using an enhanced grey wolf optimization strategy"de Cai et al. (2019). O algoritmo tem como objetivo selecionar os hiperparâmetros da rede neural KELM realizando a busca por meio da meta-heurística IGWO. É proposto aplicar este método ao problema de dados desbalanceados.

3.1 DESCRIÇÃO DO PROBLEMA

O desequilíbrio das classes em problemas de eventos raros, compromete o desempenho de modelos de classificação especialmente quando a classe majoritária predomina. Algoritmos que priorizam a classe dominante acentuam esse viés, tornando essencial adotar métodos que considerem as classes independentemente da representatividade de cada categoria da variável resposta (ZHANG; MA, 2012).

Para contornar o problema de dados desbalanceados, é proposto utilizar como classificador, o algoritmo *Kernel Extreme Learning Machine* (KELM), uma rede neural adaptada para problemas de classificação não-lineares (HUANG; ZHOU; DING, 2012). O KELM projeta o conjunto de treinamento em um espaço de alta dimensão para melhorar a assertividade em conjuntos com dados complexos (CAI et al., 2019). A seleção aleatória de pesos, o cálculo eficiente de produtos internos e regularização tornam o algoritmo eficiente (LU et al., 2017).

Conforme explicado no 2, o KELM possui hiperparâmetros que afetam significativamente o desempenho do classificador. Para otimizar os resultados, é proposto aplicar o IGWO ao KELM, como critério de busca. O *Improved Grey Wolf Optimizer* (IGWO) oferece uma hierarquia dinâmica, superando ótimos locais na seleção de hiperparâmetros (CAI et al., 2019).

Esta estratégia procura evitar que as soluções fiquem presas em ótimos locais, aprimorando a eficiência do processo global de otimização. O mecanismo de aprendizado, com reforço adaptativo do tamanho do passo e posição inicial de cada solução, melhora significativamente a eficiência do algoritmo classificador (CAI et al., 2019).

3.2 TRABALHOS CORRELATOS

Com o rápido desenvolvimento da tecnologia da informação, uma grande quantidade de dados é produzida e coletada em diferentes domínios. Algoritmos de classificação

são amplamente usados como ferramenta para extrair conhecimento de forma eficiente a partir destes. No entanto, os algoritmos de classificação tradicionais geralmente assumem que os conjuntos de treinamento são bem equilibrados, com custo de classificação errada igual (LI et al., 2019).

Li et al. (2019) revisa as abordagens para o problema de classificação de dados desbalanceados nos últimos anos. O artigo compara as soluções de acordo com a diferença do nível de pré-processamento de dados, nível de características e nível de algoritmo, respectivamente.

De acordo com Esposito et al. (2021), classificadores de aprendizado de máquina treinados em dados desbalanceados tendem a superestimar a classe majoritária, o que leva a uma maior taxa de classificação incorreta para a classe minoritária, que muitas vezes é a classe de interesse em aplicações do mundo real.

Para dados binários, o limiar de classificação é definido por padrão como $0,5$, o que frequentemente não é ideal para dados desbalanceados. O trabalho de Esposito et al. (2021) propõe um método para ajustar o limiar de decisão, demostrando uma boa estratégia para lidar com o problema de desbalanceamento de classes, dado que a maioria dos classificadores se beneficia da otimização limiar.

Embora vários artigos de revisão possam ser encontrados sobre problemas de classificação para estes casos, Rezvani e Wang (2023) apresenta uma revisão abrangente dos métodos existentes para lidar com questões relacionadas ao tema.

Abdul Bujang et al. (2023) discute as melhores práticas das características do conjunto de dados, métodos e análise comparativa de algoritmos apresentando os métodos de equilíbrio mais comuns publicados de 2015 a 2021 e destaca seu impacto na resolução do problema de dados desbalanceados em três abordagens: nível de dados, nível de algoritmo e nível híbrido.

Quando o cenário é de *big data*, Lin e Chen (2012) revisa e avalia alguns dos métodos mais importantes para a previsão de classes de dados desbalanceados de alta dimensão. A avaliação aborda questões fundamentais do problema: taxa de desequilíbrio, ruídos nos dados, complexidade de sobreposição, falta de informação e seleção de variáveis.

Alsaui et al. (2022) considera várias técnicas de reamostragem e classificação. As descobertas sugerem que o fator mais influente dos algoritmos de reamostragem é o que controla o número de amostras a serem removidas ou geradas sinteticamente, como esperado. O efeito do balanceamento é melhorar o desempenho da classificação da classe minoritária às custas da redução das previsões corretas da classe majoritária.

Além disso, quando as classes estão balanceadas parcialmente, apresentam resultados mais assertivos do que em uma reamostragem onde as classes possuem o mesmo volume. A maior melhoria na classificação foi obtida com a técnica de subamostragem

aleatória com os vizinhos mais próximos e a floresta aleatória (ALSAUI et al., 2022).

Maheshwari, Jain e Jadon (2018) estuda, discute e categoriza métodos existentes para abordar esse problema e fornecer uma revisão abrangente de métodos de aprendizado de máquina aplicados ao problema desequilíbrio de classes. O sistema de classificação proposto, que combina subamostragem aleatória com sobreamostragem SMOTE melhora o desempenho de métricas como *recall*, precisão e *F1-score* em resultados experimentais.

Neste trabalho, os conjuntos de dados desbalanceados disponíveis no repositório *Knowledge Extraction based on Evolutionary Learning* (KEEL) (ALCALÁ-FDEZ et al., 2011) são propostos como critério de comparação com outros métodos sugeridos por Jedrzejowicz e Jedrzejowicz (2021) e Nanni, Fantozzi e Lazzarini (2015a).

O repositório é uma coleção de conjuntos de dados de referência comumente usados para avaliar algoritmos de aprendizado de máquina. Foi desenvolvido pelo grupo de pesquisa *Soft Computing and Intelligent Information Systems* (SCI2S) da Universidade de Granada, na Espanha.

Com objetivo de verificar se é possível obter soluções rápidas e assertivas para este tipo de problema específico, é apresentada a aplicação do IGWO-KELM ao problema de dados desbalanceados utilizando o conjunto de amostras que contém estas características (FERNÁNDEZ; GARCÍA; LUENGO et al., 2010).

Para encontrar a configuração mais adequada dos hiperparâmetros C e γ do KELM, o IGWO é utilizado como método de busca (CAI et al., 2019). Originalmente, esses valores são escolhidos aleatoriamente ou definidos manualmente pelo analista. No entanto, automatizar e acelerar a busca por essas constantes pode ajudar o modelo a alcançar um desempenho mais robusto e preciso.

O IGWO possui vantagens como tempo de execução rápido, sem a necessidade de ajuste manual massivo devido a simplicidade do processo. Com isso, é proposto utilizar este algoritmo para selecionar uma configuração adequada e rápida de hiperparâmetros para o KELM.

3.3 CLASSIFICADOR

O ELM é um tipo de rede neural artificial que consiste em uma camada oculta intermediária entre a camada de entrada e a camada de saída. Na camada oculta, os neurônios combinam linearmente os valores de entrada ponderados, seguidos pela aplicação de uma função de ativação não linear (HUANG; ZHU; SIEW, 2006).

Os resultados são transmitidos para a camada de saída, onde ocorre outra combinação linear ponderada, seguida por uma função de ativação que gera as saídas da rede. Essa arquitetura permite que a rede aprenda relações não lineares nos dados e seja capaz

de realizar tarefas mais complexas, como classificação e regressão, com maior capacidade de representação do que uma rede com muitas camadas (HUANG; ZHU; SIEW, 2006).

Matematicamente uma rede neural com uma única camada pode ser representada por meio da equação:

$$f(x) = h(x)\bar{\beta} = H\bar{\beta} \tag{3.1}$$

Onde:

- x representa os dados de entrada;
- $h(x)$ ou $H_{m\times n}$ representa a matriz de mapeamento de características da camada oculta;
- $\bar{\beta}$ é o peso entre a camada oculta e a camada de saída.

No ELM, o beta denotado como $(\bar{\beta})$, é calculado por meio da fórmula:

$$\bar{\beta} = H^T(\frac{1}{\dot{C}} + HH^T)^{-1}T \tag{3.2}$$

O coeficiente de regularização $(\dot{C})$, adicionado à função objetivo do ELM, é utilizado para controlar o ajuste dos dados de treinamento e a complexidade do modelo. O objetivo dessa inclusão é penalizar o valor dos pesos, de modo que se mantenham pequenos, resultando em uma representação mais simples e regularizada do modelo (HUANG; ZHU; SIEW, 2006).

Vantagens que o KELM apresenta em relação ao ELM são a inicialização aleatória, a presença de poucos parâmetros ajustáveis, a convergência mais rápida, a capacidade de criar modelos mais generalizados e a menor suscetibilidade a ficar preso em ótimos locais (LU et al., 2017).

Por sua vez, o KELM utiliza o truque do kernel (HUANG; ZHOU; DING, 2012), técnica que permite tornar padrões linearmente separáveis. Esta abordagem transforma os dados originais em um espaço de maior dimensão, sem a necessidade de calcular as coordenadas explicitamente dentro deste espaço (CORTES; VAPNIK, 1995).

No modelo KELM, a função de mapeamento $h(x)$ da camada oculta, é substituída pela função kernel $K(u, v)$. Com essa alteração, a equação pode ser reformulada da seguinte maneira:

$$f(x) = h(x)H^T(\frac{1}{\dot{C}} + HH^T)^{-1}T \tag{3.3}$$

De forma matricial:

$$f(x) = \begin{pmatrix} K(x, x_1) \\ \vdots \\ K(x, x_N) \end{pmatrix}^T (\frac{1}{\dot{C}} + \Omega_{KELM})^{-1} T \tag{3.4}$$

Onde a matriz kernel Ω_{KELM} pode ser definida por meio das expressões:

$$\begin{cases} \Omega_{KELM} = HH^T \\ \Omega_{KELMi,j} = h(x_i) \cdot h(x_j) \\ \Omega_{KELMi,j} = K(x_i, x_j) \end{cases} \tag{3.5}$$

A função de base radial (RBF), também conhecida como função gaussiana, é frequentemente utilizada em algoritmos de aprendizado de máquina, como as redes neurais radiais *(RBF neural networks)*. A função gaussiana atribui pesos aos pontos de acordo com sua distância em relação a um ponto central, seguindo uma distribuição normal. Essa abordagem é eficaz para encontrar padrões complexos e não lineares nos dados. Além disso, possui propriedades matemáticas como a universalidade, que permite aproximar qualquer função contínua com precisão arbitrária (BISHOP, 2006).

A função RBF, utilizada para mapear os dados do modelo, é definida por meio da equação:

$$K(u, v) = exp(-\gamma u - v^2) \tag{3.6}$$

O parâmetro gama (γ) controla a forma, o alcance, a suavidade, o comportamento e o desempenho da função, assim como a capacidade de adaptação do modelo aos dados. De acordo com Cai (CAI et al., 2019), o desempenho do KELM é diretamente influenciado pela seleção apropriada de seus parâmetros, tais como o parâmetro de regularização $\dot{C}$ e o valor de γ.

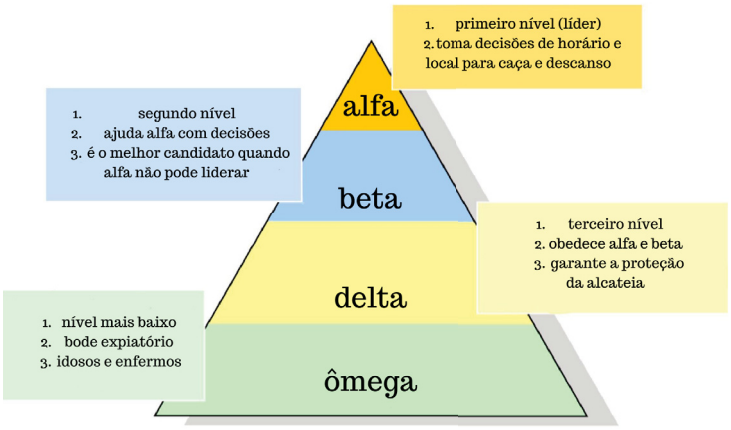
3.4 OTIMIZADOR DE HIPERPÂMETROS

O GWO (MIRJALILI; MIRJALILI; LEWIS, 2014), algoritmo baseado na hierarquia social dos lobos no momento da caça, possui uma divisão de quatro grupos de acordo com a importância de cada indivíduo. Alfa (α), beta (β), delta (δ) e ômega (ω).

O alfa é o líder da alcateia, responsável por determinar o local e momento de caçar. O lobo beta, auxilia o alfa com decisões e controle da matilha. O delta, que obedece o alfa e o beta, é encarregado de reconhecer o território, ficar de sentinela e cuidar dos

filhotes. Todos os outros lobos são colocados na categoria ômega (MIRJALILI; SAREMI et al., 2016).

FIGURA 7 – Hierarquia dos Lobos



Fonte: (CAI et al., 2019)

Cada lobo representa uma solução em potencial para o problema de otimização. A posição de cada lobo é determinada por um vetor de $\vec{D}$ dimensões, onde cada elemento do vetor representa um valor específico (MIRJALILI; MIRJALILI; LEWIS, 2014).

Durante as etapas de busca e atualização de posição dos lobos, o GWO leva em consideração as dimensões do espaço solução do problema, movendo os vetores em direção a melhores resultados (MIRJALILI; SAREMI et al., 2016). A atualização da posição dos lobos no momento da caça é representada pela equação:

$$\vec{D} = |\vec{C} \cdot \vec{X}_{presa}(t) - \vec{X}_{lobo}(t)| \tag{3.7}$$

Onde t é a iteração atual, $\vec{X}_{presa}$ e $\vec{X}_{lobo}$ indicam a posição da presa e do lobo, respectivamente. $\vec{C}$ é o fator de oscilação, calculado por meio da fórmula:

$$\vec{C} = 2\vec{r_1} \tag{3.8}$$

Para atualizar a posição dos lobos, a expressão é dada por:

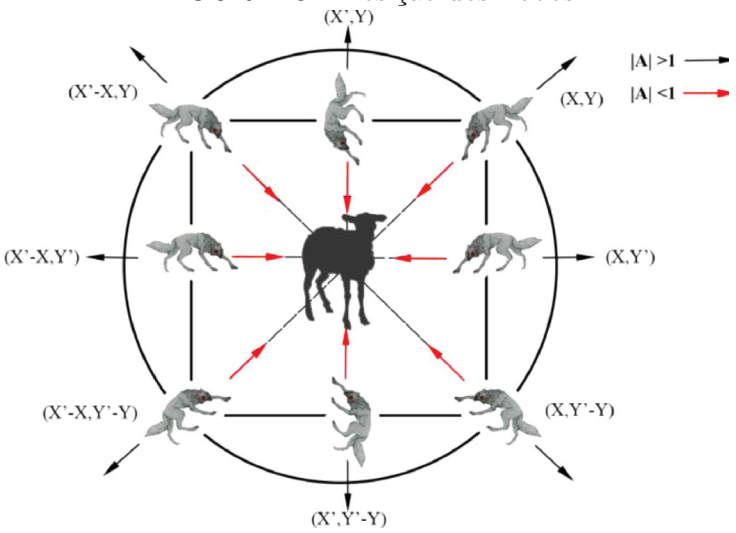
$$\vec{X}_{lobo}(t+1) = \vec{X}_{presa}(t) - \vec{A} \cdot \vec{D} \tag{3.9}$$

A é o fator de convergência determinado pela equação:

$$\vec{A} = 2a \cdot \vec{r_2} - a \tag{3.10}$$

Os vetores $\vec{r_1}$ e $\vec{r_2}$ são definidos aleatoriamente no intervalo $[0, 1]$ e a é reduzido linearmente dentro do intervalo $[0, 2]$ de acordo com o aumento das iterações. O cálculo do $\vec{A}$ e do $\vec{C}$ aproximam os lobos da melhor solução atual. De acordo com a figura abaixo, os lobos atacam quando $|\vec{A}| < 1$.

FIGURA 8 – Posição dos Lobos



Fonte: (CAI et al., 2019)

No entanto, não existe garantia de que a melhor solução atual seja a melhor solução para o problema. Alfa, beta e delta são as melhores soluções apenas para esta iteração. As equações a seguir, são utilizadas para calcular a posição dos outros lobos.

$$\begin{cases} \vec{D}_\alpha = |\vec{C_1} \cdot \vec{X}_\alpha - \vec{X}| \\ \vec{D}_\beta = |\vec{C_2} \cdot \vec{X}_\beta - \vec{X}| \\ \vec{D}_\delta = |\vec{C_3} \cdot \vec{X}_\delta - \vec{X}| \end{cases} \tag{3.11}$$

$$\begin{cases} \vec{X_1} = \vec{X}_\alpha - \vec{A_1} \cdot \vec{D}_\alpha \\ \vec{X_2} = \vec{X}_\beta - \vec{A_2} \cdot \vec{D}_\beta \\ \vec{X_3} = \vec{X}_\delta - \vec{A_3} \cdot \vec{D}_\delta \end{cases} \tag{3.12}$$

$$\vec{X}_{t+1} = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3} \tag{3.13}$$

O principal problema do GWO acontece quando o lobo alfa cai em um ótimo local. Tal resultado pode levar a alcateia toda a convergir para esta solução, perdendo a habilidade de busca global (CAI et al., 2019).

Na hiearquia social original do GWO, alfa, beta e delta são as três melhores soluções, representados como o melhor, segundo melhor e terceiro melhor lobo, respectivamente. O conjunto ômega é composto por todos os outros lobos dentro do grupo. A atualização dos lobos ômega é a baseada nas melhores soluções (MIRJALILI; MIRJALILI; LEWIS, 2014).

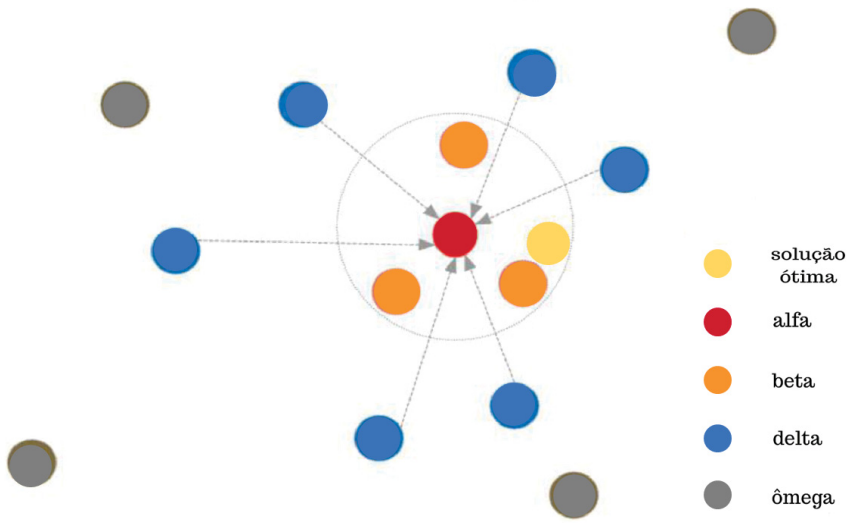
TABELA 1 – Principais diferenças entre GWO e IGWO

HIERARQUIA	GWO	IGWO
Alfa	Melhor solução atual	Melhor solução atual
Beta	Segunda melhor solução	Gerada aleatoriamente perto do alfa
Delta	Terceira melhor solução	Todos os casos restantes menos ômega
Ômega	Todas as outras soluções	Piores soluções

O IGWO propõe uma nova hierarquia com o intuito de evitar ótimos locais. O alfa permanece da mesma maneira, representando a melhor solução conhecida. O beta é caracterizado por um conjunto de lobos que fazem uma busca aleatória ao redor de alfa (CAI et al., 2019).

Ômega é o conjunto de piores soluções atualizadas aleatoriamente dentro do espaço solução global, com o objetivo de garantir a capacidade de busca global. Todos os outros lobos fazem parte do conjunto delta, que são modificados seguindo alfa dentro do espaço solução. Se algum lobo apresentar uma solução melhor que alfa, alfa é substituído (CAI et al., 2019).

FIGURA 9 – IGWO



Fonte: (CAI et al., 2019)

O IGWO possui três principais etapas: inicialização da população, atualização das posições dos lobos e atualização da hierarquia. A nova hierarquia atualiza os lobos de uma maneira diferente. Os lobos beta são representados pela fórmula do conjunto:

$$\beta_{i,j} = \alpha + 2 \cdot D \cdot r - D$$
$$i \in \{1, 2, ..., n_\beta\}$$

Onde n_β é a quantidade de lobos dentro do conjunto beta, j varia dentro do vetor $j \in \{1, 2, ..., d]\}$, sendo d a dimensão do vetor posição, r é um valor aleatório dentro do intervalo $\{0, 1\}$. Por último, D é a distância euclidiana entre alfa atual e a melhor solução dentro do conjunto delta. D pode ser calculada da seguinte maneira:

$$D = \sqrt{|2|} \sum_{j=1}^{d} (\alpha_j - \delta_{melhor,j})^2 \tag{3.14}$$

Se houver uma solução melhor que alfa dentro do conjunto beta, alfa é substituído.

$$\alpha = \begin{cases} \beta_{melhor}, \text{ se } f(\beta_{melhor}) > f(\alpha) \\ \alpha, \text{ caso contrário} \end{cases} \tag{3.15}$$

Para evitar que a população fique presa em soluções ótimas locais, os lobos que não pertencem aos conjuntos alfa e beta são divididos em duas partes. Os lobos delta são atualizados seguindo a posição do alfa, de acordo com as fórmulas apresentadas a seguir:

$$A = 2\tau r_1 - \tau \tag{3.16}$$

Sendo τ, o decrescimento linear no intervalo $\{0, 2\}$ representado pela equação:

$$\tau = 2(1 - \frac{t}{T}) \tag{3.17}$$

Em que T é a quantidade total de iterações e t a iteração atual. r_1, r_2 são valores criados aleatoriamente dentro do intervalo $\{0, 1\}$.

$$C = 2r_2 \tag{3.18}$$

$$L = |C \cdot \alpha_j - \delta_{i,j}| \tag{3.19}$$

O fator de oscilação C e o de convergência L compõem o cálculo do conjunto delta por meio da fórmula:

$${\delta_{i,j}}^{(t+1)} = \alpha_j + A \cdot L \tag{3.20}$$

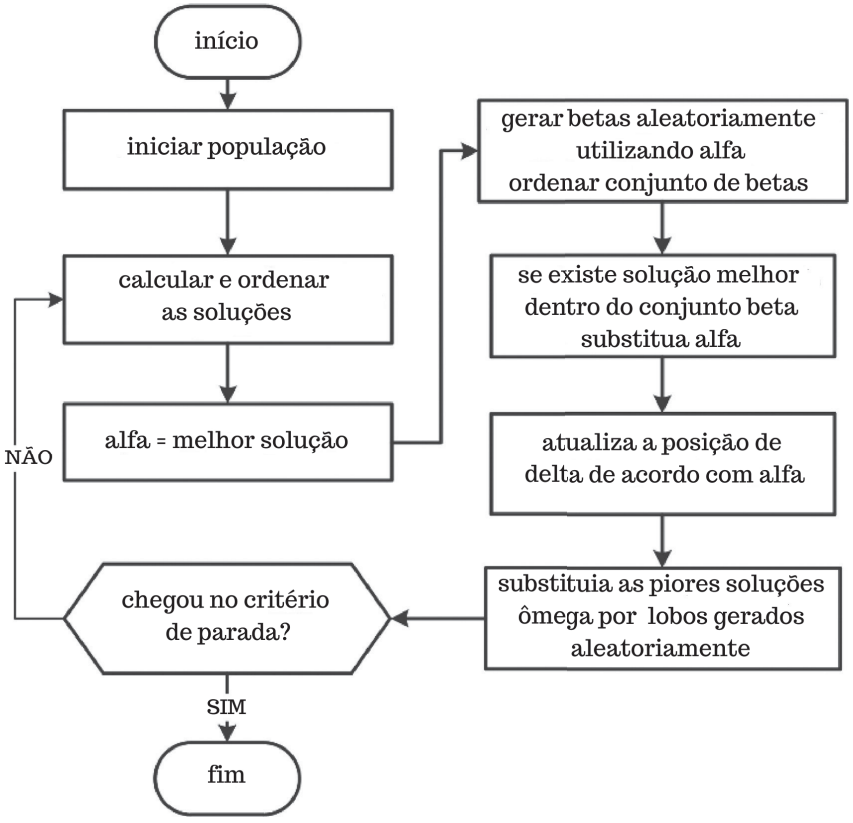
As piores soluções, representadas pelo conjunto ômega, são atualizadas de forma aleatória por meio da equação:

$$\omega_i^{(t+1)} = (UB - LB) \cdot rand(1, d) + LB$$
$$i \in \{1, 2, ..., n_\omega\}$$

Em que LB e UB são os limites superiores e inferiores do conjunto solução, respectivamente. n_ω é a quantidade de lobos dentro do conjunto ômega e d é a dimensão dos vetores de cada posição.

O fluxograma representado pela FIGURA 10 abaixo, tem como objetivo demonstrar como o algoritmo funciona na prática.

FIGURA 10 – Fluxograma IGWO



Fonte: (CAI et al., 2019)

Para validar o modelo proposto anteriormente, foram realizados experimentos computacionais. A implementação levou em consideração os seguintes pontos: tratamento inicial dos dados, criação e atualização de parâmetros, implementação da função KELM com kernel RBF e os testes em 99 bases de dados disponibilizadas por (ALCALÁ-FDEZ et al., 2011).

A função principal que implementa o algoritmo IGWO-KELM recebe um conjunto de dados de treino e teste, número máximo de iterações e o número de lobos de cada conjunto. O algoritmo aplica as etapas IGWO-KELM para otimizar os hiperparâmetros C o γ e retorna a solução alfa, conjunto que possui o maior AUC entre os modelos treinados.

Para demonstrar a implementação, o pseudocódigo abaixo descreve passo a passo as etapas do algoritmo:

Algoritmo 1: IGWO-KELM PARA DADOS DESBALANCEADOS

Entrada: n, n_β, n_ω, $maxgen$
Saída: α

```
Inicie a população de lobos X_i = [C_i, γ_i] para i = (1, 2, ..., n)
Defina n_β como o tamanho do conjunto β
Defina n_ω como o tamanho do conjunto ω
para i = 1, ..., n faça
    Treine o KELM utilizando o conjunto X_i
    Calcule o AUC do KELM para X_i
fim
para t = 1, ..., maxgen faça
    Ordene os resultados pela métrica AUC do maior para o menor
    Defina alfa como a solução com o maior AUC
    Defina o conjunto ômega como as piores n_ω soluções em X_i
    Calcule o conjunto beta utilizando a posição de alfa
    para j = 1, ..., n_β faça
        Treine o KELM utilizando β_j
        Calcule o AUC do KELM para β_j se AUC(β_j) > AUC(α) então
            α = β_j
        fim
    fim
    Calcule o conjunto delta utilizando a posição de alfa
    para k = n_β + 1, ..., n − n_ω faça
        Treine o KELM utilizando δ_k
        Calcule o AUC do KELM para δ_k
    fim
fim
```

Resultado: α

O algoritmo é de fácil compreensão, pois utiliza um número reduzido de parâmetros e operações. A implementação foi realizada na linguagem de programação Python, seguindo as diretrizes definidas neste pseudocódigo.

4 EXPERIMENTOS COMPUTACIONAIS

Neste capítulo, os experimentos computacionais realizados para avaliar o desempenho do modelo proposto e seus resultados serão apresentados e detalhados. O desbalanceamento de classes é um desafio comum em várias aplicações reais conforme exemplicados no capítulo 2. O KELM otimizado pelo algoritmo IGWO foi aplicado ao problema utilizando um conjunto de bases de dados públicas.

Os experimentos foram realizados em um conjunto de bases de dados disponíveis no repositório (ALCALÁ-FDEZ et al., 2011). Cada base é caracterizada por um conjunto de variáveis com os atributos e rótulos de cada observação, onde a distribuição da variável resposta é desbalanceada. Todas as bases disponíveis possuem a divisão em conjuntos de treino contendo 80% das observações e teste, contendo os outros 20%. As variáveis estão formatadas de maneira padronizada para facilitar o uso em experimentos.

Cada base de dados do repositório KEEL (ALCALÁ-FDEZ et al., 2011) possui um conjunto de atributos que representam as variáveis dependentes e uma variável resposta binária. Ao total 99 bases de dados foram testadas neste experimento. Para que a descrição das 99 bases fique intuitiva, o conjunto total foi dividido em quatro conjuntos, separados pela taxa da classe negativa, conforme descrito na tabela 2.

TABELA 2 – Faixas de segmentação

INTERVALO TAXA	QUANTIDADE DE BASES
[0,79% - 3,03%]	25
[3,17% - 6,78%]	25
[7,14% - 10,27%]	25
[10,36% - 65,45%]	24

A base de dados que possui a distribuição mais desbalanceada, é a abalone19. Esta tabela apresenta dados reais, com 4.174 observações sendo apenas 33 pertencentes a classe negativa. As variáveis desta base são apresentadas na tabela 3.

TABELA 3 – Variáveis abalone19

NOME	TIPO	VALORES
Sex	Categórica	[M;F;I]
Length	Real	[0,075: 0,815]
Diameter	Real	[0,055: 0,65]
Height	Real	[0,0: 1,13]
Whole_weight	Real	[0,0020: 2,8255]
Shucked_weight	Real	[0,0010: 1,488]
Viscera_weight	Real	[5,0E-4: 0,76]
Shell_weight	Real	[0,0015: 1,005]
Class	Binária	[positive, negative]

Os dados testados levaram em consideração a validação cruzada, onde o modelo foi treinado em 5 divisões de treino e teste da mesma base, considerando distribuições diferentes em cada conjunto. Cada uma das 5 bases de treinamento possui 80% dos pedidos totais e os 5 conjuntos de teste que representam os outros 20% da base possuem observações diferentes entre si.

A validação cruzada é útil para lidar com problemas de variabilidade e garantir que o modelo tenha um bom desempenho em diferentes subconjuntos de dados. Além disso, ela ajuda a evitar problemas de superajuste e subajuste do modelo (BISHOP, 2006).

A base com a taxa de balanceamento mais equilibrada, é a ecoli-0-vs-1. Esta base possui apenas 220 registros, sendo 45% pertencentes a classe positiva. As variáveis desta base são apresentadas na tabela 4.

TABELA 4 – Variáveis ecoli-0-vs-1

NOME	TIPO	VALORES
Mcg	Real	[0,0: 0,89]
Gvh	Real	[0,16: 1,0]
Lip	Real	[0,48: 1,0]
Chg	Real	[0,5: 1,0]
Aac	Real	[0,0: 0,88]
Alm1	Real	[0,03: 1,0]
Alm2	Real	[0,0: 0,99]
Class	Binária	[positive, negative]

Os parâmetros do C e γ foram definidos por meio do IGWO com o intuito de melhorar o desempenho de classificação. A avaliação do desempenho dos classificadores foi realizada por meio da métrica AUC. A AUC é particularmente relevante para avaliar o desempenho em tarefas com desbalanceamento de classes, pois leva em conta a capacidade do classificador de distinguir cada classe, independentemente da distribuição desigual na base de dados.

Cada teste foi realizado com a quantidade inicial de lobos n, betas n_β, ômega n_ω e de iterações máximas $maxgen$ definidas com o valor 10. O conjunto de lobos com os parâmetros C e γ iniciais foram definidos como $X_i \in \{2^i, 2^j\}$, onde i e j variam dentro do intervalo $\{-n, n\}$. Como critério de parada, foi utilizado o valor máximo que a métrica pode assumir, em que $AUC(\alpha) = 1$.

4.1 CONJUNTO DE DADOS

Para facilitar as análises, as 99 bases foram separadas em 4 tabelas. As principais informações sobre cada base dentro do conjunto serão listadas abaixo, de acordo com o intervalo de segmentação apresentado na tabela 2, onde:

- NOME: Nome da base de dados;

- VOLUME: Quantidade de observações dentro da base;
- TAXA: Quantidade de observações negativas dividida pelo volume total;
- VARIAVEIS: Quantidade de variáveis na base.

TABELA 5 – Tabela intervalo [0,79% - 3,03%]

NOME	VOLUME	TAXA	VARIAVEIS
abalone19	4.174	1%	8
kddcup-rootkit-imap-vs-back	2.225	1%	41
poker-8-9-vs-5	2.075	1%	10
poker-8-vs-6	1.477	1%	10
kr-vs-k-zero-vs-fifteen	2.193	1%	6
kddcup-buffer-overflow-vs-back	2.233	1%	41
kddcup-land-vs-satan	1.610	1%	41
abalone-20-vs-8-9-10	1.916	1%	8
winequality-red-3-vs-5	691	1%	11
shuttle-2-vs-5	3.316	2%	9
poker-8-9-vs-6	1.485	2%	10
winequality-white-3-9-vs-5	1.482	2%	11
kr-vs-k-zero-vs-eight	1.460	2%	6
abalone-19-vs-10-11-12-13	1.622	2%	8
kddcup-land-vs-portsweep	1.061	2%	41
winequality-red-8-vs-6-7	855	2%	11
yeast6	1.484	2%	8
abalone-17-vs-7-8-9-10	2.338	3%	8
abalone-21-vs-8	581	3%	8
kr-vs-k-three-vs-eleven	2.935	3%	6
ecoli-0-1-3-7-vs-2-6	281	3%	7
winequality-red-8-vs-6	656	3%	11
winequality-white-9-vs-4	168	3%	11
abalone-3-vs-11	502	3%	8
yeast5	1.484	3%	8

TABELA 6 – Tabela intervalo [3,17% - 6,78%]

NOME	VOLUME	TAXA	VARIAVEIS
yeast-1-2-8-9-vs-7	947	3%	8
kddcup-guess-passwd-vs-satan	1.642	3%	41
winequality-red-4	1.599	3%	11
yeast4	1.484	4%	8
kr-vs-k-one-vs-fifteen	2.244	4%	6
kr-vs-k-zero-one-vs-draw	2.901	4%	6
poker-9-vs-7	244	4%	10
car-vgood	1.728	4%	6
car-good	1.728	4%	6
flare-F	1.066	4%	11
yeast-2-vs-8	482	4%	8
yeast-1-4-5-8-vs-7	693	4%	8
shuttle-6-vs-2-3	230	4%	9
glass5	214	5%	9
lymphography-normal-fibrosis	148	5%	18
zoo-3	101	5%	16
shuttle-c2-vs-c4	129	5%	9
glass-0-1-6-vs-5	184	5%	9
dermatology-6	358	6%	34
abalone9-18	731	6%	8
ecoli4	336	6%	7
page-blocks-1-3-vs-4	472	6%	10
yeast-1-vs-7	459	7%	7
glass4	214	7%	9
shuttle-c0-vs-c4	1.829	7%	9

TABELA 7 – Tabela intervalo [7,14% - 10,27%]

NOME	VOLUME	TAXA	VARIAVEIS
ecoli-0-1-4-6-vs-5	280	7%	6
ecoli-0-1-4-7-vs-5-6	332	8%	6
cleveland-0-vs-4	173	8%	13
ecoli-0-1-vs-5	240	8%	6
glass2	214	8%	9
led7digit-0-2-4-5-6-7-8-9-vs-1	443	9%	7
glass-0-1-4-6-vs-2	205	9%	9
ecoli-0-1-4-7-vs-2-3-5-6	336	9%	7
ecoli-0-6-7-vs-5	220	9%	6
vowel0	988	9%	13
glass-0-6-vs-5	108	9%	9
glass-0-1-6-vs-2	192	9%	9
ecoli-0-3-4-7-vs-5-6	257	10%	7
ecoli-0-3-4-6-vs-5	205	10%	7
yeast-0-5-6-7-9-vs-4	528	10%	8
ecoli-0-4-6-vs-5	203	10%	6
yeast-0-3-5-9-vs-7-8	506	10%	8
ecoli-0-2-3-4-vs-5	202	10%	7
yeast-0-2-5-6-vs-3-7-8-9	1.004	10%	8
yeast-0-2-5-7-9-vs-3-6-8	1.004	10%	8
ecoli-0-3-4-vs-5	200	10%	7
yeast-2-vs-4	514	10%	8
page-blocks0	5.472	10%	10
ecoli-0-1-vs-2-3-5	244	10%	7
ecoli-0-2-6-7-vs-3-5	224	10%	7

TABELA 8 – Tabela intervalo [10,36% - 65,45%]

NOME	VOLUME	TAXA	VARIAVEIS
ecoli-0-6-7-vs-3-5	222	10%	7
ecoli3	336	10%	7
glass-0-1-5-vs-2	172	10%	9
glass-0-4-vs-5	92	11%	9
yeast3	1.484	11%	8
glass6	214	14%	9
segment0	2.308	14%	19
ecoli2	336	16%	7
new-thyroid1	215	16%	5
newthyroid2	215	16%	5
ecoli1	336	23%	7
vehicle0	846	24%	18
glass-0-1-2-3-vs-4-5-6	214	24%	9
vehicle3	846	25%	18
vehicle1	846	26%	18
vehicle2	846	26%	18
haberman	306	27%	3
yeast1	1.484	29%	8
glass0	214	33%	9
iris0	150	33%	4
pima	768	35%	8
wisconsin	683	35%	9
glass1	214	36%	9
ecoli-0-vs-1	220	65%	7

Com isso, fica claro que pelo menos 75% das bases utilizadas para os testes possuem uma taxa de desbalanceamento menor que 10%, demonstrando que esse conjunto possui exemplos suficientes para verificar a eficácia do método proposto.

4.2 RESULTADOS

Os resultados serão apresentados seguindo o padrão de segmentação por taxa exemplificado na seção anterior. A tabela 10 traz um resumo dos indicadores dentro de cada faixa.

TABELA 9 – Resumo por intervalo de taxas

INTERVALO	TAXA	AUC	TMEDIO	TMAX
[0,79% - 3,03%]	2%	94,73%	0,29	2,78
[3,17% - 6,78%]	5%	94,98%	0,09	0,69
[7,14% - 10,27%]	9%	89,14%	0,35	8,49
[10,36% - 65,45%]	24%	83,94%	0,04	0,39

Os indicadores utilizados na tabela podem ser descritos como:

- INTERVALO: Intervalo de segmentação;
- TAXA: Média da taxa da classe negativa dentro do intervalo;
- AUC: Média do AUC das melhores soluções dentro do intervalo;
- TMEDIO: Tempo médio em segundos para encontrar a melhor solução dentro do intervalo;
- TMAX: Maior tempo em segundos para encontrar a melhor solução dentro do intervalo.

A variação de aproximadamente 10 pontos percentuais de AUC entre a última e as duas primeiras segmentações, indica que o modelo, treinado com apenas 10 iterações como critério de parada, encontra soluções eficientes em casos onde a taxa é menor que 6%. Os indicadores de tempo são influenciados pela dimensão de observações no conjunto de treinamento.

Quando a segmentação é avaliada pela quantidade de observações em cada base, a métrica AUC tem uma variação menor, o que indica que o tamanho da base de dados não altera a qualidade dos resultados do modelo. No entanto, o tempo de treinamento aumenta de acordo com o volume de dados.

TABELA 10 – Resumo por intervalo de volume

INTERVALO	TAXA	AUC	TMEDIO	TMAX
[92 - 215]	12%	90,08%	0,00	0,02
[220 - 506]	11%	94,70%	0,01	0,02
[514 - 1484]	10%	86,88%	0,06	0,23
[1484 - 5472]	5%	91,43%	0,74	8,49

Os resultados apresentados a seguir, trazem detalhes a respeito de cada teste realizado, onde as colunas das tabelas 11, 12, 13 e 14 podem ser descritas como:

- NOME: Nome da base da dados;
- C: Constante de penalidade da melhor solução;
- GAMA: Parâmetro do kernel da melhor solução;
- TEMPO: Tempo em segundos da melhor solução;
- AUC: Maior AUC entre as soluções.

TABELA 11 – Resultados intervalo [0,79% - 3,03%]

NOME	C	GAMA	TEMPO	AUC
abalone19	2,000	0,004	2,78	79%
kddcup-rootkit-imap-vs-back	0,004	0,002	0,27	100%
poker-8-9-vs-5	64,000	0,004	0,20	99%
poker-8-vs-6	128,000	0,002	0,11	99%
kr-vs-k-zero-vs-fifteen	4,000	64,000	0,49	100%
kddcup-buffer-overflow-vs-back	32,000	0,002	0,25	100%
kddcup-land-vs-satan	8,000	0,004	0,16	100%
abalone-20-vs-8-9-10	0,031	0,002	0,24	95%
winequality-red-3-vs-5	0,002	0,008	0,03	99%
shuttle-2-vs-5	2,000	0,002	0,66	100%
poker-8-9-vs-6	4,000	0,008	0,09	100%
winequality-white-3-9-vs-5	16,000	0,008	0,12	78%
kr-vs-k-zero-vs-eight	0,125	64,000	0,23	100%
abalone-19-vs-10-11-12-13	0,250	0,002	0,16	77%
kddcup-land-vs-portsweep	16,000	0,002	0,05	94%
winequality-red-8-vs-6-7	0,002	0,002	0,03	83%
yeast6	0,250	0,004	0,10	99%
abalone-17-vs-7-8-9-10	0,125	0,002	0,47	79%
abalone-21-vs-8	0,031	0,004	0,02	98%
kr-vs-k-three-vs-eleven	64,000	64,000	0,77	100%
ecoli-0-1-3-7-vs-2-6	2,000	0,002	-	100%
winequality-red-8-vs-6	4,000	0,002	0,03	91%
winequality-white-9-vs-4	8,000	0,008	-	100%
abalone-3-vs-11	16,000	2,000	0,02	100%
yeast5	8,000	0,002	0,10	99%

TABELA 12 – Resultados intervalo [3,17% - 6,78%]

NOME	C	GAMA	TEMPO	AUC
yeast-1-2-8-9-vs-7	0,031	0,063	0,05	81%
kddcup-guess-passwd-vs-satan	0,500	0,002	0,13	100%
winequality-red-4	0,250	0,002	0,13	85%
yeast4	0,125	0,002	0,09	93%
kr-vs-k-one-vs-fifteen	8,000	64,000	0,56	100%
kr-vs-k-zero-one-vs-draw	0,125	128,000	0,69	100%
poker-9-vs-7	0,016	2,000	-	100%
car-vgood	0,008	128,000	0,20	100%
car-good	256,000	0,500	0,13	100%
flare-F	0,125	0,031	0,05	98%
yeast-2-vs-8	8,000	0,002	0,02	83%
yeast-1-4-5-8-vs-7	0,031	0,008	0,03	85%
shuttle-6-vs-2-3	0,008	0,002	-	100%
glass5	0,063	0,002	-	99%
lymphography-normal-fibrosis	512,000	0,002	-	100%
zoo-3	512,000	64,000	-	100%
shuttle-c2-vs-c4	2,000	0,002	-	100%
glass-0-1-6-vs-5	0,004	0,004	-	100%
dermatology-6	0,250	8,000	-	100%
abalone9-18	0,500	0,004	0,04	76%
ecoli4	2,000	0,002	-	100%
page-blocks-1-3-vs-4	4,000	0,002	0,00	98%
yeast-1-vs-7	0,250	0,031	0,02	91%
glass4	64,000	0,004	-	87%
shuttle-c0-vs-c4	0,002	0,008	0,14	100%

TABELA 13 – Resultados intervalo [7,14% - 10,27%]

NOME	C	GAMA	TEMPO	AUC
ecoli-0-1-4-6-vs-5	2,000	0,002	-	100%
ecoli-0-1-4-7-vs-5-6	0,125	0,002	-	95%
cleveland-0-vs-4	8,000	0,002	0,02	99%
ecoli-0-1-vs-5	0,004	0,031	-	100%
glass2	512,000	0,002	-	54%
led7digit-0-2-4-5-6-7-8-9-vs-1	512,000	0,125	0,01	100%
glass-0-1-4-6-vs-2	0,016	0,031	-	74%
ecoli-0-1-4-7-vs-2-3-5-6	0,063	0,002	-	91%
ecoli-0-6-7-vs-5	0,016	0,002	-	98%
vowel0	4,000	128,000	0,06	50%
glass-0-6-vs-5	4,000	0,016	-	100%
glass-0-1-6-vs-2	256,000	0,002	-	59%
ecoli-0-3-4-7-vs-5-6	0,250	0,002	-	96%
ecoli-0-3-4-6-vs-5	0,125	0,002	0,02	100%
yeast-0-5-6-7-9-vs-4	0,500	0,002	0,01	94%
ecoli-0-4-6-vs-5	0,008	0,016	-	100%
yeast-0-3-5-9-vs-7-8	0,125	0,002	0,02	81%
ecoli-0-2-3-4-vs-5	0,031	0,004	-	99%
yeast-0-2-5-6-vs-3-7-8-9	0,016	0,002	0,05	90%
yeast-0-2-5-7-9-vs-3-6-8	0,031	0,002	0,05	97%
ecoli-0-3-4-vs-5	0,031	0,004	-	100%
yeast-2-vs-4	0,063	0,002	0,02	98%
page-blocks0	0,031	0,008	8,49	57%
ecoli-0-1-vs-2-3-5	512,000	0,004	0,01	100%
ecoli-0-2-6-7-vs-3-5	0,031	0,002	-	98%

TABELA 14 – Resultados intervalo [10,36% - 65,45%]

NOME	C	GAMA	TEMPO	AUC
ecoli-0-6-7-vs-3-5	0,031	0,002	0,02	99%
ecoli3	0,031	0,002	0,02	96%
glass-0-1-5-vs-2	4,000	0,002	-	55%
glass-0-4-vs-5	2,000	0,002	-	100%
yeast3	0,250	0,002	0,11	98%
glass6	0,250	0,008	-	97%
segment0	0,125	0,008	0,39	50%
ecoli2	8,000	0,002	0,02	96%
new-thyroid1	2,000	0,002	-	100%
newthyroid2	0,002	0,002	-	100%
ecoli1	0,031	0,002	0,02	96%
vehicle0	64,000	0,002	0,05	92%
glass-0-1-2-3-vs-4-5-6	4,000	0,063	-	89%
vehicle3	2,000	0,002	0,05	55%
vehicle1	512,000	0,002	0,03	64%
vehicle2	4,000	0,002	0,05	75%
haberman	0,125	0,008	0,02	86%
yeast1	0,250	0,002	0,09	83%
glass0	0,500	0,004	0,02	67%
iris0	0,004	0,031	-	100%
pima	256,000	0,002	0,03	79%
wisconsin	0,500	0,004	0,03	100%
glass1	0,063	0,002	0,02	75%
ecoli-0-vs-1	512,000	0,002	0,02	62%

Os resultados demonstram que os piores AUCs estão diretamente relacionados aos critérios de parada. A quantidade de iterações máximas foi um limitador visando custo computacional, no entanto, o objetivo principal do teste é avaliar se o IGWO-KELM pode ser um preditor universal para dados desbalanceados. As tabelas acima mostram que o IGWO-KELM consegue resultados de AUC superiores a 90% em 70 das 99 bases, mesmo com testes limitados.

Os indicadores de tempo estão diretamente ligados a dimensão da base, onde as bases com maior volume consequentemente tomam mais tempo de treinamento. No entanto, a média de tempo para encontrar a melhor solução é de $0,2$ segundos, demonstrando que, além de eficiente para estes dados, o IGWO-KELM também é rápido.

Outras metodologias foram aplicadas ao mesmos conjuntos de dados, apresentadas nos artigos de Jedrzejowicz e Jedrzejowicz (2021) e (NANNI; FANTOZZI; LAZZARINI, 2015a). Os resultados destes estudos foram comparados com as métricas do IGWO-KELM.

4.3 COMPARAÇÃO ENTRE CLASSIFICADORES

Para critério de comparação, Jedrzejowicz e Jedrzejowicz (2021) propõe um classificador que utiliza Programação de Expressão Gênica (GEP), um método adaptado para resolver problemas de classificação. O GEP combina o poder de algoritmos genéticos e regressão simbólica para encontrar uma estrutura de programa para o problema de dados desbalanceados.

O artigo tem como foco abordar os desafios apresentados pelo conjuntos de dados desbalanceados disponibilizados no repositório (ALCALÁ-FDEZ et al., 2011), como o baixo desempenho de classificadores tradicionais e limitações de fronteiras de decisão. Os resultados serão comparados com outros classificadores apresentados no artigo, conforme indica a tabela 15.

TABELA 15 – Comparação por classificador

NOME	NBU	ENN	NCR	NONE	OSS	RENN	RUS	SMOTE	TL	GEP-i	IGWO-KELM
abalone19	0,659	0,695	0,695	0,694	0,695	0,695	0,708	0,679	0,694	0,746	0,793
poker-8-vs-6	0,531	0,437	0,439	0,438	0,427	0,437	0,456	0,500	0,437	0,756	0,985
winequality-red-8-vs-6-7	0,695	0,713	0,717	0,711	0,669	0,721	0,674	0,651	0,713	0,714	0,828
yeast5	0,989	0,987	0,986	0,986	0,986	0,987	0,976	0,982	0,986	0,969	0,995
winequality-red-4	0,694	0,659	0,653	0,648	0,651	0,660	0,626	0,653	0,650	0,695	0,849
yeast-2-vs-8	0,838	0,835	0,836	0,836	0,828	0,835	0,850	0,793	0,836	0,817	0,832
dermatology-6	0,988	0,966	0,966	0,879	0,966	0,879	0,966	0,975	0,966	0,990	1,000
ecoli4	0,950	0,916	0,916	0,920	0,920	0,916	0,916	0,934	0,917	0,963	1,000
page-blocks-1-3-vs-4	0,992	0,902	0,908	0,909	0,908	0,902	0,902	0,909	0,909	0,993	0,984
yeast-1-vs-7	0,801	0,802	0,800	0,800	0,800	0,805	0,792	0,776	0,800	0,803	0,911
glass4	0,830	0,723	0,726	0,728	0,714	0,723	0,760	0,752	0,728	0,991	0,866
ecoli-0-1-4-6-vs-5	0,880	0,858	0,858	0,858	0,853	0,858	0,861	0,862	0,858	0,884	1,000
ecoli-0-1-4-7-vs-5-6	0,958	0,948	0,948	0,949	0,945	0,948	0,826	0,943	0,949	0,882	0,954
glass-0-1-4-6-vs-2	0,720	0,685	0,690	0,695	0,680	0,699	0,618	0,703	0,696	0,757	0,736
ecoli-0-1-4-7-vs-2-3-5-6	0,960	0,922	0,924	0,926	0,926	0,915	0,908	0,927	0,925	0,848	0,907
ecoli-0-6-7-vs-5	0,915	0,874	0,879	0,884	0,871	0,874	0,838	0,883	0,883	0,916	0,981
ecoli-0-3-4-7-vs-5-6	0,941	0,925	0,925	0,924	0,919	0,925	0,928	0,906	0,925	0,911	0,961
ecoli-0-3-4-6-vs-5	0,922	0,849	0,850	0,850	0,853	0,849	0,766	0,854	0,849	0,884	1,000
ecoli-0-4-6-vs-5	0,872	0,846	0,846	0,846	0,853	0,846	0,848	0,857	0,846	0,904	1,000
yeast-0-3-5-9-vs-7-8	0,721	0,695	0,690	0,699	0,712	0,680	0,713	0,731	0,698	0,731	0,810
ecoli-0-2-3-4-vs-5	0,900	0,856	0,858	0,862	0,869	0,856	0,816	0,874	0,863	0,930	0,986
yeast-0-2-5-6-vs-3-7-8-9	0,816	0,761	0,762	0,759	0,762	0,764	0,742	0,755	0,760	0,817	0,899
yeast-0-2-5-7-9-vs-3-6-8	0,932	0,916	0,916	0,913	0,888	0,916	0,907	0,816	0,915	0,938	0,973
ecoli-0-3-4-vs-5	0,894	0,835	0,844	0,844	0,860	0,835	0,750	0,860	0,844	0,896	1,000
yeast-2-vs-4	0,864	0,833	0,835	0,833	0,833	0,831	0,822	0,834	0,835	0,875	0,976
page-blocks0	0,954	0,936	0,935	0,931	0,928	0,937	0,935	0,932	0,932	0,968	0,569

TABELA 16 – Comparação por classificador

NOME	NBU	ENN	NCR	NONE	OSS	RENN	RUS	SMOTE	TL	GEP-i	IGWO-KELM
ecoli3	0,894	0,890	0,892	0,906	0,920	0,873	0,923	0,908	0,907	0,927	0,965
glass6	0,864	0,854	0,854	0,854	0,825	0,888	0,855	0,827	0,856	0,936	0,973
segment0	0,987	0,982	0,982	0,982	0,982	0,982	0,982	0,980	0,982	0,986	0,500
ecoli2	0,941	0,928	0,929	0,928	0,928	0,927	0,921	0,942	0,928	0,918	0,958
new-thyroid1	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,978	1,000
newthyroid2	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	0,952	1,000
ecoli1	0,885	0,850	0,852	0,847	0,839	0,853	0,872	0,842	0,848	0,916	0,964
vehicle0	0,904	0,806	0,805	0,811	0,823	0,801	0,809	0,820	0,811	0,939	0,923
vehicle3	0,762	0,701	0,700	0,701	0,698	0,700	0,697	0,699	0,698	0,710	0,555
vehicle1	0,740	0,709	0,713	0,713	0,717	0,708	0,719	0,717	0,713	0,747	0,636
vehicle2	0,920	0,861	0,859	0,858	0,850	0,857	0,834	0,849	0,857	0,927	0,747
haberman	0,656	0,670	0,676	0,645	0,658	0,671	0,611	0,645	0,657	0,662	0,863
glass0	0,826	0,771	0,800	0,802	0,810	0,755	0,800	0,793	0,804	0,842	0,667
iris0	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000	1,000
pima	0,815	0,799	0,809	0,817	0,815	0,789	0,814	0,815	0,813	0,767	0,788
wisconsin	0,978	0,975	0,977	0,983	0,993	0,974	0,983	0,983	0,982	0,972	0,999
glass1	0,660	0,663	0,662	0,660	0,669	0,702	0,702	0,637	0,656	0,770	0,748

Onde cada coluna representa um classificador, conforme descrito abaixo:

- NBU: *Naive Bayes Classifier*;
- ENN: *Edited Nearest Neighbors*;
- NCR: *Neighborhood Cleaning Rule*;
- NONE: *Naive Bayes Without a Resampling*;
- OSS: *One Side Selection*;
- RENN: *Repeated Edited Nearest Neighbors*;
- RUS: *Random Under Sampling*;
- SMOTE: *Synthetic Minority Over-sampling Technique*;
- TL: *Tomek Links*;
- GEP-i: *Incremental Gene Expression Programming Classifier*;
- IGWO-KELM: *Improved Grey Wolf Optimizer Kernel Extreme Learning Machine.*

Os casos onde o AUC do modelo IGWO-KELM é maior que todos os outros resultados ocorre em 28 das 43 bases comparadas, cerca de 65% do total. Quando comparado aos outros classificadores individualmente, o IGWO-KELM tem em média resultados superiores de AUC em 76% do conjunto total de dados, conforme demonstrado na tabela 17.

TABELA 17 – Comparação individual

MÉTODO	COMPARAÇÃO
IGWO-KELM x OSS	81,40%
IGWO-KELM x SMOTE	81,40%
IGWO-KELM x ENN	79,07%
IGWO-KELM x NCR	79,07%
IGWO-KELM x NONE	79,07%
IGWO-KELM x RENN	79,07%
IGWO-KELM x RUS	79,07%
IGWO-KELM x TL	79,07%
IGWO-KELM x NBU	74,42%
IGWO-KELM x GEP-i	74,42%

4.4 COMPARAÇÃO ENTRE ALGORITMOS DE REAMOSTRAGEM

Um segundo artigo, proposto por Nanni, Fantozzi e Lazzarini (2015a), combina diferentes métodos de balanceamento de classe. Esses métodos incluem sobreamostragem, subamostragem e geração sintética de exemplos da classe minoritária. Os experimentos

mostram que a combinação de métodos pode levar a melhorias significativas na capacidade de predição da classe minoritária, em comparação com o uso de métodos individuais. O artigo traz as seguintes técnicas de amostragem:

- SVM: *Support Vector Machine*
- SV-W: *Cost Sensitive Support Vector Machine*
- CS: *Critical SMOTE*
- RB: *RUBoost*
- B-C: *Critical SMOTE RUBoost*
- B-Cov: *Cost Sensitive Critical SMOTE RUBoost*
- HE-S: *HardEnsemble RUBoost*
- HE-A: *HardEnsemble AdaBoost*
- IGWO-KELM: *Improved Grey Wolf Optimizer Kernel Extreme Learning Machine*.

As tabelas 18 e 19 trazem as comparações realizadas:

TABELA 18 – Comparação remostragem

NOME	SVM	SV-W	CS	RB	B-Cov	B-C	HE-S	HE-A	HE-FUS	IGWO-KELM
abalone19	72,52%	74,67%	73,21%	71,97%	76,59%	73,25%	79,02%	83,03%	83,38%	79,31%
yeast6	92,35%	94,44%	94,34%	92,09%	94,96%	94,36%	94,61%	93,93%	94,46%	100,00}
ecoli-0-1-3-7-vs-2-6	95,67%	95,85%	95,89%	94,19%	96,22%	95,86%	96,10%	94,73%	95,88%	100,00}
yeast5	98,97%	99,04%	98,98%	98,90%	99,00%	99,05%	99,09%	99,28%	99,22%	100,00}
yeast-1-2-8-9-vs-7	76,66%	80,18%	81,28%	75,90%	82,35%	81,28%	82,11%	80,57%	81,22%	80,51%
yeast4	87,96%	88,21%	90,34%	89,50%	90,92%	90,33%	89,62%	90,91%	90,96%	100,00}
yeast-2-vs-8	82,49%	86,30%	85,19%	83,42%	87,74%	85,14%	81,22%	82,55%	83,12%	100,00}
yeast-1-4-5-8-vs-7	68,16%	70,96%	69,74%	70,95%	72,57%	71,16%	70,14%	70,31%	70,88%	84,59%
glass5	99,81%	99,88%	99,83%	98,52%	100,00%	99,31%	99,85%	99,86%	99,85%	98,78%
shuttle-c2-vs-c4	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00}
glass-0-1-6-vs-5	98,33%	99,76%	98,90%	97,18%	99,79%	98,96%	98,92%	98,99%	98,97%	100,00}
abalone9-18	96,73%	97,62%	97,32%	95,83%	97,65%	97,31%	97,11%	96,61%	96,99%	75,75%
ecoli4	99,78%	99,75%	98,95%	98,02%	99,86%	98,92%	99,11%	98,83%	99,01%	100,00}
page-blocks-1-3-vs-4	99,85%	99,79%	99,97%	99,97%	99,98%	100,00%	99,82%	100,00%	100,00%	98,43%
yeast-1-vs-7	79,57%	86,28%	85,42%	81,55%	85,85%	85,49%	85,20%	84,91%	85,81%	91,09%
glass4	98,00%	97,04%	97,45%	96,52%	97,27%	96,31%	98,34%	98,28%	98,71%	86,59%
shuttle-c0-vs-c4	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00}
glass2	87,59%	89,57%	86,87%	88,56%	88,07%	88,49%	88,03%	85,45%	87,27%	54,49%
vowel0	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	50,00%
glass-0-1-6-vs-2	85,28%	87,96%	82,33%	87,78%	88,36%	86,36%	86,01%	80,88%	82,17%	58,57%
yeast-0-5-6-7-9-vs-4	88,04%	88,64%	87,28%	88,16%	88,83%	87,80%	87,32%	88,16%	88,75%	93,78%
yeast-2-vs-4	97,56%	98,09%	98,08%	97,38%	98,15%	98,03%	97,89%	98,25%	98,33%	97,63%
page-blocks0	97,97%	98,11%	98,04%	98,21%	98,35%	98,21%	98,12%	99,13%	99,10%	56,89%

TABELA 19 – Comparação remostragem

NOME	SVM	SV-W	CS	RB	B-Cov	B-C	HE-S	HE-A	HE-FUS	IGWO-KELM
ecoli3	95,13%	94,06%	94,25%	94,68%	95,97%	94,23%	94,99%	94,80%	95,11%	96,49%
yeast3	97,41%	97,11%	97,39%	97,09%	97,41%	97,33%	97,41%	97,44%	97,44%	100,00}
glass6	96,94%	92,18%	95,71%	97,44%	98,00%	96,81%	97,98%	98,11%	98,06%	97,30%
segment0	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	50,00%
ecoli2	96,48%	96,51%	96,51%	96,78%	96,98%	96,69%	96,43%	96,81%	97,01%	95,79%
new-thyroid1	100,00%	100,00%	100,00%	99,91%	100,00%	100,00%	100,00%	99,98%	100,00%	100,00}
newthyroid2	100,00%	100,00%	100,00%	99,81%	100,00%	100,00%	99,93%	99,93%	99,97%	100,00}
ecoli1	95,97%	95,92%	95,87%	95,94%	96,46%	95,99%	95,67%	96,05%	96,07%	96,41%
vehicle0	99,85%	99,86%	99,90%	99,81%	99,86%	99,95%	99,61%	99,85%	99,86%	92,27%
glass-0-1-2-3-vs-4-5-6	98,05%	97,83%	98,86%	98,27%	98,88%	98,52%	98,68%	98,81%	99,03%	88,64%
vehicle3	91,13%	91,75%	90,33%	90,49%	91,35%	90,31%	89,56%	89,38%	89,62%	55,47%
vehicle1	93,02%	93,85%	91,93%	92,19%	93,14%	91,92%	91,37%	90,27%	91,08%	63,64%
vehicle2	99,83%	99,87%	99,84%	99,76%	99,90%	99,84%	99,87%	99,87%	99,84%	74,67%
haberman	70,98%	69,85%	67,48%	70,43%	71,16%	69,41%	70,84%	69,75%	72,06%	86,27%
yeast1	79,63%	80,01%	79,64%	80,18%	80,72%	79,83%	80,33%	79,96%	80,52%	83,43%
glass0	85,56%	86,58%	86,34%	86,75%	86,81%	86,74%	86,57%	85,93%	86,97%	66,75%
iris0	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%	100,00%
pima	83,72%	83,06%	83,94%	82,55%	84,12%	83,99%	83,89%	83,20%	83,91%	78,78%
wisconsin	99,52%	99,52%	99,58%	99,16%	99,51%	99,59%	99,56%	99,37%	99,58%	99,88%
glass1	79,34%	79,54%	78,87%	78,89%	79,86%	78,72%	79,10%	80,90%	80,31%	74,76%
ecoli-0-vs-1	99,30%	99,27%	99,35%	99,50%	99,47%	99,36%	99,45%	99,41%	99,49%	62,30%

Quando comparado aos métodos de reamostragem apresentados por Nanni, Fantozzi e Lazzarini (2015a), mesmo em comparação individual, o IGWO-KELM apresenta pouco ganho em relação aos resultados apresentados no artigo, conforme destacado na tabela 20.

TABELA 20 – Comparação individual

MÉTODO	COMPARAÇÃO
IGWO-KELM x SVM	56,82%
IGWO-KELM x RB	56,82%
IGWO-KELM x SV_W	54,55%
IGWO-KELM x CS	52,27%
IGWO-KELM x B_C	52,27%
IGWO-KELM x HE_S	50,00%
IGWO-KELM x B_Cov	47,73%
IGWO-KELM x HE_A	47,73%
IGWO-KELM x HE_FUS	47,73%

Com isso, é possível afirmar que dentro do ambiente de experimentos deste trabalho, o IGWO-KELM traz vantagens em desempenho e tempo quando comparado com outros classificadores para resolver o problema de dados desbalanceados. No entanto, quando comparado aos métodos de reamostragem, demonstra poucos ganhos.

5 CONCLUSÃO

Este trabalho representa uma contribuição significativa para a área de aprendizado de máquina com foco na otimização de hiperparâmetros, propondo uma solução eficiente para o o problema dos dados desbalanceados. A principal motivação foi abordar a dificuldade inerente à distribuição desigual das classes, que pode comprometer a precisão da classificação.

Os experimentos computacionais com o algoritmo IGWO-KELM atingiram altas taxas de AUC nos conjuntos testados. Comparado a outros classificadores, o IGWO-KELM mostrou uma vantagem em aproximadamente 76% dos casos avaliados. Essa superioridade foi evidente mesmo em conjuntos de dados com desbalanceamento inferior a 6%, indicando a robustez do método.

A relação entre a quantidade de iterações máximas e os critérios de parada, sugere que o IGWO-KELM pode ser um preditor universal para dados desbalanceados. Mesmo com limitações computacionais, os resultados de AUC superiores a 90% em uma maioria expressiva das bases reforçam a eficiência do método.

Apesar dos resultados promissores, identificamos oportunidades de aprimoramento. A exploração de diferentes algoritmos de busca de hiperparâmetros, técnicas de reamostragem e abordagens complementares pode proporcionar melhorias adicionais. Os indicadores de tempo, embora dependentes do volume da base de dados, mostram que o IGWO-KELM também é rápido, com uma média de tempo de treinamento de apenas 0,2 segundos.

Em síntese, o IGWO-KELM emerge como uma solução eficiente e confiável para aprimorar o desempenho do classificador em cenários desafiadores de dados desbalanceados. Comparado com estudos anteriores que utilizam as mesmas referências de dados, propostos por Jedrzejowicz e Jedrzejowicz (2021) e Nanni, Fantozzi e Lazzarini (2015a), o IGWO-KELM apresenta métricas superiores. Essa comparação valida a eficácia do método proposto.

Embora o IGWO-KELM tenha apresentado resultados promissores, reconhecemos que ainda há espaço para melhorias e otimizações no método. Recomenda-se explorar diferentes configurações de hiperparâmetros, utilizando outros métodos de meta-heurística e outros classificadores. Avaliar o desempenho em outras bases de dados e considerar outras abordagens complementares, como reamostragem, também podem trazerganhos significativos aos resultados.

Este trabalho contribui para o avanço do conhecimento sobre o problema de dados não balanceados e fornece uma solução para pesquisadores e profissionais que enfrentam este desafio. A abordagem IGWO-KELM oferece uma nova alternativa eficiente

e confiável para melhorar o desempenho do modelo proposto em termos de AUC e tempo de processamento.

Os resultados demonstram que os piores AUCs estão diretamente relacionados aos critérios de parada. No entanto, a quantidade de iterações máximas foi um limitador do trabalho, criada para diminuir custo computacional, o que contribui para a conclusão de que IGWO-KELM pode ser um preditor universal para dados desbalanceados.

Foi demonstrado pelas métricas de AUC superiores a 90% que o modelo atingiu em 70 das 99 bases testadas, onde o IGWO-KELM contribui não apenas com uma abordagem prática para lidar com o problema, mas também estabelece bases para investigações futuras na busca contínua por métodos mais eficazes e robustos.

REFERÊNCIAS

ABDEL-BASSET, Mohamed et al. BSMA: A novel metaheuristic algorithm for multi-dimensional knapsack problems: Method and comprehensive analysis. **Computers Industrial Engineering**, v. 159, p. 107469, 2021.

ABDUL BUJANG, Siti Dianah et al. Imbalanced Classification Methods for Student Grade Prediction: A Systematic Literature Review. **IEEE Access**, v. 11, p. 1970–1989, 2023.

ADOMAVICIUS, G.; TUZHILIN, A. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. **IEEE Transactions on Knowledge and Data Engineering**, v. 17, n. 6, p. 734–749, 2005.

AGRAWAL, Rakesh; IMIELI'NSKI, Tomasz; SWAMI, Arun. Mining association rules between sets of items in large databases. **ACM SIGMOD Record**, ACM, v. 22, n. 2, p. 207–216, 1993.

ALCALÁ-FDEZ, Jesús et al. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. **J. Multiple Valued Log. Soft Comput.**, 2011. Disponível em: <https://sci2s.ugr.es/keel/imbalanced.php#sub10>.

ALPAYDIN, E. **Introduction to machine learning**. [S.l.]: MIT press, 2010.

ALSAUI, Abdulmohsen A. et al. Resampling Techniques for Materials Informatics: Limitations in Crystal Point Groups Classification. **Journal of Chemical Information and Modeling**, v. 62, n. 15, p. 3514–3523, 2022.

ANDRYCHOWICZ, Marcin et al. Learning to learn by gradient descent by gradient descent. In: ADVANCES in Neural Information Processing Systems. [S.l.: s.n.], 2016. P. 3981–3989.

AVCI, Deniz; DOGANTEKIN, Ahmet. An expert diagnosis system for Parkinson disease based on genetic algorithm-wavelet kernel-extreme learning machine. **Parkinson's Disease**, v. 2016, p. 9, 2016.

BANAIEDEZFOULI, Mohammad et al. R-GWO: Representative-based grey wolf optimizer for solving engineering problems. **Applied Soft Computing**, v. 106, p. 107328, 2021.

BISHOP, Christopher. **Neural Networks for Pattern Recognition**. [S.l.]: Oxford University Press, 1995.

_____. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006.

BOWLES, Michael. **Machine Learning in Python: Essential Techniques for Predictive Analysis**. [S.l.]: John Wiley & Sons, 2015.

BREIMAN, Leo. Random Forests. **Machine Learning**, Kluwer Academic Publishers, USA, v. 45, n. 1, p. 5–32, out. 2001.

BREIMAN, Leo et al. **Classification and regression trees**. [S.l.]: Chapman e Hall/CRC, 1984.

BUDA, Mateusz; MAKI, Atsuto; MAZUROWSKI, Maciej A. A systematic study of the class imbalance problem in convolutional neural networks. **Neural Networks**, v. 106, p. 249–259, 2018.

BUREZ, J.; VAN DEN POEL, D. Handling class imbalance in customer churn prediction. **Expert Systems with Applications**, v. 36, 3, Part 1, p. 4626–4636, 2009.

CAI, Z. et al. Evolving an optimal kernel extreme learning machine by using an enhanced grey wolf optimization strategy. **Expert Systems with Applications**, v. 138, 2019.

CHANDOLA, Varun; BANERJEE, Arindam; KUMAR, Vipin. Anomaly detection: A survey. **ACM Computing Surveys (CSUR)**, ACM, v. 41, n. 3, p. 15, 2009.

CHATTERJEE, Samprit; HADI, Ali S. **Regression analysis by example**. [S.l.]: John Wiley & Sons, 1998.

CHAWLA, Nitesh V. et al. SMOTE: Synthetic Minority over-Sampling Technique. **J. Artif. Int. Res.**, AI Access Foundation, El Segundo, CA, USA, v. 16, n. 1, p. 321–357, jun. 2002.

CHOLLET, François. **Deep Learning with Python**. [S.l.]: Manning Publications, 2018.

CORTES, Corinna; VAPNIK, Vladimir. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

D., Berrar. **Performance measures for binary classification**. [S.l.: s.n.], 2018. v. 1-3, p. 546–560.

DUA, Sumeet; DU, Xian. **Data Mining and Machine Learning in Cybersecurity**. [S.l.]: CRC Press, 2016. P. 331.

EIBEN, A.E.; SMITH, J.E. **Introduction to Evolutionary Computing**. [S.l.]: Springer, 2015.

ELKAN, Charles. The foundations of cost-sensitive learning. **Proceedings of the 17th International Joint Conference on Artificial Intelligence**, p. 973–978, 2001.

ESHTAY, Mohammed; FARIS, Hossam; OBEID, Nadim. Metaheuristic-based extreme learning machines: a review of design formulations and applications. **International Journal of Machine Learning and Cybernetics**, v. 10, jun. 2019.

ESPOSITO, Carmen et al. GHOST: Adjusting the Decision Threshold to Handle Imbalanced Data in Machine Learning. **Journal of Chemical Information and Modeling**, v. 61, n. 6, p. 2623–2640, 2021.

FARIS, Hossam et al. Grey wolf optimizer: a review of recent variants and applications. **Neural Computing and Applications**, v. 30, p. 413–435, 2018.

FERNÁNDEZ, Alberto; GARCÍA, Salvador; GALAR, Mikel et al. **Learning from Imbalanced Data Sets**. [S.l.]: Springer Cham, 2018.

FERNÁNDEZ, Alberto; GARCÍA, Salvador; LUENGO, Julian et al. Genetics-Based Machine Learning for Rule Induction: Taxonomy, Experimental Study and State of the Art. **IEEE Transactions on Evolutionary Computation**, 2010.

FERNÁNDEZ, Alberto; LÓPEZ, Victoria et al. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. **Knowledge-Based Systems**, v. 42, p. 97–110, 2013.

FEURER, Matthias; HUTTER, Frank. **Automated Machine Learning: Methods, Systems, Challenges**. [S.l.]: Springer International Publishing, 2019. P. 3–33.

FINN, Chelsea; ABBEEL, Pieter; LEVINE, Sergey. Model-agnostic meta-learning for fast adaptation of deep networks. In: JMLR. ORG. PROCEEDINGS of the 34th International Conference on Machine Learning-Volume 70. [S.l.: s.n.], 2017. P. 1126–1135.

GERON, Aurelien. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. 2nd. [S.l.]: O'Reilly Media, Inc., 2019.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. [S.l.]: MIT press, 2016.

GRAHAM, Paul. A plan for spam. **http://www.paulgraham.com/**, 2002.

HABEEB, Riyaz Ahamed et al. Real-time big data processing for anomaly detection: A Survey. **International Journal of Information Management**, v. 45, p. 289–307, 2019.

HAIXIANG, Guo et al. Learning from class-imbalanced data: Review of methods and applications. **Expert Systems with Applications**, v. 73, p. 220–239, 2017.

HAN, Jiawei; KAMBER, Micheline; PEI, Jian. **Data Mining: Concepts and Techniques**. 3. ed. [S.l.]: Morgan Kaufmann Publishers, 2012.

HANOON, Marwah Sattar et al. Developing machine learning algorithms for meteorological temperature and humidity forecasting at Terengganu state in Malaysia. **Scientific Reports**, v. 11, n. 1, 2021.

HASTIE, Trevor; TIBSHIRANI, Robert; FRIEDMAN, Jerome. **The Elements of Statistical Learning: Data Mining, Inference, and Prediction**. [S.l.]: Springer, 2009.

HE, Haibo; BAI, Yang et al. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In: 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence). [S.l.: s.n.], 2008. P. 1322–1328.

HE, Haibo; GARCIA, Edwardo. **Imbalanced Learning: Foundations, Algorithms, and Applications**. [S.l.]: John Wiley & Sons, 2009.

HE, Haibo; GARCIA, Edwardo A. Learning from Imbalanced Data. **IEEE Transactions on knowledge and data engineering**, IEEE, v. 21, n. 9, p. 1263–1284, 2009.

HEIDARI, Ali Asghar; PAHLAVANI, Parham. An efficient modified grey wolf optimizer with Lévy flight for optimization tasks. **Applied Soft Computing**, v. 60, p. 115–134, 2017.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the Dimensionality of Data with Neural Networks. **Science**, v. 313, n. 5786, p. 504–507, 2006.

HINTON, Geoffrey E; SEJNOWSKI, Terrence J. **Unsupervised Learning: Foundations of Neural Computation**. [S.l.]: MIT Press, 1999.

HODGE, Victoria J.; AUSTIN, Jim. A survey of outlier detection methodologies. **Artificial Intelligence Review**, v. 22, n. 2, p. 85–126, 2004.

HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. **Neural Networks**, v. 2, n. 5, p. 359–366, 1989.

HUANG, Guang-Bin; ZHOU, Hong; DING, Xue. Extreme learning machine for regression and multiclass classification. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, v. 42, n. 2, p. 513–529, 2012.

HUANG, Guang-Bin; ZHU, Qin-Yu; SIEW, Chee Kheong. Extreme learning machine: theory and applications. **Neurocomputing**, v. 70, p. 489–501, 2006.

HYNDMAN, Rob J.; ATHANASOPOULOS, George. **Forecasting: principles and practice**. 2. ed. [S.l.]: OTexts, 2018. Disponível em: <`https://otexts.com/fpp2/`>.

JAIN, Anil K; DUBES, Richard C. **Algorithms for clustering data**. [S.l.]: Prentice-Hall, Inc., 1988.

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

JAMES, Gareth et al. **An Introduction to Statistical Learning: with Applications in R**. [S.l.]: Springer, 2013.

JEDRZEJOWICZ, J.; JEDRZEJOWICZ, P. GEP-based classifier for mining imbalanced data. **Expert Systems with Applications**, v. 164, 2021.

JOLLIFFE, Ian. **Principal component analysis**. [S.l.]: Springer Science Business Media, 2011.

JORDAN, MI; MITCHELL, TM. **Machine learning: Trends, perspectives, and prospects**. [S.l.]: Science, 2015.

JURAFSKY, Daniel; MARTIN, James H. **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition**. [S.l.]: Pearson Education, 2019.

KARPAGACHELVI, S.; ARTHANARI, M.; SIVAKUMAR, M. Classification of ECG Signals Using Extreme Learning Machine. **Computer and Information Science**, v. 4, n. 1, p. 42–52, 2010.

LECUN, Yann; BENGIO, Yoshua; HINTON, Geoffrey. Deep learning. **Nature**, v. 521, n. 7553, p. 436–444, 2015.

LEIPPOLD, Markus; WANG, Qian; ZHOU, Wenyu. Machine learning in the Chinese stock market. **Journal of Financial Economics**, v. 145, 2, Part A, p. 64–82, 2022.

LI, Yan-Xia et al. Review of imbalanced data classification methods; [] **Kongzhi yu Juece/Control and Decision**, v. 34, n. 4, p. 673–688, 2019.

LIN, Wei-Jiun; CHEN, James J. Class-imbalanced classifiers for high-dimensional data. **Briefings in Bioinformatics**, v. 14, n. 1, p. 13–26, 2012. ISSN 1467-5463.

LIU, Shuo; TING, Kai Ming; ZHOU, Zhi-Hua. Exploratory undersampling for class-imbalance learning. **IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)**, IEEE, v. 39, n. 2, p. 539–550, 2009.

LU, Huijuan et al. A kernel extreme learning machine algorithm based on improved particle swarm optimization. **Memetic Computing**, v. 9, n. 2, p. 121–128, 2017.

MAHESHWARI, Satyam; JAIN, R.C.; JADON, R.S. An Insight into Rare Class Problem: Analysis and Potential Solutions. **Journal of Computer Science**, Science Publications, v. 14, n. 6, 2018.

MARR, Bernard. **Big data: Using smart big data, analytics and metrics to make better decisions and improve performance**. [S.l.]: John Wiley & Sons, 2015.

MEIDANI, Kourosh et al. Adaptive grey wolf optimizer. **Neural Computing and Applications**, Springer, v. 34, n. 23, p. 7711–7731, 2022.

MINHAS, Rashid et al. Human action recognition using extreme learning machine based on visual vocabularies. **Neurocomputing**, v. 73, n. 10, p. 1906–1917, 2010. Subspace Learning / Selected papers from the European Symposium on Time Series Prediction.

MIRJALILI, S.; MIRJALILI, S. M.; LEWIS, A. Grey Wolf Optimizer. **Advances in Engineering Software**, v. 69, p. 46–61, 2014.

MIRJALILI, Seyedali; SAREMI, Seyed et al. Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization. **Expert Systems with Applications**, v. 47, p. 106–119, 2016.

MITCHELL, Tom M. **Machine learning**. [S.l.]: McGraw-hill New York, 1997. v. 1.

MOHAMMED, Ali A et al. Human face recognition based on multidimensional PCA and extreme learning machine. **Pattern Recognition**, Elsevier, v. 44, n. 10-11, p. 2588–2597, 2011.

MURPHY, Kevin P. **Machine learning: A probabilistic perspective**. [S.l.]: MIT press, 2012.

NANNI, Loris; FANTOZZI, Carlo; LAZZARINI, Nicola. Coupling Different Methods for Overcoming the Class Imbalance Problem. **Neurocomputing**, Elsevier Science Publishers B. V., NLD, v. 158, n. 100, p. 48–61, jun. 2015.

_____. Coupling different methods for overcoming the class imbalance problem. **Neurocomputing**, v. 158, p. 48–61, 2015.

NIELSEN, Michael A. **Neural Networks and Deep Learning**. [S.l.]: Determination press, 2015.

PAN, Chen et al. Leukocyte image segmentation by visual attention and extreme learning machine. **Neural Computing and Applications**, Springer, v. 21, n. 6, p. 1217–1227, 2012.

PEDREGOSA, F. et al. Scikit-learn: Machine Learning in Python. **Journal of Machine Learning Research**, v. 12, p. 2825–2830, 2011.

PRATI, Ronaldo C; BATISTA, Gustavo EAPA; SILVA, Diego F. Class imbalance revisited: a new experimental setup to assess the performance of treatment methods. **Knowledge and Information Systems**, Springer, v. 45, n. 2, p. 247–270, 2015.

RAGHU, Maithra et al. Rapid learning or feature reuse? Towards understanding the effectiveness of MAML. **Advances in neural information processing systems**, v. 32, p. 11254–11265, 2019.

RASCHKA, Sebastian; MIRJALILI, Vahid. **Python Machine Learning**. [S.l.]: Packt Publishing, 2017.

RAWLINGS, John O; PANTULA, Sastry G; DICKEY, David A. **Applied regression analysis: a research tool**. [S.l.]: Springer, 2001.

REZVANI, Salim; WANG, Xizhao. A broad review on class imbalance learning techniques. **Applied Soft Computing**, v. 143, 2023.

RICCI, Francesco; ROKACH, Lior; SHAPIRA, Bracha. **Introduction to recommender systems handbook**. [S.l.]: Springer, 2011.

RIZQI, Fakhrur; LESTARI, Firda Fajar; SAPUTRO, Hendry. Solving School Bus Routing Problem using Genetic Algorithm-based Model. **International Journal of Engineering & Technology**, Science Publishing Corporation Inc., v. 7, p. 180–184, 3.28 2018.

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

SEIFFERT, Chris et al. RUSBoost: A hybrid approach to alleviating class imbalance. **IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans**, IEEE, v. 40, n. 1, p. 185–197, 2010.

SERRADILLA, Oscar et al. Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. **Applied Intelligence**, v. 52, jan. 2022.

SHAHVAROUGHI FARAHANI, Mohammad; RAZAVI HAJIAGHA, Saeed H. Forecasting stock price using integrated artificial neural network and metaheuristic algorithms compared to time series models. **Soft Computing**, Springer, v. 25, p. 8483–8513, 2021.

SUN, Yanmin; KAMEL, Mohamed S. et al. Cost-sensitive boosting for classification of imbalanced data. **Pattern Recognition**, v. 40, n. 12, p. 3358–3378, 2007.

SUN, Yanqing; WONG, Albert KC; KAMEL, Mohamed S. Classification of imbalanced data: A review. **International Journal of Pattern Recognition and Artificial Intelligence**, World Scientific, v. 23, n. 04, p. 687–719, 2009.

TALBI, El-Ghazali. Metaheuristics: from design to implementation. **John Wiley & Sons**, Wiley Online Library, v. 74, n. 12, p. 2511–2511, 2009.

TAMSSAOUET, Karim; DAUZÈRE-PÉRÈS, Stéphane; YUGMA, Claude. Metaheuristics for the job-shop scheduling problem with machine availability constraints. **Computers Industrial Engineering**, v. 125, p. 1–8, 2018.

TANG, Jiexiong; DENG, Chenwei; HUANG, Guang-Bin. Extreme learning machine for multilayer perceptron. **IEEE Transactions on Neural Networks and Learning Systems**, IEEE, v. 26, n. 8, p. 1890–1901, 2015.

TOPOL, Eric J. High-performance medicine: the convergence of human and artificial intelligence. **Nature medicine**, Nature Publishing Group, v. 25, n. 1, p. 44–56, 2019.

WANG, Jie-Sheng; LI, Shu-Xia. An Improved Grey Wolf Optimizer Based on Differential Evolution and Elimination Mechanism. **Scientific Reports**, Nature Publishing Group, v. 9, n. 1, p. 7181, 2019.

WANG, Jingru. A hybrid machine learning model for sales prediction. In: 2020 International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI). [S.l.: s.n.], 2020. P. 363–366.

WANG, Mingwen; ZHANG, Jinde et al. Grey wolf optimization evolving kernel extreme learning machine: Application to bankruptcy prediction. **Engineering Applications of Artificial Intelligence**, v. 63, p. 54–68, 2017.

ZANELLA, Andrea et al. Internet of Things for Smart Cities. **IEEE Internet of Things Journal**, v. 1, n. 1, p. 22–32, 2014.

ZHANG, Cha; MA, Yunqian. **Ensemble Machine Learning: Methods and Applications**. [S.l.]: Springer Publishing Company, Incorporated, 2012.

ZHOU, Zhi-Hua. **Ensemble Methods: Foundations and Algorithms**. 1st. [S.l.]: Chapman Hall/CRC, 2012.

ZHU, Aimin et al. Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. **Journal of Systems Engineering and Electronics**, v. 26, n. 2, p. 317–328, 2015.