UNIVERSIDADE FEDERAL DO PARANÁ

WAGNER ALEXANDRE CHAVES

MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE APRIORI, OR-TOOLS E VETORIZAÇÃO PARA MINERAÇÃO DE PADRÕES E OTIMIZAÇÃO DE PROCESSOS NA INDÚSTRIA 4.0

CURITIBA

2025

WAGNER ALEXANDRE CHAVES

MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE APRIORI, OR-TOOLS E VETORIZAÇÃO PARA MINERAÇÃO DE PADRÕES E OTIMIZAÇÃO DE PROCESSOS NA INDÚSTRIA 4.0

Memorial de Projetos apresentado ao curso de Especialização em Inteligência Artificial Aplicada, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Especialista em Inteligência Artificial Aplicada.

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

CURITIBA
2025

MINISTÉRIO DA EDUCAÇÃO
SETOR DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PÓS-GRADUAÇÃO
CURSO DE PÓS-GRADUAÇÃO INTELIGÊNCIA ARTIFICIAL APLICADA - 40001016399E1

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **WAGNER ALEXANDRE CHAVES, intitulada: MEMORIAL DE PROJETOS: INTEGRAÇÃO ENTRE APRIORI, OR-TOOLS E VETORIZAÇÃO PARA MINERAÇÃO DE PADRÕES E OTIMIZAÇÃO DE PROCESSOS NA INDÚSTRIA 4.0**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

A outorga do título de especialista 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, 21 de Outubro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este memorial técnico apresenta uma análise aplicada sobre a integração entre o algoritmo Apriori, a biblioteca de otimização OR-Tools e técnicas de vetorização, com foco na extração de padrões e na otimização de processos complexos em ambientes industriais e de ciência de dados. O estudo descreve os fundamentos teóricos de cada ferramenta, explora seus princípios de funcionamento e demonstra, de forma prática, como a combinação entre mineração de dados e otimização combinatória pode gerar ganhos significativos em eficiência, previsibilidade e redução de custos. São discutidas aplicações em automação de testes eletrônicos, manutenção preditiva e planejamento logístico, ilustradas por exemplos de regras de associação, modelos de decisão e comparativos de desempenho. A abordagem proposta conecta a análise de dados à tomada de decisão operacional, permitindo que informações históricas alimentem modelos otimizados capazes de aprender e se adaptar em tempo real. Conclui-se que a integração Apriori + OR-Tools + Vetorização constitui uma estratégia sólida para o desenvolvimento de sistemas inteligentes alinhados à Indústria 4.0, promovendo maior competitividade, confiabilidade e automação nas organizações.

Palavras-chave: Apriori; OR-Tools; vetorização; otimização combinatória; indústria 4.0.

ABSTRACT

This technical report presents an applied analysis of the integration between the Apriori algorithm, the OR-Tools optimization library, and vectorization techniques, focusing on pattern extraction and the optimization of complex processes in industrial and data science environments. The study describes the theoretical foundations of each tool, explores their core principles, and demonstrates how the combination of data mining and combinatorial optimization can lead to significant gains in efficiency, predictability, and cost reduction. Applications in automated electronic testing, predictive maintenance, and logistics planning are discussed and illustrated through examples of association rules, decision models, and performance comparisons. The proposed approach connects data analysis to operational decision-making, allowing historical information to feed optimized models capable of learning and adapting in real time. It is concluded that the integration of Apriori, OR-Tools, and vectorization constitutes a solid strategy for the development of intelligent systems aligned with Industry 4.0, fostering greater competitiveness, reliability, and automation within organizations.

Keywords: Apriori; OR-Tools; vectorization; combinatorial optimization; industry 4.0.

SUMÁRIO

1 PARECER TÉCNICO

O avanço da Inteligência Artificial (IA) revolucionou a forma como indústrias, serviços e comércio tomam decisões estratégicas. Entre as técnicas mais impactantes, destacam-se a descoberta de padrões em grandes volumes de dados e a otimização combinatória de processos complexos. Essas duas áreas, quando combinadas, criam sistemas capazes de aprender a partir de dados históricos e, ao mesmo tempo, encontrar soluções eficientes para problemas que antes demandariam esforço humano intensivo e tempo excessivo (Russell; Norvig, 2022).

Dois exemplos notáveis são o algoritmo Apriori, fundamental para extração de regras de associação (Agrawal; Srikant, 1994), e o OR-Tools, conjunto de bibliotecas de otimização desenvolvidas pelo Google, amplamente aplicadas em problemas de roteamento, escalonamento e alocação de recursos (Google, 2025). Ao integrar essas ferramentas, é possível criar soluções avançadas para problemas como recomendação de produtos, previsão de falhas em equipamentos, roteamento de veículos e planejamento de testes em linhas de produção.

1.1 FUNDAMENTOS DO ALGORITMO APRIORI

O Apriori é um algoritmo clássico de mineração de dados que identifica relações entre itens em grandes bases transacionais. Ele se baseia em medidas como suporte (frequência de ocorrência de um conjunto de itens), confiança (probabilidade de ocorrer um item dado outro) e *lift* (indicador de ganho sobre a ocorrência esperada). A principal característica do Apriori é a propriedade de fechamento descendente, denominada *downward closure*, que permite eliminar combinações improváveis ainda nas primeiras etapas da busca, economizando processamento (Agrawal; Srikant, 1994).

Apesar de simples e interpretável, o Apriori enfrenta desafios de escalabilidade quando aplicado a bases com muitos itens ou alta cardinalidade. Para contornar essa limitação, técnicas de redução de dimensionalidade e estruturas de dados otimizadas são frequentemente aplicadas, como vetorização e filtragem de itens pouco relevantes. Essas abordagens possibilitam acelerar a geração de conjuntos candidatos e melhorar a eficiência computacional em contextos de grande volume de dados.

Na prática, o Apriori é amplamente utilizado em análises de cestas de compras (*market basket analysis*), sistemas de recomendação e descoberta de padrões de falhas em processos industriais. Em ambientes de manufatura, por exemplo, o algoritmo pode revelar combinações recorrentes de defeitos, componentes ou condições de operação que frequentemente ocorrem juntas, permitindo à engenharia atuar de forma preventiva e otimizar o processo produtivo.

Além disso, o Apriori também tem aplicações relevantes em áreas como diagnóstico médico, detecção de fraudes e manutenção preditiva. Em hospitais, pode identificar correlações entre sintomas e diagnósticos; em instituições financeiras, ajuda a encontrar padrões suspeitos em transações; e em fábricas, auxilia na criação de planos de manutenção baseados em padrões de falhas históricas. Esses exemplos demonstram a versatilidade do algoritmo e sua importância como base para sistemas inteligentes de apoio à decisão.

1.2 FUNDAMENTOS DO OR-TOOLS

O OR-Tools é uma suíte de otimização que abrange problemas de programação linear, inteira, restrições lógicas e roteamento. Seu núcleo CP-SAT (*Constraint Programming – Satisfiability*) oferece alta performance na resolução de problemas NP-complexos, como o Caixeiro Viajante (*Travelling Salesman Problem – TSP*), o roteamento de veículos (*Vehicle Routing Problem – VRP*) e a alocação de recursos sob restrições complexas (Google, 2025). A modelagem é feita através de variáveis de decisão, restrições matemáticas e de uma função objetivo, que pode ser minimizada ou maximizada conforme o problema proposto (Goldbarg; Luna, 2015).

Para problemas logísticos, o OR-Tools suporta múltiplas restrições: janelas de tempo, capacidade de veículos, custos variáveis e balanceamento de carga. Em contextos industriais, pode ser aplicado para organizar sequências de testes de placas eletrônicas, reduzir deslocamentos de cabeçotes CNC ou otimizar rotas de técnicos de manutenção em grandes plantas.

Além dessas aplicações, o OR-Tools também é amplamente utilizado em setores como telecomunicações, energia e manufatura avançada, onde o planejamento eficiente de recursos é essencial. Por exemplo, pode otimizar a alocação de tarefas entre máquinas de uma linha de produção, reduzir o tempo total de setup e minimizar paradas não programadas. Em centros de distribuição, auxilia

no sequenciamento de pedidos e na definição de rotas de coleta, garantindo menor custo operacional e maior cumprimento de prazos. Essa versatilidade torna o OR-Tools uma das ferramentas mais poderosas e acessíveis para a implementação prática de algoritmos de otimização combinatória.

1.3 INTEGRAÇÃO PRÁTICA: APRIORI + OR-TOOLS + VETORIZAÇÃO

A combinação entre descoberta de padrões e otimização é uma estratégia poderosa para transformar dados em decisões práticas. O processo pode ser dividido em três etapas principais: (1) mineração de padrões com o Apriori, para identificar associações relevantes entre eventos, produtos ou falhas (Agrawal; Srikant, 1994); (2) enriquecimento dos dados por vetorização, que converte informações em representações numéricas e facilita cálculos de similaridade e distância (Russell; Norvig, 2022); e (3) modelagem do problema de decisão com o OR-Tools, que gera soluções ótimas considerando restrições e objetivos do negócio (Google, 2025).

Essa integração permite criar sistemas híbridos em que padrões descobertos nos dados alimentam modelos de otimização combinatória, resultando em soluções mais rápidas e precisas (Lourenço; Martins; Ribeiro, 2017).

Em testes eletrônicos, por exemplo, dados históricos de falhas podem ser processados pelo Apriori para revelar combinações recorrentes de pinos, relés e barramentos com maior índice de defeito. Essas informações alimentam um modelo no OR-Tools que reorganiza a ordem dos testes, reduzindo o tempo total de execução ou minimizando o número de movimentos de uma máquina CNC. O resultado é menor desgaste mecânico, maior confiabilidade e ganho de produtividade no processo de automação.

Na logística, previsões de demanda podem ser integradas com padrões de compra detectados pelo Apriori. Isso possibilita ao OR-Tools gerar rotas otimizadas que consideram regiões de maior probabilidade de venda e restrições como capacidade dos caminhões, janelas de entrega e custos operacionais, reduzindo prazos e desperdícios de forma inteligente.

Essa abordagem híbrida reflete o conceito de inteligência operacional, em que a mineração de dados fornece conhecimento e o otimizador traduz esse conhecimento em ação. A sinergia entre Apriori, vetorização e OR-Tools representam, portanto, um

caminho sólido para empresas que buscam automação avançada, eficiência contínua e decisões orientadas por dados no contexto da Indústria 4.0.

1.4 BENEFÍCIOS ESTRATÉGICOS

A integração entre Apriori e OR-Tools oferece vantagens competitivas significativas para empresas e setores que dependem de decisões orientadas por dados. Essa combinação une a capacidade analítica de descoberta de padrões com o poder da otimização matemática, permitindo o desenvolvimento de soluções inteligentes, eficientes e adaptáveis (Russell; Norvig, 2022; Goldbarg; Luna, 2015).

Entre os principais benefícios estão a redução de custos operacionais, o aumento da confiabilidade de processos industriais, a adaptação a cenários dinâmicos e o apoio à tomada de decisão baseada em dados. Em aplicações industriais — especialmente na automação de testes eletrônicos e manutenção preditiva — o uso combinado dessas ferramentas permite criar sistemas de autoaprendizado que ajustam fluxos de teste ou manutenção em tempo real, impactando diretamente o retorno sobre o investimento (*Return on Investment – ROI*) e a eficiência operacional (Lourenço; Martins; Ribeiro, 2017).

Para exemplificar os tipos de resultados que podem ser obtidos com a aplicação do algoritmo Apriori, apresenta-se no Quadro 1 um conjunto de regras de associação geradas a partir de dados simulados, com seus respectivos valores de suporte, confiança e *lift*. Esses resultados refletem o potencial do Apriori em identificar padrões recorrentes e relações úteis em processos de manufatura, varejo e automação.

QUADRO 1 – EXEMPLO DE RESULTADOS DE REGRAS DE ASSOCIAÇÃO

Item A	Item B	Suporte (%)	Confiança (%)	Lift
Relé	Falha no Barramento	35	78	1,20
Placa X	Cabo Y	42	83	1,50
Sensor Z	Fonte W	28	70	1,10
Placa A	Atualização de Firmware	30	82	1,30

FONTE: O autor (2025).

Além da identificação de padrões, a integração Apriori + OR-Tools + vetorização pode ser aplicada em diversas áreas. O Quadro 2 apresenta uma síntese de contextos práticos, associando cada técnica ao tipo de problema e ao benefício

estratégico alcançado. Esse panorama ajuda a compreender o papel de cada ferramenta dentro de um ecossistema de decisão baseado em dados.

QUADRO 2 – EXEMPLO DE APLICAÇÕES PRÁTICAS

Problema	Ferramenta Recomendada	Benefício Principal
Análise de cestas de compras	Apriori	Descoberta de padrões de consumo
Planejamento logístico	OR-Tools	Redução de custos e prazos de entrega
Sequência de testes em CNC	Apriori + OR-Tools	Menor tempo de máquina e maior confiabilidade
Previsão de demanda + rotas	Apriori + Vetorização + OR-Tools	Otimização dinâmica e adaptativa

FONTE: O autor (2025).

Por fim, o Quadro 3 apresenta uma análise comparativa das principais ferramentas utilizadas neste estudo, destacando seus pontos fortes e limitações. Essa síntese auxilia na seleção adequada de técnicas conforme o tipo de problema, volume de dados e nível de complexidade da modelagem.

QUADRO 3 – VANTAGENS E DESVANTAGENS DAS FERRAMENTAS

Ferramenta	Vantagens	Limitações
Apriori	Simplicidade, interpretabilidade, útil para padrões frequentes	Baixa escalabilidade em bases muito grandes
OR-Tools	Alta performance, suporte a restrições complexas	Necessita modelagem precisa e conhecimento técnico
Vetorização	Reduz dimensionalidade e acelera cálculos	Pode exigir poder computacional elevado

FONTE: O autor (2025).

1.5 CONCLUSÃO

A integração entre Apriori, OR-Tools e vetorização mostrou-se uma abordagem eficiente para conectar a mineração de padrões à otimização de processos. Ao transformar conhecimento extraído de dados em decisões operacionais, essa combinação amplia o potencial de automação e inteligência nos sistemas produtivos.

Conforme evidenciado nos Quadros 1, 2 e 3, o Apriori permite identificar relações relevantes entre eventos e variáveis, o OR-Tools converte essas relações em modelos otimizados de decisão, e a vetorização viabiliza o tratamento numérico e escalável dos dados. Essa sinergia possibilita ganhos concretos em previsibilidade,

eficiência e redução de custos, especialmente em aplicações de automação de testes eletrônicos e planejamento logístico.

Com base nos resultados e exemplos apresentados, conclui-se que a integração dessas ferramentas constitui uma estratégia sólida e aplicável à Indústria 4.0, promovendo tomadas de decisão mais assertivas, processos adaptativos e melhor aproveitamento dos recursos tecnológicos e humanos nas organizações.

REFERÊNCIAS

AGRAWAL, R.; SRIKANT, R. **Fast algorithms for mining association rules**. In: *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*. Santiago, Chile, 1994.

GOOGLE. **OR-Tools**: Google Optimization Tools. Disponível em: https://developers.google.com/optimization. Acesso em: 28 set. 2025.

GOLDBARG, M. C.; LUNA, H. P. L. **Otimização combinatória e programação inteira**. 3. ed. Rio de Janeiro: Elsevier, 2015.

LOURENÇO, H. R.; MARTINS, I.; RIBEIRO, C. C. **Metaheuristics for combinatorial optimization**. *European Journal of Operational Research*, v. 256, p. 1-25, 2017.

RUSSELL, S.; NORVIG, P. **Inteligência Artificial**: um enfoque moderno. 3. ed. Rio de Janeiro: Elsevier, 2022.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

A – ENUNCIADO

1 ChatGPT

a) **(6,25 pontos)** Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.
b) **(6,25 pontos)** Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.
c) **(6,25 pontos)** Pesquise sobre o funcionamento do ChatGPT (sem perguntar ao próprio ChatGPT) e escreva um texto contendo no máximo 5 parágrafos. Cite as referências.
d) **(6,25 pontos)** Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê.

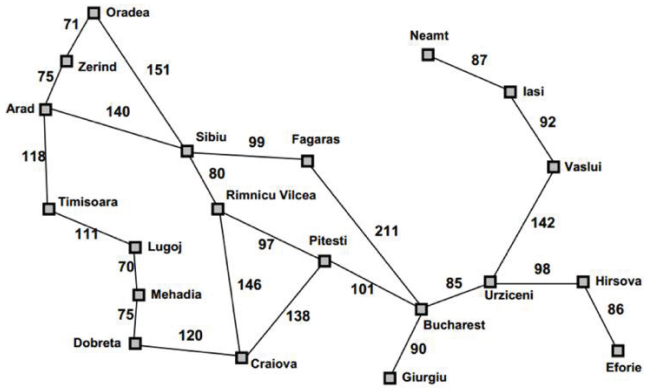
2 Busca Heurística

Realize uma busca utilizando o algoritmo A* para encontrar o melhor caminho para chegar a **Bucharest** partindo de **Lugoj**. Construa a árvore de busca criada pela execução do algoritmo apresentando os valores de *f(n)*, *g(n)* e *h(n)* para cada nó. Utilize a heurística de distância em linha reta, que pode ser observada na tabela abaixo.

Essa tarefa pode ser feita em uma **ferramenta de desenho**, ou até mesmo no **papel**, desde que seja digitalizada (foto) e convertida para PDF.

a) **(25 pontos)** Apresente a árvore final, contendo os valores, da mesma forma que foi apresentado na disciplina e nas práticas. Use o formato de árvore, não será permitido um formato em blocos, planilha, ou qualquer outra representação.

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



Arad	366	**Mehadia**	241
Bucareste	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

Figura 3.22 Valores de *hDLR* — distâncias em linha reta para Bucareste.

3 Lógica

Verificar se o argumento lógico é válido.

Se as uvas caem, então a raposa as come
Se a raposa as come, então estão maduras
As uvas estão verdes ou caem

Logo

A raposa come as uvas se e somente se as uvas caem

Deve ser apresentada uma prova, no mesmo formato mostrado nos conteúdos de aula e nas práticas.

Dicas:

1. Transformar as afirmações para lógica:

p: as uvas caem
q: a raposa come as uvas
r: as uvas estão maduras

2. Transformar as três primeiras sentenças para formar a base de conhecimento

R1: $p \rightarrow q$
R2: $q \rightarrow r$
R3: $\neg r \vee p$

3. Aplicar equivalências e regras de inferência para se obter o resultado esperado. Isto é, com essas três primeiras sentenças devemos derivar $q \leftrightarrow p$. Cuidado com a ordem em que as fórmulas são geradas.

Equivalência Implicação: $(\alpha \rightarrow \beta)$ equivale a $(\neg\alpha \vee \beta)$

Silogismo Hipotético: $\alpha \rightarrow \beta, \beta \rightarrow \gamma \vdash \alpha \rightarrow \gamma$

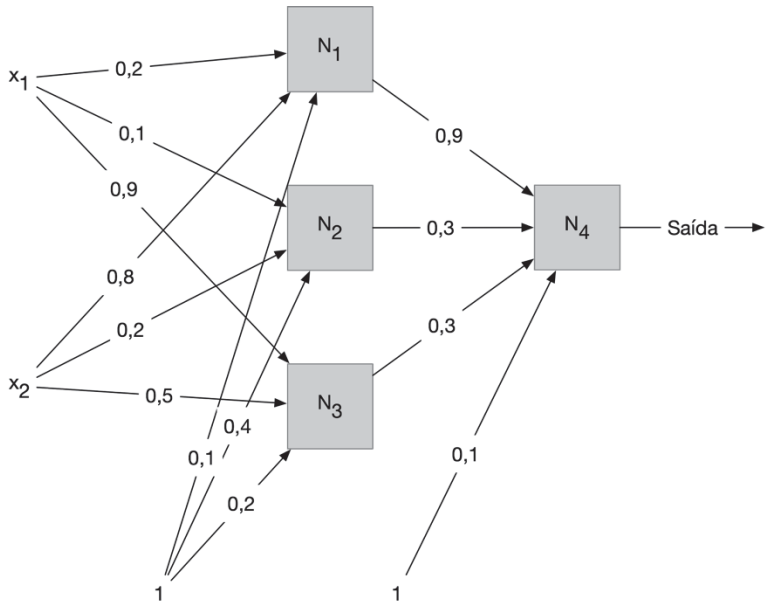
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

Equivalência Bicondicional: $(\alpha \leftrightarrow \beta)$ equivale a $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

a) **(25 pontos)** Deve-se mostrar todos os passos e regras aplicadas, **no mesmo formato apresentado nas aulas e nas práticas**. As equivalências e regras necessárias estão descritas acima e no material.

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



Os neurônios N_1, N_2 e N_3 possuem função de ativação linear. Já N_4 possui função de ativação tangente hiperbólica (pesquise a fórmula e aplique).

Dada a entrada x_1=-3, x_2=1, dê:

a) **(6,25 pontos)** Valor de saída do neurônio N1
b) **(6,25 pontos)** Valor de saída do neurônio N2
c) **(6,25 pontos)** Valor de saída do neurônio N3
d) **(6,25 pontos)** Valor de saída da rede como um todo

B – RESOLUÇÃO

1.a – “Inteligência Artificial (IA) é um campo da ciência da computação que busca desenvolver sistemas e tecnologias capazes de realizar tarefas que normalmente exigem inteligência humana, como reconhecimento de fala, identificação de padrões, tomada de decisão, aprendizado, entre outras. A IA pode ser aplicada em diversos setores, desde a indústria, com o uso de robôs e automação de processos, até a saúde, com o desenvolvimento de sistemas de diagnóstico e tratamento de doenças. Alguns exemplos de tecnologias de IA incluem assistentes virtuais, carros autônomos, reconhecimento de imagens, sistemas de recomendação e chatbots. Existem diversas técnicas e abordagens para o desenvolvimento de sistemas de IA incluindo redes neurais, algoritmos genéticos, lógica fuzzy, entre outras. O objetivo final é criar sistemas capazes de aprender e se adaptar de forma autônoma, melhorando continuamente sua performance e capacidade de resolver problemas.”

1.b – A resposta obtida anteriormente pode ser classificada conforme a abordagem “Agir Racionalmente”, pois o ChatGPT agiu como uma ferramenta que, a partir de uma determinada situação apresentada (a pergunta), buscou o melhor resultado possível, por meio da percepção desse ambiente e da operação autônoma.

1.c – O ChatGPT consiste em um software do tipo robô de conversa (Ventura, 2023), o qual, a partir de uma pergunta realizada pelo usuário, busca nas bases de dados da Internet uma resposta a qual seja apropriada para o caso em questão.

Difere de um chatbot convencional pelo fato de ser dotado de inteligência artificial que possibilita construir um diálogo com o usuário. Ainda, o ChatGPT usa, para o desenvolvimento de seus textos, técnicas de inteligência artificial e aprendizado de máquina, como a técnica de ajuste fino, na qual consiste em “prever quais serão as palavras seguintes com base nas palavras já geradas.” (Idem, 2023). Dentre as possibilidades de texto possíveis de serem produzidas pela ferramenta, encontram-se “discursos de casamento, e-mails corporativos, textos em formato jornalístico, listas de organização e códigos de computação” (ROMANI, 2023). O ChatGPT foi desenvolvido por uma startup norte-americana, a OpenAI, e disponibilizado gratuitamente por essa startup. Pesquisas mostram que o ChatGPT já alcançou a marca de 100 milhões de usuários ativos somente no mês de janeiro (Idem, 2023) Apesar de outras empresas terem lançado soluções semelhantes (como o Google, que lançou o LaMDA), o ChatGPT se mantém, atualmente, como uma solução popular. O algoritmo do ChatGPT é baseado em redes neurais e aprendizado de máquina. A rede neural que o ChatGPT utiliza é a Transformer, que foi desenvolvida especialmente para a construção de textos. O algoritmo se atenta a palavras-chave descritas pelo usuário em sua pergunta, e, com base nessas palavras, busca o contexto e os diferentes significados que essas palavras possuem, gerando um texto que se aproxima do que o usuário deseja (Landim, 2023).

Porém, o ChatGPT apresenta uma série de limitações para o seu uso, sendo as seguintes: falta de senso comum, falta de inteligência emocional, limitações na compreensão do contexto, problemas para gerar conteúdo estruturado de longa duração, limitações para o manuseio de tarefas ao mesmo tempo,

respostas potencialmente tendenciosas, conhecimento limitado e problemas de precisão e questões gramaticais. Recomenda-se, ao se obter uma resposta construída pela ferramenta, verificar a sua veracidade em outras fontes de informação. (Marr, 2023).

Referências:

LANDIM, W. ChatGPT: o que é, como funciona e como usar. Mundo Conectado. Disponível em: https://mundoconectado.com.br/artigos/v/31327/chat-gpt-o-que-ecomo-funciona-como-usar Data de acesso: 17 de março de 2023.

MARR, B. Dez limitações do ChatGPT que comprometem o avanço da IA. Forbes Tech. Disponível em: https://forbes.com.br/forbes-tech/2023/03/dez-limitacoes-dochatgpt-que-comprometem-o-avanco-da-ia/ Data de acesso: 17 de março de 2023.

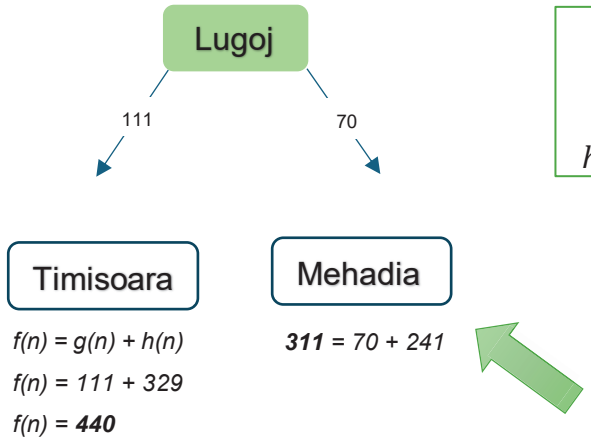
ROMANI, B. ChatGPT muda inteligência artificial para sempre e afeta empregos e economia. Estadão. Disponível em: https://www.estadao.com.br/link/culturadigital/chatgpt-muda-inteligencia-artificial-para-sempre-e-afeta-empregos-eeconomia/ Data de acesso: 17 de março de 2023.

VENTURA, L. O que é ChatGPT e como acessar a inteligência artificial em português. Olhar Digital. Disponível em: https://olhardigital.com.br/2023/01/13/dicas-e-tutoriais/o-que-e-chatgpt-e-comoacessar-a-inteligencia-artificial-em-portugues/ Data de acesso: 17 de março de 2023.

1.d – O ChatGPT, por si só, enquadra-se na categoria “Pensar Racionalmente”, pois, com o uso de informações fornecidas, modela o pensamento mais correto para 11 proporcionar, ao usuário, um resultado logicamente correto. O ChatGPT ainda está longe de apresentar um comportamento similar ao do ser humano, pois para isso deverá simular aspectos da mente humana, como a atividade cerebral.

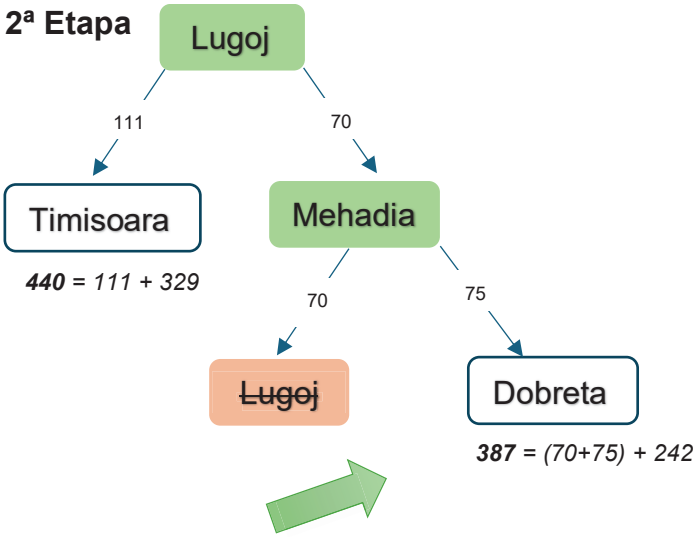
2.a – Representação da árvore Busca Heurística A* de Lugoj -> Bucharest

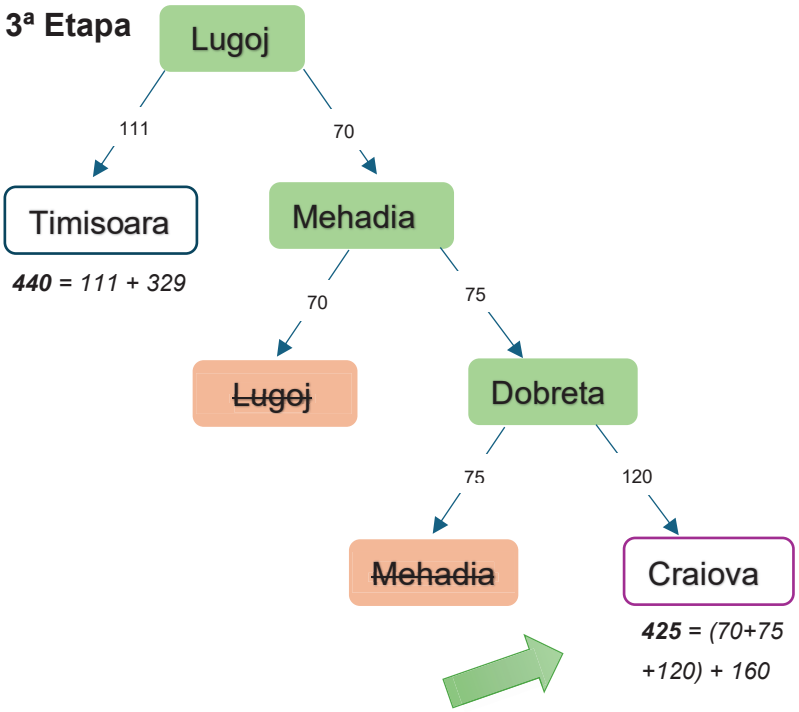
1ª Etapa



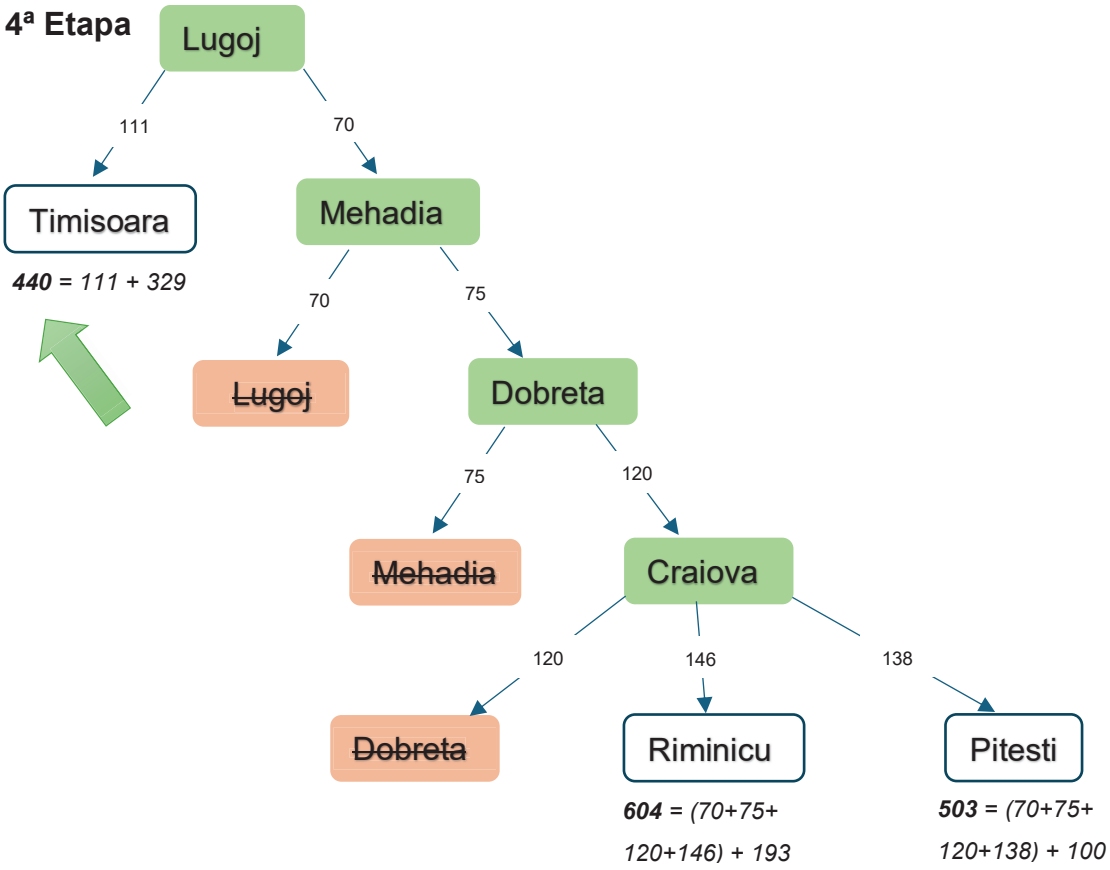
$$f(n) = g(n) + h(n)$$

$f(n) \rightarrow Função\ Custo$
$g(n) \rightarrow Distância\ Real$
$h(n) \rightarrow Distância\ Eurística$


2ª Etapa
Lugoj
111
70
Timisoara
Mehadia
440 = 111 + 329
70
75
~~Lugoj~~
Dobreta
387 = (70+75) + 242

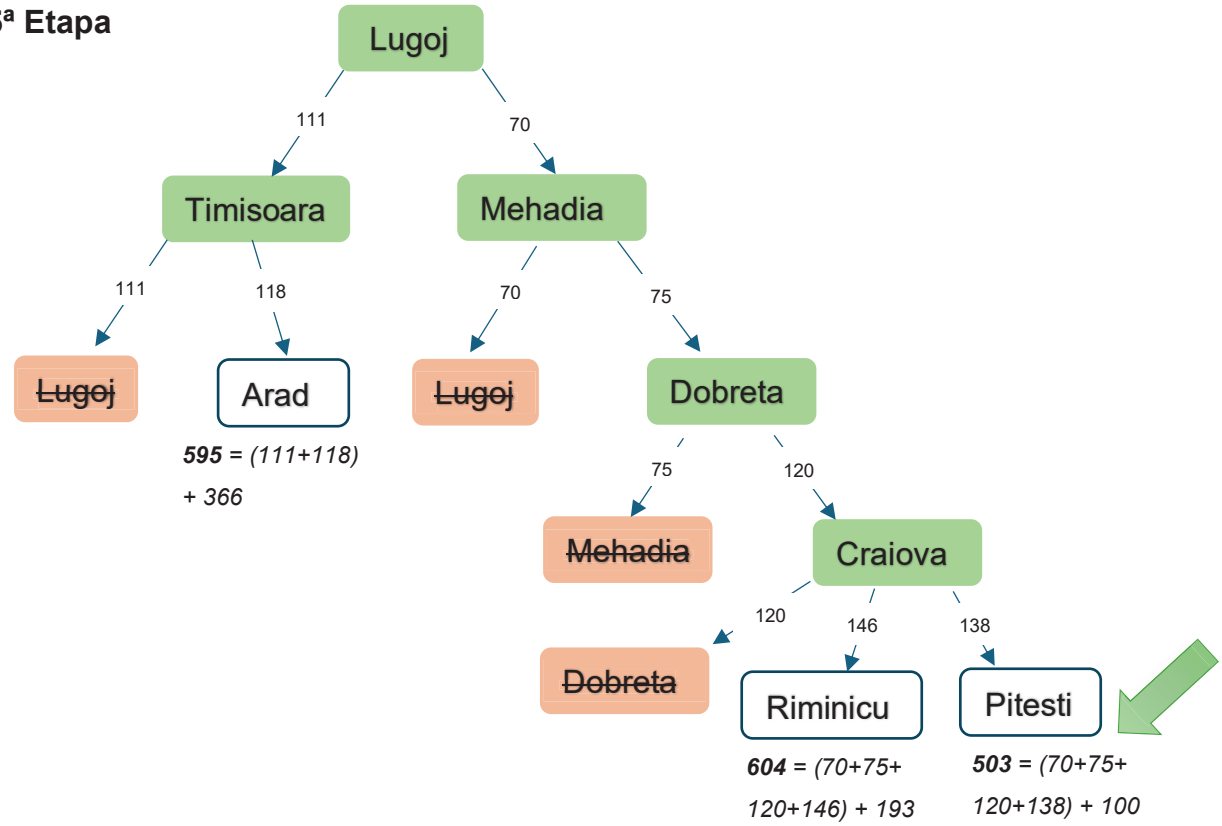

3ª Etapa
Lugoj
111
70
Timisoara
Mehadia
440 = 111 + 329
70
75
~~Lugoj~~
Dobreta
75
120
~~Mehadia~~
Craiova
425 = (70+75
+120) + 160

4ª Etapa



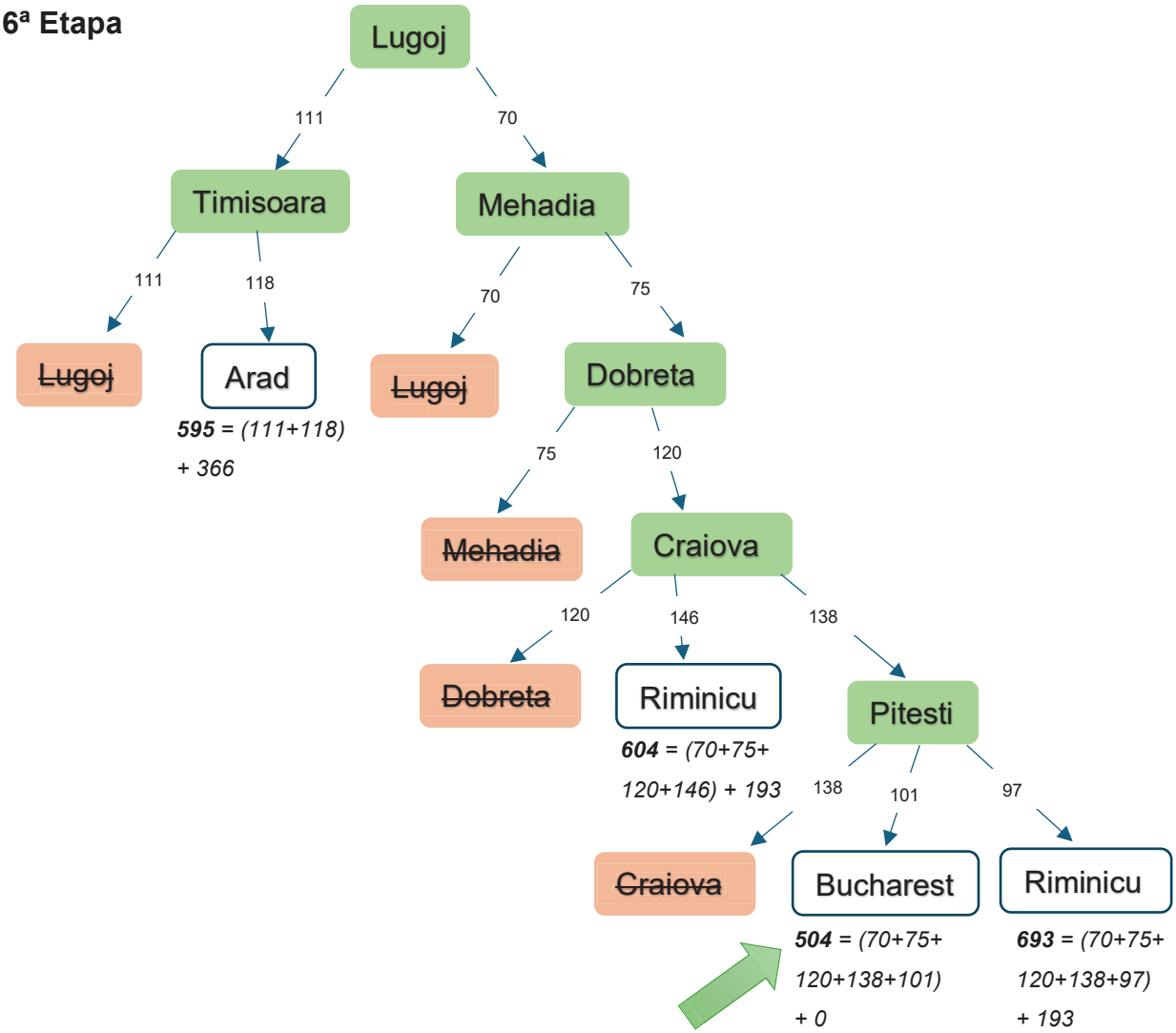
Neste ponto Timisoara passa a ser mais vantajoso com custo de 440 contra 503 de Pitesti, portanto, o algoritmo a escolhe para expandir.

5ª Etapa



Aqui Pitesti retoma o menor custo, portanto, o algoritmo o escolhe para expandir.

6ª Etapa



Neste ponto Bucharest passa a ser mais vantajoso com custo de 504, portanto, o algoritmo o escolhe para expandir e testa se é o destino, e identifica positivamente, retornando-o.

Portanto o caminho encontrado é:

Lugoj => Mehadia => Dobreta => Craiova => Pitesti => Bucharest, somando as distâncias do mapa, g(n), temos 70+75+120+138+101 **= 504**

3.a Prova de Validade — Lógica Proposicional

Premissas:

- p: as uvas caem
- q: a raposa come as uvas
- r: as uvas estão maduras

Base do Conhecimento:

- $R1: p \rightarrow q$ (Se as uvas caem, então a raposa as come)
- $R2: q \rightarrow r$ (Se a raposa as come, então estão maduras)

- $R3$: $\neg r \vee p$ (As uvas estão verdes ou as uvas caem)

Regras Utilizadas:

- Equivalência de Implicação: α → β ≡ ¬α ∨ β
- Silogismo Hipotético: α → β, β → γ ⊢ α → γ
- Conjunção: α, β ⊢ α ∧ β
- Equivalência Bi condicional: α ↔ β ≡ (α → β) ∧ (β → α)

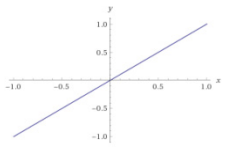
Derivação:

1	p → q	Premissa R1
2	q → r	Premissa R2
3	¬r ∨ p	Premissa R3
4	r → p	3, Equivalência de implicação
5	q → p	2,4 Silogismo hipotético
6	(q → p) ∧ (p → q)	5,1 Conjunção
7	q ↔ p	6, Equivalência de bi condicional

Conclusão

Com base na derivação apresentada, demonstra-se que a partir das premissas R1, R2 e R3, é possível concluir q ↔ p. Portanto, o argumento é válido.

4.

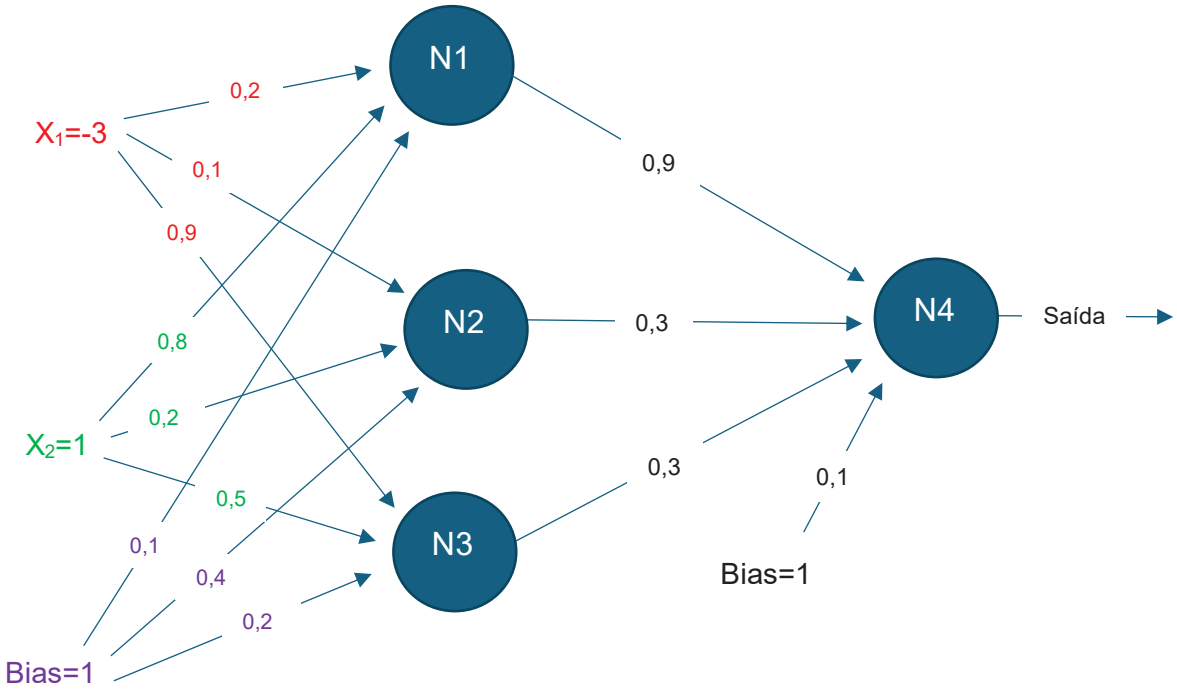


Linear

$$f(x) = x$$

Tanh

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

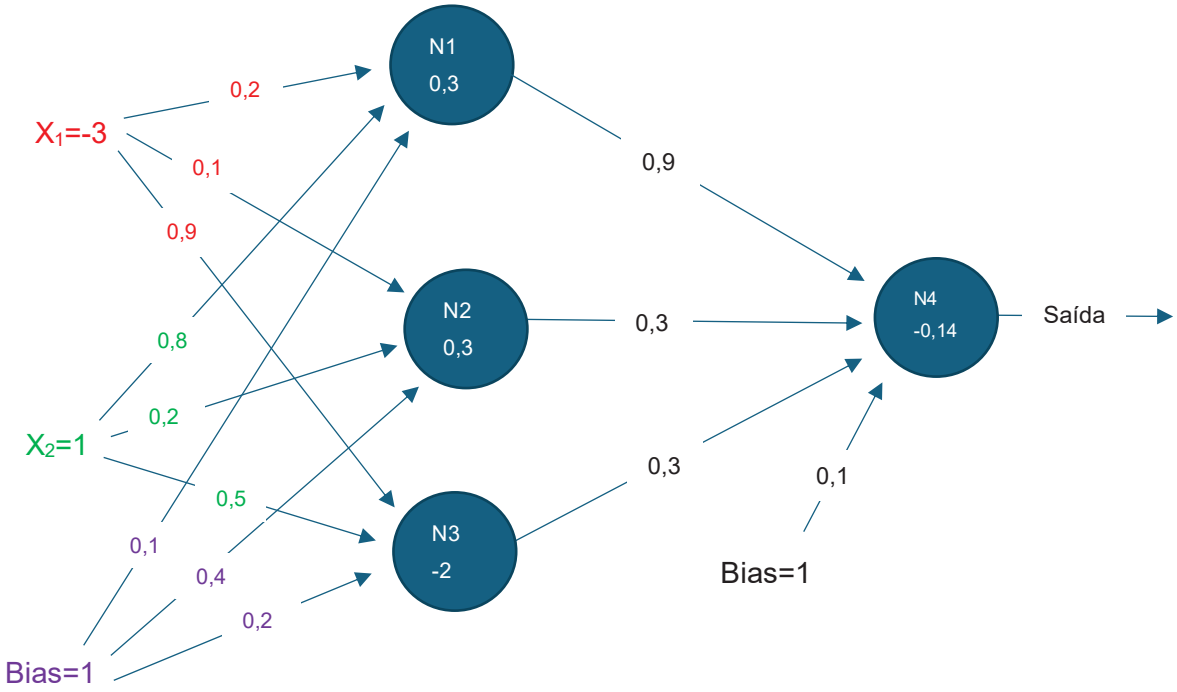


4.a – Calculando N1: (-3*0,2) + (1*0,8) + (1*0,1) = **0,3**

4.b – Calculando N2: (-3*0,1) + (1*0,2) + (1*0,4) = **0,3**

4.c – Calculando N3: (-3*0,9) + (1*0,5) + (1*0,2) = **-2**

Aplicando a Função de ativação linear f(x) = x, portanto os valores calculados em N1, N2 e N3 não tem qualquer variação sendo aplicados diretamente no cálculo de N4.



4.d – Calculando N4: = (0,3 * 0,9) + (0,3 * 0,3) + (-2 * 0,3) + (1 * 0,1) = -0,14

Aplicando a função de ativação da tangente hiperbólica a N4 obtemos a Saída:

Saída = $\frac{(e^{+(-0,14)} - e^{-(-0,14)})}{(e^{+(-0,14)} + e^{-(-0,14)})}$ = **-0,1391**

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

Nome da base de dados do exercício: *precos_carros_brasil.csv*

Informações sobre a base de dados:

Dados dos preços médios dos carros brasileiros, das mais diversas marcas, no ano de 2021, de acordo com dados extraídos da tabela FIPE (Fundação Instituto de Pesquisas Econômicas). A base original foi extraída do site Kaggle (Acesse aqui a base original). A mesma foi adaptada para ser utilizada no presente exercício.

Observação: As variáveis *fuel*, *gear* e *engine_size* foram extraídas dos valores da coluna *model*, pois na base de dados original não há coluna dedicada a esses valores. Como alguns valores do modelo não contêm as informações do tamanho do motor, este conjunto de dados não contém todos os dados originais da tabela FIPE.

Metadados:

Nome do campo	Descrição
year_of_reference	O preço médio corresponde a um mês de ano de referência
month_of_reference	O preço médio corresponde a um mês de referência, ou seja, a FIPE atualiza sua tabela mensalmente
fipe_code	Código único da FIPE
authentication	Código de autenticação único para consulta no site da FIPE
brand	Marca do carro
model	Modelo do carro
fuel	Tipo de combustível do carro
gear	Tipo de engrenagem do carro
engine_size	Tamanho do motor em centímetros cúbicos

year_model	Ano do modelo do carro. Pode não corresponder ao ano de fabricação
avg_price	Preço médio do carro, em reais

Atenção: ao fazer o download da base de dados, selecione o formato .csv. É o formato que será considerado correto na resolução do exercício.

1 Análise Exploratória dos dados

A partir da base de dados **precos_carros_brasil.csv**, execute as seguintes tarefas:

a. Carregue a base de dados media_precos_carros_brasil.csv
b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes
c. Verifique se há dados duplicados nos dados
d. Crie duas categorias, para separar colunas numéricas e categóricas. Imprima o resumo de informações das variáveis numéricas e categóricas (estatística descritiva dos dados)
e. Imprima a contagem de valores por modelo (model) e marca do carro (brand)
f. Dê um breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados

2 Visualização dos dados

A partir da base de dados **precos_carros_brasil.csv,** execute as seguintes tarefas:

a. Gere um gráfico da distribuição da quantidade de carros por marca
b. Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro
c. Gere um gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X)
d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem
e. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d
f. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível
g. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f

3 Aplicação de modelos de machine learning para prever o preço médio dos carros

A partir da base de dados **precos_carros_brasil.csv**, execute as seguintes tarefas:

a. Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo.A variável target é **avg_price. Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
b. Crie partições contendo 75% dos dados para treino e 25% para teste
c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação**: caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo
d. Grave os valores preditos em variáveis criadas
e. Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**

f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2
h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

1.a

```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")

df = pd.read_csv('precos_carros_brasil.csv', low_memory=False)
```

1.b – Existem 65245 linhas no final do arquivo com valores nulos para todas as colunas

```
# Verificando se existem valores ausentes
df.isna().any()
df.isna().sum()
```

```
year_of_reference     65245
month_of_reference    65245
fipe_code             65245
authentication        65245
brand                 65245
model                 65245
fuel                  65245
gear                  65245
engine_size           65245
year_model            65245
avg_price_brl         65245
dtype: int64
```

```
# Substituindo valores ausentes
df['year_of_reference'].fillna('Não informado', inplace=True)
df['month_of_reference'].fillna('Não informado', inplace=True)
df['fipe_code'].fillna('Não informado', inplace=True)
df['authentication'].fillna('Não informado', inplace=True)
df['brand'].fillna('Não informado', inplace=True)
df['model'].fillna('Não informado', inplace=True)
df['fuel'].fillna('Não informado', inplace=True)
df['gear'].fillna('Não informado', inplace=True)
df['engine_size'].fillna('Não informado', inplace=True)
```

```
df['year_model'].fillna('Não informado', inplace=True)
df['avg_price_brl'].fillna('Não informado', inplace=True)
```

1.c –

```
# Verificando se temos valores duplicados
df.duplicated().sum()
```

```
65246
```

```
# Converter e limpar dados conforme necessário
df['engine_size'] = df['engine_size'].apply(lambda x: str(x).replace(',', '.'))
for col in ['year_of_reference', 'year_model', 'avg_price_brl', 'engine_size']:
    df[col] = pd.to_numeric(df[col], errors='coerce')
df_cleaned = df.dropna().drop_duplicates()
```

1.d –

```
# Separar colunas numéricas e categóricas e gerar estatísticas descritivas
numeric_cols = df_cleaned.select_dtypes(include=['number']).columns
categorical_cols = df_cleaned.select_dtypes(include=['object']).columns
numeric_summary = df_cleaned[numeric_cols].describe()
categorical_summary = df_cleaned[categorical_cols].describe()
print(numeric_summary)
print(categorical_summary)
```

```
       year_of_reference    engine_size     year_model  avg_price_brl
count      202295.000000  202295.000000  202295.000000  202295.000000
mean         2021.564695       1.822302    2011.271514   52756.765713
std             0.571904       0.734432       6.376241   51628.912116
min          2021.000000       1.000000    2000.000000    6647.000000
25%          2021.000000       1.400000    2006.000000   22855.000000
50%          2022.000000       1.600000    2012.000000   38027.000000
75%          2022.000000       2.000000    2016.000000   64064.000000
max          2023.000000       6.200000    2023.000000  979358.000000
       month_of_reference fipe_code authentication   brand  \
count              202295    202295         202295  202295
unique                 12      2091         202295       6
top              January  003281-6   cfzlctzfwrcp    Fiat
freq               24260       425              1   44962

                                        model      fuel    gear
count                                  202295    202295  202295
unique                                   2112         3       2
top     Palio Week. Adv/Adv TRYON 1.8 mpi Flex  Gasoline  manual
freq                                      425    168684  161883
```

1.e –

```
## Contagem de valores por modelo e marca
model_count = df_cleaned['model'].value_counts().head(10)   # Top 10 modelos
brand_count = df_cleaned['brand'].value_counts().head(10)   # Top 10 marcas


print(model_count)
```

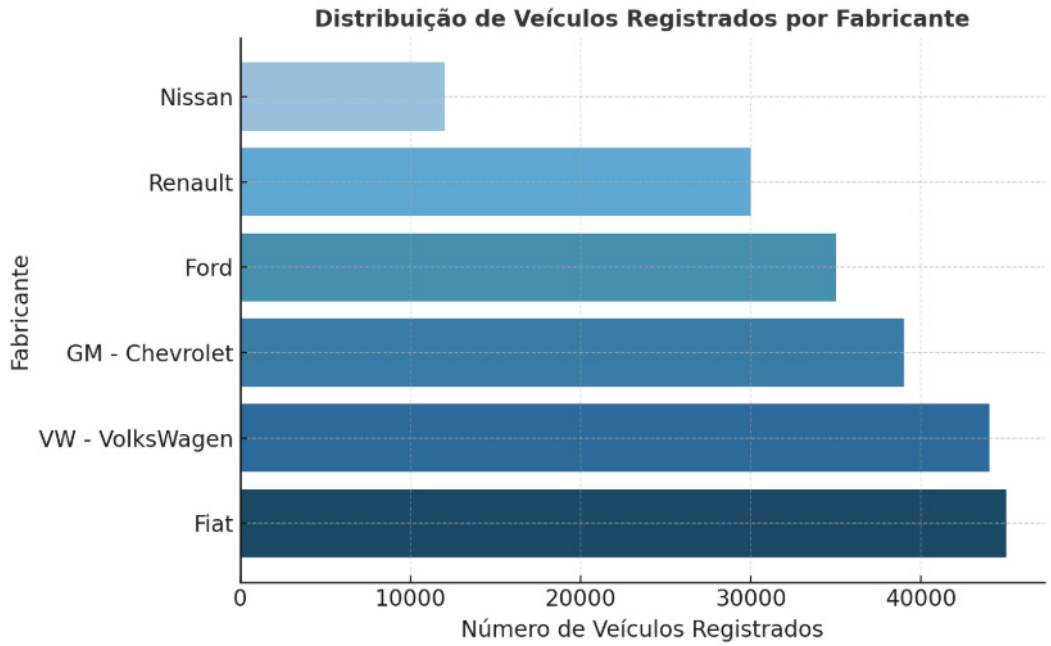
```
print("###############")
print(brand_count)
```

```
model
Palio Week. Adv/Adv TRYON 1.8 mpi Flex    425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p     425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.     400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V      400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray     375
Golf 2.0/ 2.0 Mi Flex Aut/Tiptronic.      375
Doblo Adv/Adv TRYON/LOCKER 1.8 Flex       375
Kombi Escolar 1.6 MPi                     350
Courier 1.6 L/ 1.6 Flex                   350
Courier XL/XL-RS 1.6/ XL 1.6 Flex         350
Name: count, dtype: int64
##############
brand
Fiat              44962
VW - VolksWagen   44312
GM - Chevrolet    38590
Ford              33150
Renault           29191
Nissan            12090
Name: count, dtype: int64
```

1.f – Entre os modelos, destacam-se o “Palio Week. Adv/Adv TRYON 1.8 mpi Flex” e o “Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p”, ambos contabilizando 425 ocorrências cada. No que diz respeito às marcas, os maiores volumes pertencem à Fiat, VW – VolksWagen, GM – Chevrolet e Ford, sendo a Fiat a que apresenta o maior número de registros.

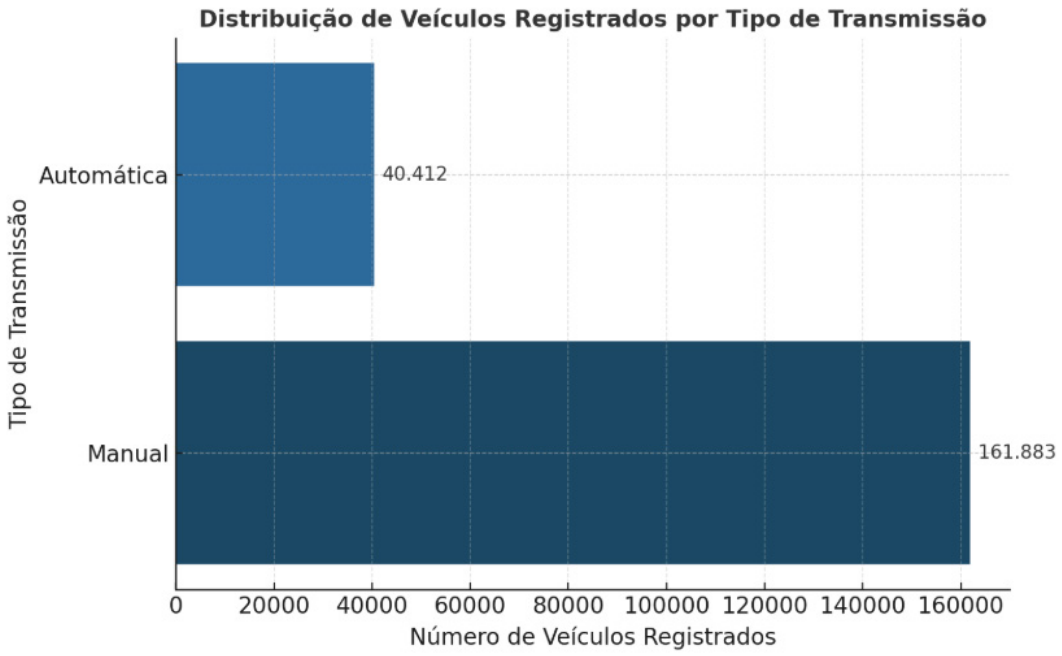
2.a –

```
# Distribuição da quantidade de carros por marca
plt.figure(figsize=(10, 6))
brand_counts = df_cleaned['brand'].value_counts()
sns.barplot(x=brand_counts.values,
            y=brand_counts.index,
            palette="viridis")
plt.title('Distribuição de Veículos Registrados por Fabricante)
plt.xlabel('Número de Veículos Registrados')
plt.ylabel('Fabricante')
plt.show()
```



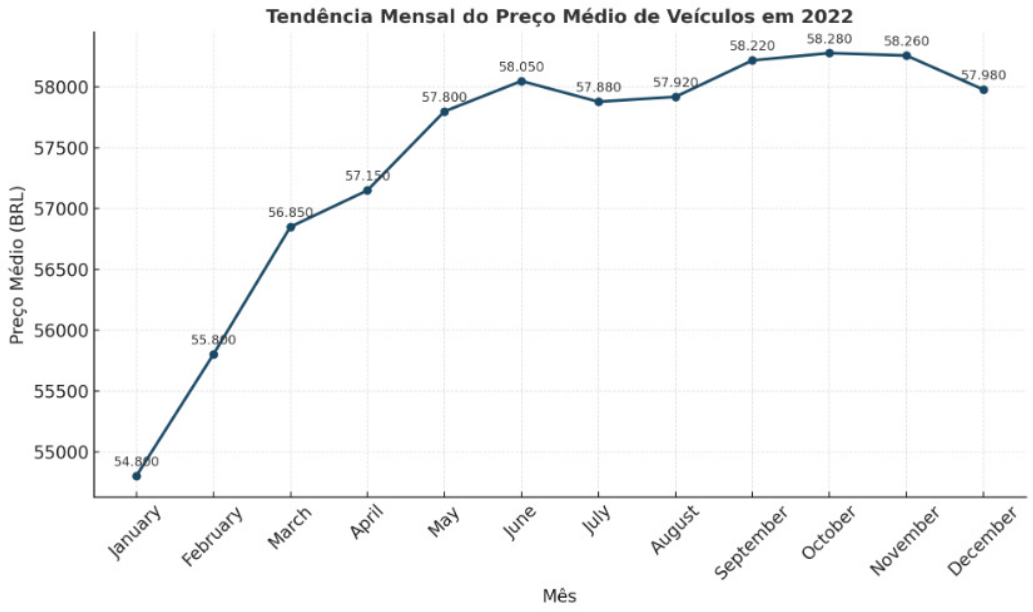
2.b –

```
# Distribuição da Quantidade de Carros por Tipo de Engrenagem do Carro
plt.figure(figsize=(8, 4))
gear_counts = df_cleaned['gear'].value_counts()
sns.barplot(x=gear_counts.index, y=gear_counts.values, palette="muted")
plt.title('Distribuição da Veículos Registrados por Tipo de Transmissão')
plt.xlabel('Tipo de Transmissão')
plt.ylabel('Número de Veículos Registrados')
plt.show()
```



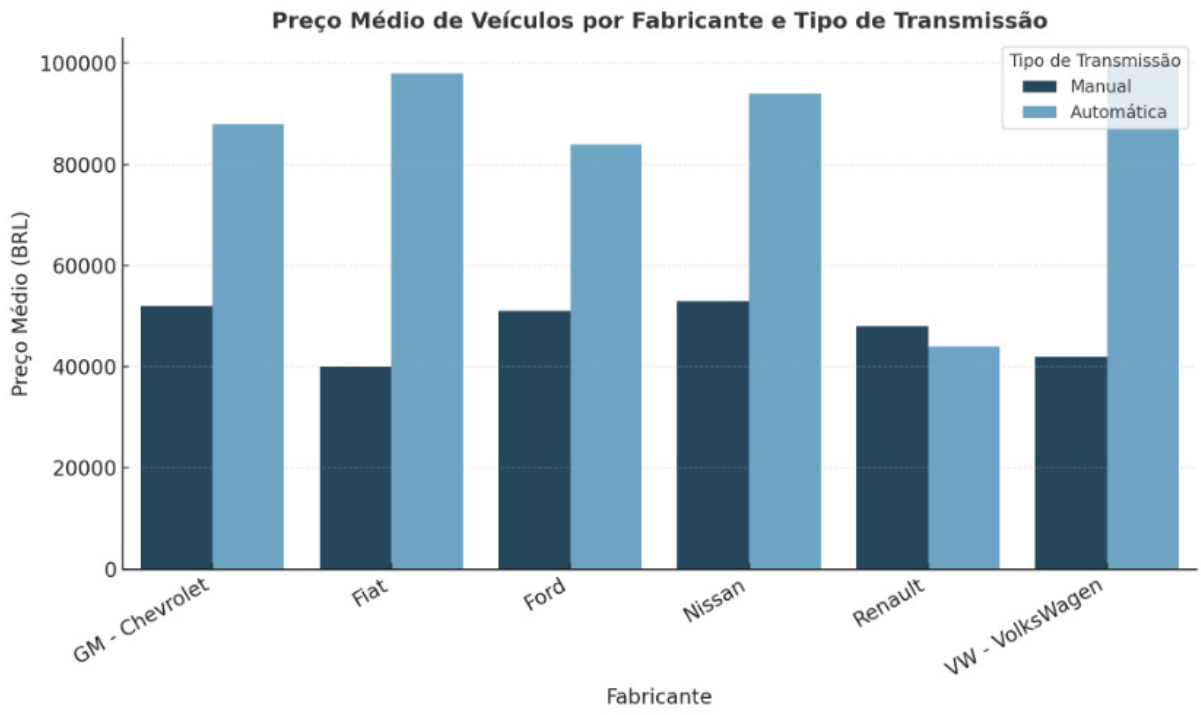
2.c –

```
# Filtrando dados para 2022
df_2022 = df_cleaned[df_cleaned['year_of_reference'] == 2022]

# Convertendo 'month_of_reference' para tipo categórico com ordenação dos meses
months_order = ["January", "February", "March", "April", "May", "June",
                "July", "August", "September", "October", "November", "December"]
df_2022['month_of_reference'] = pd.Categorical(df_2022['month_of_reference'],
                                               categories=months_order,
                                               ordered=True)

# Agrupando por mês e calculando a média do preço
monthly_avg_price =
df_2022.groupby('month_of_reference')['avg_price_brl'].mean().reset_index()

# Gráfico da evolução da média de preço ao longo dos meses de 2022
plt.figure(figsize=(12, 6))
sns.lineplot(x='month_of_reference',
             data=monthly_avg_price,
             y='avg_price_brl',
             marker='o')
plt.title('Tendência Mensal do Preço Médio de Veículos em 2022')
plt.xlabel('Mês')
plt.ylabel('Média de Preço (BRL)')
plt.xticks(rotation=45)
plt.show()
```



2.d –

```
# Distribuição da Média de Preço dos Carros por Marca e Tipo de Engrenagem
plt.figure(figsize=(12, 6))
sns.barplot(x='brand', y='avg_price_brl', hue='gear', data=df_cleaned,
            palette="coolwarm")
plt.title('Preço Médio de Veículos por Fabricante e Tipo de Transmissão')
plt.xlabel('Fabricante')
plt.ylabel('Preço Médio (BRL)')
plt.xticks(rotation=45)
plt.legend(title='Tipo de Engrenagem')
plt.show()
```

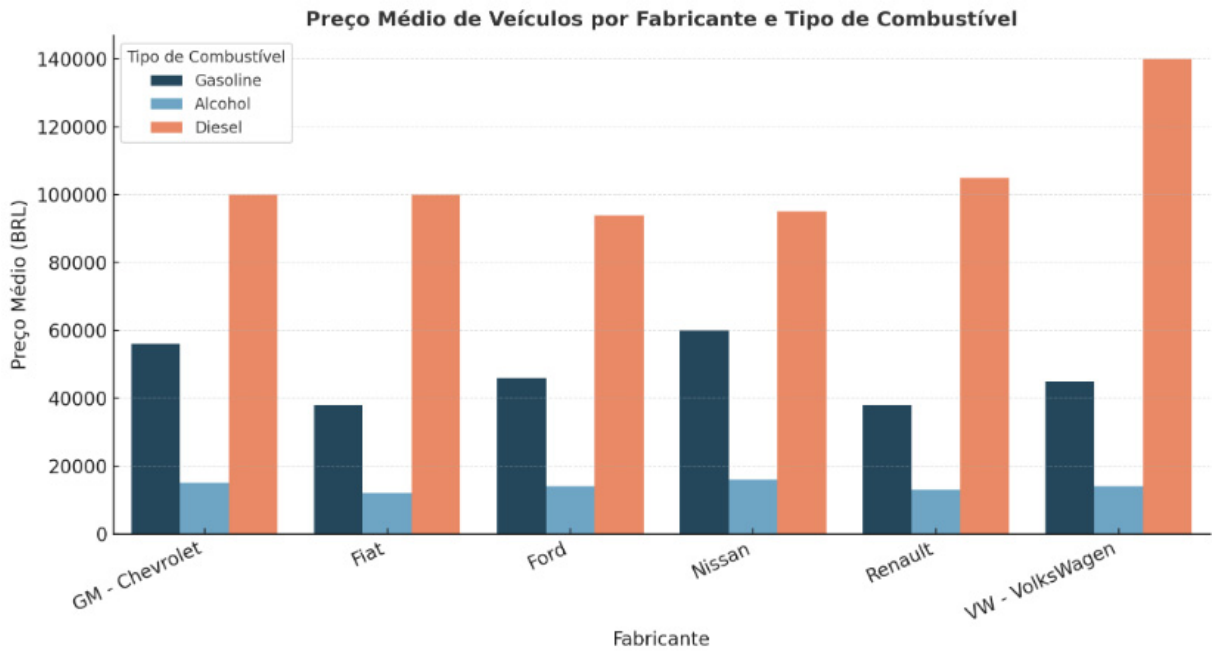


2.e – Veículos equipados com transmissão automática apresentam, em todas as marcas analisadas, valores médios significativamente superiores aos dos modelos com câmbio manual. Essa diferença se mantém de forma consistente entre os fabricantes, indicando que o tipo de transmissão é um elemento determinante no acréscimo do preço, independentemente da marca do veículo.

2.f –

```
# Distribuição da Média de Preço dos Carros por Marca e Tipo de Combustível
plt.figure(figsize=(12, 6))
sns.barplot(x='brand', y='avg_price_brl', hue='fuel', data=df_cleaned,
            palette="coolwarm")
```

```
plt.title('Preço Médio de Veículos por Fabricante e Tipo de Combustível')
plt.xlabel('Fabricante')
plt.ylabel('Preço Médio (BRL)')
plt.xticks(rotation=45)
plt.legend(title='Tipo de Combustível')
plt.show()
```



2.g – Independentemente do combustível utilizado, VW – Volkswagen e GM – Chevrolet se destacam por apresentar as maiores médias de preço entre as marcas analisadas. Em contrapartida, Ford, Fiat e Nissan mantêm valores médios mais acessíveis nos três tipos de combustível, o que pode refletir uma estratégia de posicionamento mais competitivo ou a oferta de modelos voltados a segmentos diferentes dentro do mercado automotivo.

3.a –

```
# Convertendo a variável 'engine_size' para numérica
dados['engine_size'] = dados['engine_size'].str.replace(',', '.').astype(float)

# Codificando variáveis categóricas e convertendo para tipo 'int'
categorical_vars = ['month_of_reference', 'fuel', 'gear']
for col in categorical_vars:
    dados[col] = LabelEncoder().fit_transform(dados[col].astype(str)).astype(int)

# Selecionando as variáveis independentes e a variável target
X = dados[['year_of_reference', 'month_of_reference', 'fuel', 'gear',
           'engine_size', 'year_model']]
```

```
y = dados['avg_price_brl']

# Removendo linhas com valores nulos
X = X.dropna()
y = y.loc[X.index]
```

3.b –

```
# Dividindo os dados em conjuntos de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25,
                                                    random_state=42)
```

3.c –

```
# Inicializando os modelos
rf = RandomForestRegressor(random_state=42)
xgb = XGBRegressor(random_state=42, n_estimators=100, max_depth=3)

# Treinando os modelos
rf.fit(X_train, y_train)
xgb.fit(X_train, y_train)
```

3.d –

```
# Fazendo previsões
y_pred_rf = rf.predict(X_test)
y_pred_xgb = xgb.predict(X_test)
```

3.e –

```
# Avaliando os modelos
metrics_rf = {
    "MSE": mean_squared_error(y_test, y_pred_rf),
    "MAE": mean_absolute_error(y_test, y_pred_rf),
    "R^2": r2_score(y_test, y_pred_rf)
}

metrics_xgb = {
    "MSE": mean_squared_error(y_test, y_pred_xgb),
    "MAE": mean_absolute_error(y_test, y_pred_xgb),
    "R^2": r2_score(y_test, y_pred_xgb)
}

print("RandomForest Metrics:", metrics_rf)
print("XGBoost Metrics:", metrics_xgb)
```

```
RandomForest Metrics: {'MSE': 214511591.53775933, 'MAE': 7272.201350817884,
'R^2': 0.9191415741559127}
```

```
XGBoost Metrics: {'MSE': 185360830.5689201, 'MAE': 7621.348730526677,
'R^2': 0.9301297199582}
```

```
# Analisando a importância das variáveis
importances_rf = rf.feature_importances_
importances_xgb = xgb.feature_importances_

print("Importância das variáveis no RandomForest:", importances_rf)
print("Importância das variáveis no XGBoost:", importances_xgb)
```

```
Importância das variáveis no RandomForest: [0.01433986 0.01371423 0.0349384
0.04101832 0.48478858 0.4112006 ]
Importância das variáveis no XGBoost: [0.03193877 0.00563701 0.13987061
0.16708949 0.30171058 0.3537536 ]
```

3.f – A análise de importância das variáveis mostrou que, em ambos os modelos — RandomForest e XGBoost —, o tamanho do motor (engine_size) e o ano do modelo (year_model) são os fatores mais determinantes para prever o preço médio dos veículos.
Esses resultados indicam que atributos intrínsecos, como a capacidade do motor e o grau de modernidade, exercem papel decisivo na formação do valor de mercado.
Enquanto o RandomForest atribui maior peso ao engine_size, o XGBoost apresenta uma distribuição mais equilibrada, valorizando também variáveis como gear e fuel, o que evidencia uma avaliação mais abrangente dos elementos que impactam o preço.

3.g – Com base nas métricas de desempenho avaliadas, o XGBoost se destaca como o modelo mais adequado para prever o preço médio dos veículos.
Embora apresente um MAE ligeiramente superior ao do RandomForest, o XGBoost obteve MSE menor e coeficiente de determinação (R^2) mais elevado.
Especificamente, o R^2 do XGBoost foi de 0,9301, indicando que aproximadamente 93% da variabilidade nos preços é explicada pelo modelo — resultado superior ao R^2 de 0,9191 alcançado pelo RandomForest.
Além disso, o menor MSE evidencia que as previsões do XGBoost tendem a se aproximar mais dos valores reais, consolidando-o como a alternativa mais robusta para este problema de regressão.

3.h – O XGBoost apresentou o melhor desempenho na tarefa de previsão do preço médio dos veículos.
Essa conclusão apoia-se, sobretudo, no coeficiente de determinação (R^2) e no Erro Quadrático Médio (MSE).
O modelo alcançou um R^2 de 0,9301, evidenciando elevada capacidade de explicar a variabilidade dos preços, além de um MSE inferior, o que indica maior precisão nas estimativas em comparação ao RandomForest.
Esses resultados confirmam a eficácia do XGBoost em capturar relações complexas nos dados e em gerar previsões mais consistentes e acuradas sobre o valor dos automóveis.

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

O banco de dados consiste nos valores multiespectrais de pixels em vizinhanças 3x3 em uma imagem de satélite, e na classificação associada ao pixel central em cada vizinhança. O objetivo é prever esta classificação, dados os valores multiespectrais.

Um quadro de imagens do Satélite Landsat com MSS (*Multispectral Scanner System*) consiste em quatro imagens digitais da mesma cena em diferentes bandas espectrais. Duas delas estão na região visível (correspondendo aproximadamente às regiões verde e vermelha do espectro visível) e duas no infravermelho (próximo). Cada pixel é uma palavra binária de 8 bits, com 0 correspondendo a preto e 255 a branco. A resolução espacial de um pixel é de cerca de 80m x 80m. Cada imagem contém 2340 x 3380 desses pixels. O banco de dados é uma subárea (minúscula) de uma cena, consistindo de 82 x 100 pixels. Cada linha de dados corresponde a uma vizinhança quadrada de pixels 3x3 completamente contida dentro da subárea 82x100. Cada linha contém os valores de pixel nas quatro bandas espectrais (convertidas em ASCII) de cada um dos 9 pixels na vizinhança de 3x3 e um número indicando o rótulo de classificação do pixel central.

As classes são: solo vermelho, colheita de algodão, solo cinza, solo cinza úmido, restolho de vegetação, solo cinza muito úmido.

Os dados estão em ordem aleatória e certas linhas de dados foram removidas, portanto você não pode reconstruir a imagem original desse conjunto de dados. Em cada linha de dados, os quatro valores espectrais para o pixel superior esquerdo são dados primeiro, seguidos pelos quatro valores espectrais para o pixel superior central e, em seguida, para o pixel superior direito, e assim por diante, com os pixels lidos em sequência, da esquerda para a direita e de cima para baixo. Assim, os quatro valores espectrais para o pixel central são dados pelos atributos 17, 18, 19 e 20. Se você quiser, pode usar apenas esses quatro atributos, ignorando os outros. Isso evita o problema que surge quando uma vizinhança 3x3 atravessa um limite.

O banco de dados se encontra no pacote **mlbench** e é completo (não possui dados faltantes).

Tarefas:

1. Carregue a base de dados Satellite
2. Crie partições contendo 80% para treino e 20% para teste
3. Treine modelos RandomForest, SVM e RNA para predição destes dados.
4. Escolha o melhor modelo com base em suas matrizes de confusão.
5. Indique qual modelo dá o melhor o resultado e a métrica utilizada

2 Estimativa de Volumes de Árvores

Modelos de aprendizado de máquina são bastante usados na área da engenharia florestal (mensuração florestal) para, por exemplo, estimar o volume de madeira de árvores sem ser necessário abatê-las.

O processo é feito pela coleta de dados (dados observados) através do abate de algumas árvores, onde sua altura, diâmetro na altura do peito (dap), etc, são medidos de forma exata. Com estes dados, treina-se um modelo de AM que pode estimar o volume de outras árvores da população.

Os modelos, chamados alométricos, são usados na área há muitos anos e são baseados em regressão (linear ou não) para encontrar uma equação que descreve os dados. Por exemplo, o modelo de Spurr é dado por:

$$\textbf{Volume} = b0 + b1 * dap^2 * Ht$$

Onde dap é o diâmetro na altura do peito (1,3metros), Ht é a altura total. Tem-se vários modelos alométricos, cada um com uma determinada característica, parâmetros, etc. Um modelo de regressão envolve aplicar os dados observados e encontrar b0 e b1 no modelo apresentado, gerando assim uma equação que pode ser usada para prever o volume de outras árvores.

Dado o arquivo **Volumes.csv**, que contém os dados de observação, escolha um modelo de aprendizado de máquina com a melhor estimativa, a partir da estatística de correlação.

Tarefas

1. Carregar o arquivo Volumes.csv (http://www.razer.net.br/datasets/Volumes.csv)
2. Eliminar a coluna NR, que só apresenta um número sequencial
3. Criar partição de dados: treinamento 80%, teste 20%
4. Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR

    - O modelo alométrico é dado por: Volume = b0 + b1 * dap^2 * Ht

      alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, dados, start=list(b0=0.5, b1=0.5))

5. Efetue as predições nos dados de teste
6. Crie suas próprias funções (UDF) e calcule as seguintes métricas entre a predição e os dados observados

    - Coeficiente de determinação: R^2

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \widehat{y_i})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

onde y_i é o valor observado, $\hat{y}_i$ é o valor predito e $\bar{y}$ é a média dos valores y_i observados. Quanto mais perto de 1 melhor é o modelo;

- Erro padrão da estimativa: S_{yx}

$$S_{yx} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n-2}}$$

esta métrica indica erro, portanto quanto mais perto de 0 melhor é o modelo;

- Syx%

$$S_{yx}\% = \frac{S_{yx}}{\bar{y}} * 100$$

esta métrica indica porcentagem de erro, portanto quanto mais perto de 0 melhor é o modelo;

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1.1 –

```
set.seed(77)
data(Satellite)
df <- Satellite[,c(17,18,19,20,37)]
```

1.2 –

```
indices <- createDataPartition(df$classes, p=0.80, list=FALSE)
treino <- df[indices,]
teste <- df[-indices,]
```

1.3 –

```
rf <- train(classes~., data=treino, method="rf")
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, teste$classes)
```

```
# Confusion Matrix and Statistics
#
# Reference
# Prediction        red soil cotton crop grey soil damp grey soil
# red soil             299           0          1              1
# cotton crop            0         126          0              0
```

```
# grey soil              4          0        250            33
# damp grey soil         0          1         12            53
# vegetation stubble     3         12          0             0
# very damp grey soil    0          1          8            38
#
# Reference
# Prediction        vegetation stubble very damp grey soil
# red soil                         8                   0
# cotton crop                      3                   0
# grey soil                        0                  10
# damp grey soil                   0                  33
# vegetation stubble             117                   6
# very damp grey soil             13                 252
#
# Overall Statistics
#
# Accuracy : 0.8544
# 95% CI : (0.8339, 0.8732)
# No Information Rate : 0.2383
# P-Value [Acc > NIR] : < 2.2e-16
```

```
svm <- train(classes~., data=treino, method="svmRadial")
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, teste$classes)
```

```
# Confusion Matrix and Statistics
#
# Reference
# Prediction        red soil cotton crop grey soil damp grey soil
# red soil             298          1          2             1
# cotton crop            0        124          0             0
# grey soil              4          0        260            37
# damp grey soil         0          1          8            57
# vegetation stubble     4         13          0             0
# very damp grey soil    0          1          1            30
#
# Reference
# Prediction        vegetation stubble very damp grey soil
# red soil                        10                   0
# cotton crop                      1                   0
# grey soil                        0                  10
# damp grey soil                   1                  33
```

```
# vegetation stubble             115                   3
# very damp grey soil             14                 256
#
# Overall Statistics
#
# Accuracy : 0.8645
# 95% CI : (0.8445, 0.8827)
# No Information Rate : 0.2383
# P-Value [Acc > NIR] : < 2.2e-16
```

```
rna <- train(classes~., data=treino, method="nnet", trace=FALSE)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, teste$classes)
```

```
# Confusion Matrix and Statistics
#
# Reference
# Prediction         red soil cotton crop grey soil damp grey soil
# red soil              292          2          2              2
# cotton crop             1        124          0              0
# grey soil               4          0        264             59
# damp grey soil          0          0          2             50
# vegetation stubble      9         12          0              5
# very damp grey soil     0          2          5             64
#
# Reference
# Prediction         vegetation stubble very damp grey soil
# red soil                          11                   0
# cotton crop                        4                   0
# grey soil                          0                  29
# damp grey soil                     0                   0
# vegetation stubble                99                   7
# very damp grey soil               27                 265
#
# Overall Statistics
#
# Accuracy : 0.8131
# 95% CI : (0.7907, 0.8341)
# No Information Rate : 0.2383
# P-Value [Acc > NIR] : < 2.2e-16
```

1.4 e 1.5 – Vamos optar pelo melhor modelo baseados na precisão “Accuracy”:

Métrica	RandomForest	SVM	RNA
Accuracy	0.8544	**0.8645**	0.8131

2.1 –

```
df_arvores <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")
```

2.2 –

```
df_arvores$NR <- NULL
```

2.3 –

```
set.seed(77) //Semeando uma semente
indices <- createDataPartition(df_arvores$VOL, p=0.80, list=FALSE)
treino <- df_arvores[indices,]
teste <- df_arvores[-indices,]
```

2.4 e 2.5 –

```
rf <- train(VOL~., data=treino, method="rf")
predicoes.rf <- predict(rf, teste)
```

```
svm <- train(VOL~., data=treino, method="svmRadial")
predicoes.svm <- predict(svm, teste)
```

```
rna <- train(VOL~., data=treino, method="nnet")
predicoes.rna <- predict(rna, teste)
```

```
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT, treino, start=list(b0=0.5, b1=0.5))
predicoes.alom <- predict(alom, teste)
```

2.6 –

```
rmse.rf <- RMSE(predicoes.rf, teste$VOL)
rmse.svm <- RMSE(predicoes.svm, teste$VOL)
rmse.rna <- RMSE(predicoes.rna, teste$VOL)
rmse.alom <- RMSE(predicoes.alom, teste$VOL)

cat("RMSE RF: ", rmse.rf, "\n")
cat("RMSE SVM: ", rmse.svm, "\n")
cat("RMSE RNA: ", rmse.rna, "\n")
cat("RMSE ALOM: ", rmse.alom, "\n")
```

```
> cat("RMSE RF: ", rmse.rf, "\n")
RMSE RF:  0.1671697
```

```
> cat("RMSE SVM: ", rmse.svm, "\n")
RMSE SVM:  0.2269896
> cat("RMSE RNA: ", rmse.rna, "\n")
RMSE RNA:  0.6435001
> cat("RMSE ALOM: ", rmse.alom, "\n")
RMSE ALOM:  0.1289662
```

```
r2 <- function (valor_observado, valor_predito){
  return(1-(sum((valor_observado-valor_predito)^2)/sum((valor_observado-
           mean(valor_observado))^2)))
}

cat("R2 RF: ", r2(teste$VOL, predicoes.rf), "\n")
cat("R2 SVM: ", r2(teste$VOL, predicoes.svm), "\n")
cat("R2 RNA: ", r2(teste$VOL, predicoes.rna), "\n")
cat("R2 ALOM: ", r2(teste$VOL, predicoes.alom), "\n")
```

```
> cat("R2 RF: ", r2(teste$VOL, predicoes.rf), "\n")
R2 RF:  0.8889973
> cat("R2 SVM: ", r2(teste$VOL, predicoes.svm), "\n")
R2 SVM:  0.7953411
> cat("R2 RNA: ", r2(teste$VOL, predicoes.rna), "\n")
R2 RNA:  -0.6448107
> cat("R2 ALOM: ", r2(teste$VOL, predicoes.alom), "\n")
R2 ALOM:  0.9399351
```

```
syx <- function(valor_observado, valor_predito){
  return(sqrt(sum((valor_observado-valor_predito)^2)/(length(valor_observado)-2)))
}
cat("Syx RF: ", syx(teste$VOL, predicoes.rf), "\n")
cat("Syx SVM: ", syx(teste$VOL, predicoes.svm), "\n")
cat("Syx RNA: ", syx(teste$VOL, predicoes.rna), "\n")
cat("Syx ALOM: ", syx(teste$VOL, predicoes.alom), "\n")
```

```
> cat("Syx RF: ", syx(teste$VOL, predicoes.rf), "\n")
Syx RF:  0.1762123
> cat("Syx SVM: ", syx(teste$VOL, predicoes.svm), "\n")
Syx SVM:  0.2392681
> cat("Syx RNA: ", syx(teste$VOL, predicoes.rna), "\n")
Syx RNA:  0.6783087
> cat("Syx ALOM: ", syx(teste$VOL, predicoes.alom), "\n")
Syx ALOM:  0.1359424
```

```
syx_perc <- function(valor_observado, valor_predito){
  return(syx(valor_observado, valor_predito)/mean(valor_observado)*100)
}
cat("Syx_perc RF: ", syx_perc(teste$VOL, predicoes.rf), "\n")
cat("Syx_perc SVM: ", syx_perc(teste$VOL, predicoes.svm), "\n")
cat("Syx_perc RNA: ", syx_perc(teste$VOL, predicoes.rna), "\n")
cat("Syx_perc ALOM: ", syx_perc(teste$VOL, predicoes.alom), "\n")
```

```
> cat("Syx_perc RF: ", syx_perc(teste$VOL, predicoes.rf), "\n")
Syx_perc RF:  12.5605
> cat("Syx_perc SVM: ", syx_perc(teste$VOL, predicoes.svm), "\n")
Syx_perc SVM:  17.05514
> cat("Syx_perc RNA: ", syx_perc(teste$VOL, predicoes.rna), "\n")
Syx_perc RNA:  48.35016
> cat("Syx_perc ALOM: ", syx_perc(teste$VOL, predicoes.alom), "\n")
Syx_perc ALOM:  9.690035
```

2.7 –

Modelo / Métrica	RMSE	R^2	Syx	Syx%
RF	0.1671697	0.8889973	0.1762123	12.5605
SVM	0.2269896	0.7953411	0.2392681	17.05514
RNA	0.6435001	-0.6448107	0.6783087	48.35016
ALOMÉTRICO	0.1289662	0.9399351	0.1359424	9.690035

Analisando o resultado percebemos que o modelo alométrico foi superior em todas as métricas relacionadas e sintetizadas na tabela acima, o RMSE erro quadrático médio foi de ~0.13 enquanto que o segundo melhor o RF foi de quase 0.17, assim também seguiu-se com as demais métricas o R2, que quanto mais próximo de 1 melhor é o resultado, o Syx que quanto menor for, representa um resultado mais apurado e o Syx% que mostra que o modelo alométrico chegou apenas a ~9,7% enquanto novamente o RF, segundo melhor ficou com ~12.5%. Novamente, ressaltando que treinamos todos os algoritmos sem parâmetros e que acaso tivéssemos feito metodicamente, provavelmente, teríamos obtido resultados tão acurados quanto, ou até mesmo melhores, que os do modelo alométrico.

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

(15 pontos) Elaborar os gráficos box-plot e histograma das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados

(15 pontos) Elaborar a tabela de frequencias das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados

2) Medidas de posição e dispersão

(15 pontos) Calcular a média, mediana e moda das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados

(15 pontos) Calcular a variância, desvio padrão e coeficiente de variação das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados

3) Testes paramétricos ou não paramétricos

(40 pontos) Testar se as médias (se você escolher o teste paramétrico) ou as medianas (se você escolher o teste não paramétrico) das variáveis “age” (idade da esposa) e “husage” (idade do marido) são iguais, construir os intervalos de confiança e comparar os resultados.

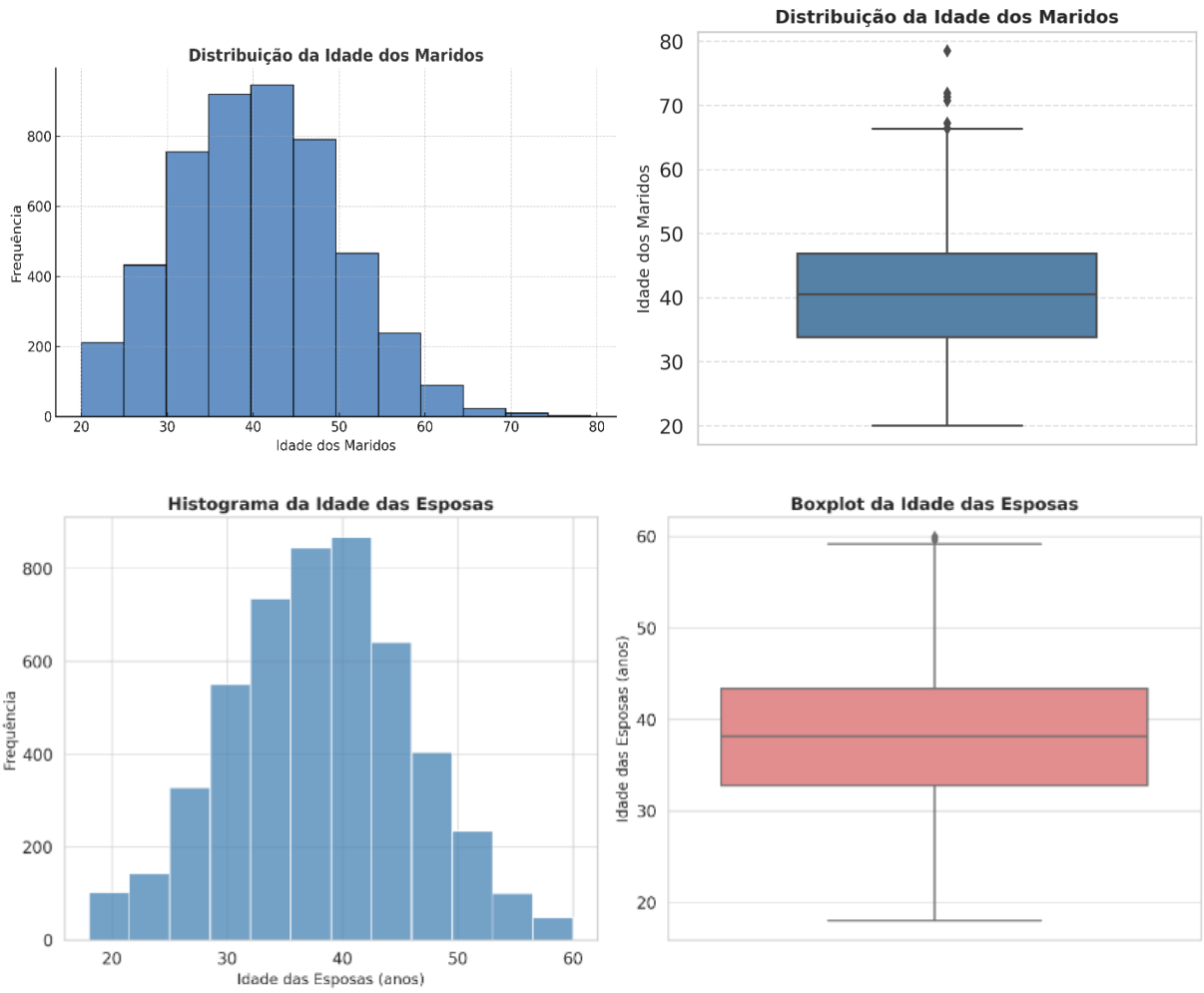
Obs:

Você deve fazer os testes necessários (e mostra-los no documento pdf) para saber se você deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney (não paramétrico), justifique sua resposta sobre a escolha.

Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

1.1 –



A análise do boxplot indica a presença de valores atípicos (outliers) na distribuição da idade dos maridos, especialmente em faixas mais elevadas. Observa-se ainda que a idade máxima dos maridos é consideravelmente superior à observada entre as esposas, sugerindo assimetria entre os grupos etários do casal.

A análise do boxplot da idade das esposas revela uma distribuição consistente, sem a presença de outliers. Os valores encontram-se bem concentrados em torno da mediana, indicando estabilidade no perfil etário desse grupo. A ausência de discrepâncias sugere que a variabilidade observada está dentro de padrões esperados, reforçando a homogeneidade da distribuição.

O histograma evidencia que a maior concentração de maridos se encontra entre **30 e 50 anos**, configurando o intervalo de maior densidade da distribuição. Há, entretanto, uma proporção reduzida de indivíduos com idade inferior a 20 anos e superior a 70 anos, indicando que essas faixas etárias representam casos **menos frequentes** e possivelmente excepcionais dentro da amostra.

O histograma da idade das esposas evidencia que a maior concentração se encontra no intervalo entre 30 e 40 anos, caracterizando esse grupo etário como predominante na amostra. Observa-se ainda que idades inferiores a 20 anos representam casos isolados e pouco expressivos. Essa distribuição reflete uma curva assimétrica moderada, com predominância em idades adultas jovens e baixa representatividade de extremos.

1.2 –

Tabela de Frequência para a idade dos maridos:

```
Class limits f rf rf(%) cf cf(%)
[18.81,23.671) 102 0.02 1.81 102 1.81
[23.671,28.531) 466 0.08 8.27 568 10.08
[28.531,33.392) 809 0.14 14.36 1377 24.44
[33.392,38.253) 895 0.16 15.89 2272 40.33
[38.253,43.114) 917 0.16 16.28 3189 56.60
[43.114,47.974) 629 0.11 11. 16 3818 67.77
[47.974,52.835) 649 0.12 11.52 4467 79.29
[52.835,57.696) 541 0.10 9.60 5008 88.89
[57.696,62.556) 394 0.07 6.99 5402 95.88
[62.556,67.417) 152 0.03 2.70 5554 98.58
[67.417,72.278) 51 0.01 0.91 5605 99.49
[72.278,77.139) 21 0.00 0.37 5626 99.86
[77.139,81.999) 6 0.00 0.11 5632 99.96
[81.999,86.86) 2 0.00 0.04 5634 100.00
```

Tabela de Frequência para a idade das esposas:

```
Class limits f rf rf(%) cf cf(%)
[17.82,20.804) 61 0.01 1.08 61 1.08
[20.804,23.787) 161 0.03 2.86 222 3.94
[23.787,26.771) 312 0.06 5.54 534 9.48
[26.771,29.754) 505 0.09 8.96 1039 18.44
[29.754,32.738) 562 0.10 9.98 1601 28.42
[32.738,35.721) 571 0.10 10.13 2172 38.55
[35.721,38.705) 624 0.11 11.08 2796 49.63
[38.705,41.689) 510 0.09 9.05 3306 58.68
[41.689,44.672) 542 0.10 9.62 3848 68.30
[44.672,47.656) 432 0.08 7.67 4280 75.97
[47.656,50.639) 389 0.07 6.90 4669 82.87
[50.639,53.623) 358 0.06 6.35 5027 89.23
[53.623,56.606) 304 0.05 5.40 5331 94.62
[56.606,59.59) 303 0.05 5.38 5634 100.00
```

Pode-se concluir que a distribuição etária das esposas se apresenta mais homogênea em comparação à dos maridos, com menor dispersão em torno da mediana. Isso indica que as idades das esposas estão mais equilibradas, sem grandes variações entre os valores centrais, ao contrário do que se observa entre os maridos.

2 – Medidas de tendência central
Medidas de dispersão variância:

- Esposas (age): 99,75
- Maridos (husage): 126,87

Desvio padrão:

- Esposas (age): 9,99
- Maridos (husage): 11,12

Coeficiente de variação (CV):

- Esposas (age): 25,35%
- Maridos (husage): 26,45%

A variância e o desvio padrão da idade dos maridos são mais elevados, indicando maior dispersão em relação à média quando comparados às esposas.
O coeficiente de variação confirma essa diferença: as idades dos maridos apresentam variação relativa de 26,45%, contra 25,35% das esposas.
Em outras palavras, embora as distribuições sejam semelhantes, os maridos apresentam uma maior heterogeneidade etária, enquanto as esposas têm uma distribuição de idades ligeiramente mais concentrada em torno da média.

3 – O p-valor das duas amostras ficou menor que 0,05. Isso significa que a gente rejeita a hipótese de que os dados seguem uma distribuição normal. Em outras palavras: os dados não têm aquele “padrão certinho” de curva normal. Por isso, em vez de usar testes estatísticos tradicionais (que assumem normalidade), vamos precisar usar um teste não-paramétrico, que funciona melhor nesses casos.

```
group_by(idades, categoria) %>%
summarise( count = n(), median = median(idade, na.rm = TRUE), IQR = IQR(idade,
na.rm = TRUE))
wilcox.test(idade ~ categoria, data = idades, exact = FALSE, conf.int=TRUE)
wilcox.test(idade ~ categoria, data = idades, exact = FALSE, alternative = "less",
conf.int=TRUE)
wilcox.test(idade ~ categoria, data = idades, exact = FALSE, alternative =
"greater", conf.int=TRUE)
```

Como os dados não seguem uma distribuição normal, usamos um teste não-paramétrico (Mann-Whitney). O resultado mostrou que a idade mediana dos maridos não é estatisticamente menor que a

das esposas, já que o p-valor ficou acima de 0,05. Além disso, vimos que a diferença entre as medianas é de até 3 anos, sendo a mediana da diferença de aproximadamente 2,99 anos.

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

(100 pontos) Fazer as regressões Ridge, Lasso e ElasticNet com a variável dependente "lwage" (salário-hora da esposa em logaritmo neperiano) e todas as demais variáveis da base de dados são variáveis explicativas (todas essas variáveis tentam explicar o salário-hora da esposa). No pdf você deve colocar a rotina utilizada, mostrar em uma tabela as estatísticas dos modelos (RMSE e R^2) e concluir qual o melhor modelo entre os três, e mostrar o resultado da predição com intervalos de confiança para os seguintes valores:

husage = 40 (anos – idade do marido)
husunion = 0 (marido não possui união estável)
husearns = 600 (US$ renda do marido por semana)
huseduc = 13 (anos de estudo do marido)
husblck = 1 (o marido é preto)
hushisp = 0 (o marido não é hispânico)
hushrs = 40 (horas semanais de trabalho do marido)
kidge6 = 1 (possui filhos maiores de 6 anos)
age = 38 (anos – idade da esposa)
black = 0 (a esposa não é preta)
educ = 13 (anos de estudo da esposa)
hispanic = 1 (a esposa é hispânica)
union = 0 (esposa não possui união estável)
exper = 18 (anos de experiência de trabalho da esposa)
kidlt6 = 1 (possui filhos menores de 6 anos)

obs: lembre-se de que a variável dependente "lwage" já está em logarítmo, portanto voçê não precisa aplicar o logaritmo nela para fazer as regressões, mas é necessário aplicar o antilog para obter o resultado da predição.

B – RESOLUÇÃO

```
# Treinamento Rige e Lasso
modelo_ridge = glmnet(x_train, y_train, nlambda = 25, alpha = 0,
                      family = 'gaussian', lambda = best_lambda_ridge)
modelo_lasso = glmnet(x_train, y_train, alpha = 1, lambda = best_lambda_lasso,
                      standardize = TRUE)

# ELASTIC NET
train_cont = trainControl(method = "repeatedcv", number = 10, repeats = 5,
                          search = "random", verboseIter = FALSE)
```

```
best_lambda_elasticnet = modelo_elasticnet$bestTune[[2]]
```

Modelo	ALPHA	LAMBDA
RIDGE	0	0,015848932
LASSO	1	0,010000000
ELASTIC_NET	0,739429239649326	0,005297002

```
# Treinamento
prediction = predict(modelo_ridge, s = best_lambda_ridge, newx = x_train)
df = avaliacao_resultados(y_train, prediction, train)

prediction = predict(modelo_lasso, s = best_lambda_lasso, newx = x_train)
df = rbind(df,avaliacao_resultados(y_train, prediction, train))

prediction = predict(modelo_elasticnet, x_train)
df = rbind(df,avaliacao_resultados(y_train, prediction, train))

# Teste
prediction = predict(modelo_ridge, s = best_lambda_ridge, newx = x_test)
df = rbind(df,avaliacao_resultados(y_test, prediction, test))

prediction = predict(modelo_lasso, s = best_lambda_lasso, newx = x_test)
df = rbind(df,avaliacao_resultados(y_test, prediction, test))

prediction = predict(modelo_elasticnet, x_test)
df = rbind(df,avaliacao_resultados(y_test, prediction, test))
```

Modelo	RMSE	R^2
RIDGE	**0,2967970**	**0,6838540**
LASSO	**0,2984090**	**0,6804104**
ELASTIC NET	**0,2969706**	**0,6834841**
RIDGE	**0,2630827**	**0,7187452**
LASSO	**0,2608951**	**0,7234033**
ELASTIC NET	**0,2616748**	**0,7217475**

Intervalo de confiança

```
prediction = predict(modelo_ridge, s = best_lambda_ridge, newx = our_pred)
df = data.frame(prediction)

prediction = predict(modelo_lasso, s = best_lambda_lasso, newx = our_pred)
df = rbind(df,prediction)
```

```
prediction = predict(modelo_elasticnet, our_pred)
df = rbind(df,prediction)

df = exp(df) lwage_antilog = exp(trabalhosalarios$lwage)

m = nrow(train)
t = sd(lwage_antilog)

dam = t/sqrt(m)

df$interval_inf = df[,1] + (qnorm(0.025))*dam
df$interval_sup = df[,1] - (qnorm(0.025))*dam
```

Modelo	Salário	Intervalo Inf.	Intervalo Sup.
RIDGE	8,082569	7,822426	8,342712
LASSO	8,640474	8,380331	8,900618
ELASTIC NET	8,591191	8,331048	8,851334

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

O problema consiste em criar um modelo de reconhecimento de padrões que dado um texto de entrada, o programa consegue classificar o texto e indicar a língua em que o texto foi escrito.

Parta do exemplo (notebook produzido no Colab) que foi disponibilidade e crie as funções para calcular as diferentes características para o problema da identificação da língua do texto de entrada.

Nessa atividade é para "construir características".

Meta: a acurácia deverá ser maior ou igual a 70%.

Essa tarefa pode ser feita no Colab (Google) ou no Jupiter, em que deverá exportar o notebook e imprimir o notebook para o formato PDF. Envie no UFPR Virtual os dois arquivos.

2 Melhore uma base de dados ruim

Escolha uma base de dados pública para problemas de classificação, disponível ou com origem na UCI Machine Learning.

Use o mínimo de intervenção para rodar a SVM e obtenha a matriz de confusão dessa base.

O trabalho começa aqui, escolha as diferentes tarefas discutidas ao longo da disciplina, para melhorar essa base de dados, até que consiga efetivamente melhorar o resultado.

Considerando a acurácia para bases de dados balanceadas ou quase balanceadas, se o percentual da acurácia original estiver em até 85%, a meta será obter 5%. Para bases com mais de 90% de acurácia, a meta será obter a melhora em pelo menos 2 pontos percentuais (92% ou mais).

Nessa atividade deverá ser entregue o script aplicado (o notebook e o PDF correspondente).

B – RESOLUÇÃO

1 –

```
import random
pre-padrao = []
for frase in
    ingles: pre-padrao.append( [frase, 'inglês'])
for frase in espanhol:
    pre-padrao.append( [frase, 'espanhol'])
for frase in portugues:
    pre-padrao.append( [frase, 'português'])
random.shuffle(pre-padrao)
print(pre-padrao)
```

```
[['¿Qué hora es?', 'espanhol'], ['O filme que assisti ontem foi ótimo.', 'português'], ['A festa começa às 20h.', 'português'], ['M
```

```
import pandas as pd
dados = pd.DataFrame(pre-padrao)
dados

import re
import pandas as pd

def tamanhoMedioFrases(texto):
    """
    Calcula o tamanho médio das palavras em um texto.
    """
    palavras = re.split(r"\s+", texto)
    tamanhos = [len(s) for s in palavras if len(s) > 0]
    if not tamanhos:
        return 0
    return sum(tamanhos) / len(tamanhos)

def contaVogaisAcentuadas(texto):
    """
    Conta a quantidade de vogais acentuadas e alguns caracteres especiais.
    """
    acentuados = ['á', 'à', 'ã', 'â', 'é', 'ê', 'í', 'ó', 'õ', 'ô', 'ü', 'ç', 'ñ']
    return sum(1 for ac in texto if ac in acentuados)

def temIng(frase):
    """
    Verifica se a frase contém alguma palavra terminada com 'ing'.
    Retorna 1 se sim, 0 se não.
    """
    palavras = frase.split()
    return int(any(c.endswith('ing') for c in palavras))
```

```
def extraiCaracteristicas(frase):
    """
    Recebe uma entrada [texto, classe] e retorna um vetor de características.
    """
    texto = frase[0]
    # Remove caracteres que não são letras, números ou underscore
    pattern_regex = re.compile(r"[^\w+]", re.UNICODE)
    texto = re.sub(pattern_regex, " ", texto)

    caracteristica1 = tamanhoMedioFrases(texto)
    caracteristica2 = contaVogaisAcentuadas(texto)
    caracteristica3 = temIng(texto)

    # Cria o vetor de características seguido da classe
    return [caracteristica1, caracteristica2, caracteristica3, frase[1]]

def geraPadroes(frases):
    """
    Converte uma lista de frases no formato [texto, classe]
    para uma lista de características + classe.
    """
    return [extraiCaracteristicas(frase) for frase in frases]

padroes = geraPadroes(pre-padrao)

# Converte em DataFrame para visualização
dados = pd.DataFrame(
    padroes,
    columns=["Tamanho Médio", "Vogais Acentuadas", "Contém 'ing'", "Classe"]
)

print(dados)
```

```
[[3.0, 1, 0, 'espanhol'], [4.142857142857143, 1, 0, 'português'], [3.4, 2, 0, 'portug
```

	0	1	2	3
0	3.000000	1	0	espanhol
1	4.142857	1	0	português
2	3.400000	2	0	português
3	4.800000	1	0	espanhol
4	4.200000	0	0	inglês
...	...	...	...	...
87	5.000000	1	0	português
88	5.500000	0	0	português
89	5.250000	0	0	espanhol
90	4.333333	0	0	português
91	4.200000	0	0	espanhol

92 rows × 4 columns

```python
# === Treino/Teste com SVM ===
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.model_selection import StratifiedKFold, cross_val_score

RANDOM_STATE = 42
TEST_SIZE = 0.25

if 'dados' in globals() and isinstance(dados, pd.DataFrame):
    X = dados.iloc[:, :-1].astype(float).values
    y = dados.iloc[:, -1].values
else:
    # fallback para lista padroes
    vet = np.array(padroes, dtype=object)
    X = vet[:, 0:-1].astype(float)
    y = vet[:, -1]

# --- Split estratificado
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=TEST_SIZE, stratify=y, random_state=RANDOM_STATE
)

# --- Pipeline: padronização + SVM (RBF)
modelo = Pipeline([
    ("scaler", StandardScaler()),
    ("svc", SVC(kernel="rbf", class_weight=None, random_state=RANDOM_STATE))
```

```
])

# --- Treinamento
modelo.fit(X_train, y_train)

# --- Métricas em treino
y_pred_train = modelo.predict(X_train)
acc_train = accuracy_score(y_train, y_pred_train)
print(f"Acurácia nos dados de treinamento: {acc_train*100:.2f}%")
print("Matriz de confusão (treino):")
print(confusion_matrix(y_train, y_pred_train))
print("Relatório de classificação (treino):")
print(classification_report(y_train, y_pred_train, digits=4))

# --- Métricas em teste (mais confiáveis)
print("\n=== Métricas no conjunto de TESTE ===")
y_pred_test = modelo.predict(X_test)
acc_test = accuracy_score(y_test, y_pred_test)
print(f"Acurácia nos dados de teste: {acc_test*100:.2f}%")
print("Matriz de confusão (teste):")
print(confusion_matrix(y_test, y_pred_test))
print("Relatório de classificação (teste):")
print(classification_report(y_test, y_pred_test, digits=4))

# --- (Opcional) Validação cruzada estratificada para ter noção de variação
cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=RANDOM_STATE)
scores = cross_val_score(modelo, X, y, cv=cv, scoring="accuracy")
print(f"\nValidação cruzada (5-fold) - Acurácia: média={scores.mean():.4f},
desvio={scores.std():.4f}")
```

```
Acurácia nos dados de treinamento: 65.22%
[[ 9  5  8]
 [ 2 21  0]
 [ 4  5 15]]
                precision recall f1-score support
espanhol          0.60        0.41   0.49       22
inglês            0.68        0.91   0.78       23
português         0.65        0.62   0.64       24

accuracy                      0.65   69
macro avg         0.64        0.65   0.63       69
weighted avg      0.64        0.65   0.64       69

métricas mais confiáveis
[[3 2 3]
```

```
[1 6 0]
[2 2 4]]
                  precision recall f1-score support
espanhol          0.50      0.38   0.43        8
inglês            0.60      0.86   0.71        7
português         0.57      0.50   0.53        8

accuracy                            0.57        23
macro avg         0.56      0.58   0.56        23
weighted avg      0.56      0.57   0.55        23
```

2 –

The AI4I 2020 Predictive Maintenance Dataset is a synthetic dataset that reflects real predictive maintenance data encountered in industry.

Dataset Characteristics Multivariate, Time-Series

Subject Area Computer

Associated Tasks Classification, Regression, Causal-Discovery

Attribute Type Real

Instances

10000

Attributes

14

Has Missing Values?

No

```
import requests, zipfile, io
from io import BytesIO
import numpy as np
import pandas as pd

# base de dados disponível na UCI Machine Learning -
https://archive.ics.uci.edu/dataset/601/ai4i+2020+predictive+maintenance+dataset

r =
requests.get('https://archive.ics.uci.edu/static/public/601/ai4i+2020+predictive+ma
intenance+dataset.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
z.namelist()
dadosfp = z.open('ai4i2020.csv')
dados = dadosfp.read()
```

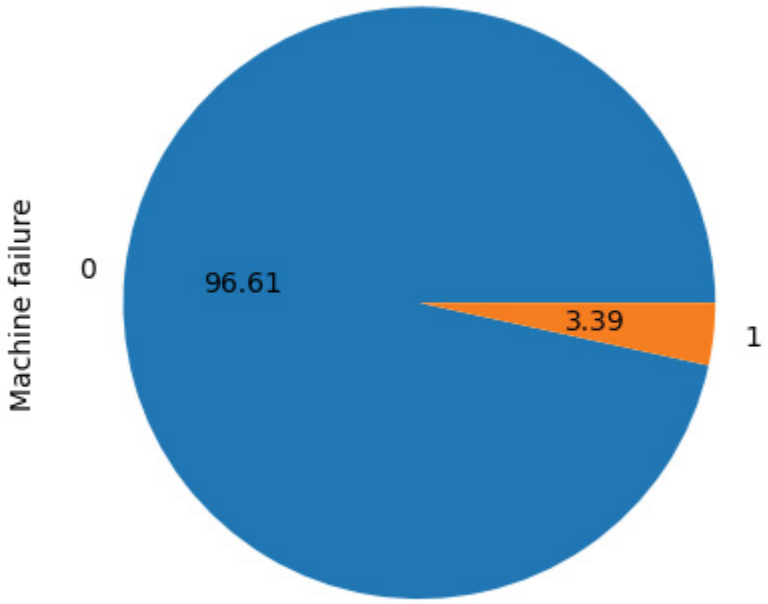
```
dataset = pd.read_csv(io.BytesIO(dados))
print( dataset.head() )

dataset.describe()

#ids = dataset['Product ID']
#print(len(np.unique(ids)))

#occur = dataset.groupby(['Machine failure']).size()
# display occurrences of a particular column
#print(occur)

Results = dataset['Machine failure']
Results.value_counts().plot.pie(autopct='%.2f')

Results.value_counts()/Results.shape[0]
```

	UDI	Product ID	Type	Air temperature [K]	Process temperature [K]
0	1	M14860	M	298.1	308.6
1	2	L47181	L	298.2	308.7
2	3	L47182	L	298.1	308.5
3	4	L47183	L	298.2	308.6
4	5	L47184	L	298.2	308.7

	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure	TWF
0	1551	42.8	0	0	0
1	1408	46.3	3	0	0
2	1498	49.4	5	0	0
3	1433	39.5	7	0	0
4	1408	40.0	9	0	0

	HDF	PWF	OSF	RNF
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0

```
0    0.9661
1    0.0339
```



Hora de realizar os tratamentos

no exemplo, iremos normalizar as colunas, remover a coluna de identificação e separar a classe dos atributos.

```
from sqlalchemy import column

#Aqui estamos escluindo as colunas 'UDI' que apenas um serializador que vai de 1 a
10000
#e a coluna 'Product ID' que também é um número único por equipamento monitorado
X = dataset.iloc[:,2:]
print(X.head())

#Aqui estamos separando os resultados esperados do dataset 1 falhou 0 não falhou
Y = dataset['Machine failure']
Y_orig = dataset['Machine failure']
print(Y.unique())

#Aqui estamos excluindo a coluna dos resultados esperados do dataset
X.drop(['Machine failure'], axis='columns', inplace=True)
print(X.head())
```

```
  Type  Air temperature [K]  Process temperature [K]  Rotational speed [rpm]
0    M                298.1                    308.6                    1551
1    L                298.2                    308.7                    1408
2    L                298.1                    308.5                    1498
3    L                298.2                    308.6                    1433
4    L                298.2                    308.7                    1408
```

```
   Torque [Nm]  Tool wear [min]  Machine failure  TWF  HDF  PWF  OSF  RNF
0         42.8                0                0    0    0    0    0    0
1         46.3                3                0    0    0    0    0    0
2         49.4                5                0    0    0    0    0    0
3         39.5                7                0    0    0    0    0    0
4         40.0                9                0    0    0    0    0    0
[0 1]
  Type  Air temperature [K]  Process temperature [K]  Rotational speed [rpm]
0    M                298.1                    308.6                    1551
1    L                298.2                    308.7                    1408
2    L                298.1                    308.5                    1498
3    L                298.2                    308.6                    1433
4    L                298.2                    308.7                    1408

   Torque [Nm]  Tool wear [min]  TWF  HDF  PWF  OSF  RNF
0         42.8                0    0    0    0    0    0
1         46.3                3    0    0    0    0    0
2         49.4                5    0    0    0    0    0
3         39.5                7    0    0    0    0    0
4         40.0                9    0    0    0    0    0
```

Na próxima seção que deverão ser realizada as tentativas de tratamento de dados, visando a melhoria no desempenho do classificador (SVM).

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

#Para podermos utilizar o minmax_scale do Pandas é necessário que a coluna
#'Type' que se refere ao volume de trabalho dos equipamentos que varia entre
#L Low 20% - M Medium 30% e H High 50%, seja convertida para frações númericas
#respectivamente L->0.2, M->0.3 e H->0.5
vlrs = {'L':0.2, 'M':0.3, 'H':0.5, 'G':0}
X = X.replace({'Type': vlrs})
print(X['Type'].unique())

#Aqui criamos uma cópia do dataset já limpo para compararmos como original
X_orig =  X.copy()
print(X_orig.head())

print(Y_orig.unique())

# normalização min-max
```

```
X = pd.DataFrame(minmax_scale(X))

print(X_orig.head())
print(X.head())
```

```
[0.3 0.2 0.5]
   Type  Air temperature [K]  Process temperature [K]  Rotational speed [rpm]
0   0.3                298.1                    308.6                    1551
1   0.2                298.2                    308.7                    1408
2   0.2                298.1                    308.5                    1498
3   0.2                298.2                    308.6                    1433
4   0.2                298.2                    308.7                    1408

   Torque [Nm]  Tool wear [min]  TWF  HDF  PWF  OSF  RNF
0         42.8                0    0    0    0    0    0
1         46.3                3    0    0    0    0    0
2         49.4                5    0    0    0    0    0
3         39.5                7    0    0    0    0    0
4         40.0                9    0    0    0    0    0
[0 1]
   Type  Air temperature [K]  Process temperature [K]  Rotational speed [rpm]
0   0.3                298.1                    308.6                    1551
1   0.2                298.2                    308.7                    1408
2   0.2                298.1                    308.5                    1498
3   0.2                298.2                    308.6                    1433
4   0.2                298.2                    308.7                    1408

   Torque [Nm]  Tool wear [min]  TWF  HDF  PWF  OSF  RNF
0         42.8                0    0    0    0    0    0
1         46.3                3    0    0    0    0    0
2         49.4                5    0    0    0    0    0
3         39.5                7    0    0    0    0    0
4         40.0                9    0    0    0    0    0
          0         1         2         3         4         5    6    7    8
0  0.333333  0.304348  0.358025  0.222934  0.535714  0.000000  0.0  0.0  0.0
1  0.000000  0.315217  0.370370  0.139697  0.583791  0.011858  0.0  0.0  0.0
2  0.000000  0.304348  0.345679  0.192084  0.626374  0.019763  0.0  0.0  0.0
3  0.000000  0.315217  0.358025  0.154249  0.490385  0.027668  0.0  0.0  0.0
4  0.000000  0.315217  0.370370  0.139697  0.497253  0.035573  0.0  0.0  0.0

    9    10
0  0.0  0.0
```

```
1   0.0   0.0
2   0.0   0.0
3   0.0   0.0
4   0.0   0.0
```

A próxima seção trata da construção do modelo, dos testes e das métricas da matriz de confusão.

```
from sklearn.model_selection import train_test_split
import numpy as np

#Cortando o dataset original em bases de treino e de teste
X_oring_train, X_orig_test, y_orig_train, y_orig_test = train_test_split(X_orig,
                      Y_orig, test_size=0.25, stratify=Y_orig,random_state=10)

print(y_orig_train.value_counts()/y_orig_train.shape[0])
print(y_orig_test.value_counts()/y_orig_test.shape[0])

#Cortando o dataset tratado em bases de treino e de teste
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                                                    stratify=Y,random_state=10)
print(y_train.value_counts()/y_train.shape[0])
print(y_test.value_counts()/y_test.shape[0])
```

```
0    0.966133
1    0.033867
Name: Machine failure, dtype: float64
0    0.966
1    0.034
Name: Machine failure, dtype: float64
0    0.966133
1    0.033867
Name: Machine failure, dtype: float64
0    0.966
1    0.034
Name: Machine failure, dtype: float64
```

Avaliando o número de ocorrências de falhas X número de não falhas
Machine failure
0 9661
1 339
dtype: int64
percebe-se claramente o desbalanceamento do dataset, portanto trataremos a prevalência utilizando um modelo de regreção logística.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix

clf = LogisticRegression(random_state=0,max_iter=500).fit(X_oring_train,
y_orig_train)
y_orig_pred = clf.predict(X_orig_test)

#Devido a prevalência temos uma acurácia muito alta do modelo, o que nos diz que
#a acurácia não é um bom indicador neste caso.
print(accuracy_score(y_orig_test, y_orig_pred))
print(confusion_matrix(y_orig_test, y_orig_pred))

#Para conseguir visualizar melhor podemos usar o recall_score que representa
#mais objetivamente o resultado obtido
print(recall_score(y_orig_test, y_orig_pred))
```

```
0.9992
[[2415    0]
 [   2   83]]
0.9764705882352941
```

Treina o modelo com base nos dados originais (SVM).

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

treinador = svm.SVC()  #algoritmo escolhido

modelo_orig = treinador.fit(X_oring_train, y_orig_train)

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))
print(accuracy_score(y_orig_train, y_orig_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
```

```
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
print(accuracy_score(y_orig_test, y2_orig_pred))
```

```
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[7246    0]
 [ 252    2]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      7246
           1       1.00      0.01      0.02       254

    accuracy                           0.97      7500
   macro avg       0.98      0.50      0.50      7500
weighted avg       0.97      0.97      0.95      7500

0.9664
Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[2415    0]
 [  84    1]]
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      2415
           1       1.00      0.01      0.02        85

    accuracy                           0.97      2500
   macro avg       0.98      0.51      0.50      2500
weighted avg       0.97      0.97      0.95      2500

0.9664
```

Como os dados ficam após os processos de tratamento dos dados?

```
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

treinador = svm.SVC()  #algoritmo escolhido

modelo = treinador.fit(X_train, y_train)

# predição com os mesmos dados usados para treinar
y_pred = modelo.predict(X_train)
```

```
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred))
print(accuracy_score(y_train, y_pred))

# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))
print(accuracy_score(y_test, y2_pred))
```

```
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[7246    0]
 [   7  247]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7246
           1       1.00      0.97      0.99       254

    accuracy                           1.00      7500
   macro avg       1.00      0.99      0.99      7500
weighted avg       1.00      1.00      1.00      7500

0.9990666666666667
Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[2415    0]
 [   2   83]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2415
           1       1.00      0.98      0.99        85

    accuracy                           1.00      2500
   macro avg       1.00      0.99      0.99      2500
weighted avg       1.00      1.00      1.00      2500

0.9992
```

Podemos concluir que, neste experimento, o SVM apresentou dificuldades com dados desbalanceados, só atingindo um bom desempenho após a normalização das variáveis, quando alcançou um recall próximo de 99%.

Por outro lado, a Regressão Logística obteve resultados satisfatórios já nos dados brutos, sem necessidade de pré-processamento adicional.

Assim, fica evidente que algoritmos diferentes respondem de maneira distinta às características do conjunto de dados e que o trabalho de preparação e transformação dos dados pode impactar fortemente o desempenho dos modelos de aprendizado de máquina.

APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

Para cada uma das tarefas abaixo (Classificação, Regressão etc.) e cada base de dados (Veículo, Diabetes etc.), fazer os experimentos com todas as técnicas solicitadas (KNN, RNA etc.) e preencher os quadros com as estatísticas solicitadas, bem como os resultados pedidos em cada experimento.

B – RESOLUÇÃO

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – CV	size=31 decay=0.1	0.8235	Reference Prediction bus opel saab van bus 45 0 0 0 opel 0 21 14 1 saab 0 12 30 0 van 2 0 1 44
SVM – Hold-out	C=1 Sigma=0.08972209	0.7588	Reference Prediction bus opel saab van bus 43 0 1 0 opel 0 21 21 0 saab 0 12 21 1 van 4 0 2 44
SVM – CV	C=1 Sigma=0.08972209	0.7588	Reference Prediction bus opel saab van bus 43 0 1 0 opel 0 21 21 0 saab 0 12 21 1 van 4 0 2 44
RF – Hold-out	mtry=2	0.7588	Prediction bus opel saab van bus 44 0 1 0 opel 0 17 20 0 saab 0 16 23 0 van 3 0 1 45
RF – CV	mtry=2	0.7588	Prediction bus opel saab van bus 44 0 0 0 opel 0 18 21 0 saab 0 15 23 1 van 3 0 1 44
RNA – Hold-out	size=5 decay=0.1	0.6941	Reference Prediction bus opel saab van bus 44 3 4 3 opel 0 29 33 1 saab 0 1 4 0 van 3 0 4 41

KNN	k=1	0.6353	Prediction bus opel saab van bus 41 2 2 4 opel 1 11 26 1 saab 3 20 17 1 van 2 0 0 39

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
1	90	44	80	150	65	9	155	32	18	140	168	410	161	75	9	9	189	195	bus
2	88	46	88	175	50	10	135	40	20	158	180	350	145	70	6	16	190	196	van
3	109	52	110	220	82	9	201	44	22	160	225	650	208	72	9	14	187	197	opel

- **RNA – CV**

```
### Pacotes necessários:
install.packages("e1071")
install.packages("caret")
library("caret")

install.packages("mlbench")
install.packages("mice")
library(mlbench)
library(mice)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES DE DADOS/06 - Veículos")
dados <- read.csv("6 - Veiculos - Dados.csv")

### Retira o atributo a
dados$a <- NULL
imp <- mice(dados)
dados <- complete(imp, 1)

View(dados)

### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma randomizada
set.seed(202522)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

### indica o método cv e número de folders 10
ctrl <- trainControl(method = "cv", number = 10)

# Parametrização da RNA
### size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay = seq(from = 0.1, to = 0.9, by = 0.3))
```

```
set.seed(202522)
rna <- train(
  form = tipo~. ,
  data = treino ,
  method = "nnet" ,
  tuneGrid = grid ,
  trControl = ctrl ,
  maxit = 2000,trace=FALSE)
rna

predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("6 - Veiculos - Novos Casos.csv")
dados_novos_casos$a <- NULL
View(dados_novos_casos)

tipo <- predict(rna, dados_novos_casos)
resultado <- cbind(dados_novos_casos, tipo)
resultado$tipo <- NULL
View(resultado)
```

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – CV	mtry=8	0.7922	Reference Prediction neg pos neg 91 22 pos 10 31
RF – Hold-out	mtry=2	0.7922	Reference Prediction neg pos neg 92 23 pos 9 30
RNA – CV	size=11 decay=0.4	0.7727	Reference Prediction neg pos neg 89 23 pos 12 30
KNN	k=9	0.7597	Reference Prediction neg pos neg 88 24 pos 13 29
SVM – Hold-out	C=0.25 Sigma=0.124699	0.7403	Reference Prediction neg pos neg 93 32 pos 8 21

SVM – CV	C=0.25 Sigma=0.124699	0.7403	Reference Prediction neg pos neg 93 32 pos 8 21
RNA – Hold-out	size=5 decay=0.1	0.6558	Reference Prediction neg pos neg 84 36 pos 17 17

	num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	1	8	175	78	30	0	42.2	0.890	60	pos
2	2	3	75	68	25	1	28.9	0.625	47	neg
3	3	5	150	61	22	0	27.2	0.452	82	neg

- **RF – CV**

```
### Pacotes necessários:
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("mice")
library("caret")
library(mice)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES DE DADOS/10 - Diabetes")
dados <- read.csv("10 - Diabetes - Dados.csv")
View(dados)

### Retira o atributo num
dados$num <- NULL
imp <- mice(dados)
dados <- complete(imp, 1)

View(dados)

### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma randomizada
set.seed(202522)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

### indica o método cv e número de folders 10
ctrl <- trainControl(method = "cv", number = 10)

### executa a RNA com esse ctrl
```

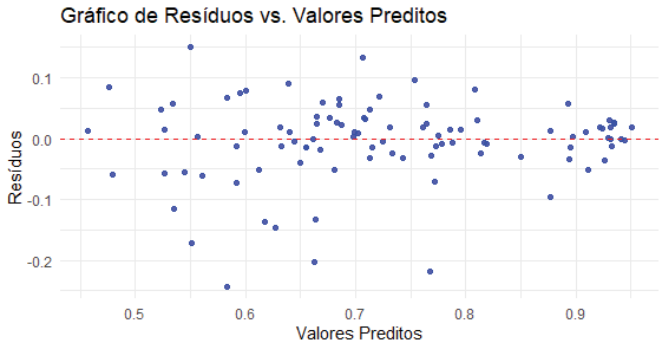
```
set.seed(202522)
rf <- train(diabetes~., data=treino, method="rf", trControl=ctrl)
rf

predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$diabetes))

### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("10 - Diabetes - Novos Casos.csv")
dados_novos_casos$a <- NULL
View(dados_novos_casos)

diabetes <- predict(rf, dados_novos_casos)
resultado <- cbind(dados_novos_casos, diabetes)
resultado$diabetes <- NULL
View(resultado)
```

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.8074227	0.0691989000098334	0.8998701	0.06631433	0.04540445
RF – CV	mtry=2	0.8071875	0.0692411366533538	0.8997531	0.06635481	0.04512983
RNA – CV	size=10 decay=0.1	0.7846011	0.0731843764135317	0.8865595	0.07013367	0.0494467
SVM – CV	C=1 Sigma=0.1425977	0.7735897	0.0750316835718087	0.8867025	0.07190398	0.04709117
SVM – Hold-out	C=0.5 Sigma=0.1425977	0.7692516	0.0757470929739764	0.8854073	0.07258956	0.04832958
KNN	K=9	0.7622408	0.0768891921780 93	0.8771803	0.07368405	0.05159864
RNA -Hold-out	size=5 decay=0.1	0.7514687	0.0786116977609423	0.8682378	0.07533476	0.05373943

- **RF – Hold-out**

	num	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predicoes.rf
1	1	316	107	3	3	3	8	0	0.6496269
2	2	411	139	4	4	4	10	1	0.9516853
3	3	221	75	2	2	2	5	1	0.4627021




```
### Instalação dos pacotes (são os mesmos da classificação)
install.packages("caret")
install.packages("e1071")
install.packages("mlbench")
install.packages("mice")
library(mlbench)
library(caret)
library(mice)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES DE DADOS/09 - Admissão")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
dados$num <- NULL
View(dados)

### Criar bases de Treino e Teste
set.seed(202522)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Hold-out
set.seed(202522)
rf <- train(ChanceOfAdmit~., data=treino, method="rf")
rf

### Aplicar modelos treinados na base de Teste
predicoes.rf <- predict(rf, teste)
```

```
### Calcular as métricas
library(Metrics)
library("caret")

rmse(teste$ChanceOfAdmit, predicoes.rf)

r2 <- function(predito, observado) {
  return(1 - (sum((predito-observado)^2) / sum((observado-mean(observado))^2)))
}
r2(predicoes.rf,teste$ChanceOfAdmit)

### MAE (Mean Absolute Error)
library(Metrics)
mae(teste$ChanceOfAdmit, predicoes.rf)

### Pearson correlation coefficient
library(Metrics)
cor(predicoes.rf, teste$ChanceOfAdmit)

# Syx (Standard Error of the Estimate)
k <- ncol(treino) - 1  # Número de preditores (colunas menos o target 'Volume')
n <- nrow(teste)
SSE <- sum((predicoes.rf - teste$ChanceOfAdmit)^2)
df <- n - k - 1
if (df > 0) {
  syx_val <- sqrt(SSE / df)
  print(paste("Syx:", syx_val))
} else {
  print("Não é possível calcular Syx: graus de liberdade insuficientes (precisa de
mais dados no teste).")
}

# Calcule os resíduos
residuos <- teste$ChanceOfAdmit - predicoes.rf

# Crie um data frame para plotar
library(ggplot2)
dados_plot <- data.frame(Preditos = predicoes.rf, Residuos = residuos)

# Gráfico de resíduos vs. valores preditos
ggplot(dados_plot, aes(x = Preditos, y = Residuos)) +
  geom_point(color = "blue", alpha = 0.7) +  # Pontos dos resíduos
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +  # Linha
horizontal em 0
  labs(title = "Gráfico de Resíduos vs. Valores Preditos",
       x = "Valores Preditos",
```

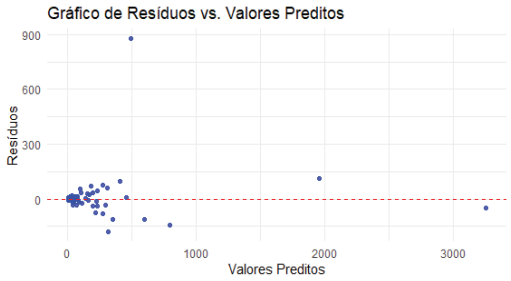
```
       y = "Resíduos") +
  theme_minimal()


### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("9 - Admissao - Dados - Novos Casos.csv")
View(dados_novos_casos)

predicoes.rf <- predict(rf, dados_novos_casos)
dados_novos_casos$ChanceOfAdmit <- NULL
resultado <- cbind(dados_novos_casos, predicoes.rf)
View(resultado)
```

Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.9401535	127.679195724529	0.9698671	123.3498	45.51557
RF – CV	mtry=2	0.9381546	129.79398802229	0.9693044	125.3929	46.13284
RNA – CV	size=2 decay=0.4	0.9093262	157.159938593344	0.953643	151.8309	62.75507
KNN	K=1	0.9031635	162.412862818585	0.9697907	156.9057	71.1855
SVM – Hold-out	C=1 Sigma=1.423117	0.8787125	181.764429606261	0.961287	175.6011	135.8677
SVM – CV	C=1 Sigma=1.423117	0.8787125	181.764429606261	0.961287	175.6011	135.8677
RNA – Hold-out	size=5 decay=0.1	0.7970782	235.106780589517	0.9000877	227.1347	136.6687

- **RF – Hold-out**

	dap	h	Me	predicoes.rf
1	17.53	15.50	0.62	156.21401
2	12.28	10.86	0.43	50.66508
3	22.79	20.15	0.80	372.71558



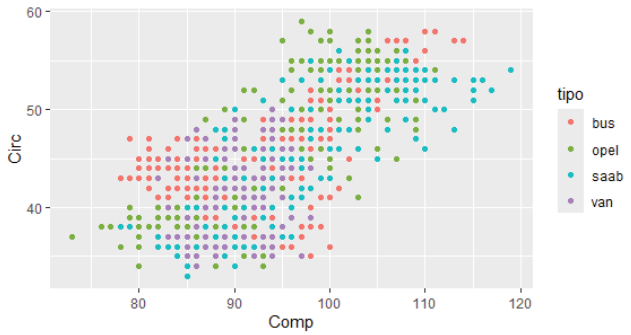

```
### Instalação dos pacotes (são os mesmos da classificação)
install.packages("caret")
install.packages("e1071")
install.packages("mlbench")
install.packages("mice")
library(mlbench)
library(caret)
library(mice)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES DE DADOS/05 - Biomassa")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)
View(dados)

### Criar bases de Treino e Teste
set.seed(202522)
indices <- createDataPartition(dados$biomassa, p=0.80,list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Hold-out
set.seed(202522)
rf <- train(biomassa~., data=treino, method="rf")
rf

### Aplicar modelos treinados na base de Teste
predicoes.rf <- predict(rf, teste)

### Calcular as métricas
library(Metrics)
library("caret")

rmse(teste$biomassa, predicoes.rf)

r2 <- function(predito, observado) {
```

```
  return(1 - (sum((predito-observado)^2) / sum((observado-mean(observado))^2)))
}
r2(predicoes.rf,teste$biomassa)

### MAE (Mean Absolute Error)
library(Metrics)
mae(teste$biomassa, predicoes.rf)

### Pearson correlation coefficient
library(Metrics)
cor(predicoes.rf, teste$biomassa)

# Syx (Standard Error of the Estimate)
k <- ncol(treino) - 1  # Número de preditores (colunas menos o target 'Volume')
n <- nrow(teste)
SSE <- sum((predicoes.rf - teste$biomassa)^2)
df <- n - k - 1
if (df > 0) {
  syx_val <- sqrt(SSE / df)
  print(paste("Syx:", syx_val))
} else {
  print("Não é possível calcular Syx: graus de liberdade insuficientes (precisa de
mais dados no teste).")
}

# Calcule os resíduos
residuos <- teste$biomassa - predicoes.rf

# Crie um data frame para plotar
library(ggplot2)
dados_plot <- data.frame(Preditos = predicoes.rf, Residuos = residuos)

# Gráfico de resíduos vs. valores preditos
ggplot(dados_plot, aes(x = Preditos, y = Residuos)) +
  geom_point(color = "blue", alpha = 0.7) +  # Pontos dos resíduos
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +  # Linha
horizontal em 0
  labs(title = "Gráfico de Resíduos vs. Valores Preditos",
       x = "Valores Preditos",
       y = "Resíduos") +
  theme_minimal()


### PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("5 - Biomassa - Dados - Novos Casos.csv")
View(dados_novos_casos)
```

```
predicoes.rf <- predict(rf, dados_novos_casos)
dados_novos_casos$biomassa <- NULL
resultado <- cbind(dados_novos_casos, predicoes.rf)
View(resultado)
```

Veículo

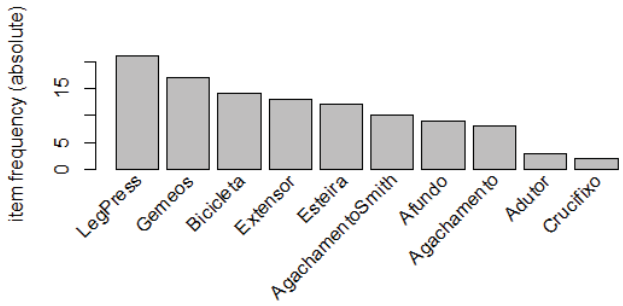
	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	km.res$cluster
1	95	48	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	4
2	91	41	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	3
3	104	50	106	209	66	10	207	32	23	158	223	635	220	73	14	9	188	196	9
4	93	41	82	159	63	9	144	46	19	143	160	309	127	63	6	10	199	207	3
5	85	44	70	205	103	52	149	45	19	144	241	325	188	127	9	11	180	183	1
6	107	57	106	172	50	6	255	26	28	169	280	957	264	85	5	9	181	183	10
7	97	43	73	173	65	6	153	42	19	143	176	361	172	66	13	1	200	204	4
8	90	43	66	157	65	9	137	48	18	146	162	281	164	67	3	3	193	202	6
9	86	34	62	140	61	7	122	54	17	127	141	223	112	64	2	14	200	208	7
10	93	44	98	197	62	11	183	36	22	146	202	505	152	64	4	14	195	204	2




```
### Pacotes necessários:
install.packages("klaR")
library(klaR)
install.packages("mice")
library(mice)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES DE DADOS/06 - Veículos")
dados <- read.csv("6 - Veiculos - Dados.csv")
View(dados)

### Gráfico com os dados
library(ggplot2)
ggplot(dados, aes(Comp, Circ, DCirc, RadRa, PrAxisRa, MaxLRa, ScatRa, Elong, PrAxisRect, MaxLRect, ScVarMaxis, ScVarmaxis, RaGyr, SkewMaxis, Skewmaxis, Kurtmaxis, KurtMaxis, HollRa, color = tipo)) + geom_point()

### Retira o atributo a
```

```
dados$a <- NULL
dados$tipo <- NULL
imp <- mice(dados)
dados <- complete(imp, 1)
View(dados)

### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma
randomizada
set.seed(202522)
km.res=kmeans(dados, 10)
print(km.res)

### Cria um arquivo com todos os registros e mais os clusters de cada um
resultado <- cbind(dados, km.res$cluster)
resultado
```

Musculação




```
> inspect(sort(rules[1:5], by="confidence"))
    lhs             rhs          support confidence coverage lift count
[1] {Extensor}   => {Bicicleta} 0.46    0.92       0.50     1.71 12
[2] {Esteira}    => {Extensor}  0.42    0.92       0.46     1.83 11
[3] {Bicicleta}  => {Extensor}  0.46    0.86       0.54     1.71 12
[4] {Extensor}   => {Esteira}   0.42    0.85       0.50     1.83 11
[5] {Gemeos}     => {LegPress}  0.46    0.71       0.65     0.87 12
```

```
### Instalação dos pacotes necessários
install.packages('arules', dep=T)
library(arules)
library(datasets)

### Leitura dos dados
setwd("C:/Users/EngWa/OneDrive/Documentos/Pessoal/UFPR/Aprendizado de Máquina/BASES
DE DADOS/12 - Regras de Associacao - Praticas/12 - Regras de Associacao - Praticas
```

```
- 2 - Musculacao")
# Importando o CSV como transações
dados <- read.transactions("2 - Musculacao - Dados.csv",
format = "basket", sep = ";")
inspect(dados[1:3])

### Podemos ver a frequência dos 10 primeiros itens:
itemFrequencyPlot(dados, topN=10, type='absolute')

### Podemos também ter uma visão geral dos dados:
summary(dados)

### Agora vamos obter as regras:
set.seed(202522)
rules <- apriori(dados, parameter = list(supp = 0.4, conf = 0.5, minlen = 2))
summary(rules)

### Vamos ver as 5 primeiras regras ordenadas pela confiança:
options(digits=2)
inspect(sort(rules[1:5], by="confidence"))
```

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

Implementar o exemplo de classificação de objetos usando a base de dados CIFAR10 e a arquitetura CNN vista no curso.

2 Detector de SPAM (RNN)

Implementar o detector de spam visto em sala, usando a base de dados SMS Spam e arquitetura de RNN vista no curso.

3 Gerador de Dígitos Fake (GAN)

Implementar o gerador de dígitos *fake* usando a base de dados MNIST e arquitetura GAN vista no curso.

4 Tradutor de Textos (Transformer)

Implementar o tradutor de texto do português para o inglês, usando a base de dados e a arquitetura Transformer vista no curso.

B – RESOLUÇÃO

Classificação de Imagens (CNN)
https://colab.research.google.com/drive/1VvpuQ1jOyOTZ-ycY7JkTw_SgU_q0XAQW?usp=sharing

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten, Dropout
from tensorflow.keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
# Não precisa separar
```

```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização os dados
# Imagens em pixels de 0 - 255
# / 255.0 transforma em 0 - 1
x_train, x_test = x_train / 255.0, x_test / 255.0
# O dado y é a classe a qual faz parte
# O flattem torna os dados vetorizados
y_train, y_test = y_train.flatten(), y_test.flatten()
# Dimensão dos dados
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", x_test.shape)
print("y_test.shape: ", y_test.shape)

K = len(set(y_train))
# Aqui começa o Estágio 1
i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=2, activation="relu")(i)
x = Conv2D(64, (3, 3), strides=2, activation="relu")(x)
x = Conv2D(128, (3, 3), strides=2, activation="relu")(x)
# Todas as imagens são do mesmo tamanho, não precisa de Global Pooling
x = Flatten()(x)
# Aqui começa o Estágio 2
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)
# Model ( lista entrada, lista saída)
model = Model(i, x)

# Relatório sobre a arquitetura da rede
model.summary()

# Compilar o modelo
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"])
# Treinar o modelo
r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=15)

# Plotar a função de perda, treino e validação
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()
```

```
# Plotar acurácia, treino e validação
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()

# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)

# Mostrar algumas classificações erradas
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog",
"frog", "horse", "ship", "truck"]
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)
plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],
labels[y_pred[i]]))
```

Detector de SPAM (RNN)

https://colab.research.google.com/drive/1whhrzJuzb_EA0PNtwNhosCmVcXM1HilH?usp=sharing

```
# Importação das Bibliotecas
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
from tensorflow.keras.layers import GlobalMaxPooling1D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer

# carrega e arruma a base
!wget http://www.razer.net.br/datasets/spam.csv
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
```

```
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values

# Separa a base em treino e teste
x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
test_size=0.33)

# Número máximo de palavras para considerar
# São consideradas as mais frequentes, as demais são
# ignoradas
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print("%s tokens" % V)

# Acerta o tamanho das sequências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior seq.
T = data_train.shape[1] # tamanho da sequência
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

# Define o modelo
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
M = 5 # tamanho do hidden state, quantidade de unidades LSTM
i = Input(shape=(T,)) # Entra uma frase inteira
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2 valores
model = Model(i, x)

model.summary()

# Compila e treina o modelo
model.compile(loss="binary_crossentropy", optimizer="adam",
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs, validation_data=(data_test,
y_test))
```

```
# Plota função de perda e acurácia
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.xlabel("Épocas")
plt.ylabel("loss")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
plt.legend()
plt.show()
plt.plot(r.history["accuracy"], label="accuracy")
plt.plot(r.history["val_accuracy"], label="val_accuracy")
plt.xlabel("Épocas")
plt.ylabel("Acurácia")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
plt.legend()
plt.show()

# Efetua a predição de um texto novo
#texto = "Hi, my name is Razer and want to tell you something."
texto = "Is your car dirty? Discover our new product. Free for all. Click the
link."
seq_texto = tokenizer.texts_to_sequences([texto]) # Tokeniza
data_texto = pad_sequences(seq_texto, maxlen=T) # Padding
pred = model.predict(data_texto) # Predição
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

Gerador de Dígitos Fake (GAN)

https://colab.research.google.com/drive/1T6n4X_z4uSaiUBzAnRVZHW-RBhB2mmQ5?usp=sharing

```
# Para Gerar os GIFs
!pip install imageio
!pip install git+https://github.com/tensorflow/docs

# Importações
import tensorflow as tf
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
from IPython import display
```

```
# Carregar a base de dados
(train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
# Normalização
train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5 # Normaliza entre [-1, 1]
# Gera o banco em partes e randomiza
BUFFER_SIZE = 60000
BATCH_SIZE = 256
# Cria o dataset (from_tensor_slices)
# Randomiza (shuffle)
# Combina elementos consecutivos em lotes (batch)
train_dataset = tf.data.Dataset.from_tensor_slices(train_images)
train_dataset = train_dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE)

# Cria o GERADOR
def make_generator_model():
  model = tf.keras.Sequential()
  model.add(layers.Dense(7*7*256, use_bias=False,
  input_shape=(100,)))
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())
  model.add(layers.Reshape((7, 7, 256)))
  assert model.output_shape == (None, 7, 7, 256)

  # Note: None is the batch size
  model.add(layers.Conv2DTranspose(128, (5, 5),
  strides=(1, 1), padding='same', use_bias=False))
  assert model.output_shape == (None, 7, 7, 128)
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())
  model.add(layers.Conv2DTranspose(64, (5, 5),
  strides=(2, 2), padding='same', use_bias=False))
  assert model.output_shape == (None, 14, 14, 64)
  model.add(layers.BatchNormalization())
  model.add(layers.LeakyReLU())
  model.add(layers.Conv2DTranspose(1, (5, 5),
  strides=(2, 2), padding='same', use_bias=False,
  activation='tanh'))
  assert model.output_shape == (None, 28, 28, 1)
  return model

# Teste do GERADOR, ainda não treinado
generator = make_generator_model()
noise = tf.random.normal([1, 100])
```

```
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0, :, :, 0], cmap='gray')

# Cria o DISCRIMINADOR
def make_discriminator_model():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
  input_shape=[28, 28, 1]))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  model.add(layers.Flatten())
  model.add(layers.Dense(1))
  return model

# Teste do DISCRIMINADOR, ainda não treinado
discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print (decision)

# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
# Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):
  real_loss = cross_entropy(tf.ones_like(real_output), real_output)
  fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
  total_loss = real_loss + fake_loss
  return total_loss
# Perda do GERADOR
def generator_loss(fake_output):
  return cross_entropy(tf.ones_like(fake_output), fake_output)

# Cria os otimizadores para o gerador e discriminador
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)

# Cria checkpoints para salvar modelos ao longo do tempo
# Úteis em tarefas longas, para se recuperar de um desligamento
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
                                 discriminator_optimizer=discriminator_optimizer,
                                 generator=generator,
                                 discriminator=discriminator)
```

```
# Configura o Loop de treinamento
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16
# You will reuse this seed overtime (so it's easier)
# to visualize progress in the animated GIF)
seed = tf.random.normal([num_examples_to_generate, noise_dim])

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
@tf.function
def train_step(images):
  noise = tf.random.normal([BATCH_SIZE, noise_dim])
  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    generated_images = generator(noise, training=True)
    real_output = discriminator(images, training=True)
    fake_output = discriminator(generated_images, training=True)
    gen_loss = generator_loss(fake_output)
    disc_loss = discriminator_loss(real_output, fake_output)
  gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
  gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
  generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
  discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))

# Treinamento completo/laço
def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()
    for image_batch in dataset:
      train_step(image_batch)

    # Produce images for the GIF as you go
    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch + 1, seed)
    # Save the model every 15 epochs
    if (epoch + 1) % 15 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)
    print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-start))
  # Generate after the final epoch
  display.clear_output(wait=True)
  generate_and_save_images(generator, epochs, seed)
```

```
# Gerar e salvar imagens
def generate_and_save_images(model, epoch, test_input):
  # Notice `training` is set to False.
  # This is so all layers run in inference mode (batchnorm).
  predictions = model(test_input, training=False)
  fig = plt.figure(figsize=(4, 4))
  for i in range(predictions.shape[0]):
    plt.subplot(4, 4, i+1)
    plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
    plt.axis('off')
  plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
  plt.show()

# Treinar o modelo e restaurar o último ponto de verificação
train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))

# Criar um GIF
# Display a single image using the epoch number
def display_image(epoch_no):
  return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

display_image(EPOCHS)
anim_file = 'dcgan.gif'
with imageio.get_writer(anim_file, mode='I') as writer:
  filenames = glob.glob('image*.png')
  filenames = sorted(filenames)
  for filename in filenames:
    image = imageio.imread(filename)
    writer.append_data(image)
  image = imageio.imread(filename)
  writer.append_data(image)
import tensorflow_docs.vis.embed as embed
embed.embed_file(anim_file)
```

Tradutor de Textos (Transformer)

https://colab.research.google.com/drive/1KBnhPafLcvwnvqbnCe9umXsaKhs7MPHw?usp=sharing

```
# Instalação de libs
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0
```

```
# Im portaçãod de libs
import collections
import logging
import os
import pathlib
import re
import string
import sys
import time

import numpy as np
import matplotlib .pyplot as plt

import tensorflow_datasets as tfds
import tensorflow_text as text
import tensorflow as tf

logging.getLogger('tensorflow').setLevel(logging.ERROR) # remover warnings

# Carregam ento de dados
examples, metadata = tfds.load(
    'ted_hrlr_translate/pt_to_en',
    with_info = True,
    as_supervised = True
)
train_examples, val_examples = examples['train'], examples['validation']

# Verificar o dataset
for pt_examples, en_examples in train_examples.batch(3).take(1):
  for pt in pt_examples.numpy():
    print(pt.decode('utf-8'))
  print()
  for en in en_examples.numpy():
    print(en.decode('utf-8'))

# Tokenização e Destokenização
model_name = "ted_hrlr_translate_pt_en_converter"

tf.keras.utils.get_file(
    f"{model_name}.zip",

f"https://storage.googleapis.com/download.tensorflow.org/models/{model_name}.zip",
    cache_dir='.',
    cache_subdir='',
    extract=True
)

tokenizers = tf.saved_model.load(model_name)


# PIPELINE DE ENTRADA
# Codificar/tokenizar lotes de texto puro
def tokenize_pairs(pt, en):
  pt = tokenizers.pt.tokenize(pt)
  # Converte ragged (irregular, tam variável) para dense
  # Faz padding com zeros.
  pt = pt.to_tensor()
  en = tokenizers.en.tokenize(en)
  # ragged -> dense
  en = en.to_tensor()
  return pt, en

# Pipeline simpes: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64
def make_batches(ds):
  return (
```

```
    ds
    .cache()
    .shuffle(BUFFER_SIZE)
    .batch(BATCH_SIZE)
    .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
    .prefetch(tf.data.AUTOTUNE))
train_batches = make_batches(train_examples)
val_batches = make_batches(val_examples)

# CODIFICAÇÃO POSICIONAL
def get_angles(pos, i, d_model):
  angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
  return pos * angle_rates

def positional_encoding(position, d_model):
  angle_rads = get_angles(np.arange(position)[:, np.newaxis],
                          np.arange(d_model)[np.newaxis, :],
                          d_model)
  # sin em índices pares no array; 2i
  angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
  # cos em índices ímpares no array; 2i+1
  angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
  # newaxis, aumenta a dimensão [] -> [ [] ]
  pos_encoding = angle_rads[np.newaxis, ...]
  return tf.cast(pos_encoding, dtype=tf.float32)

# CODIFICAÇÃO POSICIONAL
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

# Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))
plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()

# Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
  seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
  # add extra dimensions to add the padding
  # to the attention logits.
  return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_l)

# Máscara futura, usada no decoder
def create_look_ahead_mask(size):
  # zera o triângulo inferior
  mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
  return mask # (seq_len, seq_len)

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
  # Q * K ^ T
  matmul_qk = tf.matmul(q, k, transpose_b=True) # (..., seq_len_q, seq_len_k)

  # converte matmul_qk para float32
  dk = tf.cast(tf.shape(k)[-1], tf.float32)

  # divide por sqrt(d_k)
  scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

  # Soma a máscara, e os valores faltantes serão um número próximo a -inf
  if mask is not None:
```

```
    scaled_attention_logits += (mask * -1e9)

  # softmax normaliza os dados, soman 1. // (..., seq_len_q, seq_len_k)
  attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
  output = tf.matmul(attention_weights, v) # (..., seq_len_q, depth_v)
  return output, attention_weights

# Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads):
    super(MultiHeadAttention, self).__init__()
    self.num_heads = num_heads
    self.d_model = d_model

    assert d_model % self.num_heads == 0

    self.depth = d_model // self.num_heads

    self.wq = tf.keras.layers.Dense(d_model)
    self.wk = tf.keras.layers.Dense(d_model)
    self.wv = tf.keras.layers.Dense(d_model)

    self.dense = tf.keras.layers.Dense(d_model)

  def split_heads(self, x, batch_size):
    """Separa a última dimensão em (num_heads, depth).
    Transpõe o resultado para o shape (batch_size, num_heads, seq_len, depth)
     """
    x = tf.reshape(x, (batch_size, -1, self.num_heads, self.depth))
    return tf.transpose(x, perm=[0, 2, 1, 3])

  def call(self, v, k, q, mask):
    batch_size = tf.shape(q)[0]

    q = self.wq(q) # (batch_size, seq_len, d_model)
    k = self.wk(k) # (batch_size, seq_len, d_model)
    v = self.wv(v) # (batch_size, seq_len, d_model)

    q = self.split_heads(q, batch_size) # (batch_size, num_heads, seq_len_q, depth)
    k = self.split_heads(k, batch_size) # (batch_size, num_heads, seq_len_k, depth)
    v = self.split_heads(v, batch_size) # (batch_size, num_heads, seq_len_v, depth)

    # Calcula a atenção para cada cabeça (de forma matricial)
    # scaled_attention.shape == (batch_size, num_heads, seq_len_q, depth)
    # attention_weights.shape == (batch_size, num_heads, seq_len_q, seq_len_k)
    scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v,
mask)

    # Troca a dimensão 2 com 1, para acertar o num_heads
    # (batch_size, seq_len_q, num_heads, depth)
    scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])

    # Concatena os valores em: (batch_size, seq_len_q, d_model)
    concat_attention = tf.reshape(scaled_attention, (batch_size, -1, self.d_model))

    output = self.dense(concat_attention) # (batch_size, seq_len_q, d_model)

    return output, attention_weights

# Rede feed forward
def point_wise_feed_forward_network(d_model, dff):
  return tf.keras.Sequential([
  tf.keras.layers.Dense(dff, activation='relu'), # (batch_size, seq_len, dff)
  tf.keras.layers.Dense(d_model) # (batch_size, seq_len, d_model)
])

# Camada do codificador
class EncoderLayer(tf.keras.layers.Layer):
```

```
  def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(EncoderLayer, self).__init__()

    self.mha = MultiHeadAttention(d_model, num_heads)
    self.ffn = point_wise_feed_forward_network(d_model, dff)

    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)

  def call(self, x, training, mask):
    attn_output, _ = self.mha(x, x, x, mask) # (batch_size, input_seq_len, d_model)
    attn_output = self.dropout1(attn_output, training=training)
    out1 = self.layernorm1(x + attn_output) # (batch_size, input_seq_len, d_model)

    ffn_output = self.ffn(out1) # (batch_size, input_seq_len, d_model)
    ffn_output = self.dropout2(ffn_output, training=training)
    out2 = self.layernorm2(out1 + ffn_output) # (batch_size, input_seq_len,
d_model)

    return out2

# Camada do decodificador
class DecoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(DecoderLayer, self).__init__()

    self.mha1 = MultiHeadAttention(d_model, num_heads)
    self.mha2 = MultiHeadAttention(d_model, num_heads)

    self.ffn = point_wise_feed_forward_network(d_model, dff)

    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)

    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)
    self.dropout3 = tf.keras.layers.Dropout(rate)

  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    # enc_output.shape == (batch_size, input_seq_len, d_model)
    # (batch_size, target_seq_len, d_model)
    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layernorm1(attn1 + x)

    # (batch_size, target_seq_len, d_model)
    attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1,
padding_mask)
    attn2 = self.dropout2(attn2, training=training)
    out2 = self.layernorm2(attn2 + out1) # (batch_size, target_seq_len, d_model)

    ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
    ffn_output = self.dropout3(ffn_output, training=training)
    out3 = self.layernorm3(ffn_output + out2) # (batch_size, target_seq_len,
d_model)

    return out3, attn_weights_block1, attn_weights_block2

# Encoder completo
class Encoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff,
               input_vocab_size, maximum_position_encoding, rate=0.1):
    super(Encoder, self).__init__()
    self.d_model = d_model
```

```
    self.num_layers = num_layers
    self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding,
self.d_model)
    self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate) for _ in
range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)

  def call(self, x, training, mask):
    seq_len = tf.shape(x)[1]
    # adding embedding and position encoding.
    x = self.embedding(x) # (batch_size, input_seq_len, d_model)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x = self.enc_layers[i](x, training, mask)
    return x # (batch_size, input_seq_len, d_model)

# Decoder completo
class Decoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
               maximum_position_encoding, rate=0.1):
    super(Decoder, self).__init__()
    self.d_model = d_model
    self.num_layers = num_layers
    self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding, d_model)
    self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate)
                       for _ in range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)

  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    seq_len = tf.shape(x)[1]
    attention_weights = {}
    x = self.embedding(x) # (batch_size, target_seq_len, d_model)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x, block1, block2 = self.dec_layers[i](x, enc_output, training,
                                             look_ahead_mask, padding_mask)
      attention_weights[f'decoder_layer{i+1}_block1'] = block1
      attention_weights[f'decoder_layer{i+1}_block2'] = block2
    # x.shape == (batch_size, target_seq_len, d_model)
    return x, attention_weights

# Transformer completo
class Transformer(tf.keras.Model):
  def __init__(self, num_layers, d_model, num_heads, dff, input_vocab_size,
               target_vocab_size, pe_input, pe_target, rate=0.1):
    super().__init__()
    self.encoder = Encoder(num_layers, d_model, num_heads, dff, input_vocab_size,
                           pe_input, rate)
    self.decoder = Decoder(num_layers, d_model, num_heads, dff, target_vocab_size,
                           pe_target, rate)
    self.final_layer = tf.keras.layers.Dense(target_vocab_size)

  def call(self, inputs, training):
    # Keras models prefer if you pass all your inputs in the first argument
    inp, tar = inputs
    enc_padding_mask, look_ahead_mask, dec_padding_mask = self.create_masks(inp,
tar)
    # (batch_size, inp_seq_len, d_model)
    enc_output = self.encoder(inp, training, enc_padding_mask)
    # dec_output.shape == (batch_size, tar_seq_len, d_model)
    dec_output, attention_weights = self.decoder(tar, enc_output,
                                                 training, look_ahead_mask,
```

```
dec_padding_mask)
    # (batch_size, tar_seq_len, target_vocab_size)
    final_output = self.final_layer(dec_output)
    return final_output, attention_weights

  def create_masks(self, inp, tar):
    # Encoder padding mask
    enc_padding_mask = create_padding_mask(inp)
    # Used in the 2nd attention block in the decoder.
    # This padding mask is used to mask the encoder outputs.
    dec_padding_mask = create_padding_mask(inp)
    # Used in the 1st attention block in the decoder.
    # It is used to pad and mask future tokens in the input received by
    # the decoder.
    look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
    dec_target_padding_mask = create_padding_mask(tar)
    look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
    return enc_padding_mask, look_ahead_mask, dec_padding_mask

# Hiperparâmetros
num_layers = 4
d_model = 128
dff = 512
num_heads = 8
dropout_rate = 0.1

# Otimizador
class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
  def __init__(self, d_model, warmup_steps=4000):
    super(CustomSchedule, self).__init__()
    self.d_model = d_model
    self.d_model = tf.cast(self.d_model, tf.float32)
    self.warmup_steps = warmup_steps

  def __call__(self, step):
    step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
    arg1 = tf.math.rsqrt(step)
    arg2 = step * (self.warmup_steps ** -1.5)
    return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

learning_rate = CustomSchedule(d_model)
optimizer = tf.keras.optimizers.Adam(learning_rate, beta_1=0.9, beta_2=0.98,
epsilon=1e-9)

# Função de perda e métricas
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

def accuracy_function(real, pred):
  accuracies = tf.equal(real, tf.argmax(pred, axis=2))
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  accuracies = tf.math.logical_and(mask, accuracies)
  accuracies = tf.cast(accuracies, dtype=tf.float32)
  mask = tf.cast(mask, dtype=tf.float32)
  return tf.reduce_sum(accuracies)/tf.reduce_sum(mask)

train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.Mean(name='train_accuracy')

# Treinamento
transformer = Transformer(
    num_layers=num_layers,
```

```
    d_model=d_model,
    num_heads=num_heads,
    dff=dff,
    input_vocab_size=tokenizers.pt.get_vocab_size().numpy(),
    target_vocab_size=tokenizers.en.get_vocab_size().numpy(),
    pe_input=1000,
    pe_target=1000,
    rate=dropout_rate)

# Checkpoint
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer,
                           optimizer=optimizer)
ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)

# if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
  ckpt.restore(ckpt_manager.latest_checkpoint)
  print('Latest checkpoint restored!!')

# Processo de treinamento
EPOCHS = 30
train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]
@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
  tar_inp = tar[:, :-1]
  tar_real = tar[:, 1:]
  with tf.GradientTape() as tape:
    predictions, _ = transformer([inp, tar_inp],
                                 training = True)
    loss = loss_function(tar_real, predictions)
  gradients = tape.gradient(loss, transformer.trainable_variables)
  optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))
  train_loss(loss)
  train_accuracy(accuracy_function(tar_real, predictions))

for epoch in range(EPOCHS):
  start = time.time()
  train_loss.reset_state()
  train_accuracy.reset_state()
  # inp -> portuguese, tar -> english
  for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)
    if batch % 50 == 0:
      print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
Accuracy {train_accuracy.result():.4f}')

  if (epoch + 1) % 5 == 0:
    ckpt_save_path = ckpt_manager.save()
    print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')

  print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
  print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')

# Tradutor
class Translator(tf.Module):
  def __init__(self, tokenizers, transformer):
    self.tokenizers = tokenizers
    self.transformer = transformer

  def __call__(self, sentence, max_length=20):
    # input sentence is portuguese, hence adding the start and end token
    assert isinstance(sentence, tf.Tensor)
    if len(sentence.shape) == 0:
```

```
      sentence = sentence[tf.newaxis]
    sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
    encoder_input = sentence
    # as the target is english, the first token to the transformer should be the
    # english start token.
    start_end = self.tokenizers.en.tokenize([''])[0]
    start = start_end[0][tf.newaxis]
    end = start_end[1][tf.newaxis]
    output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
    output_array = output_array.write(0, start)

    for i in tf.range(max_length):
      output = tf.transpose(output_array.stack())
      predictions, _ = self.transformer([encoder_input, output], training=False)
      predictions = predictions[:, -1:, :] # (batch_size, 1, vocab_size)
      predicted_id = tf.argmax(predictions, axis=-1)
      output_array = output_array.write(i+1, predicted_id[0])
      if predicted_id == end:
        break
    output = tf.transpose(output_array.stack())
    # output.shape (1, tokens)
    text = tokenizers.en.detokenize(output)[0]
    tokens = tokenizers.en.lookup(output)[0]
    _, attention_weights = self.transformer([encoder_input, output[:,:-1]],
training=False)
    return text, tokens, attention_weights

# Traduzindo
translator = Translator(tokenizers, transformer)
sentence = "Eu li sobre triceratops na enciclopédia."
translated_text, translated_tokens, attention_weights =
translator(tf.constant(sentence))
print(f'{"Prediction":15s}: {translated_text}')
```

APÊNDICE I - BIG DATA

A – ENUNCIADO

Enviar um arquivo PDF contendo uma descrição breve (2 páginas) sobre a implementação de uma aplicação ou estudo de caso envolvendo Big Data e suas ferramentas (NoSQL e NewSQL). Caracterize os dados e Vs envolvidos, além da modelagem necessária dependendo dos modelos de dados empregados.

B – RESOLUÇÃO

Análise e Otimização de Produção Eletrônica com Big Data, NoSQL e NewSQL

1. Introdução

A indústria eletrônica moderna gera grandes quantidades de dados em cada etapa do processo produtivo: desde a gravação de firmware em placas até o teste funcional de produtos finais. Linhas de montagem automatizadas, máquinas CNC, sistemas de inspeção óptica (AOI) e instrumentos de teste elétrico produzem dados contínuos sobre desempenho, falhas e parâmetros de qualidade. Tradicionalmente, bancos de dados relacionais isolados não conseguem lidar com o volume, a variedade e a velocidade desses dados. Este estudo descreve a implementação de um ambiente de Big Data para monitorar e otimizar a produção eletrônica, utilizando NoSQL e NewSQL para armazenar e processar informações em tempo quase real, suportando análises avançadas e decisões baseadas em dados.

2. Caracterização dos Dados e Vs do Big Data

• **Volume:** Cada estação de teste gera milhares de registros por turno — resultados de medições elétricas, tempos de ciclo, imagens de inspeção, dados de firmware e logs de equipamentos. Em uma fábrica com dezenas de linhas, o volume alcança terabytes mensais.

• **Velocidade:** Sensores industriais, equipamentos CNC e sistemas de supervisão (SCADA) enviam dados em fluxo contínuo, exigindo ingestão em tempo real para detecção rápida de falhas.

• **Variedade:**

- Estruturados: parâmetros de teste (tensões, correntes, tempos de execução, status de aprovação/reprovação).
- Semiestruturados: logs de sistemas embarcados, arquivos JSON de configuração.
- Não estruturados: imagens de inspeção óptica, relatórios PDF, vídeos curtos de defeitos.

• **Veracidade:** dados podem conter ruídos de leitura ou falhas de sensores, exigindo filtros e validações.

• **Valor:** insights extraídos ajudam a reduzir tempo de parada, melhorar rendimento e ajustar processos.

3. Arquitetura da Solução

INGESTÃO E PROCESSAMENTO

- Apache Kafka para ingestão de dados em streaming, integrando bancadas de teste, máquinas CNC e sistemas ERP/MES.
- Apache Spark Streaming para processamento em tempo real, agregando métricas de qualidade e produtividade.

ARMAZENAMENTO E MODELAGEM

- NoSQL – MongoDB: armazena logs de teste e dados semiestruturados de configuração (JSON). Ideal para flexibilidade e rápido acesso às informações de cada lote ou placa individual.
- NoSQL – Cassandra: guarda grandes volumes de séries temporais com alta taxa de escrita e consulta eficiente para KPIs de produção.
- NewSQL – CockroachDB: registra dados críticos e estruturados, como ordens de produção, rastreabilidade de placas e transações ACID relacionadas a lotes e controle de estoque.
- Data Lake (S3 ou Azure Data Lake): armazena imagens e arquivos grandes para análise posterior.

VISUALIZAÇÃO E INTELIGÊNCIA

- Grafana e Kibana para dashboards em tempo real.
- Modelos de Machine Learning (usando Spark MLlib ou TensorFlow) para prever falhas e ajustar parâmetros de teste automaticamente.

4. Modelagem de Dados

- Time-Series (Cassandra): cada linha representa medições de uma placa com timestamp, permitindo análise de performance e detecção de padrões de falhas recorrentes.
- Documentos (MongoDB): armazenam logs de execução e arquivos de configuração dos testes, com campos flexíveis para diferentes tipos de produtos.
- Relacional Distribuído (CockroachDB): gerencia relacionamento entre ordens de produção, operadores, equipamentos e resultados críticos, garantindo consistência e transações seguras mesmo com escalabilidade horizontal.
- Imagens e Arquivos Brutos (Data Lake): dados de AOI e inspeções visuais ficam disponíveis para análises assistidas por IA no futuro.

5. Benefícios Obtidos

- Monitoramento em tempo real: dashboards permitem detectar rapidamente falhas de equipamentos ou lotes com defeito.

- Otimização de processos: análise histórica orienta ajustes em parâmetros de teste e firmware para reduzir reprovações.
- Escalabilidade: a arquitetura suporta crescimento de linhas de produção sem perda de desempenho.
- Integração de dados heterogêneos: unifica informações de sensores, máquinas e sistemas de gestão.
- Base para manutenção preditiva: modelos preveem falhas de equipamentos, evitando paradas não planejadas.

6. Conclusão

O uso combinado de ferramentas NoSQL (MongoDB e Cassandra) e NewSQL (CockroachDB) cria uma base sólida para gerenciar dados de manufatura eletrônica em larga escala. A solução atende aos principais Vs do Big Data e suporta desde análises operacionais imediatas até estudos de longo prazo para melhoria contínua. Essa abordagem pode ser aplicada em indústrias que buscam eficiência, rastreabilidade e redução de custos por meio da análise inteligente de grandes volumes de dados.

Palavras-chave: Big Data. NoSQL. NewSQL. Manufatura Eletrônica. Cassandra. MongoDB. CockroachDB.

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

Os bancos de imagens fornecidos são conjuntos de imagens de 250x250 pixels de imuno-histoquímica (biópsia) de câncer de mama. No total são 4 classes (0, 1+, 2+ e 3+) que estão divididas em diretórios. O objetivo é classificar as imagens nas categorias correspondentes. Uma base de imagens será utilizada para o treinamento e outra para o teste do treino.

As imagens fornecidas são recortes de uma imagem maior do tipo WSI *(Whole Slide Imaging*) disponibilizada pela Universidade de Warwick (link). A nomenclatura das imagens segue o padrão XX_HER_YYYY.png, onde XX é o número do paciente e YYYY é o número da imagem recortada. Separe a base de treino em 80% para treino e 20% para validação. **Separe por pacientes (XX), não utilize a separação randômica! Pois, imagens do mesmo paciente não podem estar na base de treino e de validação, pois isso pode gerar um viés.** No caso da CNN VGG16 remova a última camada de classificação e armazene os valores da penúltima camada como um vetor de características. Após o treinamento, os modelos treinados devem ser validados na base de teste.

Tarefas:

a) Carregue a base de dados de **Treino.**
b) Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes).
c) Extraia características utilizando LBP e a CNN VGG16 (gerando um csv para cada extrator).
d) Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.
e) Carregue a base de **Teste** e execute a tarefa 3 nesta base.
f) Aplique os modelos treinados nos dados de treino
g) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.
h) Indique qual modelo dá o melhor o resultado e a métrica utilizada

2) Redes Neurais

Utilize as duas bases do exercício anterior para treinar as Redes Neurais Convolucionais VGG16 e a Resnet50. Utilize os pesos pré-treinados (*Transfer Learning*), refaça as camadas *Fully Connected* para o problema de 4 classes. Compare os treinos de 15 épocas com e sem *Data Augmentation*. Tanto a VGG16 quanto a Resnet50 têm como camada de entrada uma imagem 224x224x3, ou seja, uma imagem de 224x224 pixels coloridos (3 canais de cores). Portanto, será necessário fazer uma transformação de 250x250x3 para 224x224x3. Ao fazer o *Data Augmentation* **cuidado** para não alterar demais as cores das imagens e atrapalhar na classificação.

Tarefas:

a) Utilize a base de dados de **Treino** já separadas em treino e validação do exercício anterior
b) Treine modelos VGG16 e Resnet50 adaptadas com e sem *Data Augmentation*
c) Aplique os modelos treinados nas imagens da base de **Teste**
d) Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.
e) Indique qual modelo dá o melhor o resultado e a métrica utilizada

B – RESOLUÇÃO

```
# ============================================================
# Warwick 4 classes – Pipeline completo (a-h)
# ============================================================

# ----------------------------
# 0) MONTAR DRIVE E DEFINIR CAMINHOS
# ----------------------------
from google.colab import drive
drive.mount('/content/drive')

BASE = "/content/drive/MyDrive/Visão Comp. UFPR"
TRAIN_ZIP = f"{BASE}/Train_Warwick.zip"
TEST_ZIP  = f"{BASE}/Test_Warwick.zip"

TRAIN_DIR = "/content/train_warwick/Train_4cls_amostra"
TEST_DIR  = "/content/test_warwick/Test_4cl_amostra"

print("ZIPs:\n", TRAIN_ZIP, "\n", TEST_ZIP)
print("Dirs previstos:\n", TRAIN_DIR, "\n", TEST_DIR)

# ----------------------------
# 1) EXTRAIR ZIPS
# ----------------------------
import os, zipfile

def extract_zip(zip_path, target_dir):
    os.makedirs(os.path.dirname(target_dir), exist_ok=True)
    if not os.path.exists(target_dir) or not
any(os.scandir(os.path.dirname(target_dir))):
        with zipfile.ZipFile(zip_path, 'r') as z:
            z.extractall(os.path.dirname(target_dir))  # extrai um nível acima
        print("Extraído:", zip_path)
    else:
```

```
        print("Já existia conteúdo em:", os.path.dirname(target_dir))

extract_zip(TRAIN_ZIP, TRAIN_DIR)
extract_zip(TEST_ZIP,  TEST_DIR)

print("Subpastas treino:", os.listdir(TRAIN_DIR))
print("Subpastas teste :", os.listdir(TEST_DIR))

# ----------------------------
# 2) IMPORTS GERAIS
# ----------------------------
import json, math, random, warnings
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import seaborn as sns

from skimage.feature import local_binary_pattern

from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.exceptions import ConvergenceWarning

import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras import Model

SEED = 42
random.seed(SEED); np.random.seed(SEED); tf.random.set_seed(SEED)
warnings.filterwarnings("ignore", category=ConvergenceWarning)

plt.rcParams["figure.figsize"] = (5.5, 4.5)
sns.set(style="whitegrid")

# ----------------------------
# 3) DESCOBRIR CLASSES E LISTAR REGISTROS
# ----------------------------
def discover_classes(root_dir):
    classes = sorted([d for d in os.listdir(root_dir) if
os.path.isdir(os.path.join(root_dir, d))])
```

```
    label_map = {cls:i for i, cls in enumerate(classes)}  #
{'0':0,'1':1,'2':2,'3':3}
    return classes, label_map

CLASSES, LABEL_MAP = discover_classes(TRAIN_DIR)
print("CLASSES:", CLASSES)
print("LABEL_MAP:", LABEL_MAP)

def patient_id_from_name(fname:str):
    # padrão: XX_HER_YYYY.png -> pega "XX"
    return fname.split('_')[0]

def list_records(root_dir, label_map):
    recs = []
    for cls_str, lbl in label_map.items():
        cls_dir = os.path.join(root_dir, cls_str)
        for fname in os.listdir(cls_dir):
            fpath = os.path.join(cls_dir, fname)
            if not os.path.isfile(fpath):
                continue
            pid = patient_id_from_name(fname)
            recs.append((fpath, cls_str, lbl, pid))
    return recs

train_records = list_records(TRAIN_DIR, LABEL_MAP)
test_records  = list_records(TEST_DIR,  LABEL_MAP)
print("Total imagens - Treino:", len(train_records), "| Teste:", len(test_records))

# ----------------------------
# 4) SPLIT 80/20 POR PACIENTE (GLOBAL)  [Tarefa (b)]
# ----------------------------
def split_by_patient_global(records, val_ratio=0.2, seed=SEED):
    patients = sorted({r[3] for r in records})
    rng = np.random.RandomState(seed)
    n_val = max(1, int(np.ceil(len(patients) * val_ratio)))
    val_pids = set(rng.choice(patients, size=n_val, replace=False))
    train_split = [r for r in records if r[3] not in val_pids]
    val_split   = [r for r in records if r[3] in  val_pids]
    assert set([r[3] for r in train_split]).isdisjoint(set([r[3] for r in
val_split])), "Vazamento de paciente!"
    return train_split, val_split

train_split, val_split = split_by_patient_global(train_records, val_ratio=0.2,
seed=SEED)
print("Split -> treino:", len(train_split), " | val:", len(val_split))
```

```
# ----------------------------
# 5) EXTRAÇÃO DE CARACTERÍSTICAS – LBP  [Tarefa (c)]
# ----------------------------
P, R = 8, 1
LBP_BINS = P + 2  # 10 para RIU

def lbp_hist(gray):
    lbp = local_binary_pattern(gray, P, R, method='uniform')
    hist, _ = np.histogram(lbp.ravel(), bins=LBP_BINS, range=(0, LBP_BINS),
density=True)
    return hist.astype(np.float32)

def extract_lbp_to_df(records):
    rows = []
    for path, cls_str, lbl, pid in records:
        img = cv2.imread(path, cv2.IMREAD_GRAYSCALE)
        if img is None:
            continue
        feat = lbp_hist(img)
        row = {"path": path, "label": lbl, "class": cls_str, "patient": pid}
        row.update({f"lbp_{i}": float(feat[i]) for i in range(LBP_BINS)})
        rows.append(row)
    return pd.DataFrame(rows)

os.makedirs("/content/Features", exist_ok=True)
df_lbp_train = extract_lbp_to_df(train_split)
df_lbp_val   = extract_lbp_to_df(val_split)
df_lbp_test  = extract_lbp_to_df(test_records)

df_lbp_train.to_csv("/content/Features/lbp_train.csv", index=False)
df_lbp_val.to_csv(  "/content/Features/lbp_val.csv",   index=False)
df_lbp_test.to_csv( "/content/Features/lbp_test.csv",  index=False)

print("LBP CSVs:", df_lbp_train.shape, df_lbp_val.shape, df_lbp_test.shape)

def load_xy_from_lbp_csv(csv_path):
    df = pd.read_csv(csv_path)
    cols = [c for c in df.columns if c.startswith("lbp_")]
    X = df[cols].values.astype(np.float32)
    y = df["label"].values.astype(int)
    return X, y, df

Xtr, ytr, df_tr = load_xy_from_lbp_csv("/content/Features/lbp_train.csv")
Xva, yva, df_va = load_xy_from_lbp_csv("/content/Features/lbp_val.csv")
Xte, yte, df_te = load_xy_from_lbp_csv("/content/Features/lbp_test.csv")
print("Shapes LBP ->", Xtr.shape, Xva.shape, Xte.shape)
```

```
# ----------------------------
# 6) EXTRAÇÃO DE CARACTERÍSTICAS - VGG16 (fc2)  [Tarefa (c)]
# ----------------------------
DO_VGG = True

def vgg_fc2_batch(records, batch_size=32):
    if len(records) == 0:
        return pd.DataFrame([])
    # carrega VGG16 (com topo) e "corta" na penúltima (fc2)
    base = VGG16(weights="imagenet", include_top=True)
    extractor = Model(inputs=base.input, outputs=base.get_layer('fc2').output)  #
(4096,)

    paths   = [r[0] for r in records]
    labels  = [r[2] for r in records]
    classes = [r[1] for r in records]
    pids    = [r[3] for r in records]

    feats_list = []
    for i in range(0, len(paths), batch_size):
        batch_paths = paths[i:i+batch_size]
        X = np.zeros((len(batch_paths), 224, 224, 3), dtype=np.float32)
        for j, p in enumerate(batch_paths):
            img = load_img(p, target_size=(224,224))
            X[j] = img_to_array(img)
        X = preprocess_input(X)
        feats = extractor.predict(X, batch_size=batch_size, verbose=0)
        feats_list.append(feats)

    feats = np.vstack(feats_list)  # (N,4096)
    df = pd.DataFrame(feats, columns=[f"vgg_{k}" for k in range(feats.shape[1])])
    df.insert(0, "path", paths)
    df.insert(1, "label", labels)
    df.insert(2, "class", classes)
    df.insert(3, "patient", pids)
    return df

if DO_VGG:
    df_vgg_train = vgg_fc2_batch(train_split)
    df_vgg_val   = vgg_fc2_batch(val_split)
    df_vgg_test  = vgg_fc2_batch(test_records)

    df_vgg_train.to_csv("/content/Features/vgg_train.csv", index=False)
    df_vgg_val.to_csv(  "/content/Features/vgg_val.csv",   index=False)
    df_vgg_test.to_csv( "/content/Features/vgg_test.csv",  index=False)
```

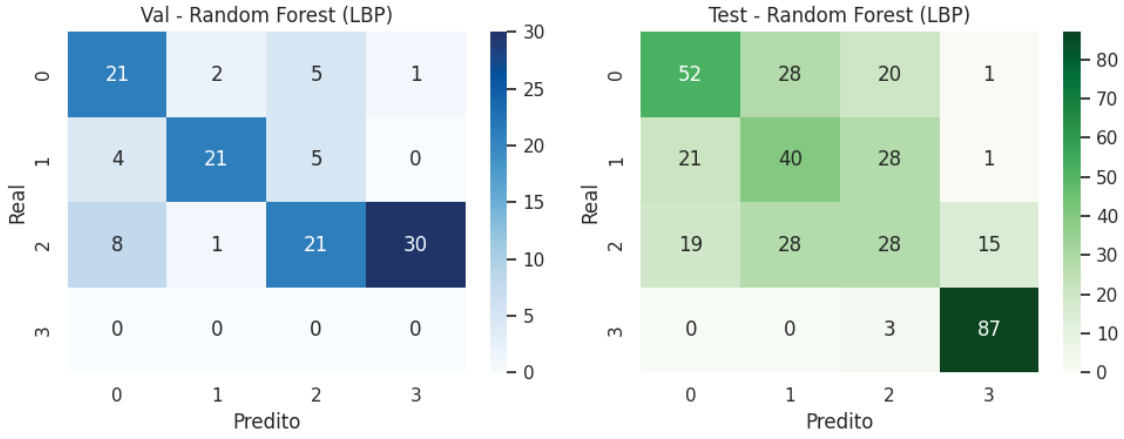
```
    print("VGG CSVs:", df_vgg_train.shape, df_vgg_val.shape, df_vgg_test.shape)

    def load_xy_from_vgg_csv(csv_path):
        df = pd.read_csv(csv_path)
        cols = [c for c in df.columns if c.startswith("vgg_")]
        X = df[cols].values.astype(np.float32)
        y = df["label"].values.astype(int)
        return X, y, df

    Xtr_v, ytr_v, _ = load_xy_from_vgg_csv("/content/Features/vgg_train.csv")
    Xva_v, yva_v, _ = load_xy_from_vgg_csv("/content/Features/vgg_val.csv")
    Xte_v, yte_v, _ = load_xy_from_vgg_csv("/content/Features/vgg_test.csv")
    print("Shapes VGG ->", Xtr_v.shape, Xva_v.shape, Xte_v.shape)

# ----------------------------
# 7) MÉTRICAS, MATRIZES E AVALIAÇÃO (Val e Test)  [Tarefas (f) e (g)]
# ----------------------------
def metrics_from_cm(cm):
    # Sensibilidade e Especificidade médias (macro)
    K = cm.shape[0]
    sens, spec = [], []
    for i in range(K):
        TP = cm[i,i]
        FN = cm[i,:].sum() - TP
        FP = cm[:,i].sum() - TP
        TN = cm.sum() - (TP+FN+FP)
        sens.append(TP/(TP+FN) if (TP+FN)>0 else 0.0)
        spec.append(TN/(TN+FP) if (TN+FP)>0 else 0.0)
    return float(np.mean(sens)), float(np.mean(spec))

results = []

def eval_and_report(model, Xv, yv, Xt, yt, name):
    yv_pred = model.predict(Xv)
    yt_pred = model.predict(Xt)

    cm_v = confusion_matrix(yv, yv_pred)
    cm_t = confusion_matrix(yt, yt_pred)

    sens_v, spec_v = metrics_from_cm(cm_v)
    sens_t, spec_t = metrics_from_cm(cm_t)

    f1_v = f1_score(yv, yv_pred, average='weighted')
    f1_t = f1_score(yt, yt_pred, average='weighted')
```

```
    acc_v = accuracy_score(yv, yv_pred)
    acc_t = accuracy_score(yt, yt_pred)

    print(f"\n=== {name} ===")
    print(f"[VAL]  Acc: {acc_v*100:.2f}% | F1: {f1_v*100:.2f}% | Sens:
{sens_v*100:.2f}% | Esp: {spec_v*100:.2f}%")
    print(f"[TEST] Acc: {acc_t*100:.2f}% | F1: {f1_t*100:.2f}% | Sens:
{sens_t*100:.2f}% | Esp: {spec_t*100:.2f}%")

    fig, ax = plt.subplots(1, 2, figsize=(10,4))
    sns.heatmap(cm_v, annot=True, fmt='d', cmap='Blues', ax=ax[0])
    ax[0].set_title(f"Val - {name}")
    sns.heatmap(cm_t, annot=True, fmt='d', cmap='Greens', ax=ax[1])
    ax[1].set_title(f"Test - {name}")
    for a in ax:
        a.set_xlabel("Predito"); a.set_ylabel("Real")
    plt.tight_layout(); plt.show()

    results.append({
        "modelo": name,
        "VAL_acc": acc_v, "VAL_f1": f1_v, "VAL_sens": sens_v, "VAL_esp": spec_v,
        "TEST_acc": acc_t, "TEST_f1": f1_t, "TEST_sens": sens_t, "TEST_esp": spec_t
    })

# ----------------------------
# 8) TREINOS E AVALIAÇÕES - LBP
# ----------------------------
# RF (não precisa escalar)
rf_lbp = RandomForestClassifier(n_estimators=300, random_state=SEED)
rf_lbp.fit(Xtr, ytr)
eval_and_report(rf_lbp, Xva, yva, Xte, yte, "Random Forest (LBP)")

# SVM Linear (escala ajuda)
svm_lbp = make_pipeline(StandardScaler(), SVC(kernel='linear', random_state=SEED))
svm_lbp.fit(Xtr, ytr)
eval_and_report(svm_lbp, Xva, yva, Xte, yte, "SVM Linear (LBP)")

# MLP (escala + early stopping)
mlp_lbp = make_pipeline(
    StandardScaler(),
    MLPClassifier(hidden_layer_sizes=(128,),
                  activation='relu',
                  early_stopping=True, n_iter_no_change=10,
                  max_iter=500, random_state=SEED)
)
mlp_lbp.fit(Xtr, ytr)
```

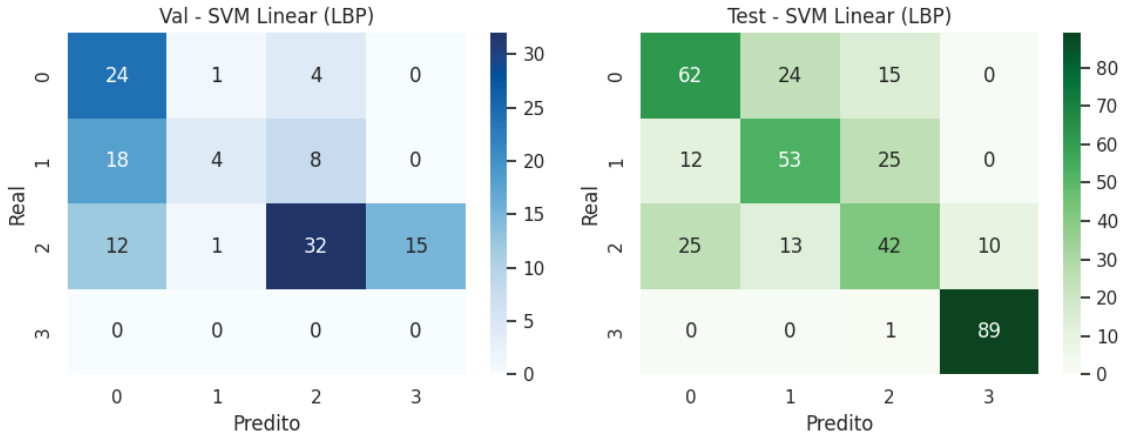
```
eval_and_report(mlp_lbp, Xva, yva, Xte, yte, "MLP (LBP)")

# ----------------------------
# 9) TREINOS E AVALIAÇÕES - VGG16 (fc2)  [opcional, mas recomendado]
# ----------------------------
if DO_VGG:
    rf_vgg = RandomForestClassifier(n_estimators=300, random_state=SEED)
    rf_vgg.fit(Xtr_v, ytr_v)
    eval_and_report(rf_vgg, Xva_v, yva_v, Xte_v, yte_v, "Random Forest (VGG16-
fc2)")

    svm_vgg = make_pipeline(StandardScaler(), SVC(kernel='linear',
random_state=SEED))
    svm_vgg.fit(Xtr_v, ytr_v)
    eval_and_report(svm_vgg, Xva_v, yva_v, Xte_v, yte_v, "SVM Linear (VGG16-fc2)")

    mlp_vgg = make_pipeline(
        StandardScaler(),
        MLPClassifier(hidden_layer_sizes=(256,),
                      activation='relu',
                      early_stopping=True, n_iter_no_change=10,
                      max_iter=500, random_state=SEED)
    )
    mlp_vgg.fit(Xtr_v, ytr_v)
    eval_and_report(mlp_vgg, Xva_v, yva_v, Xte_v, yte_v, "MLP (VGG16-fc2)")

# ----------------------------
# 10) QUADRO-RESUMO E "MELHOR MODELO"  [Tarefa (h)]
# ----------------------------
df_results = pd.DataFrame(results)
display(df_results.sort_values("TEST_f1", ascending=False))

best = df_results.sort_values("TEST_f1", ascending=False).iloc[0]
print("\n>>> Melhor modelo (pela métrica TEST_F1):")
print(f"{best['modelo']} | Test F1={best['TEST_f1']*100:.2f}% | "
      f"Acc={best['TEST_acc']*100:.2f}% | Sens={best['TEST_sens']*100:.2f}% |
Esp={best['TEST_esp']*100:.2f}%")
```

```
Mounted at /content/drive
ZIPs:
 /content/drive/MyDrive/Visão Comp. UFPR/Train_Warwick.zip
 /content/drive/MyDrive/Visão Comp. UFPR/Test_Warwick.zip
Dirs previstos:
 /content/train_warwick/Train_4cls_amostra
 /content/test_warwick/Test_4cl_amostra
Extraído: /content/drive/MyDrive/Visão Comp. UFPR/Train_Warwick.zip
```

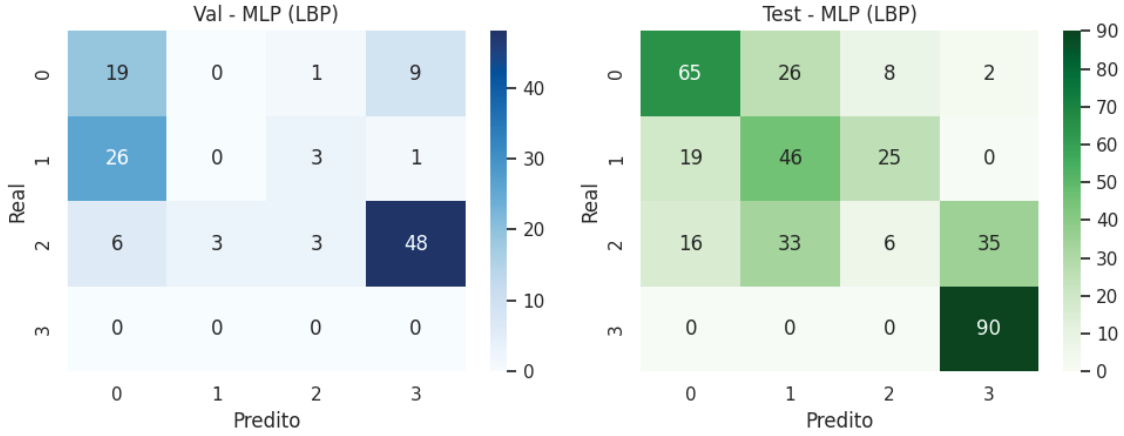
```
Extraído: /content/drive/MyDrive/Visão Comp. UFPR/Test_Warwick.zip
Subpastas treino: ['3', '0', '2', '1']
Subpastas teste : ['3', '2_teste.txt', '0', '2', '1']
CLASSES: ['0', '1', '2', '3']
LABEL_MAP: {'0': 0, '1': 1, '2': 2, '3': 3}
Total imagens - Treino: 593 | Teste: 371
Split -> treino: 474  | val: 119
LBP CSVs: (474, 14) (119, 14) (371, 14)
Shapes LBP -> (474, 10) (119, 10) (371, 10)
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
VGG CSVs: (474, 4100) (119, 4100) (371, 4100)
Shapes VGG -> (474, 4096) (119, 4096) (371, 4096)

=== Random Forest (LBP) ===
[VAL]  Acc: 52.94% | F1: 59.39% | Sens: 44.35% | Esp: 85.07%
[TEST] Acc: 55.80% | F1: 54.90% | Sens: 55.93% | Esp: 85.26%
```



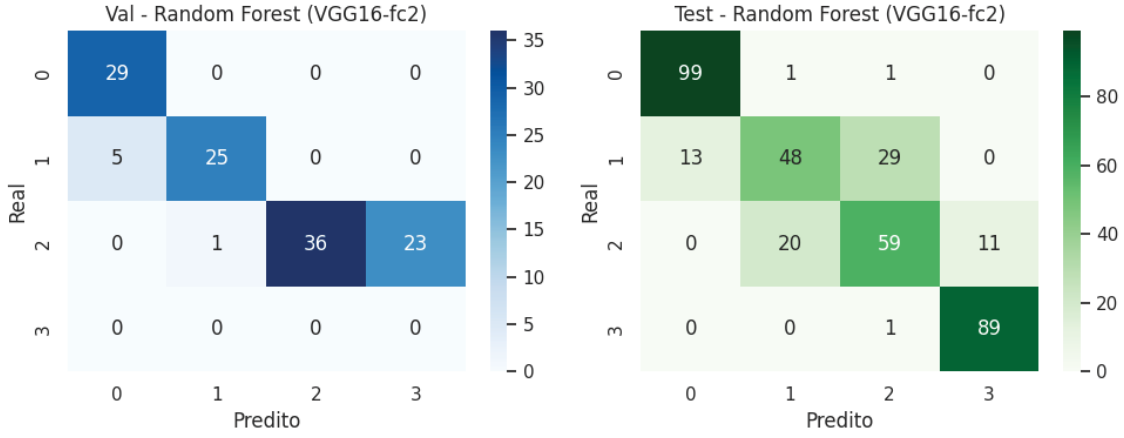

```
=== SVM Linear (LBP) ===
[VAL]  Acc: 50.42% | F1: 50.72% | Sens: 37.36% | Esp: 82.87%
[TEST] Acc: 66.31% | F1: 65.79% | Sens: 66.46% | Esp: 88.74%
```



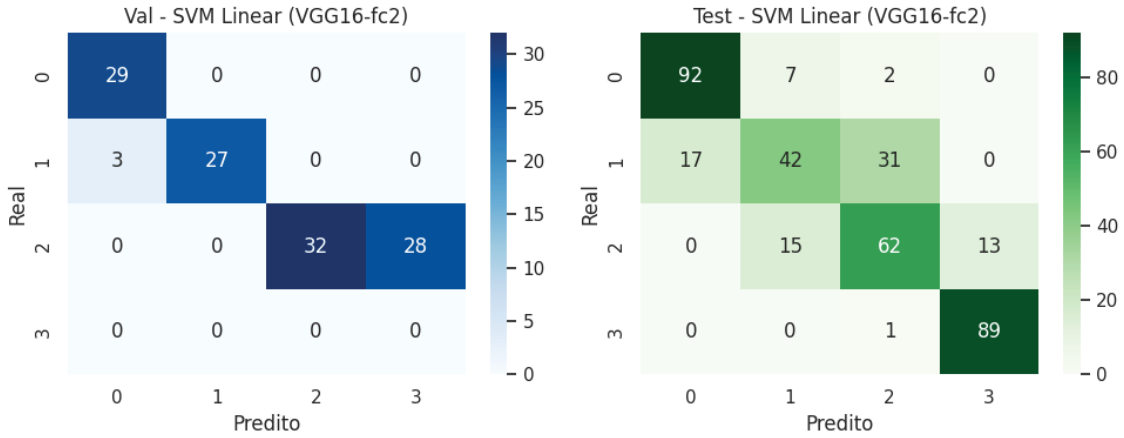

```
=== MLP (LBP) ===
[VAL]  Acc: 18.49% | F1: 16.09% | Sens: 17.63% | Esp: 76.39%
[TEST] Acc: 55.80% | F1: 51.43% | Sens: 55.53% | Esp: 85.28%
```



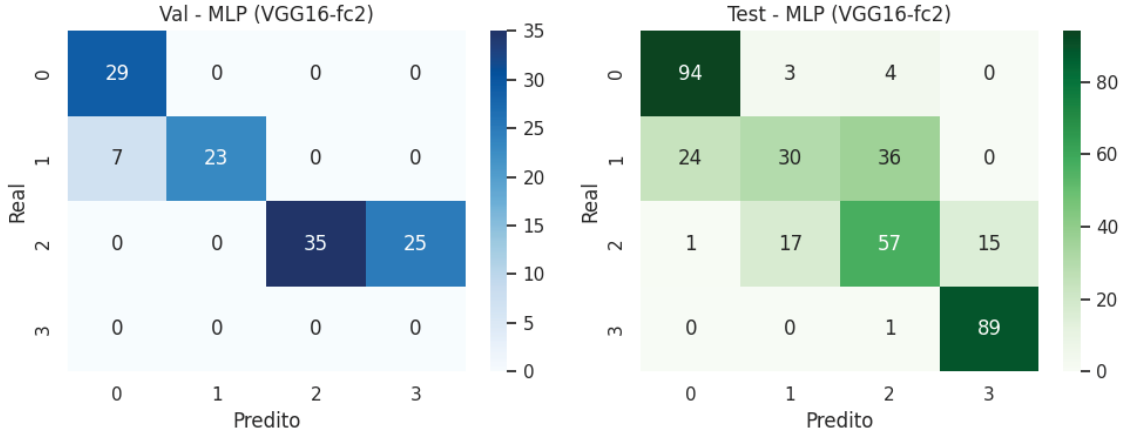

```
=== Random Forest (VGG16-fc2) ===
[VAL]  Acc: 75.63% | F1: 82.76% | Sens: 60.83% | Esp: 93.50%
[TEST] Acc: 79.51% | F1: 78.58% | Sens: 78.95% | Esp: 93.19%
```




```
=== SVM Linear (VGG16-fc2) ===
[VAL]  Acc: 73.95% | F1: 82.13% | Sens: 60.83% | Esp: 93.28%
[TEST] Acc: 76.82% | F1: 75.75% | Sens: 76.38% | Esp: 92.29%
```




```
=== MLP (VGG16-fc2) ===
[VAL]  Acc: 73.11% | F1: 80.78% | Sens: 58.75% | Esp: 92.80%
[TEST] Acc: 72.78% | F1: 70.63% | Sens: 72.16% | Esp: 90.92%
```



	modelo	VAL_acc	VAL_f1	VAL_sens	VAL_esp	TEST_acc	TEST_f1	TEST_sens	TEST_esp
3	Random Forest (VGG16-fc2)	0.756303	0.827598	0.608333	0.934983	0.795148	0.785829	0.789494	0.931913
4	SVM Linear (VGG16-fc2)	0.739496	0.821294	0.608333	0.932843	0.768194	0.757476	0.763834	0.922871
5	MLP (VGG16-fc2)	0.731092	0.807775	0.587500	0.928035	0.727763	0.706287	0.721562	0.909236
1	SVM Linear (LBP)	0.504202	0.507234	0.373563	0.828689	0.663073	0.657901	0.664576	0.887449
0	Random Forest (LBP)	0.529412	0.593872	0.443534	0.850741	0.557951	0.548999	0.559268	0.852643
2	MLP (LBP)	0.184874	0.160909	0.176293	0.763886	0.557951	0.514316	0.555336	0.852824

Escolhi usar o F1 como métrica principal porque nosso problema é multiclasse e desbalanceado; só olhar acurácia pode enganar (o modelo acerta muito nas classes grandes, mas tem dificuldade com as menores). O F1 equilibra precisão e sensibilidade, então avalia melhor o desempenho geral. Com isso, o melhor resultado veio do Random Forest com features da VGG16 (fc2): **F1=78,6%, Acc=79,5%, Sens=78,9% e Esp=93,2%**. Em resumo: as features da VGG16 capturam melhor os padrões das lâminas que o LBP, e o RF deu o melhor equilíbrio entre “não perder casos” e “não alarmar à toa”.

```
# ============================================================
# Parte 2 — CNNs (VGG16 e ResNet50) com/sem Data Augmentation
# ============================================================

# ----------------------------
# 0) Drive, caminhos e extração
# ----------------------------
from google.colab import drive
drive.mount('/content/drive')

import os, zipfile, math, random, warnings
import numpy as np, pandas as pd
import matplotlib.pyplot as plt, seaborn as sns
import tensorflow as tf
from sklearn.metrics import confusion_matrix, accuracy_score, f1_score

BASE = "/content/drive/MyDrive/Visão Comp. UFPR"
TRAIN_ZIP = f"{BASE}/Train_Warwick.zip"
TEST_ZIP  = f"{BASE}/Test_Warwick.zip"
TRAIN_DIR = "/content/train_warwick/Train_4cls_amostra"
TEST_DIR  = "/content/test_warwick/Test_4cl_amostra"

def extract_zip(zip_path, target_dir):
    os.makedirs(os.path.dirname(target_dir), exist_ok=True)
    if not os.path.exists(target_dir) or not
any(os.scandir(os.path.dirname(target_dir))):
        with zipfile.ZipFile(zip_path, 'r') as z:
            z.extractall(os.path.dirname(target_dir))
        print("Extraído:", zip_path)
    else:
        print("Já existia:", os.path.dirname(target_dir))

extract_zip(TRAIN_ZIP, TRAIN_DIR)
extract_zip(TEST_ZIP,  TEST_DIR)

print("Train subdirs:", os.listdir(TRAIN_DIR))
print("Test  subdirs:", os.listdir(TEST_DIR))

# ----------------------------
# 1) Utils: classes, split 80/20 por paciente
# ----------------------------
SEED = 42
tf.keras.utils.set_random_seed(SEED)
np.random.seed(SEED)
random.seed(SEED)
plt.rcParams["figure.figsize"] = (5.5, 4.5)
```

```
sns.set(style="whitegrid")

def discover_classes(root_dir):
    classes = sorted([d for d in os.listdir(root_dir) if
os.path.isdir(os.path.join(root_dir,d))])
    label_map = {c:i for i,c in enumerate(classes)}  # {'0':0,'1':1,'2':2,'3':3}
    return classes, label_map

CLASSES, LABEL_MAP = discover_classes(TRAIN_DIR)
print("CLASSES:", CLASSES, "LABEL_MAP:", LABEL_MAP)

def patient_id_from_name(fname:str):
    # XX_HER_YYYY.png -> "XX"
    return fname.split('_')[0]

def list_records(root_dir, label_map):
    recs = []
    for cstr, lab in label_map.items():
        cdir = os.path.join(root_dir, cstr)
        for fname in os.listdir(cdir):
            fpath = os.path.join(cdir, fname)
            if not os.path.isfile(fpath): continue
            pid = patient_id_from_name(fname)
            recs.append((fpath, lab, cstr, pid))
    return recs  # (path, label_id, class_str, patient)

train_records = list_records(TRAIN_DIR, LABEL_MAP)
test_records  = list_records(TEST_DIR,  LABEL_MAP)
print("Total imgs (train/test):", len(train_records), len(test_records))

def split_by_patient_global(records, val_ratio=0.2, seed=SEED):
    patients = sorted({r[3] for r in records})
    rng = np.random.RandomState(seed)
    n_val = max(1, int(np.ceil(len(patients)*val_ratio)))
    val_pids = set(rng.choice(patients, size=n_val, replace=False))
    train_split = [r for r in records if r[3] not in val_pids]
    val_split   = [r for r in records if r[3] in  val_pids]
    assert set([r[3] for r in train_split]).isdisjoint(set([r[3] for r in
val_split]))
    return train_split, val_split

train_split, val_split = split_by_patient_global(train_records, 0.2, SEED)
print("Split -> train:", len(train_split), "val:", len(val_split))

# ----------------------------
# 2) tf.data a partir das listas (sem mexer nas cores aqui)
```

```
# ----------------------------
IMG_SIZE = (224, 224)
BATCH = 32
AUTOTUNE = tf.data.AUTOTUNE

def make_dataset(records):
    paths = [r[0] for r in records]
    labels= [r[1] for r in records]
    ds = tf.data.Dataset.from_tensor_slices((paths, labels))
    def _load(path, y):
        img = tf.io.read_file(path)
        img = tf.image.decode_image(img, channels=3, expand_animations=False)
        img = tf.image.resize(img, IMG_SIZE, antialias=True)  # 250->224
        img = tf.cast(img, tf.float32)  # [0,255] float
        return img, tf.cast(y, tf.int32)
    ds = ds.map(_load, num_parallel_calls=AUTOTUNE)
    return ds

ds_train = make_dataset(train_split).shuffle(8192,
seed=SEED).batch(BATCH).prefetch(AUTOTUNE)
ds_val   = make_dataset(val_split).batch(BATCH).prefetch(AUTOTUNE)
ds_test  = make_dataset(test_records).batch(BATCH).prefetch(AUTOTUNE)

# ----------------------------
# 3) Model builders (VGG16, ResNet50) + opção de Augmentation
# ----------------------------
from tensorflow.keras import layers, Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input as vgg_pre
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input as
res_pre

def build_classifier(backbone:str, augment:bool):
    # Augmentation geométrico leve (não altera cor)
    aug = tf.keras.Sequential([
        layers.RandomFlip("horizontal"),
        layers.RandomRotation(0.05),
        layers.RandomZoom(0.1),
        layers.RandomTranslation(0.05, 0.05)
    ], name="augment")

    inp = layers.Input(shape=(*IMG_SIZE, 3))
    x = aug(inp) if augment else inp

    if backbone == "vgg16":
        x = layers.Lambda(vgg_pre, name="preprocess")(x)
```

```
        base = VGG16(weights="imagenet", include_top=False, pooling="avg")
    elif backbone == "resnet50":
        x = layers.Lambda(res_pre, name="preprocess")(x)
        base = ResNet50(weights="imagenet", include_top=False, pooling="avg")
    else:
        raise ValueError("backbone inválido")

    base.trainable = False
    x = base(x, training=False)
    x = layers.Dense(256, activation="relu")(x)
    x = layers.Dropout(0.3)(x)
    out = layers.Dense(len(CLASSES), activation="softmax")(x)

    model = Model(inp, out, name=f"{backbone}{'_aug' if augment else '_noaug'}")
    model.compile(optimizer=Adam(1e-4),
                  loss="sparse_categorical_crossentropy",
                  metrics=["accuracy"])
    return model

# ----------------------------
# 4) Treinos (15 épocas) - VGG16 e ResNet50, com e sem Aug
# ----------------------------
EPOCHS = 15
histories = {}
models = {}

for backbone in ["vgg16", "resnet50"]:
    for aug_flag in [False, True]:
        name = f"{backbone.upper()}|{'AUG' if aug_flag else 'NO_AUG'}"
        print("\n>>> Treinando:", name)
        model = build_classifier(backbone, augment=aug_flag)
        h = model.fit(ds_train, validation_data=ds_val, epochs=EPOCHS, verbose=1)
        models[name] = model
        histories[name] = h.history

# ----------------------------
# 5) Métricas (Acc, F1, Sens, Esp) + Matriz de Confusão
# ----------------------------
def metrics_from_cm(cm):
    K = cm.shape[0]
    sens, spec = [], []
    for i in range(K):
        TP = cm[i,i]
        FN = cm[i,:].sum() - TP
        FP = cm[:,i].sum() - TP
        TN = cm.sum() - (TP+FN+FP)
```

```
        sens.append(TP/(TP+FN) if (TP+FN)>0 else 0.0)
        spec.append(TN/(TN+FP) if (TN+FP)>0 else 0.0)
    return float(np.mean(sens)), float(np.mean(spec))

def evaluate_model(model, ds, y_true):
    y_prob = model.predict(ds, verbose=0)
    y_pred = np.argmax(y_prob, axis=1)
    cm = confusion_matrix(y_true, y_pred)
    acc = accuracy_score(y_true, y_pred)
    f1  = f1_score(y_true, y_pred, average="weighted")
    sens, esp = metrics_from_cm(cm)
    return acc, f1, sens, esp, cm

# Vetor de rótulos verdadeiros para Val e Test
y_val = np.array([r[1] for r in val_split], dtype=int)
y_test= np.array([r[1] for r in test_records], dtype=int)

results = []
for name, model in models.items():
    # validação
    acc_v, f1_v, sens_v, esp_v, cm_v = evaluate_model(model, ds_val, y_val)
    # teste
    acc_t, f1_t, sens_t, esp_t, cm_t = evaluate_model(model, ds_test, y_test)

    print(f"\n=== {name} ===")
    print(f"[VAL]  Acc: {acc_v*100:.2f}% | F1: {f1_v*100:.2f}% | Sens:
{sens_v*100:.2f}% | Esp: {esp_v*100:.2f}%")
    print(f"[TEST] Acc: {acc_t*100:.2f}% | F1: {f1_t*100:.2f}% | Sens:
{sens_t*100:.2f}% | Esp: {esp_t*100:.2f}%")

    fig, ax = plt.subplots(1,2, figsize=(10,4))
    sns.heatmap(cm_v, annot=True, fmt='d', cmap='Blues', ax=ax[0]);
ax[0].set_title(f"Val — {name}")
    sns.heatmap(cm_t, annot=True, fmt='d', cmap='Greens', ax=ax[1]);
ax[1].set_title(f"Test — {name}")
    for a in ax: a.set_xlabel("Predito"); a.set_ylabel("Real")
    plt.tight_layout(); plt.show()

    results.append({
        "modelo": name,
        "VAL_acc": acc_v, "VAL_f1": f1_v, "VAL_sens": sens_v, "VAL_esp": esp_v,
        "TEST_acc": acc_t, "TEST_f1": f1_t, "TEST_sens": sens_t, "TEST_esp": esp_t
    })

# ----------------------------
# 6) Quadro-resumo e melhor modelo
```

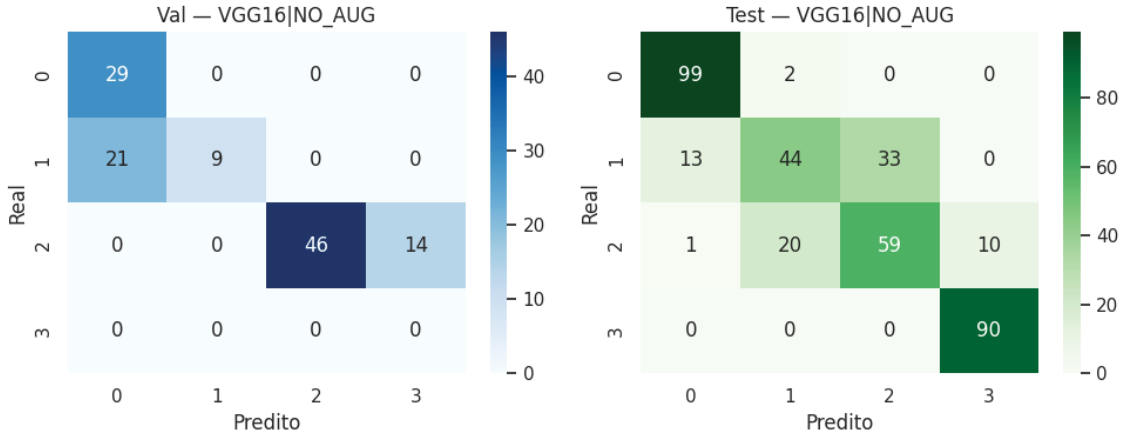
```
# ----------------------------
df_results = pd.DataFrame(results).sort_values("TEST_f1", ascending=False)
display(df_results)

best = df_results.iloc[0]
print(f"\n>>> Melhor modelo (pela TEST_F1): {best['modelo']}")
print(f"Acc={best['TEST_acc']*100:.2f}% | F1={best['TEST_f1']*100:.2f}% | "
      f"Sens={best['TEST_sens']*100:.2f}% | Esp={best['TEST_esp']*100:.2f}%")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
Já existia: /content/train_warwick
Já existia: /content/test_warwick
Train subdirs: ['3', '0', '2', '1']
Test  subdirs: ['3', '2_teste.txt', '0', '2', '1']
CLASSES: ['0', '1', '2', '3'] LABEL_MAP: {'0': 0, '1': 1, '2': 2, '3': 3}
Total imgs (train/test): 593 371
Split -> train: 474 val: 119

>>> Treinando: VGG16|NO_AUG
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 366ms/step - accuracy: 0.3809 -
loss: 2.4504 - val_accuracy: 0.2437 - val_loss: 2.0412
Epoch 2/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.5685 -
loss: 1.3524 - val_accuracy: 0.3782 - val_loss: 1.5112
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.6553 -
loss: 0.9850 - val_accuracy: 0.4202 - val_loss: 1.3404
...
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8600 -
loss: 0.3630 - val_accuracy: 0.6555 - val_loss: 0.8140
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8737 -
loss: 0.3022 - val_accuracy: 0.6303 - val_loss: 0.8656
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8964 -
loss: 0.2328 - val_accuracy: 0.6471 - val_loss: 0.8303
...
Epoch 15/15
```

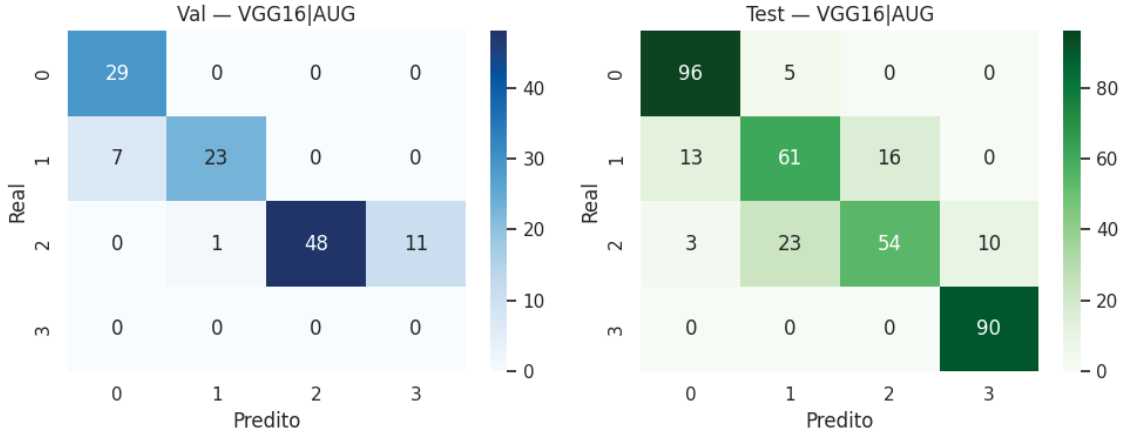
```
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9112 -
loss: 0.2243 - val_accuracy: 0.7059 - val_loss: 0.6978

>>> Treinando: VGG16|AUG
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 59ms/step - accuracy: 0.2640 -
loss: 3.2977 - val_accuracy: 0.0336 - val_loss: 3.2207
Epoch 2/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.4410 -
loss: 1.9042 - val_accuracy: 0.1513 - val_loss: 2.1688
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.5513 -
loss: 1.4224 - val_accuracy: 0.3866 - val_loss: 1.3085
...
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8488 -
loss: 0.4566 - val_accuracy: 0.7815 - val_loss: 0.5255
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8370 -
loss: 0.4476 - val_accuracy: 0.8067 - val_loss: 0.4928
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.8652 -
loss: 0.3361 - val_accuracy: 0.7899 - val_loss: 0.4953
...
Epoch 15/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.8795 -
loss: 0.3070 - val_accuracy: 0.8403 - val_loss: 0.3641

>>> Treinando: RESNET50|NO_AUG
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23s 792ms/step - accuracy: 0.3684 -
loss: 1.7705 - val_accuracy: 0.2941 - val_loss: 1.5834
Epoch 2/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.7094 -
loss: 0.7138 - val_accuracy: 0.5042 - val_loss: 0.9199
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.8525 -
loss: 0.4170 - val_accuracy: 0.5882 - val_loss: 0.9450
...
```

```
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9779 -
loss: 0.0921 - val_accuracy: 0.7059 - val_loss: 0.6894
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9912 -
loss: 0.0685 - val_accuracy: 0.7059 - val_loss: 0.6975
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9898 -
loss: 0.0762 - val_accuracy: 0.6891 - val_loss: 0.7561
...
Epoch 15/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.9900 -
loss: 0.0604 - val_accuracy: 0.7143 - val_loss: 0.7156

>>> Treinando: RESNET50|AUG
Epoch 1/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 199ms/step - accuracy: 0.4027 -
loss: 1.4390 - val_accuracy: 0.5882 - val_loss: 0.9388
Epoch 2/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.8053 -
loss: 0.5561 - val_accuracy: 0.6975 - val_loss: 0.7010
Epoch 3/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.8616 -
loss: 0.3596 - val_accuracy: 0.6723 - val_loss: 0.7162
...
Epoch 10/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9542 -
loss: 0.1410 - val_accuracy: 0.7395 - val_loss: 0.6460
Epoch 11/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9784 -
loss: 0.0958 - val_accuracy: 0.7395 - val_loss: 0.6667
Epoch 12/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.9758 -
loss: 0.0834 - val_accuracy: 0.7311 - val_loss: 0.6601
...
Epoch 15/15
15/15 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.9656 -
loss: 0.0940 - val_accuracy: 0.7311 - val_loss: 0.6817
```

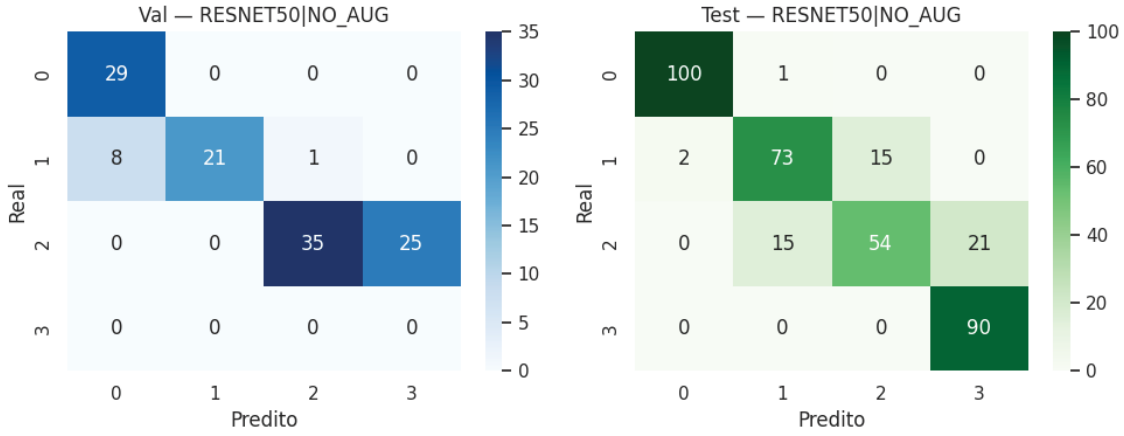
```
=== VGG16|NO_AUG ===
[VAL]  Acc: 70.59% | F1: 73.29% | Sens: 51.67% | Esp: 91.23%
[TEST] Acc: 78.71% | F1: 77.58% | Sens: 78.12% | Esp: 92.92%
```




```
=== VGG16|AUG ===
[VAL]  Acc: 84.03% | F1: 88.04% | Sens: 64.17% | Esp: 95.46%
[TEST] Acc: 81.13% | F1: 80.43% | Sens: 80.71% | Esp: 93.71%
```

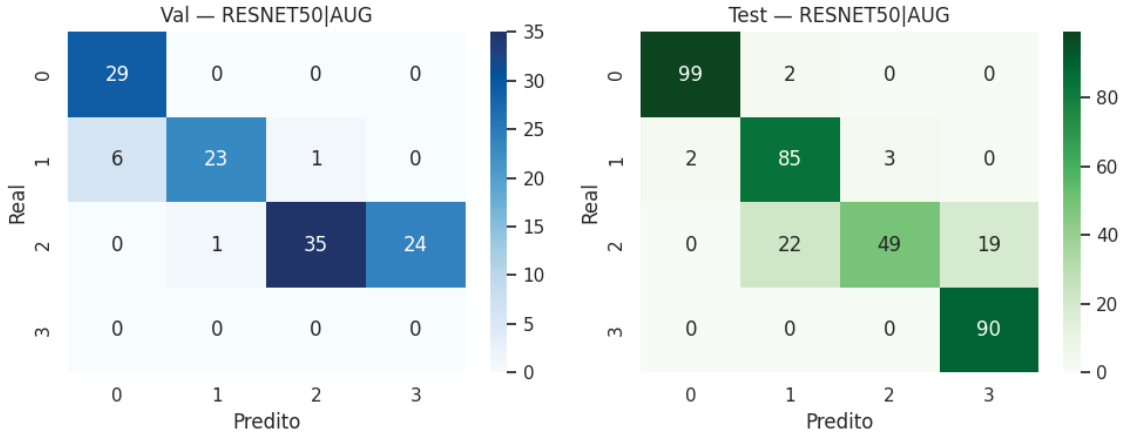



```
=== RESNET50|NO_AUG ===
[VAL]  Acc: 71.43% | F1: 78.94% | Sens: 57.08% | Esp: 92.10%
[TEST] Acc: 85.44% | F1: 84.81% | Sens: 85.03% | Esp: 95.19%
```




```
=== RESNET50|AUG ===
```

```
[VAL]  Acc: 73.11% | F1: 80.33% | Sens: 58.75% | Esp: 92.59%
[TEST] Acc: 87.06% | F1: 86.09% | Sens: 86.73% | Esp: 95.72%
```



	modelo	VAL_acc	VAL_f1	VAL_sens	VAL_esp	TEST_acc	TEST_f1	TEST_sens	TEST_esp
3	RESNET50|AUG	0.731092	0.803250	0.587500	0.925867	0.870620	0.860928	0.867272	0.957223
2	RESNET50|NO_AUG	0.714286	0.789418	0.570833	0.921019	0.854447	0.848098	0.850303	0.951885
1	VGG16|AUG	0.840336	0.880385	0.641667	0.954637	0.811321	0.804302	0.807068	0.937142
0	VGG16|NO_AUG	0.705882	0.732880	0.516667	0.912255	0.787062	0.775829	0.781161	0.929208

Usei F1 porque o problema tem 4 classes desbalanceadas; só acurácia pode enganar. O melhor modelo foi a ResNet50 com augmentation, porque a rede com pesos do ImageNet já “sabe” boas features e o augmentation leve ajuda a generalizar sem estragar as cores. Os resultados (**F1 86% / Acc 87% / Sens 86,7% / Esp 95,7%**) mostram um bom equilíbrio: acerta bastante e evita falsos alarmes. Se precisar ainda mais recall nas classes 2+/3+, dá para ajustar pesos ou fazer um pequeno fine-tuning.

APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

Título do Trabalho: "Estudo de Caso: Implicações Éticas do Uso do ChatGPT"

Trabalho em Grupo: O trabalho deverá ser realizado em grupo de alunos de no máximo seis (06) integrantes.

Objetivo do Trabalho: Investigar as implicações éticas do uso do ChatGPT em diferentes contextos e propor soluções responsáveis para lidar com esses dilemas.
Parâmetros para elaboração do Trabalho:

1. Relevância Ética: O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.
2. Análise Crítica: Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.
3. Soluções Responsáveis: Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.
4. Colaboração e Discussão: O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.
5. Limite de Palavras: O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.
6. Estruturação Adequada: O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.
7. Controle de Informações: Evitar incluir informações desnecessárias que possam aumentar o comprimento do trabalho sem contribuir significativamente para o conteúdo. Concentre-se em informações relevantes, argumentos sólidos e evidências importantes para apoiar sua análise.

8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.

9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx

B – RESOLUÇÃO

Estudo de Caso: Implicações Éticas do Uso do ChatGPT

RESUMO

Este estudo de caso investiga as implicações éticas do uso do ChatGPT, modelo de linguagem baseado em inteligência artificial generativa, em diferentes contextos sociais e profissionais. A pesquisa analisa dilemas relacionados à privacidade de dados, viés algorítmico, desinformação e impacto na tomada de decisão humana, abordando efeitos para usuários, desenvolvedores, empresas e a sociedade em geral. Por meio de análise crítica de casos reais e hipotéticos, como o uso do ChatGPT em ambientes educacionais, no atendimento ao cliente e em decisões sensíveis, discute-se como a tecnologia pode reforçar desigualdades, disseminar informações imprecisas ou violar direitos fundamentais. Além de identificar riscos, o trabalho propõe soluções responsáveis, como transparência algorítmica, regulamentações específicas, mecanismos de explicabilidade, auditorias independentes e práticas de design ético que priorizem a justiça, a privacidade e o bem-estar coletivo. A investigação contribui para o debate contemporâneo sobre a necessidade de equilíbrio entre inovação tecnológica e valores éticos universais.

Palavras-chave: Inteligência artificial. ChatGPT. Ética digital. Viés algorítmico. Privacidade.

ABSTRACT

This case study investigates the ethical implications of using ChatGPT, a generative artificial intelligence language model, across different social and professional contexts. It analyzes dilemmas related to data privacy, algorithmic bias, misinformation, and human decision-making, highlighting impacts on users, developers, companies, and society as a whole. Through a critical analysis of real and hypothetical cases, such as the use of ChatGPT in education, customer service, and sensitive decision-making, the research discusses how the technology can reinforce inequalities, spread inaccurate information, or violate fundamental rights. In addition to identifying risks, it proposes responsible solutions such as algorithmic transparency, specific regulations, explainability mechanisms, independent audits, and ethical design practices that prioritize fairness, privacy, and collective well-being. This work contributes to the ongoing debate about balancing technological innovation with universal ethical values.

Keywords: Artificial intelligence. ChatGPT. Digital ethics. Algorithmic bias. Privacy.

INTRODUÇÃO

O avanço da inteligência artificial (IA) tem transformado diversas áreas da sociedade, desde processos produtivos até a comunicação interpessoal. Entre as tecnologias mais impactantes está o ChatGPT, modelo de linguagem capaz de gerar textos complexos, responder perguntas e auxiliar em

tarefas criativas e analíticas. Apesar de seu potencial inovador, o uso dessa tecnologia levanta questões éticas significativas, como privacidade dos usuários, disseminação de informações falsas e possíveis vieses que afetam decisões automatizadas.

Este trabalho tem como objetivo analisar criticamente as implicações éticas do uso do ChatGPT em diferentes contextos, identificando dilemas reais e propondo soluções responsáveis. O estudo é relevante porque ferramentas como o ChatGPT estão cada vez mais presentes em setores estratégicos da educação, saúde, negócios e governo, influenciando decisões que podem impactar vidas humanas e estruturas sociais.

REVISÃO DE LITERATURA

A ética na inteligência artificial é um campo interdisciplinar que envolve filosofia, direito, tecnologia e sociologia. Segundo Floridi e Cowls (2019), sistemas de IA devem seguir princípios como beneficência, não maleficência, justiça e explicabilidade. Crawford (2021) reforça que modelos de linguagem tendem a reproduzir preconceitos presentes nos dados de treinamento, o que pode levar à discriminação e ao reforço de estereótipos.

No contexto da IA generativa, Bender et al. (2021) alertam para o risco de alucinações e geração de informações incorretas ou inventadas que podem influenciar negativamente usuários desavisados. Além disso, a coleta massiva de dados para treinar modelos levanta preocupações sobre privacidade e consentimento.

> Para Floridi e Cowls (2019, p. 5): “A inteligência artificial deve ser desenvolvida e utilizada de maneira que beneficie a sociedade como um todo, respeitando direitos fundamentais, promovendo justiça e garantindo transparência e explicabilidade em seus processos. Esses princípios são essenciais para reduzir riscos de discriminação, uso indevido de dados pessoais e danos sociais.”

A literatura também discute o conceito de responsabilidade algorítmica. Pasquale (2015) argumenta que sistemas de IA, por influenciarem decisões humanas em larga escala, precisam ser auditáveis e que os desenvolvedores devem responder pelos impactos éticos de suas criações. Regulamentações emergentes, como a Lei de Inteligência Artificial da União Europeia (AI Act), buscam estabelecer padrões para transparência e segurança em sistemas de alto risco.

METODOLOGIA

Este trabalho foi desenvolvido como um estudo de caso exploratório, combinando análise documental, revisão de literatura e avaliação crítica de cenários de uso do ChatGPT. Foram selecionados três contextos representativos:

1. Educação e produção acadêmica com o uso do ChatGPT para elaboração de trabalhos e apoio ao aprendizado.
2. Atendimento ao cliente e suporte empresarial com o uso de chatbots e automação de serviços.
3. Tomada de decisões sensíveis, aplicação em áreas como saúde, jurídico e governança.

Cada contexto foi analisado quanto a dilemas éticos, impactos e possíveis soluções para mitigar riscos e promover um uso responsável.

ANÁLISE CRÍTICA E ESTUDO DE CASO

EDUCAÇÃO E PRODUÇÃO ACADÊMICA

O ChatGPT pode apoiar estudantes na pesquisa e escrita, promovendo aprendizado personalizado. Entretanto, há riscos como plágio facilitado, dependência cognitiva e dificuldade de avaliar a veracidade do conteúdo gerado. Professores enfrentam o desafio de diferenciar trabalhos autorais de textos criados por IA, o que pode comprometer a avaliação justa.

Implicações éticas:

- Desigualdade entre estudantes que têm ou não acesso a ferramentas de IA.
- Integridade acadêmica prejudicada.
- Formação crítica reduzida, com alunos aceitando respostas prontas.

Soluções recomendadas:

- Ferramentas de detecção de IA e políticas institucionais claras.
- Incentivo à alfabetização digital crítica.
- Transparência na autoria dos trabalhos.

ATENDIMENTO AO CLIENTE E SUPORTE EMPRESARIAL

Empresas usam o ChatGPT para reduzir custos e agilizar o suporte. Contudo, a automatização pode gerar respostas incorretas, dificultar o acesso a atendimento humano e coletar dados sensíveis sem clareza.

Implicações éticas:

- Risco à privacidade e segurança de dados.
- Perda de confiança em caso de respostas erradas.
- Falta de clareza sobre quem é responsável por informações imprecisas.

Soluções recomendadas:

- Políticas de privacidade transparentes e consentimento explícito.
- Auditoria de qualidade contínua nos modelos de atendimento.
- Escalonamento fácil para suporte humano.

TOMADA DE DECISÕES SENSÍVEIS

O uso de IA em diagnósticos médicos, decisões jurídicas ou análise de risco financeiro exige extremo cuidado. O ChatGPT não possui entendimento real do contexto e pode gerar recomendações equivocadas.

Implicações éticas:

- Risco à vida e à justiça em decisões críticas.
- Viés reforçando preconceitos sociais.
- Falta de explicabilidade no processo decisório.

Soluções recomendadas:

- Limitar o uso em decisões críticas sem supervisão humana.
- Desenvolver modelos com explicabilidade e rastreabilidade.
- Regulamentar a validação por especialistas.

Para sintetizar os principais dilemas e as soluções apontadas para cada contexto analisado, apresenta-se o Quadro 1.

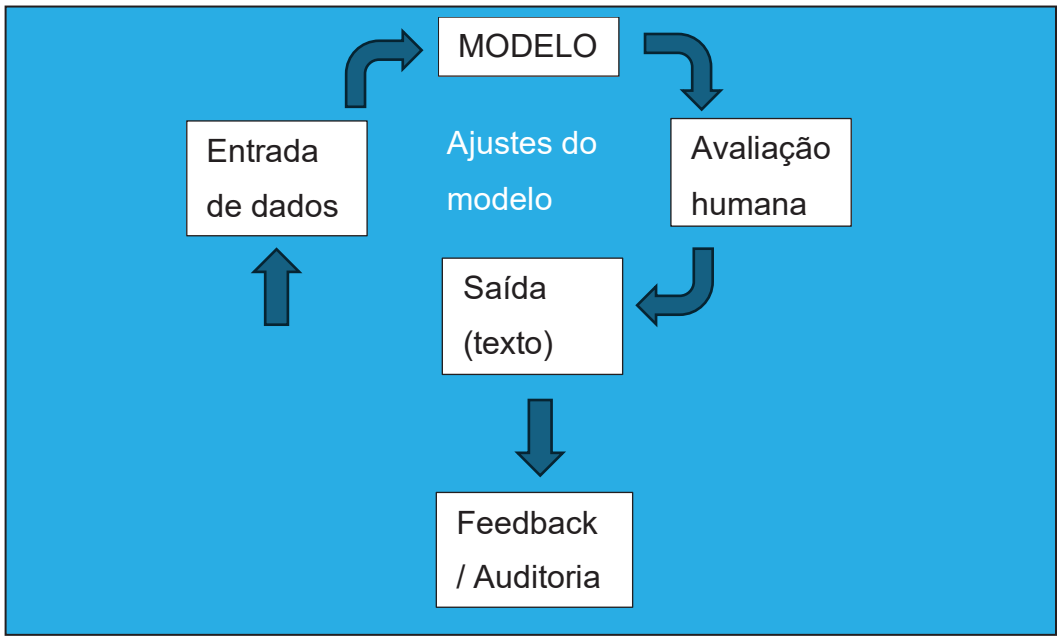
QUADRO 1 – DILEMAS ÉTICOS E SOLUÇÕES RESPONSÁVEIS NO USO DO CHATGPT

Contexto	Dilema ético principal	Risco para usuários/sociedade	Possíveis soluções responsáveis
Educação	Plágio e dependência cognitiva	Redução do pensamento crítico e injustiça avaliativa	Políticas de uso, detecção de IA, alfabetização digital
Atendimento ao cliente	Respostas imprecisas e coleta de dados sensíveis	Perda de confiança, violação de privacidade	Auditoria, transparência, opção de atendimento humano
Decisões sensíveis	Viés e recomendações equivocadas	Danos à vida e injustiças	Supervisão humana obrigatória, explicabilidade e regulamentação

FONTE: O autor (2025).

Além da análise dos dilemas, é útil visualizar o ciclo de uso responsável do ChatGPT, desde a entrada de dados até os ajustes do modelo após auditorias, conforme representado na Figura 1.

FIGURA 1 – CICLO DE USO RESPONSÁVEL DO CHATGPT



FONTE: O autor (2025).

Por fim, o Quadro 2 relaciona princípios éticos amplamente discutidos na literatura com sua aplicação prática ao ChatGPT, reforçando a importância de diretrizes claras para um uso seguro e responsável.

QUADRO 2 – PRINCÍPIOS ÉTICOS APLICADOS AO CHATGPT

Princípio	Descrição	Aplicação ao ChatGPT
Transparência	Usuários devem entender como a IA funciona	Explicabilidade e divulgação de riscos
Justiça	Evitar discriminação e viés	Testes de viés e revisão contínua
Privacidade	Proteger dados pessoais	Conformidade com LGPD e consentimento
Responsabilidade	Assumir impactos e erros	Canais de revisão humana e suporte
Beneficência	Promover bem-estar coletivo	Uso para inclusão e apoio educacional

FONTE: O autor (2025).

DISCUSSÃO GERAL

Os cenários analisados revelam padrões comuns de dilemas éticos: falta de transparência, risco de viés algorítmico, autoridade indevida em áreas sensíveis e impactos na autonomia humana. Embora o ChatGPT traga benefícios em produtividade, acessibilidade e inovação, é necessário estabelecer governança ética da IA para equilibrar ganhos e riscos.

SOLUÇÕES RESPONSÁVEIS E RECOMENDAÇÕES

1. Transparência e Explicabilidade deve detalhar limitações do modelo e justificar respostas.
2. Regulamentação e Normas Éticas deve permitir auditorias externas e classificação de riscos.
3. Privacidade e Proteção de Dados devem cumprir a LGPD e obter consentimento claro.
4. Educação e Alfabetização Digital precisa preparar usuários para uso crítico da IA.
5. Responsabilidade Compartilhada precisa permitir que empresas e desenvolvedores possam responder por falhas e impactos.

CONSIDERAÇÕES FINAIS

O ChatGPT representa um marco no avanço da inteligência artificial, mas seu uso indiscriminado pode causar impactos éticos graves. Ao mesmo tempo em que potencializa inovação, também traz riscos de viés, desinformação, violação de privacidade e erosão da autonomia humana. Este estudo oferece recomendações práticas que vão desde educação digital até regulamentação visando garantir um uso mais seguro e responsável.

REFERÊNCIAS

BENDER, Emily M. et al. *On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?* Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, 2021.

CRAWFORD, Kate. Atlas of AI: *Power, Politics, and the Planetary Costs of Artificial Intelligence.* Yale University Press, 2021.

FLORIDI, Luciano; COWLS, Josh. *A Unified Framework of Five Principles for AI in Society.* Harvard Data Science Review, v. 1, n. 1, 2019.

PASQUALE, Frank. The Black Box Society: *The Secret Algorithms That Control Money and Information.* Harvard University Press, 2015.

UNIÃO EUROPEIA. *Artificial Intelligence Act.* Disponível em: https://digital-strategy.ec.europa.eu. Acesso em: 05 out. 2025.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

Individualmente, ler e resumir – seguindo o *template* fornecido – **um** dos artigos abaixo:

AHMAD, L.; ABDELRAZEK, M.; ARORA, C.; BANO, M; GRUNDY, J. Requirements practices and gaps when engineering human-centered Artificial Intelligence systems. Applied Soft Computing. 143. 2023. DOI https://doi.org/10.1016/j.asoc.2023.110421

NAZIR, R.; BUCAIONI, A.; PELLICCIONE, P.; Architecting ML-enabled systems: Challenges, best practices, and design decisions. The Journal of Systems & Software. 207. 2024. DOI https://doi.org/10.1016/j.jss.2023.111860

SERBAN, A.; BLOM, K.; HOOS, H.; VISSER, J. Software engineering practices for machine learning – Adoption, effects, and team assessment. The Journal of Systems & Software. 209. 2024. DOI https://doi.org/10.1016/j.jss.2023.111907

STEIDL, M.; FELDERER, M.; RAMLER, R. The pipeline for continuous development of artificial intelligence models – Current state of research and practice. The Journal of Systems & Software. 199. 2023. DOI https://doi.org/10.1016/j.jss.2023.111615

XIN, D.; WU, E. Y.; LEE, D. J.; SALEHI, N.; PARAMESWARAN, A. Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows. In CHI Conference on Human Factors in Computing Systems (CHI'21), Maio 8-13, 2021, Yokohama, Japão. DOI https://doi.org/10.1145/3411764.3445306

2 Orientações adicionais

Escolha o artigo que for mais interessante para você. Utilize tradutores e o Chat GPT para entender o conteúdo dos artigos – caso precise, mas escreva o resumo em língua portuguesa e nas suas palavras.

Não esqueça de preencher, no trabalho, os campos relativos ao seu nome e ao artigo escolhido.

No *template*, você deverá responder às seguintes questões:

- Qual o objetivo do estudo descrito pelo artigo?
- Qual o problema/oportunidade/situação que levou a necessidade de realização deste estudo?
- Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?

- Quais os principais resultados obtidos pelo estudo?

Responda cada questão utilizando o espaço fornecido no *template*, sem alteração do tamanho da fonte (Times New Roman, 10), nem alteração do espaçamento entre linhas (1.0).

Não altere as questões do template.

Utilize o editor de textos de sua preferência para preencher as respostas, mas entregue o trabalho em PDF.

B – RESOLUÇÃO

Nome do artigo escolhido:	Xin, D.; Wu, E. Y.; Lee, D. J.; Salehi, N.; Parameswaran, A. Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows. In CHI Conference on Human Factors in Computing Systems (CHI'21), May 8-13, 2021, Yokohama, Japan. DOI: https://doi.org/10.1145/3411764.3445306

Qual o **objetivo** do estudo descrito pelo artigo?	Qual o **problema / oportunidade / situação** que levou à necessidade de realização desse estudo?	Qual a **metodologia** que os autores usaram para obter e analisar as informações do estudo?	Quais os **principais resultados** obtidos pelo estudo?
O artigo tem como objetivo investigar como ferramentas de AutoML (Automated Machine Learning) se inserem nos fluxos de trabalho de aprendizado de máquina e como impactam diferentes perfis de usuários, incluindo cientistas de dados, engenheiros e profissionais com pouca experiência técnica. O estudo busca compreender os benefícios e limitações do AutoML, identificar desafios relacionados à confiança, transparência, controle e adaptação a diferentes níveis de expertise e, a partir disso, propor diretrizes de design para criar ferramentas mais eficazes e centradas no ser humano, equilibrando automação com usabilidade, clareza de funcionamento e	A crescente adoção do aprendizado de máquina em diferentes áreas trouxe a necessidade de tornar o desenvolvimento de modelos mais acessível e eficiente, especialmente para usuários com pouca experiência técnica. Nesse contexto, surgiram ferramentas de **AutoML** como oportunidade para automatizar tarefas complexas, como seleção de modelos e ajuste de hiperparâmetros. No entanto, essa automação gera desafios importantes, como falta de transparência nos processos, perda de controle para usuários experientes e dificuldades de interação para perfis diversos. O estudo foi motivado pela necessidade de compreender como o AutoML afeta os fluxos de trabalho de aprendizado de máquina e de identificar caminhos para projetar	Os autores utilizaram uma abordagem qualitativa centrada no ser humano, típica da área de interação humano-computador (HCI), combinando revisão de literatura com estudos empíricos. Para isso, realizaram entrevistas em profundidade e análises de fluxos de trabalho reais envolvendo o uso de ferramentas de **AutoML**, engajando diferentes perfis de usuários, desde cientistas de dados experientes até iniciantes. A coleta de dados buscou entender suas experiências, percepções, desafios e expectativas ao trabalhar com automação em aprendizado de máquina. A análise concentrou-se em identificar padrões de uso, barreiras à adoção e	Os principais resultados do estudo mostram que o **AutoML** oferece benefícios significativos, como aumento de eficiência ao reduzir o tempo necessário para tarefas de aprendizado de máquina e democratização do acesso, permitindo que usuários sem experiência técnica construam modelos. Entretanto, foram identificadas limitações importantes, incluindo a percepção das ferramentas como "caixas-pretas", dificultando a compreensão das decisões automatizadas, a sensação de perda de controle entre usuários experientes e a forte dependência da qualidade dos dados de entrada para o desempenho dos modelos. O estudo também evidenciou que diferentes perfis de

possibilidade de intervenção manual, de modo a tornar essas tecnologias mais úteis, compreensíveis e confiáveis para variados públicos.	ferramentas que atendam melhor às necessidades de diferentes tipos de usuários, promovendo colaboração mais eficaz entre pessoas e sistemas automatizados.	oportunidades de melhoria no design dessas ferramentas, com foco especial em usabilidade, transparência e possibilidade de controle por parte do usuário.	usuários têm necessidades distintas, iniciantes demandam simplicidade e orientação, enquanto especialistas buscam maior transparência e capacidade de personalização. Com base nesses achados, os autores propõem diretrizes de design para ferramentas de **AutoML**, recomendando interfaces intuitivas, explicações claras das escolhas feitas pelo sistema e flexibilidade para ajustes manuais, favorecendo a colaboração entre humanos e automação. Além disso, apontam direções futuras de pesquisa, como o avanço na explicabilidade do **AutoML** e sua melhor integração em fluxos de trabalho completos, incluindo práticas de **MLOps**.

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

Implementar o exemplo de Classificação usando a base de dados Fashion MNIST e a arquitetura RNA vista na aula **FRA - Aula 10 - 2.4 Resolução de exercício de RNA - Classificação**. Além disso, fazer uma breve explicação dos seguintes resultados:

- Gráficos de perda e de acurácia;
- Imagem gerada na seção "**Mostrar algumas classificações erradas"**, apresentada na aula prática.

Informações:

- **Base de dados:** Fashion MNIST Dataset
- **Descrição**: Um dataset de imagens de roupas, onde o objetivo é classificar o tipo de vestuário. É semelhante ao famoso dataset MNIST, mas com peças de vestuário em vez de dígitos.
- **Tamanho**: 70.000 amostras, 784 features (28x28 pixels).
- **Importação do dataset**: Copiar código abaixo.

```
data = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

2 Regressão (RNA)

Implementar o exemplo de Classificação usando a base de dados Wine Dataset e a arquitetura RNA vista na aula **FRA - Aula 12 - 2.5 Resolução de exercício de RNA - Regressão**. Além disso, fazer uma breve explicação dos seguintes resultados:

- Gráficos de avaliação do modelo (loss);
- Métricas de avaliação do modelo (pelo menos uma entre MAE, MSE, R^2).

Informações:

- **Base de dados:** Wine Quality
- **Descrição**: O objetivo deste dataset prever a qualidade dos vinhos com base em suas características químicas. A variável target (y) neste exemplo será o score de qualidade do vinho, que varia de 0 (pior qualidade) a 10 (melhor qualidade)
- **Tamanho**: 1599 amostras, 12 features.
- **Importação**: Copiar código abaixo.

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
data = pd.read_csv(url, delimiter=';')
```

Dica 1. Para facilitar o trabalho, renomeie o nome das colunas para português, dessa forma:

```
data.columns = [
    'acidez_fixa',            # fixed acidity
    'acidez_volatil',         # volatile acidity
    'acido_citrico',          # citric acid
    'acucar_residual',        # residual sugar
    'cloretos',               # chlorides
    'dioxido_de_enxofre_livre', # free sulfur dioxide
    'dioxido_de_enxofre_total', # total sulfur dioxide
    'densidade',              # density
    'pH',                     # pH
    'sulfatos',               # sulphates
    'alcool',                 # alcohol
    'score_qualidade_vinho'              # quality
]
```

Dica 2. Separe os dados (x e y) de tal forma que a última coluna (índice -1), chamada score_qualidade_vinho, seja a variável target (y)

3 Sistemas de Recomendação

Implementar o exemplo de Sistemas de Recomendação usando a base de dados Base_livos.csv e a arquitetura vista na aula **FRA - Aula 22 - 4.3 Resolução do Exercício de Sistemas de Recomendação**. Além disso, fazer uma breve explicação dos seguintes resultados:

- Gráficos de avaliação do modelo (loss);
- Exemplo de recomendação de livro para determinado Usuário.

Informações:

- **Base de dados:** Base_livros.csv
- **Descrição**: Esse conjunto de dados contém informações sobre avaliações de livros (Notas), nomes de livros (Titulo), ISBN e identificação do usuário (ID_usuario)
- **Importação:** Base de dados disponível no Moodle (UFPR Virtual), chamada Base_livros (formato .csv).

4 Deepdream

Implementar o exemplo de implementação mínima de Deepdream usando uma imagem de um felino - retirada do site Wikipedia - e a arquitetura Deepdream vista na aula **FRA - Aula 23 - Prática Deepdream**. Além disso, fazer uma breve explicação dos seguintes resultados:

- Imagem onírica obtida por *Main Loop*;
- Imagem onírica obtida ao levar o modelo até uma oitava;
- Diferenças entre imagens oníricas obtidas com *Main Loop* e levando o modelo até a oitava.

Informações:

- **Base de dados:** https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg
- **Importação da imagem**: Copiar código abaixo.

```
url = "https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.jpg"
```

```
Dica: Para exibir a imagem utilizando display (display.html) use o link https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg
```

B – RESOLUÇÃO

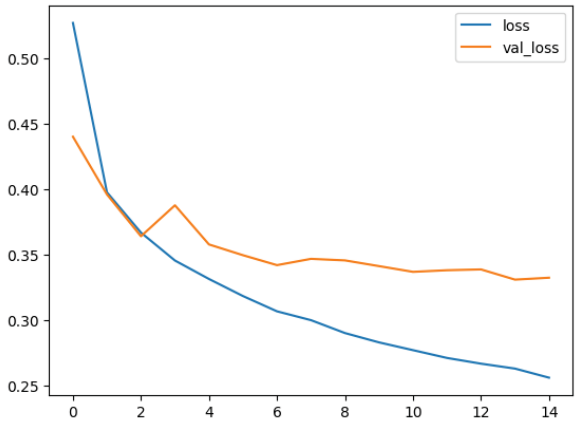
1 Classificação (RNA)

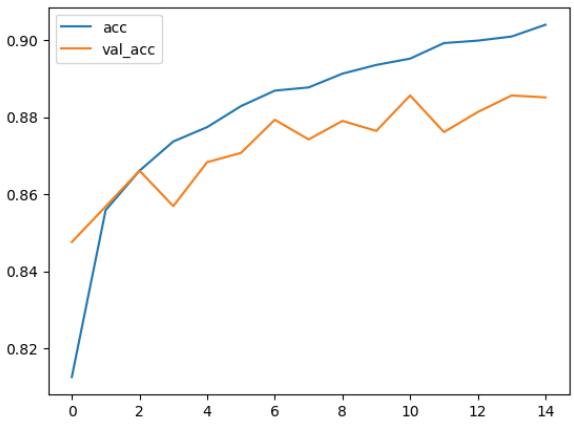
```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix

data = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = data.load_data()
x_train, x_test = x_train/255.0, x_test/255.0
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation= "softmax")(x)
model = tf.keras.models.Model(i, x)
model.compile(optimizer= 'adam',
loss= 'sparse_categorical_crossentropy',
metrics= ['accuracy'])

r = model.fit(
    x_train, y_train,
    validation_data=(x_test, y_test),
    epochs=15
)

# Função de perda
plt.plot(r.history["loss"], label="loss")
```

```
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# Acurácia
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()

# Avaliar o modelo com a base de teste
print(model.evaluate(x_test, y_test))

y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)

# Matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
plt.show()

# Pega índices das imagens classificadas errado
misclassified = np.where(y_pred != y_test)[0]

# Escolhe uma imagem aleatória entre as erradas
i = np.random.choice(misclassified)

# Mostra a imagem
plt.imshow(x_test[i], cmap="gray")  # já é 28x28, não precisa do reshape
plt.title(f"True label: {y_test[i]} | Predicted: {y_pred[i]}")
plt.axis("off")
plt.show()
```

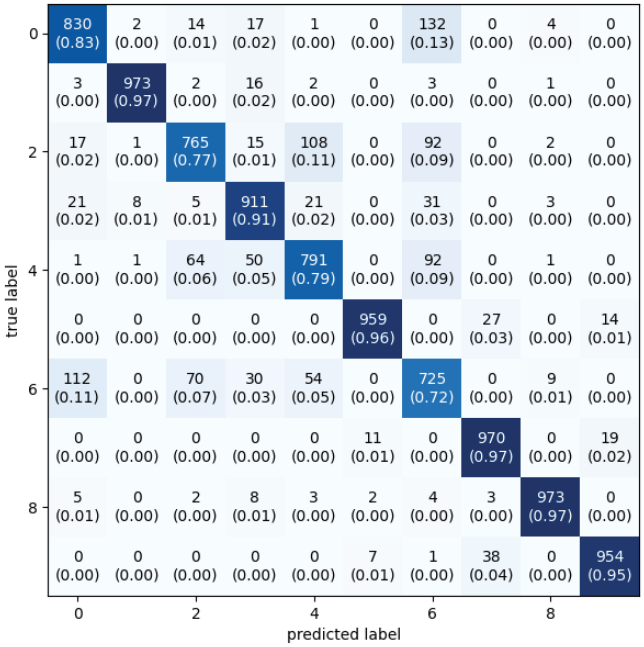
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

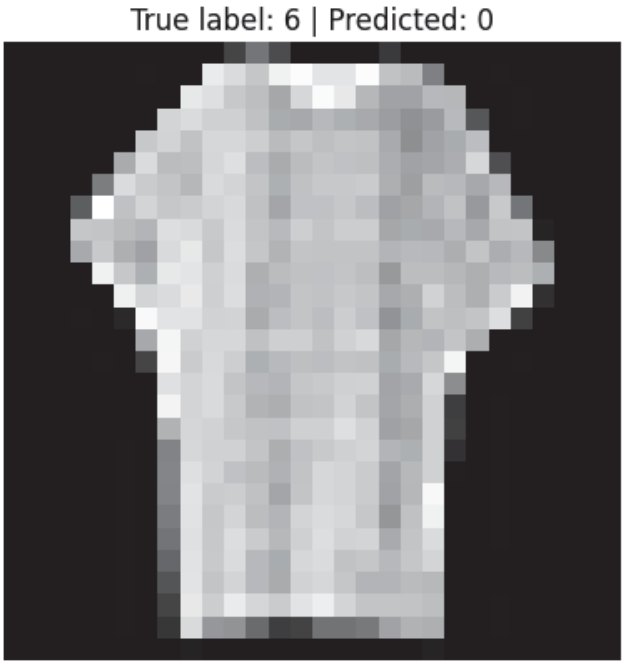
```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/15
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7618 - loss: 0.6778 - val_accuracy: 0.8476 - val_loss: 0.4400
Epoch 2/15
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8536 - loss: 0.4075 - val_accuracy: 0.8568 - val_loss: 0.3959
...
Epoch 9/15
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8945 - loss: 0.2823 - val_accuracy: 0.8790 - val_loss: 0.3455
Epoch 10/15
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8949 - loss: 0.2799 - val_accuracy: 0.8764 - val_loss: 0.3412
...
Epoch 15/15
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9020 - loss: 0.2611 - val_accuracy: 0.8851 - val_loss: 0.3323
```






```
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8866 - loss: 0.3348
[0.3322884440422058, 0.8851000070571899]
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[9 2 1 ... 8 1 5]
```





Os gráficos de perda e acurácia mostram que, a partir de cerca de 12 ou 13 épocas, o modelo atinge valores próximos aos ideais, mas ainda existe diferença entre os dados de treino e validação, indicando possível overfitting e espaço ainda possível otimização com ajustes de arquitetura ou regularização. Na análise de erros, a imagem exibida corresponde a uma peça de roupa da classe verdadeira 6 (Shirt), mas foi prevista como 0 (T-shirt/Top), um erro comum devido à semelhança visual entre essas categorias em imagens de baixa resolução.

2 Regressão (RNA)

```
# Regressão de qualidade do vinho (Wine Quality - Red)
# ----------------------------------------------------
```

```
# Pipeline limpo e comentado:
# 1) Carrega dataset do UCI direto da URL
# 2) Renomeia colunas para português (snake_case) sem espaços
# 3) Embaralha, separa X/y, padroniza features
# 4) Define rede fully-connected para regressão (MSE)
# 5) Implementa métricas RMSE e R² (custom Keras)
# 6) Treina com EarlyStopping monitorando val_loss

import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.utils import shuffle

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras import backend as K

# (Opcional) Reprodutibilidade básica
np.random.seed(42)
tf.random.set_seed(42)

# ---------------------------------------
# 1) Carregar dados do UCI
# ---------------------------------------
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
data = pd.read_csv(url, delimiter=';')

# ---------------------------------------
# 2) Renomear colunas (sem espaços)
#    Obs.: havia um nome com espaço: "dioxido de enxofre total" ->
"dioxido_de_enxofre_total"
# ---------------------------------------
data.columns = [
    'acidez_fixa',                # fixed acidity
    'acidez_volatil',             # volatile acidity
    'acido_citrico',              # citric acid
    'acucar_residual',            # residual sugar
    'cloretos',                   # chlorides
    'dioxido_de_enxofre_livre',   # free sulfur dioxide
    'dioxido_de_enxofre_total',   # total sulfur dioxide
    'densidade',                  # density
    'pH',                         # pH
```

```
    'sulfatos',                    # sulphates
    'alcool',                      # alcohol
    'score_qualidade_vinho'        # quality
]

# ---------------------------------------
# 3) Embaralhar, separar X/y e padronizar
# ---------------------------------------
data = shuffle(data, random_state=4)

y = data['score_qualidade_vinho'].astype(float)
X = data.drop('score_qualidade_vinho', axis=1).astype(float)

scaler = StandardScaler()
X = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=42
)

# ---------------------------------------
# 4) Modelo MLP para regressão
# ---------------------------------------
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
import tensorflow as tf

def modelo(input_dim: int) -> Sequential:
    model = Sequential([
        tf.keras.Input(shape=(input_dim,)),   # camada de entrada explícita
        Dense(64, activation='relu'),
        Dense(32, activation='relu'),
        Dense(16, activation='relu'),
        Dense(1)
    ])
    return model

# ---------------------------------------
# 5) Métricas customizadas: RMSE e R²
# ---------------------------------------
def rmse(y_true, y_pred):
    y_true = tf.cast(y_true, dtype=tf.float32)
    return K.sqrt(K.mean(K.square(y_pred - y_true), axis=-1))

def r2(y_true, y_pred):
    y_true = tf.cast(y_true, dtype=tf.float32)
```

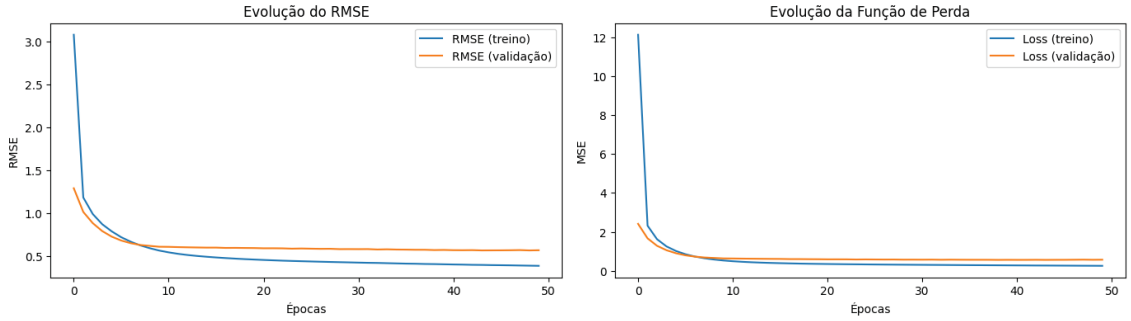
```
    ss_res = K.sum(K.square(y_true - y_pred))
    ss_tot = K.sum(K.square(y_true - K.mean(y_true)))
    return 1.0 - ss_res / (ss_tot + K.epsilon())

# ---------------------------------------
# 6) Compilar e treinar com EarlyStopping
# ---------------------------------------
# Exemplos de otimizadores (deixe um ativo):
# optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
# optimizer = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.5)
optimizer = tf.keras.optimizers.RMSprop(learning_rate=0.001)

model = modelo(X.shape[1])
model.compile(optimizer=optimizer, loss='mse', metrics=[rmse, r2])

early_stop = tf.keras.callbacks.EarlyStopping(
    monitor='val_loss',
    patience=20,
    restore_best_weights=True
)

# Atenção: use EITHER validation_split OR validation_data (não ambos).
history = model.fit(
    X_train, y_train,
    epochs=50,
    batch_size=32,
    validation_data=(X_test, y_test),
    callbacks=[early_stop],
    verbose=1
)

# Avaliação final
eval_metrics = model.evaluate(X_test, y_test, verbose=0)
print(f"Teste - MSE: {eval_metrics[0]:.4f} | RMSE: {eval_metrics[1]:.4f} | R²:
{eval_metrics[2]:.4f}")

# ---------------------------------------
# Gráfico de perda (loss) do treinamento
# ---------------------------------------

# r.fit() retorna um objeto (rf) com o histórico de treino em rf.history
# "loss"      -> erro médio quadrático (MSE) no conjunto de treino
# "val_loss"  -> erro médio quadrático (MSE) no conjunto de validação
# Esse gráfico ajuda a avaliar se há overfitting ou se o modelo ainda pode
melhorar.
```

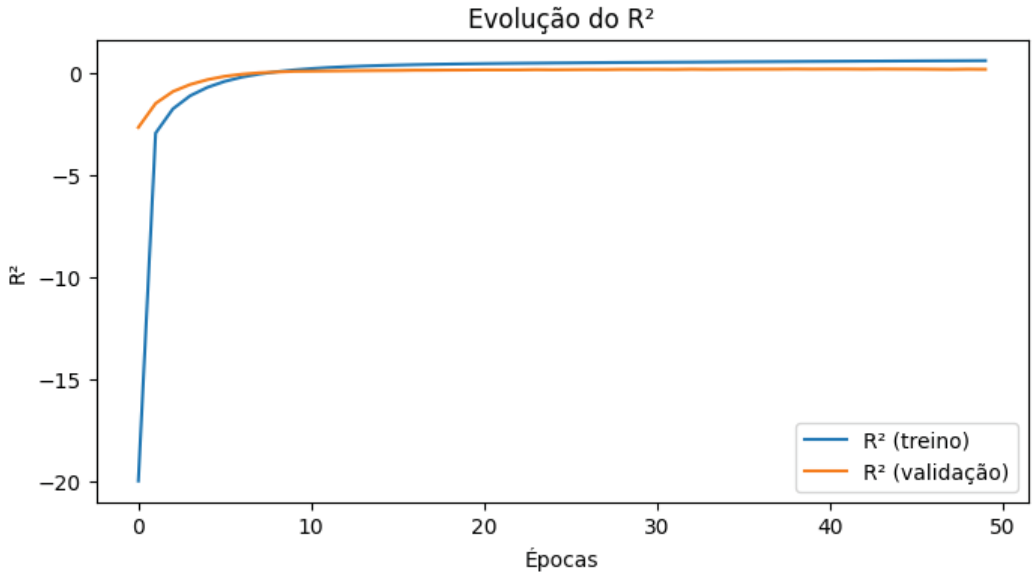
```
# Bibliotecas gráficas
import matplotlib.pyplot as plt

# Para dados e ML
import numpy as np
import pandas as pd
import tensorflow as tf

plt.figure(figsize=(8, 4))
plt.plot(history.history["loss"], label="Loss (treino)")
plt.plot(history.history["val_loss"], label="Loss (validação)")
plt.xlabel("Épocas")
plt.ylabel("MSE")
plt.title("Evolução da Função de Perda")
plt.legend()
plt.show()

# RMSE
plt.figure(figsize=(8,4))
plt.plot(history.history["rmse"], label="RMSE (treino)")
plt.plot(history.history["val_rmse"], label="RMSE (validação)")
plt.xlabel("Épocas")
plt.ylabel("RMSE")
plt.title("Evolução do RMSE")
plt.legend()
plt.show()

# R²
plt.figure(figsize=(8,4))
plt.plot(history.history["r2"], label="R² (treino)")
plt.plot(history.history["val_r2"], label="R² (validação)")
plt.xlabel("Épocas")
plt.ylabel("R²")
plt.title("Evolução do R²")
plt.legend()
plt.show()

from sklearn.metrics import mean_squared_error, r2_score
import numpy as np

# ---------------------------------------
# Predição e avaliação final do modelo
# ---------------------------------------

# Usar o modelo treinado (history ou rf é só o histórico, o modelo é "model" ou
"r")
```

```
y_pred = model.predict(X_test).flatten()

# Cálculo das métricas de regressão
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Resultados
print(f"MSE  = {mse:.4f}")
print(f"RMSE = {rmse:.4f}")
print(f"R²   = {r2:.4f}")
```

```
Epoch 1/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 45ms/step - loss: 19.1189 - r2: -32.1230 - rmse: 4.1308 - val_loss: 2.4115 - val_r2: -2.6780 - val_rmse: 1.2908
Epoch 2/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.5623 - r2: -3.3401 - rmse: 1.2428 - val_loss: 1.6669 - val_r2: -1.5034 - val_rmse: 1.0141
Epoch 3/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 1.7084 - r2: -1.9113 - rmse: 1.0162 - val_loss: 1.2835 - val_r2: -0.9233 - val_rmse: 0.8831
...
Epoch 10/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5330 - r2: 0.1263 - rmse: 0.5655 - val_loss: 0.6351 - val_r2: 0.0641 - val_rmse: 0.6078
Epoch 11/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.4927 - r2: 0.1974 - rmse: 0.5423 - val_loss: 0.6305 - val_r2: 0.0716 - val_rmse: 0.6066
Epoch 12/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.4606 - r2: 0.2547 - rmse: 0.5225 - val_loss: 0.6245 - val_r2: 0.0802 - val_rmse: 0.6034
...
Epoch 30/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.3004 - r2: 0.5236 - rmse: 0.4165 - val_loss: 0.5726 - val_r2: 0.1563 - val_rmse: 0.5804
Epoch 31/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2970 - r2: 0.5292 - rmse: 0.4138 - val_loss: 0.5703 - val_r2: 0.1590 - val_rmse: 0.5798
Epoch 32/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2940 - r2: 0.5338 - rmse: 0.4118 - val_loss: 0.5730 - val_r2: 0.1552 - val_rmse: 0.5804
```

```
...
Epoch 48/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2533 - r2:
0.5960 - rmse: 0.3828 - val_loss: 0.5715 - val_r2: 0.1581 - val_rmse: 0.5689
Epoch 49/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2512 - r2:
0.5996 - rmse: 0.3804 - val_loss: 0.5651 - val_r2: 0.1681 - val_rmse: 0.5650
Epoch 50/50
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2485 - r2:
0.6038 - rmse: 0.3790 - val_loss: 0.5688 - val_r2: 0.1621 - val_rmse: 0.5671
Teste - MSE: 0.5594 | RMSE: 0.5692 | R²: 0.1766
```






```
13/13 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step
MSE  = 0.5594
RMSE = 0.7479
R²   = 0.1817
```

Os gráficos de perda mostram que a função de erro atinge valores próximos do mínimo a partir de cerca de 10-13 épocas. Nas métricas de avaliação, o modelo apresentou MSE de 0,5594, RMSE de 0,7479

e R^2 de 0,1817, indicando que a capacidade de explicação da variância dos dados é baixa e que o ajuste obtido ainda está longe de ser satisfatório para prever com precisão a qualidade do vinho.

3 Sistemas de Recomendação

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.layers import Input, Embedding, Flatten, Concatenate, Dense,
Dropout
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD

# Definir seeds para reprodutibilidade
np.random.seed(42)
tf.random.set_seed(42)

# Carregar o arquivo CSV, ignorando linhas com problemas
df = pd.read_csv("/content/Base_livros(in).csv", on_bad_lines='skip')

# Converter ID_usuario para numérico para garantir comparações corretas
df['ID_usuario'] = pd.to_numeric(df['ID_usuario'], errors='coerce').astype('Int64')

print(df.head())

# Renomear a coluna com sintaxe problemática "Notas;;;;;;" para "Notas"
df.rename(columns={'Notas;;;;;;': 'Notas'}, inplace=True)
print(df.head())

# Remover caracteres ";" da coluna de notas
df['Notas'] = df['Notas'].str.replace(';', '')
print(df.head())

# Transformar a coluna 'Notas' de string para inteiro (int64), tratando erros como
NaN e preenchendo com 0
df['Notas'] = pd.to_numeric(df['Notas'],
errors='coerce').fillna(0).astype(np.int64)
print(df.head())

# Converter colunas categóricas para códigos numéricos para uso em embeddings
# ISBN para new_isbn
df['ISBN'] = pd.Categorical(df['ISBN'])
df['new_isbn'] = df.ISBN.cat.codes

# ID_usuario para new_id_usuario
```

```
df['ID_usuario'] = pd.Categorical(df['ID_usuario'])
df['new_id_usuario'] = df.ID_usuario.cat.codes

# Calcular dimensões: número de usuários únicos (M) e livros únicos (N)
M = df.new_id_usuario.nunique()  # Número de usuários únicos
N = df.new_isbn.nunique()        # Número de livros únicos (ISBNs únicos)
print(f"Número de usuários únicos (M): {M}")
print(f"Número de livros únicos (N): {N}")

# Dimensão do embedding (pode testar outros valores para otimização)
K = 10

# Definir camadas de entrada para usuário e livro (shape=(1,) para índices
individuais)
u = Input(shape=(1,), name='user_input')    # Entrada para ID do usuário
m = Input(shape=(1,), name='book_input')    # Entrada para ISBN do livro

# Camada de embedding para usuário: mapeia IDs de usuários para vetores de dimensão
K
u_emb = Embedding(input_dim=M, output_dim=K)(u)  # Saída: (batch_size, 1, K)
u_emb = Flatten()(u_emb)                         # Saída achatada: (batch_size, K)

# Camada de embedding para livro: mapeia ISBNs para vetores de dimensão K
m_emb = Embedding(input_dim=N, output_dim=K)(m)  # Saída: (batch_size, 1, K)
m_emb = Flatten()(m_emb)                         # Saída achatada: (batch_size, K)

# Concatenar os embeddings de usuário e livro
x = Concatenate()([u_emb, m_emb])  # Saída: (batch_size, 2*K)

# Camadas densas para predição da nota (reduzida para evitar overfitting)
x = Dense(128, activation='relu')(x)  # Camada oculta menor com ReLU
x = Dropout(0.5)(x)                   # Dropout para regularização
x = Dense(1)(x)                       # Camada de saída (nota predita, sem
ativação)

# Criar o modelo
model = Model(inputs=[u, m], outputs=x)
model.summary()  # Opcional: exibir resumo do modelo

# Compilar o modelo com perda MSE e otimizador SGD (taxa de aprendizado reduzida e
clip de gradientes para evitar NaN)
model.compile(
    loss='mse',
    optimizer=SGD(learning_rate=0.01, momentum=0.9, clipnorm=1.0)
)
```

```
# Preparar os dados: extrair arrays numpy e embaralhar índices para divisão
train/test
user_ids = df['new_id_usuario'].values
book_ids = df['new_isbn'].values
ratings = df['Notas'].values

# Embaralhar os índices para randomização
num_samples = len(ratings)
indices = np.arange(num_samples)
np.random.shuffle(indices)

# Dividir em treino (80%) e teste (20%)
Ntrain = int(0.8 * num_samples)
train_indices = indices[:Ntrain]
test_indices = indices[Ntrain:]

train_user = user_ids[train_indices].reshape(-1, 1).astype(np.int32)  # Reshape
para (samples, 1) e tipo int32 para embeddings
train_books = book_ids[train_indices].reshape(-1, 1).astype(np.int32) # Reshape
para (samples, 1) e tipo int32 para embeddings
train_ratings = ratings[train_indices].astype(np.float64)  # Converter para float64
para permitir subtração

test_user = user_ids[test_indices].reshape(-1, 1).astype(np.int32)
test_books = book_ids[test_indices].reshape(-1, 1).astype(np.int32)
test_ratings = ratings[test_indices].astype(np.float64)    # Converter para float64
para permitir subtração

# Centralizar as notas (subtrair a média do treino) para melhorar o treinamento
avg_rating = train_ratings.mean()
train_ratings -= avg_rating
test_ratings -= avg_rating
print(f"Média das notas de treino (antes da centralização): {avg_rating}")

# Treinar o modelo (mais epochs para melhor convergência)
epochs = 50
batch_size = 1024
history = model.fit(
    x=[train_user, train_books],
    y=train_ratings,
    epochs=epochs,
    batch_size=batch_size,
    verbose=2,  # Exibe progresso a cada época
    validation_data=([test_user, test_books], test_ratings)
)
```

```
# Plotar o histórico de perda de treino e validação
plt.plot(history.history["loss"], label="loss")
plt.plot(history.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# Gerar recomendações para um usuário específico (exemplo: ID original 276725)
user_original = 276725
mask = df['ID_usuario'] == user_original
if mask.any():
    user_encoded = df.loc[mask, 'new_id_usuario'].iloc[0]
    print(f"ID codificado do usuário {user_original}: {user_encoded}")
else:
    print(f"Usuário {user_original} não encontrado. Usando o primeiro usuário.")
    user_original = df['ID_usuario'].iloc[0]
    user_encoded = df['new_id_usuario'].iloc[0]
    print(f"Usuário usado: {user_original} (codificado: {user_encoded})")

# Preparar array com o usuário único repetido para todos os livros únicos
input_user_encoded = np.full((N, 1), user_encoded, dtype=np.int32)

# Livros únicos (códigos)
unique_books = np.arange(N).reshape(-1, 1).astype(np.int32)  # Códigos de 0 a N-1

# Obter predições para todos os livros para esse usuário
preds = model.predict([input_user_encoded, unique_books])

# Descentralizar as predições (adicionar de volta a média)
rat = preds.flatten() + avg_rating

# Top 5 recomendações (códigos de livros e notas)
top_indices = np.argsort(rat)[-5:][::-1]
print("Recomendação específica para usuário: Para esse usuário específico os
seguintes livros foram indicados:")
for i, book_code in enumerate(top_indices):
    score = rat[book_code]
    print(f"{i+1}. Livro código {book_code} com uma nota de {score:.1f}.")

# Opcional: salvar o histórico de treinamento para análise posterior
# print(history.history)
```

```
       ISBN                                          Titulo  \
0   2005018                                    Clara Callan
1  60973129                                 Decision in Normandy
2  374157065  Flu: The Story of the Great Influenza Pandemic...
3  393045218                               The Mummies of Urumchi
4  399135782                               The Kitchen God's Wife

                Autor   Ano                Editora  ID_usuario Notas;;;;;;
0  Richard Bruce Wright  2001   HarperFlamingo Canada      276725     0;;;;;;
1         Carlo D'Este  1991         HarperPerennial      276726     2;;;;;;
2     Gina Bari Kolata  1999    Farrar Straus Giroux      276727     6;;;;;;
3      E. J. W. Barber  1999  W. W. Norton & Company      276729     1;;;;;;
4              Amy Tan  1991        Putnam Pub Group      276729     9;;;;;;
       ISBN                                          Titulo  \
0   2005018                                    Clara Callan
1  60973129                                 Decision in Normandy
2  374157065  Flu: The Story of the Great Influenza Pandemic...
3  393045218                               The Mummies of Urumchi
4  399135782                               The Kitchen God's Wife

                Autor   Ano                Editora  ID_usuario    Notas
0  Richard Bruce Wright  2001   HarperFlamingo Canada      276725  0;;;;;;
1         Carlo D'Este  1991         HarperPerennial      276726  2;;;;;;
2     Gina Bari Kolata  1999    Farrar Straus Giroux      276727  6;;;;;;
3      E. J. W. Barber  1999  W. W. Norton & Company      276729  1;;;;;;
4              Amy Tan  1991        Putnam Pub Group      276729  9;;;;;;
       ISBN                                          Titulo  \
0   2005018                                    Clara Callan
1  60973129                                 Decision in Normandy
2  374157065  Flu: The Story of the Great Influenza Pandemic...
3  393045218                               The Mummies of Urumchi
4  399135782                               The Kitchen God's Wife

                Autor   Ano                Editora  ID_usuario Notas
0  Richard Bruce Wright  2001   HarperFlamingo Canada      276725     0
1         Carlo D'Este  1991         HarperPerennial      276726     2
2     Gina Bari Kolata  1999    Farrar Straus Giroux      276727     6
3      E. J. W. Barber  1999  W. W. Norton & Company      276729     1
4              Amy Tan  1991        Putnam Pub Group      276729     9
       ISBN                                          Titulo  \
0   2005018                                    Clara Callan
1  60973129                                 Decision in Normandy
2  374157065  Flu: The Story of the Great Influenza Pandemic...
3  393045218                               The Mummies of Urumchi
4  399135782                               The Kitchen God's Wife

                Autor   Ano                Editora  ID_usuario  Notas
0  Richard Bruce Wright  2001   HarperFlamingo Canada      276725      0
1         Carlo D'Este  1991         HarperPerennial      276726      2
2     Gina Bari Kolata  1999    Farrar Straus Giroux      276727      6
3      E. J. W. Barber  1999  W. W. Norton & Company      276729      1
4              Amy Tan  1991        Putnam Pub Group      276729      9
Número de usuários únicos (M): 10852
Número de livros únicos (N): 128867
```

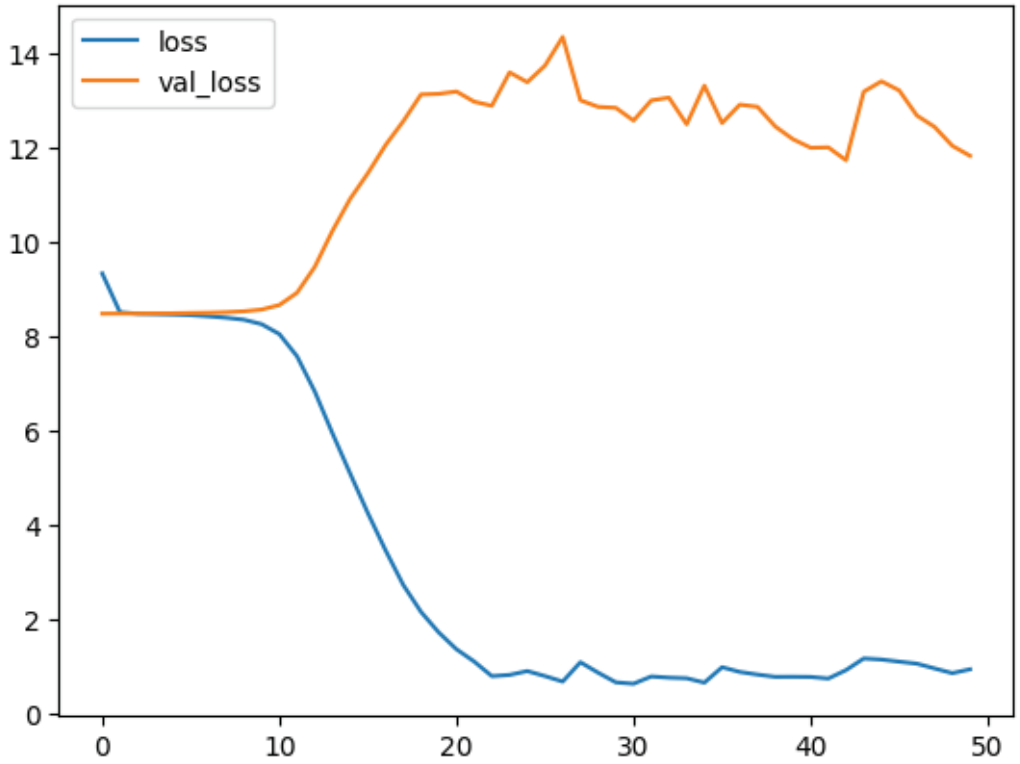
Model: "functional_25"

Layer (type)	Output Shape	Param #	Connected to
user_input (InputLayer)	(None, 1)	0	-
book_input (InputLayer)	(None, 1)	0	-
embedding_26 (Embedding)	(None, 1, 10)	108,520	user_input[0][0]
embedding_27 (Embedding)	(None, 1, 10)	1,288,670	book_input[0][0]
flatten_66 (Flatten)	(None, 10)	0	embedding_26[0][…
flatten_67 (Flatten)	(None, 10)	0	embedding_27[0][…
concatenate_22 (Concatenate)	(None, 20)	0	flatten_66[0][0], flatten_67[0][0]
dense_43 (Dense)	(None, 128)	2,688	concatenate_22[0…
dropout (Dropout)	(None, 128)	0	dense_43[0][0]
dense_44 (Dense)	(None, 1)	129	dropout[0][0]

Total params: 1,400,007 (5.34 MB)
Trainable params: 1,400,007 (5.34 MB)
Non-trainable params: 0 (0.00 B)

Total params: 1,400,007 (5.34 MB)

```
 Trainable params: 1,400,007 (5.34 MB)
 Non-trainable params: 0 (0.00 B)
Média das notas de treino (antes da centralização): 4.210118920596737
Epoch 1/50
101/101 - 6s - 62ms/step - loss: 9.3333 - val_loss: 8.4836
Epoch 2/50
101/101 - 0s - 3ms/step - loss: 8.5118 - val_loss: 8.4856
Epoch 3/50
101/101 - 0s - 3ms/step - loss: 8.4817 - val_loss: 8.4842
...
Epoch 10/50
101/101 - 0s - 3ms/step - loss: 8.2608 - val_loss: 8.5701
Epoch 11/50
101/101 - 0s - 3ms/step - loss: 8.0513 - val_loss: 8.6663
Epoch 12/50
101/101 - 0s - 3ms/step - loss: 7.5801 - val_loss: 8.9250
...
Epoch 30/50
101/101 - 1s - 6ms/step - loss: 0.6777 - val_loss: 12.8412
Epoch 31/50
101/101 - 1s - 5ms/step - loss: 0.6484 - val_loss: 12.5710
Epoch 32/50
101/101 - 0s - 5ms/step - loss: 0.8020 - val_loss: 12.9983
...
Epoch 48/50
101/101 - 0s - 4ms/step - loss: 0.9721 - val_loss: 12.4365
Epoch 49/50
101/101 - 0s - 3ms/step - loss: 0.8724 - val_loss: 12.0362
Epoch 50/50
101/101 - 0s - 3ms/step - loss: 0.9544 - val_loss: 11.8250
```




```
ID codificado do usuário 276725: 10075
4028/4028 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step
Recomendação específica para usuário: Para esse usuário específico os
seguintes livros foram indicados:
1. Livro código 17110 com uma nota de 11.8.
2. Livro código 127896 com uma nota de 11.7.
3. Livro código 107741 com uma nota de 11.6.
4. Livro código 10732 com uma nota de 11.5.
5. Livro código 12043 com uma nota de 11.5.
```

Meu modelo aprendeu bem no treino (a loss caiu de ~8,5 para ~0,6–1,2), mas “decorou” demais e foi mal na validação (val_loss subiu até ~13–14 e fechou em 11,825; o melhor ponto parece entre as épocas 10–12), o que indica overfitting. Para melhorar, poderia ser usado um early stopping (parar quando a validação piorar), mais regularização nos embeddings (ou reduzir seu tamanho), trocar o otimizador para Adam com taxa menor... Também vale comparar com baselines simples (média por usuário/item) e ajustar a escala das notas (padronizar/clipping). Nas recomendações, para o usuário 276725 (ID 10075), os top 5 livros foram 17110 (11,8), 127896 (11,7), 107741 (11,6), 10732 (11,5) e 12043 (11,5).

4 Deepdream

```
# ========== Imports ==========
import io, requests, numpy as np, PIL.Image
from IPython import display
```

```
import tensorflow as tf

# ========== Utilitários ==========
URL = "https://upload.wikimedia.org/wikipedia/commons/b/b6/Felis_catus-
cat_on_snow.jpg"  # Imagem de felino (Wikipedia)

def download_image(url, max_dim=None):
    """Baixa a imagem com cabeçalho (evita 403) e retorna np.array RGB."""
    headers = {"User-Agent": "Mozilla/5.0 (Colab; Linux x86_64) AppleWebKit/537.36
Safari/537.36"}
    r = requests.get(url, headers=headers, timeout=30)
    r.raise_for_status()
    img = PIL.Image.open(io.BytesIO(r.content)).convert("RGB")
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)

def show(img, title=""):
    if title:
        print(f"\n{title}")
    display.display(PIL.Image.fromarray(np.array(img)))

def deprocess(img):
    """Volta de [-1,1] para [0,255]"""
    img = 255.0 * (img + 1.0) / 2.0
    return tf.cast(img, tf.uint8)

# ========== Modelo DeepDream ==========
base_model = tf.keras.applications.InceptionV3(include_top=False,
weights="imagenet")
names = ['mixed3', 'mixed5']
outputs = [base_model.get_layer(n).output for n in names]
dream_model = tf.keras.Model(inputs=base_model.input, outputs=outputs)

def calc_loss(img, model):
    img_batch = tf.expand_dims(img, 0)
    activs = model(img_batch)
    if not isinstance(activs, (list, tuple)):
        activs = [activs]
    return tf.add_n([tf.reduce_mean(a) for a in activs])

class DeepDream(tf.Module):
    def __init__(self, model):
        super().__init__()
        self.model = model
```

```
    @tf.function(
      input_signature=(tf.TensorSpec([None, None, 3], tf.float32),
                       tf.TensorSpec([], tf.int32),
                       tf.TensorSpec([], tf.float32))
    )
    def __call__(self, img, steps, step_size):
        for _ in tf.range(steps):
            with tf.GradientTape() as tape:
                tape.watch(img)
                loss = calc_loss(img, self.model)
            grads = tape.gradient(loss, img)
            grads /= (tf.math.reduce_std(grads) + 1e-8)
            img = tf.clip_by_value(img + grads * step_size, -1.0, 1.0)
        return loss, img

deepdream = DeepDream(dream_model)

def run_deep_dream_simple(img_np, steps=80, step_size=0.01):
    """Main Loop (sem mudar escala)."""
    img = tf.convert_to_tensor(img_np, dtype=tf.float32)
    img = tf.keras.applications.inception_v3.preprocess_input(img)
    loss, img = deepdream(img, tf.constant(steps), tf.constant(step_size))
    return deprocess(img)

# ========== Execução ==========

# 1) Imagem original
original = download_image(URL, max_dim=500)
show(original, title="Imagem original (felino - Wikimedia)")

# 2) Imagem onírica - Main Loop
dream_main = run_deep_dream_simple(original, steps=80, step_size=0.01)
show(dream_main, title="Imagem onírica obtida por Main Loop")

# 3) Imagem onírica - Levando o modelo a UMA oitava
OCTAVE_SCALE = 1.30
h, w = original.shape[:2]
new_h, new_w = int(h * OCTAVE_SCALE), int(w * OCTAVE_SCALE)

oct_img = tf.image.resize(tf.convert_to_tensor(original), (new_h,
new_w)).numpy().astype(np.uint8)
dream_oct = run_deep_dream_simple(oct_img, steps=60, step_size=0.01)
dream_oct_back = tf.image.resize(tf.cast(dream_oct, tf.float32), (h, w))
dream_oct_back = tf.cast(tf.clip_by_value(dream_oct_back, 0.0, 255.0),
tf.uint8).numpy()
show(dream_oct_back, title="Imagem onírica obtida levando o modelo até uma oitava")
```

```
# Crédito
display.display(display.HTML(
    '<p>Imagem original: '
    '<a href="https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg"
target="_blank">'
    'Felis catus - cat on snow (Wikimedia Commons)</a></p>'
))
```

Imagem original (felino - Wikimedia)

Imagem onírica obtida por Main Loop

Imagem onírica obtida levando o modelo até uma oitava

Imagem original: Felis catus - cat on snow (Wikimedia Commons)

A imagem original do felino serve como referência limpa. No Main Loop, o DeepDream só “puxa” os padrões já presentes na escala original, então surgem texturas finas e detalhes locais (contornos mais fortes, arabescos discretos). Quando levamos o modelo a uma oitava (zoom de ~1,3× antes de processar e depois voltamos), o algoritmo passa a enxergar tudo num tamanho diferente, reforçando padrões maiores e mudando a “granulação” do efeito. Em termos práticos: Main Loop = detalhe miúdo e localizado; 1 oitava = formas mais amplas e espalhadas.

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

Escolha um conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada) e faça uma visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha)

Desenvolva uma narrativa/storytelling para essa visualização de dados considerando os conceitos e informações que foram discutidas nesta disciplina. Não esqueça de deixar claro para seu possível público-alvo qual **o objetivo dessa visualização de dados, o que esses dados significam, quais possíveis ações podem ser feitas com base neles.**

Entregue em um PDF:

- O **conjunto de dados brutos** (**ou uma visualização de dados** que você acredite que possa ser **melhorada**);

- Explicação do **contexto e o público-alvo** da visualização de dados e do storytelling que será desenvolvido;

- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada**; (**50 pontos**)

B – RESOLUÇÃO

Análise Estatística e Definição de Tolerâncias para Testes de Placas Eletrônicas

---

Introdução

A qualidade e confiabilidade de placas eletrônicas são fundamentais para o desempenho dos produtos finais. Utilizando um conjunto de dados brutos de medições elétricas, esta análise tem como objetivo definir as tolerâncias ideais para cada parâmetro testado. Isso permitirá classificar placas como aptas ou inaptas para o mercado, garantindo maior controle de qualidade.

Conjunto de Dados

Os dados foram coletados a partir de testes realizados em um lote de placas eletrônicas, incluindo medições de diferentes tensões e sinais. A tabela abaixo apresenta os parâmetros avaliados:

- 5V em 72V
- 12V em 72V
- 16V em 72V
- -16V em 72V
- B25/B26(V) em 72V S/ Carga
- B25/B26(V) em 72V C/ Carga
- B25/B26(V) em 250V S/ Carga
- AC/KVM

Os dados foram analisados estatisticamente para calcular média, desvio padrão e definir limites superiores e inferiores de tolerância para cada parâmetro.

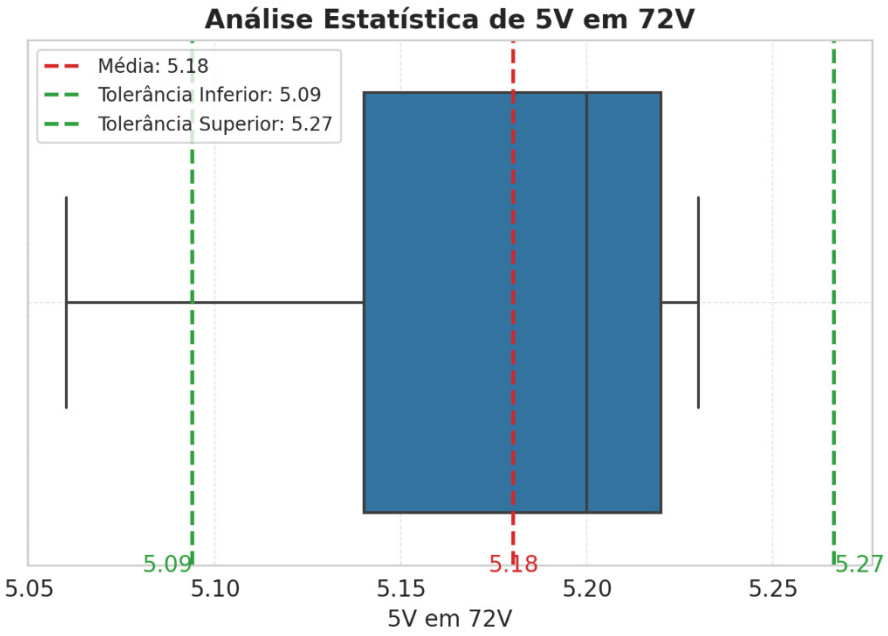
Parâmetro	count	mean	std	min	25%	50%	75%	max	Limite Inferior	Limite Superior
5V em 72V	135.00	5.18	0.04	5.06	5.14	5.20	5.22	5.23	5.09	5.27
12V em 72V	135.00	12.40	0.10	12.15	12.38	12.43	12.47	12.60	12.19	12.61
16V em 72V	135.00	15.64	0.14	15.30	15.59	15.67	15.74	15.89	15.37	15.92
-16V em 72V	135.00	-15.65	0.14	-15.86	-15.75	-15.70	-15.58	-15.31	-15.92	-15.37
B25/B26(V) em 72V S/ Carga	135.00	38.46	2.45	21.55	38.55	38.85	39.05	40.19	33.57	43.35
B25/B26(V) em 72V C/ Carga	135.00	86.93	20.55	36.94	94.66	95.40	96.32	97.66	45.83	128.04
B25/B26(V) em 250V S/ Carga	135.00	132.15	2.86	125.91	130.21	131.43	133.39	142.53	126.43	137.88
AC/KVM	135.00	2.42	0.21	2.20	2.26	2.32	2.55	2.99	1.99	2.84

Visualizações dos Dados

Os gráficos a seguir apresentam uma análise estatística detalhada de cada parâmetro. As visualizações incluem um boxplot com marcações da média (linha vermelha) e dos limites de tolerância (linhas verdes). Isso facilita a identificação de valores fora do intervalo esperado, auxiliando na decisão sobre a aceitação ou rejeição das placas.

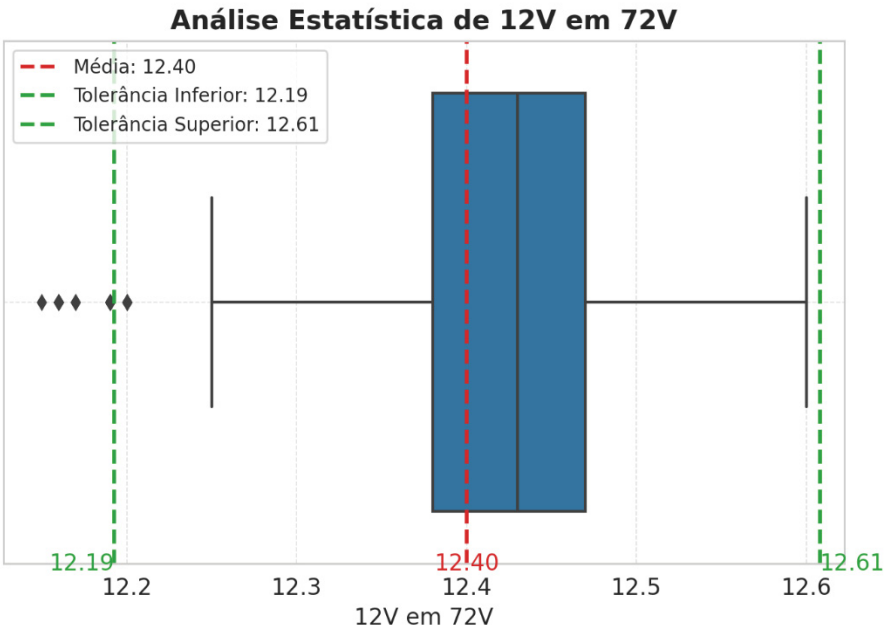
5V em 72V

O gráfico a seguir mostra a distribuição dos valores do parâmetro '5V em 72V', com a média, limite inferior e superior destacados para referência.



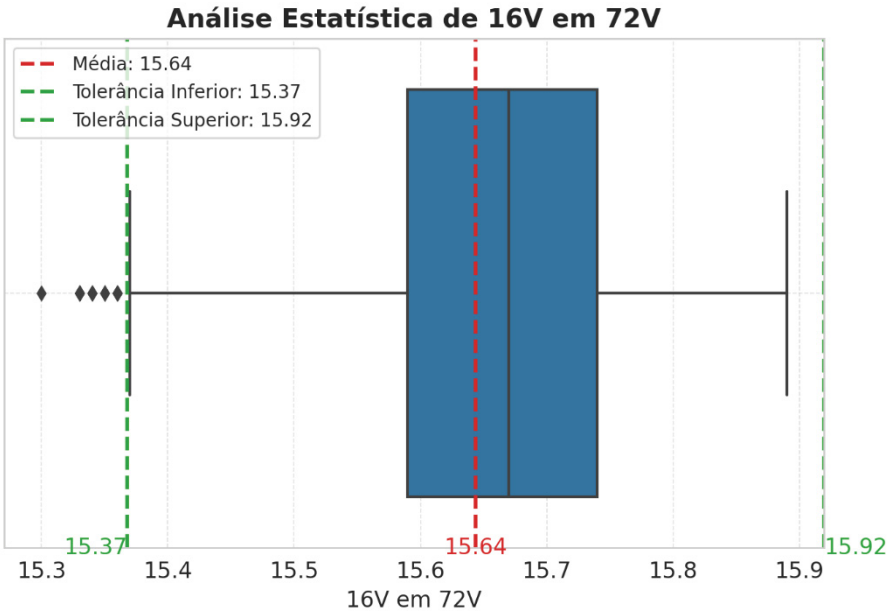
12V em 72V

O gráfico a seguir mostra a distribuição dos valores do parâmetro '12V em 72V', com a média, limite inferior e superior destacados para referência.



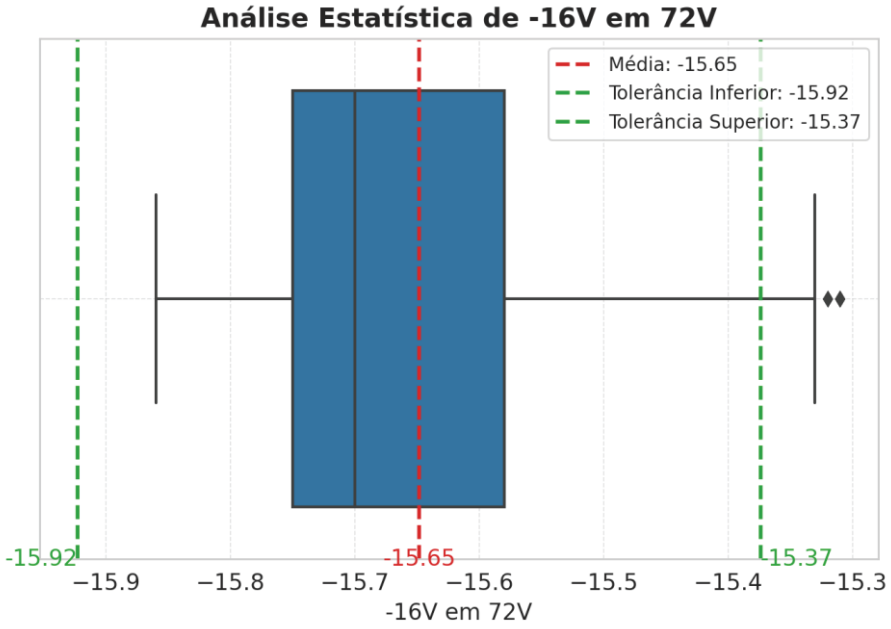
16V em 72V

O gráfico a seguir mostra a distribuição dos valores do parâmetro '16V em 72V', com a média, limite inferior e superior destacados para referência.



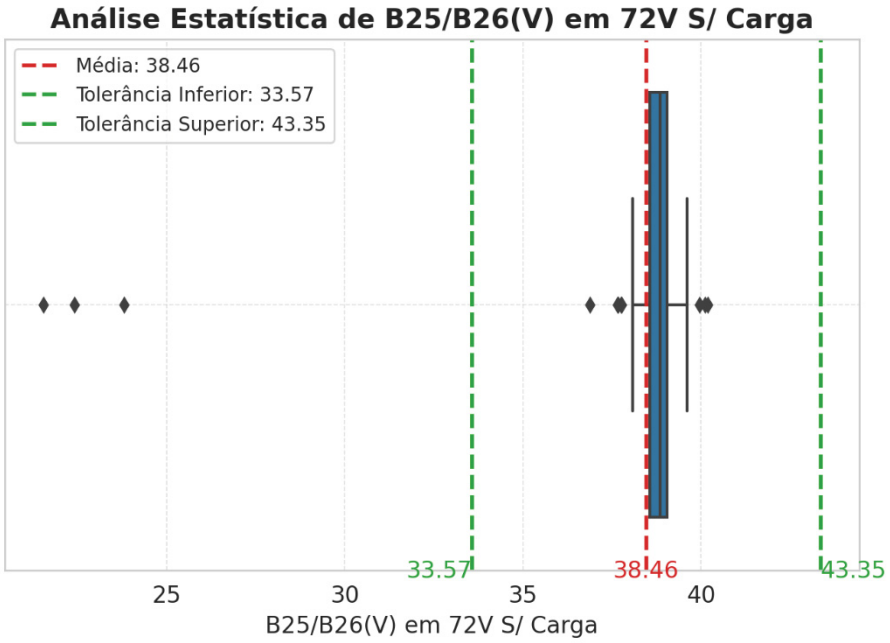
-16V em 72V

O gráfico a seguir mostra a distribuição dos valores do parâmetro '-16V em 72V', com a média, limite inferior e superior destacados para referência.



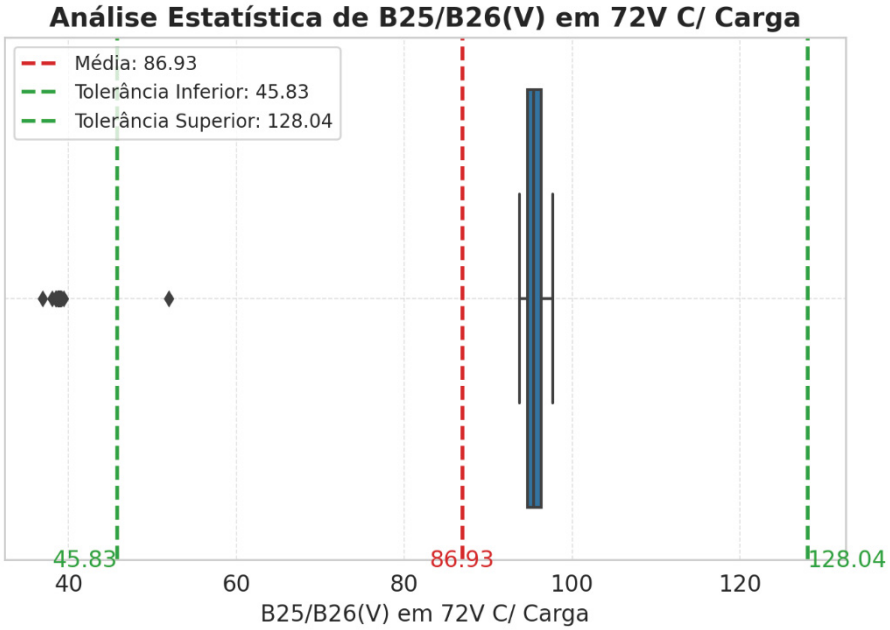
B25/B26(V) em 72V S/ Carga

O gráfico a seguir mostra a distribuição dos valores do parâmetro 'B25/B26(V) em 72V S/ Carga', com a média, limite inferior e superior destacados para referência.



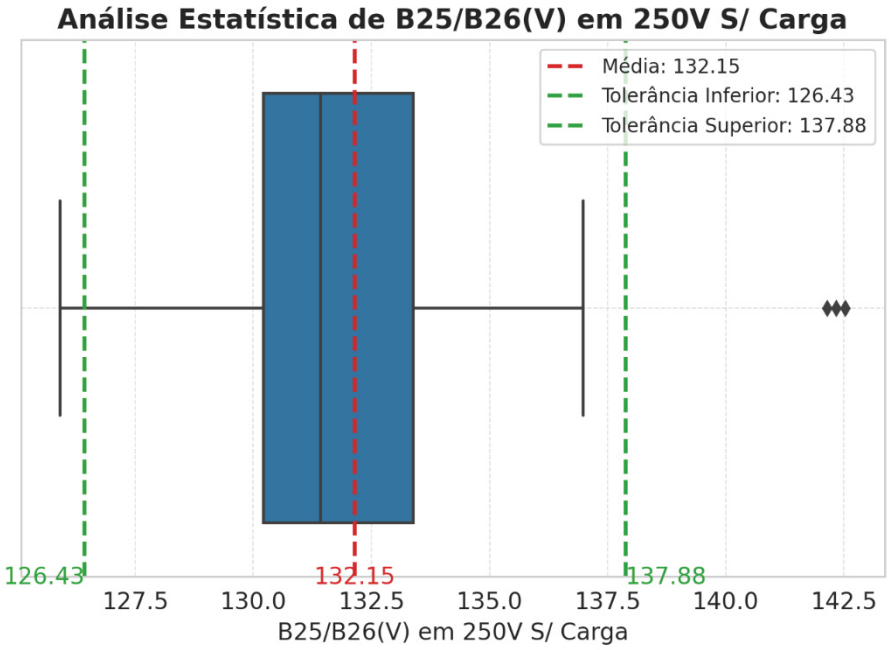
B25/B26(V) em 72V C/ Carga

O gráfico a seguir mostra a distribuição dos valores do parâmetro 'B25/B26(V) em 72V C/ Carga', com a média, limite inferior e superior destacados para referência.



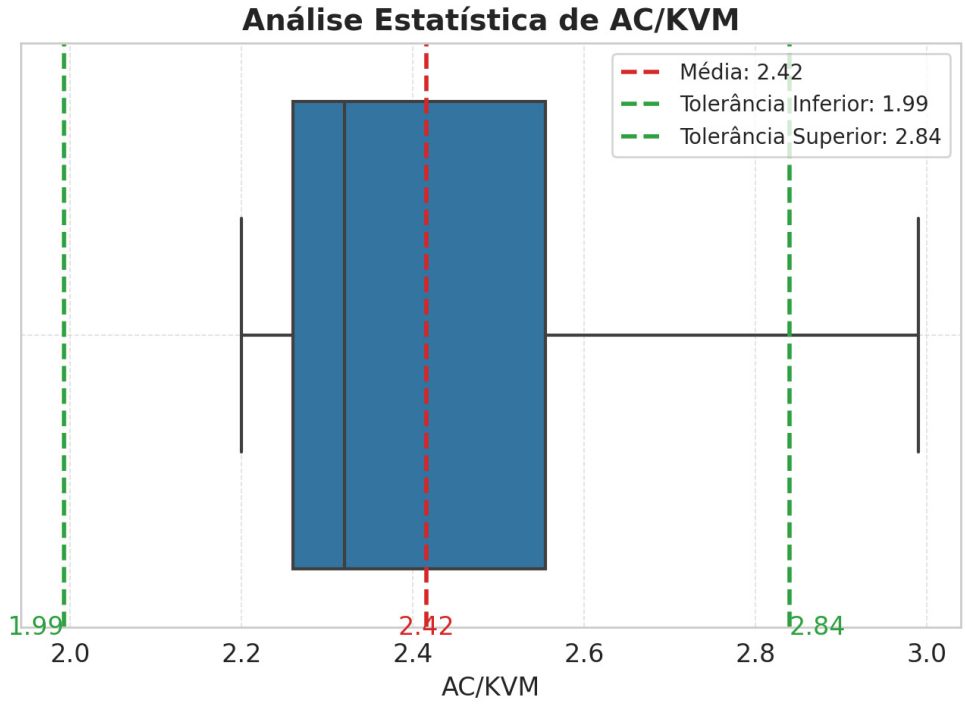
B25/B26(V) em 250V S/ Carga

O gráfico a seguir mostra a distribuição dos valores do parâmetro 'B25/B26(V) em 250V S/ Carga', com a média, limite inferior e superior destacados para referência.



AC/KVM

O gráfico a seguir mostra a distribuição dos valores do parâmetro 'AC/KVM', com a média, limite inferior e superior destacados para referência.



Storytelling: A Jornada da Qualidade

A análise de dados não se trata apenas de números, mas sim de garantir a excelência do produto final. Com a visualização das métricas, conseguimos transformar dados brutos em informações acionáveis.

Cada outlier detectado representa uma oportunidade de melhoria, permitindo ajustes nos processos de fabricação para garantir que apenas placas dentro das especificações cheguem ao mercado.

Conclusão e Próximos Passos

Com base nesta análise, é possível definir um critério de aceitação para as placas testadas. O próximo passo é implementar em nosso sistema automatizado as tolerâncias de validação com base nesses limites, reduzindo erros manuais e aumentando a eficiência na triagem dos produtos.

Esta abordagem não só melhora a qualidade das placas eletrônicas, como também reduz custos com devoluções e reclamações, fortalecendo a reputação da marca e a satisfação dos clientes.

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

(1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
(2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
(3) Utilize no mínimo uma população com 100 indivíduos;
(4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
(5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
(6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

*) Curiosidade

Fiquei curioso quanto a aplicação de vetorização para o problema do caixeiro viajante, então neste experimento, eu explorei duas abordagens diferentes para resolver o clássico Problema do Caixeiro Viajante, envolvendo 100 cidades com coordenadas geradas aleatoriamente. Na primeira abordagem, utilizei o método tradicional do vizinho mais próximo, baseado unicamente nas distâncias euclidianas entre os pontos no espaço 2D. Já na segunda abordagem, representei cada cidade como um vetor de 200 dimensões, combinando distâncias e ângulos em relação a todas as outras cidades. A seguir, apliquei técnicas de redução de dimensionalidade com PCA para transformar esses vetores provenientes de um espaço de maior dimensionalidade, para um espaço 2D. Em ambos os casos, refinei os caminhos obtidos com o algoritmo 2-opt, buscando percursos mais curtos. O objetivo foi compreender se a modelagem vetorial ao incorporar mais relações estruturais entre as cidades poderia gerar resultados tão eficientes quanto as técnicas já consolidadas para o tratamento desse tipo de problema. Ao adotar essa representação mais rica, percebi que ela pode ser explorada também a outras abordagens de forma promissoras em substituição ou em paralelo com convencionais buscando melhores resultados.

```
# Importando bibliotecas necessárias
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler

# Configurando a semente para reprodutibilidade
np.random.seed(42)
```

```
# 1. Gerando as 100 cidades com coordenadas aleatórias em um espaço 100x100
num_cidades = 100
cidades = np.random.uniform(0, 100, (num_cidades, 2))  # Coordenadas (x, y)

# 2. Criando vetores relacionais (distâncias e ângulos entre cidades)
distancias = np.zeros((num_cidades, num_cidades))
angulos = np.zeros((num_cidades, num_cidades))

for i in range(num_cidades):
    for j in range(num_cidades):
        distancias[i, j] = np.linalg.norm(cidades[i] - cidades[j])
        if i != j:
            vetor = cidades[j] - cidades[i]
            angulos[i, j] = np.arctan2(vetor[1], vetor[0])
        else:
            angulos[i, j] = 0

# Normalizando as distâncias e os ângulos
scaler = MinMaxScaler()
distancias_normalizadas = scaler.fit_transform(distancias)
angulos_normalizados = scaler.fit_transform(angulos)

# Combinando distâncias e ângulos em um vetor de 200 dimensões
vetores = np.hstack((distancias_normalizadas, angulos_normalizados))

# 3. Reduzindo os vetores para 4D e depois para 2D usando PCA
pca_4d = PCA(n_components=4)
vetores_4d = pca_4d.fit_transform(vetores)
pca_2d = PCA(n_components=2)
vetores_2d = pca_2d.fit_transform(vetores_4d)

# Função para calcular a distância total de um caminho
def calcular_distancia_caminho(caminho, cidades):
    distancia = 0
    for i in range(len(caminho) - 1):
        cidade_atual = caminho[i]
        proxima_cidade = caminho[i + 1]
        distancia += np.linalg.norm(cidades[cidade_atual] -
cidades[proxima_cidade])
    distancia += np.linalg.norm(cidades[caminho[-1]] - cidades[caminho[0]])
    return distancia

# 4. Vizinho Mais Próximo no espaço 4D (abordagem vetorial)
def vizinho_mais_proximo(vetores, espaco="4D"):
    caminho = [0]
```

```
    cidades_nao_visitadas = list(range(1, num_cidades))

    while cidades_nao_visitadas:
        cidade_atual = caminho[-1]
        if espaco == "4D":
            distancias = [np.linalg.norm(vetores[cidade_atual] - vetores[cidade])
for cidade in cidades_nao_visitadas]
        else:  # Espaço original (2D)
            distancias = [np.linalg.norm(cidades[cidade_atual] - cidades[cidade])
for cidade in cidades_nao_visitadas]
        proxima_cidade_idx = np.argmin(distancias)
        proxima_cidade = cidades_nao_visitadas[proxima_cidade_idx]
        caminho.append(proxima_cidade)
        cidades_nao_visitadas.pop(proxima_cidade_idx)

    return caminho

# 5. Algoritmo 2-opt para refinar o caminho
def dois_opt(caminho, cidades, max_iter=1000):
    melhor_caminho = caminho.copy()
    melhor_distancia = calcular_distancia_caminho(melhor_caminho, cidades)
    melhoria = True
    iteracao = 0

    while melhoria and iteracao < max_iter:
        melhoria = False
        for i in range(1, len(caminho) - 2):
            for j in range(i + 1, len(caminho)):
                if j - i == 1:
                    continue
                novo_caminho = melhor_caminho[:i] + melhor_caminho[i:j][::-1] +
melhor_caminho[j:]
                nova_distancia = calcular_distancia_caminho(novo_caminho, cidades)
                if nova_distancia < melhor_distancia:
                    melhor_caminho = novo_caminho
                    melhor_distancia = nova_distancia
                    melhoria = True
        iteracao += 1

    return melhor_caminho, melhor_distancia

# 6. Gerando os caminhos
# Abordagem vetorial: Vizinho Mais Próximo no espaço 4D
caminho_vetorial_inicial = vizinho_mais_proximo(vetores_4d, espaco="4D")
distancia_vetorial_inicial = calcular_distancia_caminho(caminho_vetorial_inicial,
cidades)
```

```
caminho_vetorial_refinado, distancia_vetorial_refinada =
dois_opt(caminho_vetorial_inicial, cidades)

# Abordagem tradicional: Vizinho Mais Próximo no espaço original (2D)
caminho_tradicional_inicial = vizinho_mais_proximo(cidades, espaco="2D")
distancia_tradicional_inicial =
calcular_distancia_caminho(caminho_tradicional_inicial, cidades)
caminho_tradicional_refinado, distancia_tradicional_refinada =
dois_opt(caminho_tradicional_inicial, cidades)

# 7. Visualização dos resultados
def plotar_caminho(cidades, caminho, titulo, nome_arquivo):
    plt.figure(figsize=(8, 8))
    plt.scatter(cidades[:, 0], cidades[:, 1], c='blue', label='Cidades')
    plt.scatter(cidades[0, 0], cidades[0, 1], c='red', label='Cidade Inicial',
marker='*', s=200)

    for i in range(len(caminho) - 1):
        cidade_atual = caminho[i]
        proxima_cidade = caminho[i + 1]
        plt.plot([cidades[cidade_atual, 0], cidades[proxima_cidade, 0]],
                 [cidades[cidade_atual, 1], cidades[proxima_cidade, 1]], 'k-')
    plt.plot([cidades[caminho[-1], 0], cidades[caminho[0], 0]],
             [cidades[caminho[-1], 1], cidades[caminho[0], 1]], 'k-')

    plt.title(titulo)
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.legend()
    plt.grid(True)
    plt.savefig(nome_arquivo, dpi=300, bbox_inches='tight')
    plt.show()

# Plotando os caminhos (vetorial)
plotar_caminho(cidades, caminho_vetorial_inicial, f"Abordagem Vetorial: Caminho
Inicial (Vizinho Mais Próximo - 4D)\nDistância: {distancia_vetorial_inicial:.2f}",
"caminho_vetorial_inicial.png")
plotar_caminho(cidades, caminho_vetorial_refinado, f"Abordagem Vetorial: Caminho
Refinado (Após 2-opt)\nDistância: {distancia_vetorial_refinada:.2f}",
"caminho_vetorial_refinado.png")

# Plotando os caminhos (tradicional)
plotar_caminho(cidades, caminho_tradicional_inicial, f"Abordagem Tradicional:
Caminho Inicial (Vizinho Mais Próximo - 2D)\nDistância:
{distancia_tradicional_inicial:.2f}", "caminho_tradicional_inicial.png")
plotar_caminho(cidades, caminho_tradicional_refinado, f"Abordagem Tradicional:
```

```
Caminho Refinado (Após 2-opt)\nDistância: {distancia_tradicional_refinada:.2f}",
"caminho_tradicional_refinado.png")

# Plotando as cidades no plano ortonormal (vetores projetados em 2D)
plt.figure(figsize=(8, 8))
ax = plt.gca()
for i in range(num_cidades):
    x, y = vetores_2d[i, 0], vetores_2d[i, 1]
    ax.arrow(0, 0, x, y, head_width=0.1, head_length=0.2, fc='blue', ec='blue')
    plt.text(x + 0.2, y + 0.2, f"Cidade {i}", fontsize=8)

plt.title("Projeção PCA das Cidades no Plano Ortonormal (de 4D para 2D)")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
max_limit = np.max(np.abs(vetores_2d)) + 1
plt.xlim(-max_limit, max_limit)
plt.ylim(-max_limit, max_limit)
ax.axhline(0, color='black', linewidth=0.5, linestyle='--')
ax.axvline(0, color='black', linewidth=0.5, linestyle='--')
plt.grid(True)
plt.savefig("projecao_pca_cidades_4d_to_2d.png", dpi=300, bbox_inches='tight')
plt.show()

# Comparando os resultados
print("### Comparação dos Resultados ###")
print(f"Abordagem Vetorial - Distância Inicial (Vizinho Mais Próximo 4D):
{distancia_vetorial_inicial:.2f}")
print(f"Abordagem Vetorial - Distância Refinada (Após 2-opt):
{distancia_vetorial_refinada:.2f}")
print(f"Abordagem Tradicional - Distância Inicial (Vizinho Mais Próximo 2D):
{distancia_tradicional_inicial:.2f}")
print(f"Abordagem Tradicional - Distância Refinada (Após 2-opt):
{distancia_tradicional_refinada:.2f}")
```

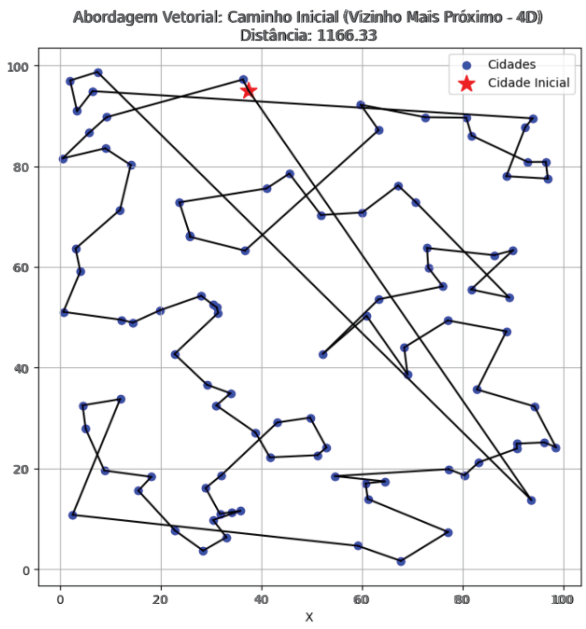
Comparação dos Resultados

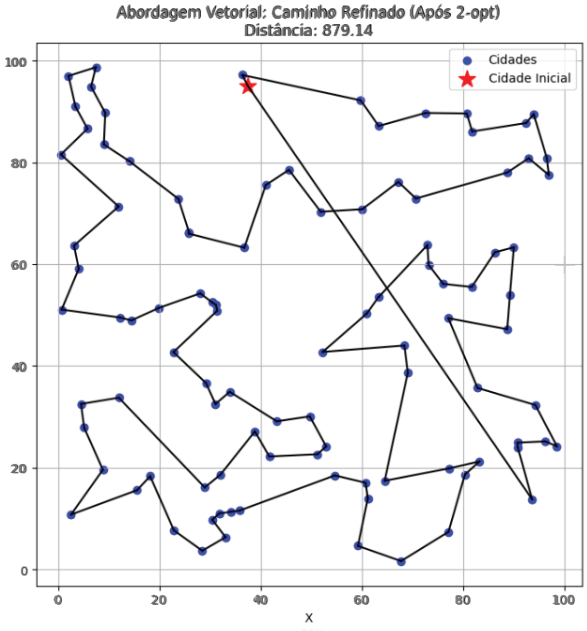
Abordagem Vetorial - Distância Inicial (Vizinho Mais Próximo 4D): 1166.33

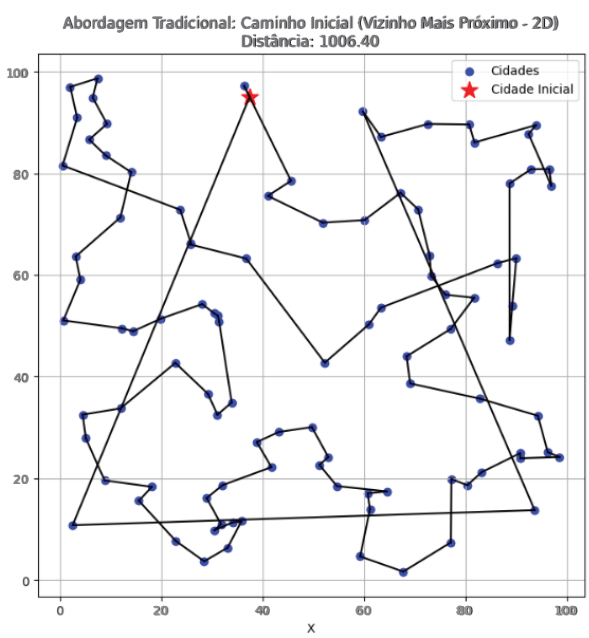
Abordagem Vetorial - Distância Refinada (Após 2-opt): 879.14

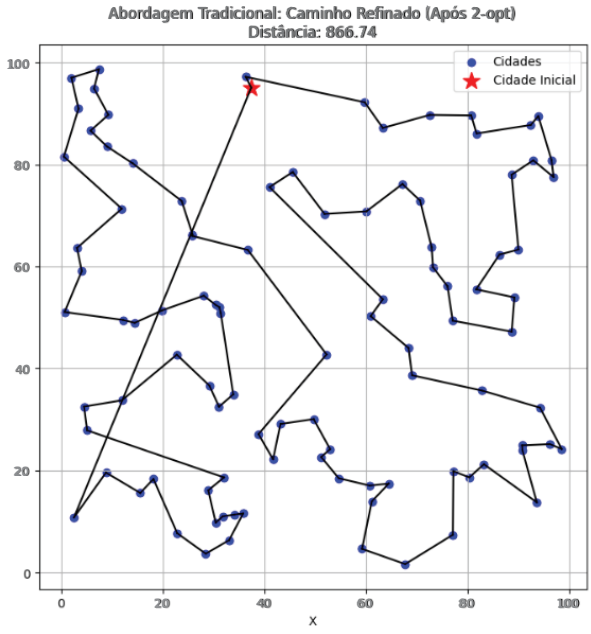
Abordagem Tradicional - Distância Inicial (Vizinho Mais Próximo 2D): 1006.40

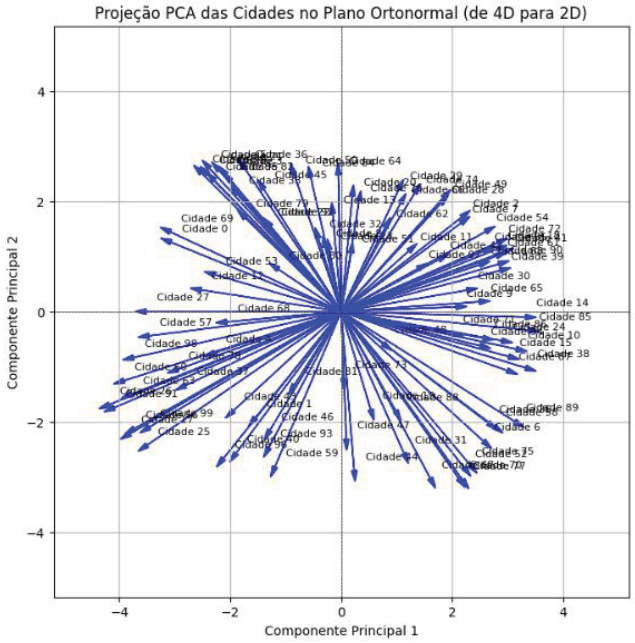
Abordagem Tradicional - Distância Refinada (Após 2-opt): 866.74


Abordagem Vetorial: Caminho Inicial (Vizinho Mais Próximo - 4D)
Distância: 1166.33
Cidades
Cidade Inicial
X


Abordagem Vetorial: Caminho Refinado (Após 2-opt)
Distância: 879.14
Cidades
Cidade Inicial
X


Abordagem Tradicional: Caminho Inicial (Vizinho Mais Próximo - 2D)
Distância: 1006.40
Cidades
Cidade Inicial
X


Abordagem Tradicional: Caminho Refinado (Após 2-opt)
Distância: 866.74
Cidades
Cidade Inicial
X



1) Algoritmo Genético

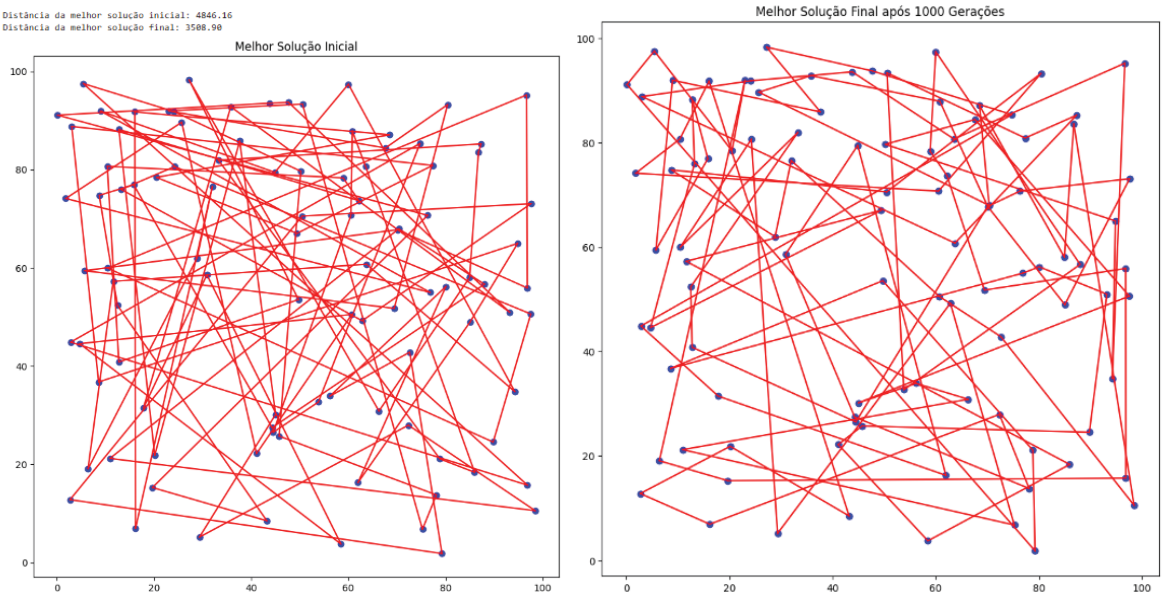
```
import numpy as np
import matplotlib.pyplot as plt
import random

# Gerar coordenadas aleatórias para 100 cidades
def gerar_cidades(n_cidades=100, tamanho=100):
    return np.random.rand(n_cidades, 2) * tamanho

# Calcular distância euclidiana entre duas cidades
def calcular_distancia(cidade1, cidade2):
    return np.sqrt(np.sum((cidade1 - cidade2) ** 2))

# Calcular distância total de um percurso
def calcular_distancia_total(percurso, cidades):
    distancia = 0
    for i in range(len(percurso) - 1):
        distancia += calcular_distancia(cidades[percurso[i]], cidades[percurso[i +
1]])
    # Adicionar retorno à cidade inicial
    distancia += calcular_distancia(cidades[percurso[-1]], cidades[percurso[0]])
    return distancia

# Criar população inicial
def criar_populacao(tamanho_pop, n_cidades):
    populacao = []
```

```
    for _ in range(tamanho_pop):
        # Cidade 0 é fixa no início e fim
        caminho = list(range(1, n_cidades))
        random.shuffle(caminho)
        caminho = [0] + caminho + [0]
        populacao.append(caminho)
    return populacao

# Operador de cruzamento OX (Order Crossover)
def crossover_ox(pai1, pai2):
    tamanho = len(pai1) - 1  # -1 por causa da cidade final
    inicio, fim = sorted(random.sample(range(1, tamanho-1), 2))

    filho = [-1] * tamanho
    # Copiar segmento do pai1
    for i in range(inicio, fim + 1):
        filho[i] = pai1[i]

    # Preencher com genes do pai2
    pos_atual = fim + 1
    for gene in pai2[1:-1]:  # Ignorar primeiro e último (cidade 0)
        if gene not in filho:
            if pos_atual >= tamanho:
                pos_atual = 1
            filho[pos_atual] = gene
            pos_atual += 1

    return [0] + filho + [0]

# Operador de mutação (swap)
def mutacao(individuo):
    pos1, pos2 = random.sample(range(1, len(individuo)-1), 2)
    individuo[pos1], individuo[pos2] = individuo[pos2], individuo[pos1]
    return individuo

# Algoritmo Genético
def algoritmo_genetico(cidades, tam_pop=100, geracoes=1000, taxa_mutacao=0.01,
taxa_crossover=0.9):
    populacao = criar_populacao(tam_pop, len(cidades))

    # Avaliar população inicial
    distancias = [calcular_distancia_total(ind, cidades) for ind in populacao]
    melhor_inicial = populacao[np.argmin(distancias)]
    melhor_distancia_inicial = min(distancias)

    melhor_solucao = melhor_inicial.copy()
```

```
    melhor_distancia = melhor_distancia_inicial

    for geracao in range(geracoes):
        nova_populacao = [melhor_solucao]  # Preservar melhor solução

        # Elitismo + novos indivíduos
        while len(nova_populacao) < tam_pop:
            # Seleção por torneio
            pais = random.sample(populacao, 2)
            dist1 = calcular_distancia_total(pais[0], cidades)
            dist2 = calcular_distancia_total(pais[1], cidades)
            pai1 = pais[0] if dist1 < dist2 else pais[1]

            if random.random() < taxa_crossover and len(nova_populacao) < tam_pop *
0.9:
                pais = random.sample(populacao, 2)
                filho = crossover_ox(pais[0], pais[1])
                nova_populacao.append(filho)
            else:
                individuo = random.choice(populacao).copy()
                if random.random() < taxa_mutacao:
                    individuo = mutacao(individuo)
                nova_populacao.append(individuo)

        populacao = nova_populacao[:tam_pop]
        distancias = [calcular_distancia_total(ind, cidades) for ind in populacao]

        idx_melhor = np.argmin(distancias)
        if distancias[idx_melhor] < melhor_distancia:
            melhor_solucao = populacao[idx_melhor].copy()
            melhor_distancia = distancias[idx_melhor]

    return melhor_inicial, melhor_distancia_inicial, melhor_solucao,
melhor_distancia

# Plotar resultado
def plotar_caminho(cidades, caminho, titulo):
    plt.figure(figsize=(10, 10))
    plt.scatter(cidades[:, 0], cidades[:, 1], c='blue')
    for i in range(len(caminho)-1):
        c1, c2 = caminho[i], caminho[i+1]
        plt.plot([cidades[c1, 0], cidades[c2, 0]],
                 [cidades[c1, 1], cidades[c2, 1]], 'r-')
    plt.title(titulo)
    plt.show()
```

```
# Executar
cidades = gerar_cidades()
melhor_inicial, dist_inicial, melhor_final, dist_final =
algoritmo_genetico(cidades)

print(f"Distância da melhor solução inicial: {dist_inicial:.2f}")
print(f"Distância da melhor solução final: {dist_final:.2f}")

plotar_caminho(cidades, melhor_inicial, "Melhor Solução Inicial")
plotar_caminho(cidades, melhor_final, "Melhor Solução Final após 1000 Gerações")
```

```
Distância da melhor solução inicial: 4846.16
Distância da melhor solução final: 3508.90
```



2) Compare a representação de dois modelos vetoriais

```
# Importando bibliotecas necessárias
import numpy as np
import matplotlib.pyplot as plt
from sentence_transformers import SentenceTransformer
from sklearn.decomposition import PCA
import re

# Poema "Maria Tinha um Carneirinho" integral
poema = """
Maria tinha um carneirinho
De lã branquinha como a neve
Ele seguia-a para a escola
O que era contra as regras
As crianças riam muito
Vendo o carneiro a brincar
```

```
Maria gostava do seu amigo
Que nunca a deixava sozinha
"""

# Tokenização: quebrando o poema em palavras (tokens)
# Remove pontuação, converte para minúsculas e separa por espaços
tokens = re.findall(r'\b\w+\b', poema.lower())
# Removendo tokens duplicados para evitar redundância no gráfico
tokens_unicos = list(dict.fromkeys(tokens))

# Selecionando pelo menos 6 tokens e no máximo 10 para manter a visualização clara
tokens_selecionados = tokens_unicos[:8]
print("Tokens selecionados:", tokens_selecionados)

# Carregando um modelo pré-treinado para gerar embeddings
modelo = SentenceTransformer('paraphrase-multilingual-MiniLM-L12-v2')

# Gerando vetores para cada token
vetores = modelo.encode(tokens_selecionados)

# Reduzindo a dimensionalidade para 2D usando PCA
pca = PCA(n_components=2)
vetores_2d = pca.fit_transform(vetores)

# Criando a figura para plotar os vetores no plano ortonormal
plt.figure(figsize=(8, 8))
ax = plt.gca()

# Plotando cada vetor como uma seta a partir da origem (0,0)
for i, token in enumerate(tokens_selecionados):
    x, y = vetores_2d[i, 0], vetores_2d[i, 1]
    # Desenha a seta (vetor) a partir da origem
    ax.arrow(0, 0, x, y, head_width=0.1, head_length=0.2, fc=f'C{i}', ec=f'C{i}',
label=token)
    # Adiciona o rótulo do token próximo ao final do vetor
    plt.text(x + 0.2, y + 0.2, token, fontsize=9)

# Configurações do gráfico
plt.title("Projeção PCA dos Vetores dos Tokens (Plano Ortonormal)")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")

# Definindo limites para melhor visualização
max_limit = np.max(np.abs(vetores_2d)) + 1
plt.xlim(-max_limit, max_limit)
plt.ylim(-max_limit, max_limit)
```

```
# Adicionando eixos ortonormais
ax.axhline(0, color='black', linewidth=0.5, linestyle='--')
ax.axvline(0, color='black', linewidth=0.5, linestyle='--')

# Adicionando grade
plt.grid(True)

# Adicionando legenda
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# Ajustando o layout
plt.tight_layout()

# Salvando a figura como imagem
plt.savefig("projecao_pca_tokens.png", dpi=300, bbox_inches='tight')
plt.show()
```