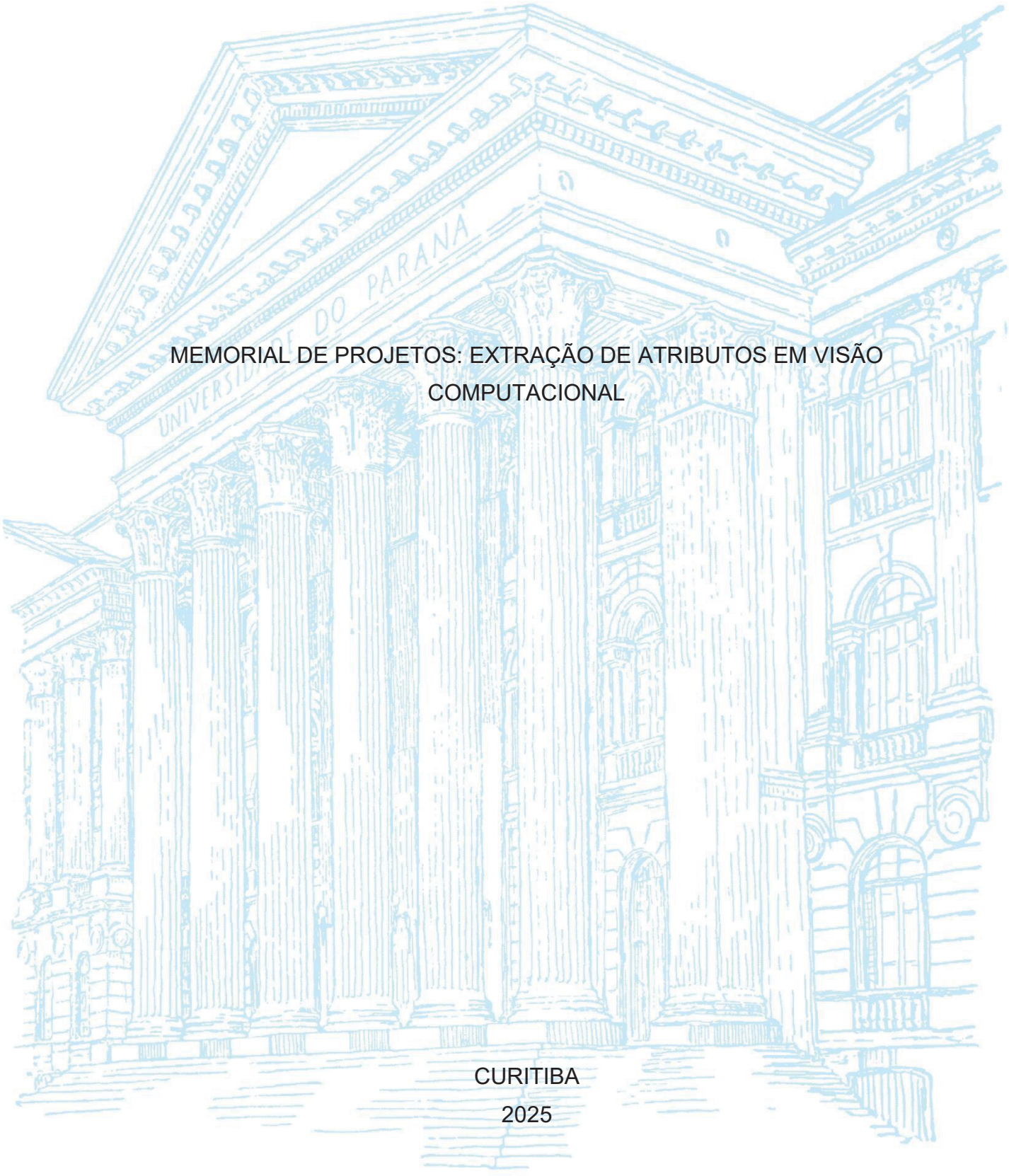
UNIVERSIDADE FEDERAL DO PARANÁ

EVERTON BELARMINO DA SILVA



MEMORIAL DE PROJETOS: EXTRAÇÃO DE ATRIBUTOS EM VISÃO COMPUTACIONAL

CURITIBA

2025

EVERTON BELARMINO DA SILVA

MEMORIAL DE PROJETOS: EXTRAÇÃO DE ATRIBUTOS EM VISÃO COMPUTACIONAL

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 **EVERTON BELARMINO DA SILVA**, intitulada: **MEMORIAL DE PROJETOS: EXTRAÇÃO DE ATRIBUTOS EM VISÃO COMPUTACIONAL**, 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, 29 de Setembro de 2025.

RAZER ANTHON NIZER ROJAS MONTANO
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

AGRADECIMENTOS

A realização deste trabalho acadêmico representa a concretização de um ciclo de muito aprendizado e dedicação, e seria impossível sem o apoio e a contribuição de pessoas essenciais.

Primeiramente, desejo expressar minha profunda gratidão à minha família. Seu amor incondicional, compreensão e incentivo constante foram o alicerce fundamental para que eu pudesse superar os desafios e manter o foco em meus objetivos. A paciência e o suporte nos momentos de maior exigência foram inestimáveis.

Aos meus professores, que com sua paixão pelo ensino, conhecimento vasto e orientação precisa, guiaram-me ao longo desta jornada acadêmica. Em especial, agradeço ao Professor Dr. Razer Anthom N. R. Montaño, pela confiança depositada em meu trabalho, pela valiosa mentoria e pelas discussões enriquecedoras que foram cruciais para o desenvolvimento deste estudo. Seu direcionamento foi um pilar fundamental.

Por fim, estendo meus agradecimentos aos colegas de trabalho e colegas de sala pelo ambiente colaborativo e pelas trocas de ideias que tanto agregaram que, de diversas formas, contribuíram para o sucesso deste projeto. A colaboração e o apoio de todos foram indispensáveis para a conclusão bem-sucedida deste trabalho.

A cada um de vocês, meu sincero muito obrigado.

RESUMO

A extração de atributos é um pilar da Visão Computacional, servindo como uma ponte que transforma dados brutos de imagem — volumosos e não estruturados — em representações compactas e significativas. Este processo é essencial para que as máquinas possam executar tarefas complexas como reconhecimento de objetos, segmentação e navegação autônoma. A principal motivação para essa extração é a *maldição da dimensionalidade*, um fenômeno onde a altíssima dimensão dos dados de imagem prejudica a eficiência dos algoritmos de aprendizado de máquina, tornando os dados esparsos e os modelos menos precisos. Historicamente, a área evoluiu de métodos manuais e heurísticos para abordagens automáticas impulsionadas pelo *Deep Learning*, que representou uma revolução. As redes neurais profundas são capazes de aprender hierarquias de características ricas e complexas diretamente dos dados, superando as limitações dos métodos antigos que descartavam muita informação para manter a tratabilidade computacional. No entanto, o poder e a escala das técnicas modernas de *Deep Learning* trazem desafios éticos e práticos significativos e interconectados. Questões como o viés algorítmico (importado de grandes conjuntos de dados), a privacidade, a segurança contra ataques adversariais, a falta de interpretabilidade dos modelos (o problema da *caixa-preta*) e o alto consumo de energia formam uma teia complexa de consequências. A busca por maior precisão muitas vezes agrava esses outros problemas. Portanto, embora a evolução da extração de atributos represente um notável avanço tecnológico, ela impõe a responsabilidade de gerenciar suas profundas e interligadas consequências sociais e técnicas.

Palavras-chave: visão computacional; pixel; extração de atributos; aprendizagem profunda; maldição da dimensionalidade.

ABSTRACT

Feature extraction is a cornerstone of Computer Vision, serving as a bridge that transforms raw image data — which is bulky and unstructured — into compact and meaningful representations. This process is essential for machines to perform complex tasks such as object recognition, segmentation, and autonomous navigation. The primary motivation for this extraction is the "curse of dimensionality," a phenomenon where the extremely high dimension of image data impairs the efficiency of machine learning algorithms, making the data sparse and the models less accurate. Historically, the field has evolved from manual and heuristic methods to automated approaches driven by Deep Learning, which marked a revolution. Deep neural networks are capable of learning rich and complex hierarchies of features directly from data, overcoming the limitations of older methods that discarded too much information to maintain computational tractability. However, the power and scale of modern Deep Learning techniques bring significant and interconnected ethical and practical challenges. Issues such as algorithmic bias (imported from large datasets), privacy, security against adversarial attacks, the lack of model interpretability (the "black box" problem), and high energy consumption form a complex web of consequences. The pursuit of greater accuracy often exacerbates these other problems. Therefore, although the evolution of feature extraction represents a remarkable technological advancement, it imposes the responsibility of managing its profound and interlinked social and technical consequences.

Keywords: computer vision; pixel; feature extraction; deep learning; curse of dimensionality.

SUMÁRIO

1 PARECER TÉCNICO

A extração de atributos constitui um pilar fundamental na Visão Computacional, servindo como a ponte conceitual e algorítmica entre os dados brutos de imagem, inerentemente não estruturados e de alta dimensionalidade, e as representações significativas e de alto nível que permitem a compreensão computacional (Szeliski, 2022). Esta etapa é crucial para a capacidade dos sistemas de entender o mundo visual, possibilitando a execução de tarefas complexas como reconhecimento de objetos, detecção de padrões, segmentação de imagens e navegação autônoma (Forsyth; Ponce, 2012).

Historicamente, a área evoluiu de métodos manuais e heurísticos, que demandavam grande expertise humana para o *design* de características, para abordagens automáticas e hierárquicas impulsionadas pelo *Deep Learning*. Essa transição marcou uma revolução na forma como as máquinas interpretam o conteúdo visual, expandindo significativamente as fronteiras da aplicabilidade da Visão Computacional em diversos setores, desde a medicina à automação industrial (Szeliski, 2022).

Na sua essência, a Visão Computacional busca desenvolver os métodos algorítmicos e as representações que permitem a uma máquina adquirir, processar, analisar e compreender dados visuais para produzir representações, descrições e interpretações significativas do mundo (Szeliski, 2022). Os dados brutos, na forma de imagens digitais, são matrizes de valores de pixel que, embora contenham toda a informação visual, são excessivamente volumosos e redundantes para uso direto em tarefas de alto nível (Bishop, 2006). A extração de atributos é, portanto, o processo transformacional que converte esses dados brutos em um formato mais compacto, informativo e tratável computacionalmente.

Este processo pode ser formalmente entendido como um passo de concentração de informação (Bishop, 2006). Ele visa reduzir a dimensionalidade massiva dos dados de imagem para um conjunto gerenciável de vetores de atributos, descartando informações redundantes e focando em entidades que são invariantes a condições variáveis, como ponto de vista e iluminação (Forsyth; Ponce, 2012). Esta transformação é o que permite que sistemas computacionais realizem tarefas de nível médio e alto, como segmentação, descrição de objetos e reconhecimento de cenas, que estão no cerne da Visão Computacional (Szeliski, 2022). Sem essa etapa de abstração, os algoritmos de aprendizado de máquina seriam sobrecarregados pela complexidade e pela escala dos dados de pixel brutos, tornando tarefas como o reconhecimento de objetos praticamente inviáveis.

A principal motivação teórica para a extração de atributos reside no fenômeno conhecido como a maldição da dimensionalidade (*curse of dimensionality*) (Liu; Lin, 2023). Este termo descreve uma série de problemas que surgem ao trabalhar com dados em espaços de alta dimensão. O espaço de todas as imagens possíveis é um exemplo primordial de um espaço de altíssima dimensão; uma imagem modesta de 256x256 pixels em tons de cinza já corresponde a um ponto em um espaço com 65.536 dimensões. Nesses espaços vastos, os dados tornam-se inerentemente esparsos (Liu; Lin, 2023). A distância entre quaisquer dois pontos tende a se tornar quase uniforme, tornando ineficazes os métodos baseados em vizinhança como o k-NN, além disso a quantidade de dados necessária para cobrir adequadamente o espaço e treinar um modelo generalizável cresce exponencialmente com o número de dimensões.

Na prática, isso significa que, à medida que a dimensionalidade dos dados aumenta, o desempenho dos classificadores e de outros modelos de aprendizado de máquina pode se degradar, um fenômeno contraintuitivo que destaca a necessidade de uma representação mais eficiente (Van Der Maaten; Postma; Van Den Herik, 2009). A extração de atributos aborda diretamente essa maldição, seu objetivo é mapear os dados de um espaço de alta dimensão para um espaço de baixa dimensão que seja mais eficaz para a tarefa em questão, eliminando informações redundantes ou irrelevantes e preservando as características mais salientes e discriminativas (Van Der Maaten; Postma; Van Den Herik, 2009). Ao fazer isso, não apenas se reduz o custo computacional e de armazenamento, mas também melhora a eficiência e a precisão dos algoritmos subsequentes, tornando o aprendizado de máquina em dados visuais uma tarefa tratável.

A extração de atributos é um processo que cria um novo conjunto de atributos a partir do conjunto original, isso é feito derivando informações do conjunto de características existente para construir um novo subespaço de atributos de menor dimensão (Raschka; Mirjalili, 2017). As novas características são tipicamente combinações ou transformações das características originais, este processo de transformação visa capturar a essência da informação em uma forma mais compacta.

A trajetória histórica da extração de atributos pode ser interpretada como uma busca contínua para otimizar a tensão fundamental entre a preservação da informação e a tratabilidade computacional. Os primeiros métodos, desenvolvidos em uma era de recursos computacionais limitados, focavam intensamente na criação de representações extremamente compactas. Isso era alcançado através de heurísticas e modelos matemáticos que, por necessidade, descartavam informações visuais que eram difíceis de modelar ou computacionalmente caras para processar. A revolução do *Deep Learning*, impulsionada pelo poder computacional massivo dos GPUs, representa uma mudança fundamental neste equilíbrio. Em vez de descartar a complexidade, as redes neurais profundas são capazes de aprender transformações muito mais ricas e sutis diretamente dos dados, preservando uma quantidade de informação que antes era inacessível. Essa evolução não é apenas sobre novos algoritmos, mas sobre uma mudança filosófica no que é considerado computacionalmente viável preservar do sinal visual original de alta dimensão.

Os avanços técnicos na extração de atributos, especialmente aqueles impulsionados pelo *Deep Learning*, não existem em um vácuo. Eles têm um impacto profundo e transformador no mundo real, impulsionando inovações em setores críticos. No entanto, o poder sem precedentes dessas tecnologias também traz consigo um conjunto de desafios éticos, sociais e práticos significativos que exigem uma análise cuidadosa, como as questões de viés algorítmico, privacidade, segurança e os custos ocultos de interpretabilidade e consumo de energia. Estes desafios não são problemas independentes, mas sim facetas profundamente interconectadas de uma mesma questão central: as consequências de construir sistemas poderosos, escaláveis, mas fundamentalmente opacos e famintos por dados. Uma solução de um determinado domínio muitas vezes exacerba um problema em outro, criando um complexo problema de otimização multiobjetivo para o campo como um todo.

A busca pela precisão levou ao aumento da escala dos modelos em conjuntos de dados massivos; esse aumento de escala leva diretamente a custos energéticos mais altos (Strubell; Ganesh; Mccallum, 2019). O uso de conjuntos de dados massivos coletados da web melhora a generalização, mas também importa vieses sociais em escala (Mehrabi *et al*., 2021) e cria enormes desafios de privacidade (Liu *et al*., 2021).

A complexidade que permite alta precisão torna os modelos caixas-pretas, criando o problema da interpretabilidade (Guidotti *et al*., 2018), e essa falta de interpretabilidade torna mais difícil diagnosticar e mitigar o viés. As mesmas fronteiras de decisão complexas e não lineares que tornam os modelos precisos também criam a vulnerabilidade a ataques adversariais, com a intenção de explorar pequenas lacunas não intuitivas para humanos na compreensão do modelo (Liu *et al*., 2020).

Portanto, é inadequado tratar essas questões como uma lista de verificação de problemas separados, elas formam uma *teia* de consequências interconectadas que derivam da própria natureza do paradigma do *Deep Learning.* A jornada da extração de atributos em Visão Computacional é uma narrativa notável de evolução científica e tecnológica, ela traça um caminho desde os primeiros dias de engenharia meticulosa e manual, onde a inteligência humana era diretamente codificada em heurísticas para extrair características como bordas e cantos, até a era atual do *Deep Learning*, onde representações visuais ricas e hierárquicas são aprendidas automaticamente a partir de vastas quantidades de dados, no entanto, este poder recém-descoberto não vem sem responsabilidades e desafios profundos.

REFERÊNCIAS

BISHOP, C. M. **Pattern Recognition and Machine Learning**. New York: Springer, 2006.

FORSYTH, D. A.; PONCE, J. **Computer Vision: A Modern Approach**. 2. ed. New Jersey: Pearson, 2012.

GUIDOTTI, R. et al. **A survey of methods for explaining black box models**. ACM Computing Surveys (CSUR), v. 51, n. 5, p. 1-42, 2018.

LIU, B; LIN, Y. **Robust meta gradient learning for high-dimensional data with noisy-label ignorance**. PLOS ONE, v. 18, n. 12, p. e0295678, 2023.

LIU, B. et al. **When machine learning meets privacy: A survey and outlook**. ACM Computing Surveys (CSUR), v. 54, n. 2, p. 1-36, 2021.

LIU, N. et al. **Adversarial attacks and defenses: An interpretation perspective**. SIGKDD Explorations, v. 22, n. 2, p. 28-40, 2020.

MEHRABI, N. et al. A Survey on Bias and Fairness in Machine Learning. **ACM Computing Surveys**, v. 54, n. 6, p. 1-35, 2021.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning**. 2. ed. Birmingham: Packt Publishing, 2017.

STRUBELL, E; GANESH, A; MCCALLUM, A. **Energy and policy considerations for deep learning in NLP**. In: ANNUAL MEETING OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (ACL), 57., 2019, Florence. Proceedings [...]. Florence: ACL, 2019. p. 3645-3650.

SZELISKI, R. **Computer Vision: Algorithms and Application**s. 2. ed. New York: Springer, 2022.

VAN DER MAATEN, L; POSTMA, E; VAN DEN HERIK, J. **Dimensionality Reduction: A Comparative Review**. Journal of Machine Learning Research, v. 10, p. 66-71, 2009.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

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.

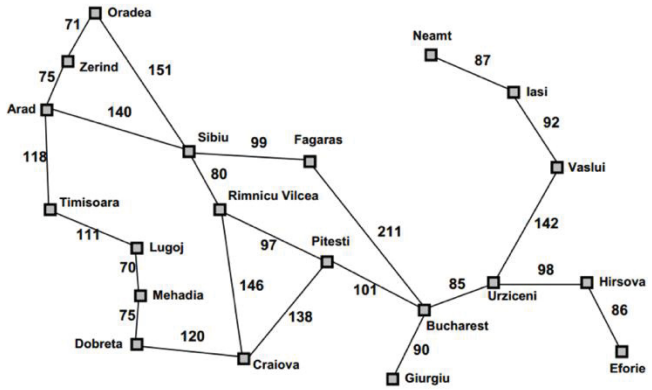
2 Busca Heurística

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

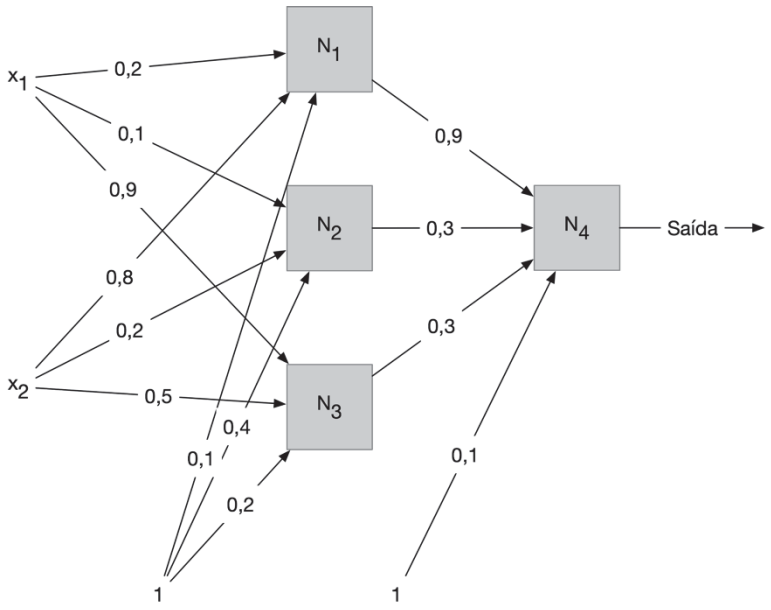
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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 se concentra no desenvolvimento de sistemas e algoritmos que podem realizar tarefas que normalmente exigiria inteligência humana. Isso inclui habilidades como aprendizado, raciocínio, resolução de problemas, reconhecimento de padrões, compreensão de linguagem natural e tomada de decisões. Os sistemas de IA são projetados para processar grandes quantidades de dados, identificar padrões e tomar decisões ou realizar ações com base nesses padrões. Eles podem usar uma variedade de técnicas, incluindo aprendizado de máquina, redes neurais, lógica simbólica, algoritmos de otimização e muito mais. A IA tem aplicações em uma ampla gama de áreas, incluindo reconhecimento de fala, visão computacional, processamento de linguagem natural, diagnóstico médico, automação industrial, finanças, jogos, entretenimento e muito mais. Nos últimos anos, tem havido avanços significativos na IA, impulsionados pelo aumento da capacidade computacional, melhores algoritmos e grandes conjuntos de dados disponíveis para treinamento de modelos de IA.

b) Pensar como humanos: Com base na resposta fornecida pelo ChatGPT, a partir das IAs são implementados sistemas que visam realizar tarefas que exigem inteligência humana: "(...) podem realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui habilidades como aprendizado, raciocínio, resolução de problemas, reconhecimento de padrões, compreensão de linguagem natural e tomada de decisões.". A ideia de implementar uma inteligência humana pode implicar em pensar como humanos.

Agir racionalmente: Na resposta é mencionado que IAs são projetadas para processar grande volumes de dados e a partir deles tomar decisões ou realizar ações: "Os sistemas de IA são projetados para processar grandes quantidades de dados, identificar padrões e tomar decisões ou realizar ações com base nesses padrões.". Portanto, uma IA vai dar um resultado conforme os dados que foram utilizados para análise e aprendizado, sendo adaptável conforme a situação.

Pensar racionalmente: Na resposta do ChatGPT, é citada a possibilidade de aplicação em diversas áreas e como os avanços obtidos nos últimos anos vão de encontro a melhores algoritmos e aumento da capacidade computacional, pode-se relacionar à abordagem de pensar racionalmente. Nessa abordagem, a IA busca modelar o processo de raciocínio correto e depende de poder computacional, premissas corretas e algoritmos que consigam resolver os problemas para um resultado logicamente certo.

Agir como humanos: Apesar de o ChatGPT falar em uso de inteligência humana nas IAs levando à abordagem de pensar como humanos “(...) podem realizar tarefas que normalmente exigiria inteligência humana”, para que uma IA pense como humano, todo o processo de pensamento deve ser mapeado: introspecção, experimentos psicológicos, imagens cerebrais e ainda o fator pessoal. Quando todo esse processo for determinado, então poderemos ter IAs com pensamento humano. Com isso, podemos dizer que as IAs existentes, por mais que consigam realizar tarefas humanas, não pensam como humanos, e sim imitam seu comportamento.

c) ChatGPT é um aplicativo desenvolvido pela OpenAI. Usando os modelos de linguagem GPT, ele pode responder suas perguntas, escrever textos, redigir e-mails, manter uma conversa, explicar código em diferentes linguagens de programação, traduzir linguagem natural em código e muito mais- ou pelo menos tentar- tudo baseado na linguagem natural em que você o alimenta.

ChatGPT usa aprendizado profundo, um subconjunto de aprendizado de máquina, para produzir texto semelhante ao humano por meio de redes neurais transformadoras. O transformador prevê o texto– incluindo a próxima palavra, frase ou parágrafo– com base na sequência típica de seus dados de treinamento. O treinamento começa com dados genéricos e depois passa para dados mais personalizados para uma tarefa específica. O ChatGPT foi treinado com texto online para aprender a linguagem humana e, em seguida, usou transcrições para aprender o básico das conversas.

O ChatGPT é ajustado a partir do GPT-3.5, um modelo de linguagem treinado para produzir texto. Foi otimizado para diálogo usando Aprendizado por Reforço com Feedback Humano (RLHF)– um método que usa demonstrações humanas e comparações de preferências para orientar o modelo em direção ao comportamento desejado.

Os treinadores humanos fornecem conversas e classificam as respostas. Esses modelos de recompensa ajudam a determinar as melhores respostas. Para continuar treinando o chatbot, os usuários podem votar positivamente ou negativamente em sua resposta clicando nos ícones de polegar para cima ou polegar para baixo ao lado da resposta. Os usuários também podem fornecer feedback adicional por escrito para melhorar e ajustar o diálogo futuro.

Referências:

GUINNESS, Harris. How does ChatGPT work?. 2023. Disponível em Acesso em 20 fev. 2024.

HETLER, Amanda. Definition: ChatGPT. 2023. Disponível em Acesso em 22 fev. 2024.

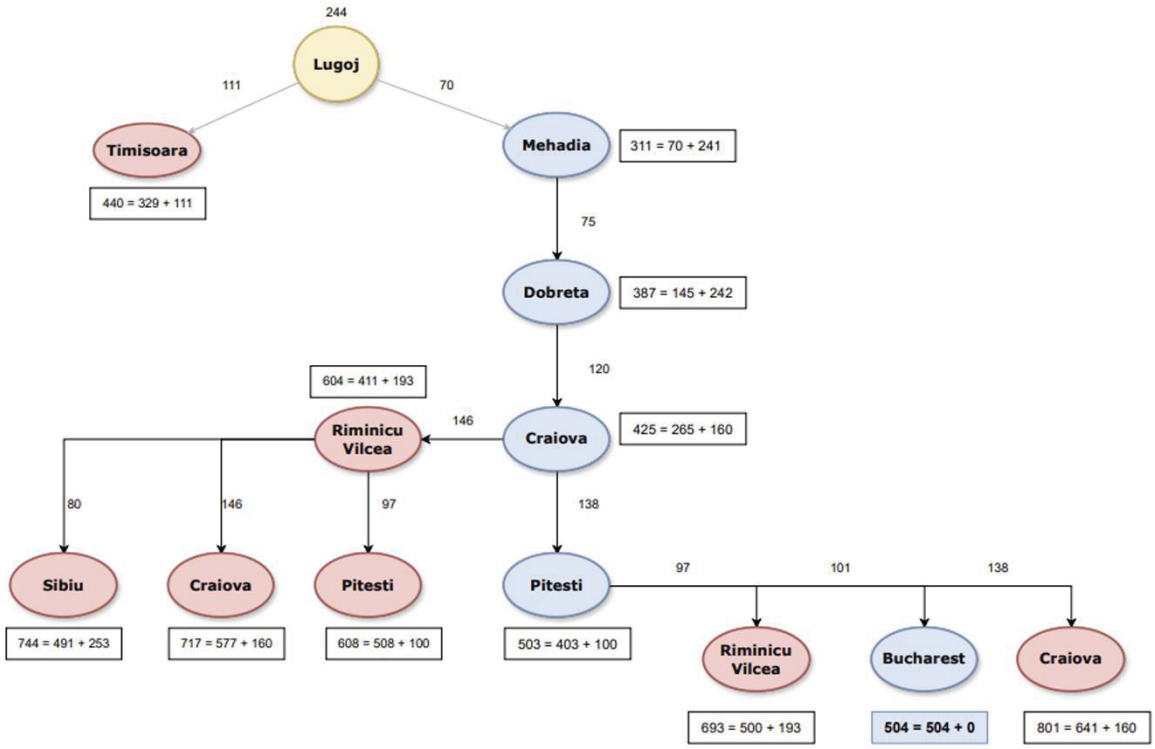
OPENAI. What is ChatGPT?. 2024. Disponível em Acesso em 23 fev. 2024.

d) Agir como Humanos: Essa abordagem se enquadra com a forma como o ChatGPT funciona, pois o objetivo não é definir o que é pensamento nem implementar algum processo cognitivo, já que não é necessário verificar respostas corretas, basta que ele consiga imitar o comportamento humano, além de utilizar habilidades como aprendizado, raciocínio e linguagem natural, sendo esta última a principal utilizada por ele gerando respostas plausíveis e que podem facilmente serem identificadas como "escritas por um ser humano".

Agir racionalmente: O ChatGPT também pode ser classificado dentro dessa abordagem, pois tem como objetivo implementar agentes que respondem a situações e buscam o melhor resultado possível, isso pode ser demonstrado na representação do conhecimento e raciocínio para que tome boas decisões além das habilidades descritas como resoluções de problemas, reconhecimento de padrões e tomada de decisões. O ChatGPT, sendo um sistema treinado com uma base de dados, consegue utilizar esse arcabouço de informações para gerar respostas, na grande maioria das vezes, corretas.

2- a) Abaixo está a melhor rota escolhida pelo algoritmo de busca heurística aplicando o algoritmo A*, representada em azul.

FIGURA 1 – RESULTADO DA BUSCA HEURÍSTICA



FONTE: O autor (2025).

3- Legenda:

p: Uvas caem

q: Raposa come as uvas

r : Uvas estão maduras

Base de Conhecimento (BC):

$R1$: $p \rightarrow q$ (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)

$R4$: $r \rightarrow p$ COND, R3

$R5$: $q \rightarrow p$ SH, R2, R4

$R6$: $(q \rightarrow p) \wedge (p \rightarrow q)$ CONJ, R5, R1

$R7$: $q \leftrightarrow p)$ BICOND, R6

Logo, $q \leftrightarrow p$ (A raposa come as uvas se e somente se as uvas caem) pode ser derivado a partir da base de conhecimento (BC).

$BC \vdash q \leftrightarrow p$

4- Os neurônios N1, N2 e N3 possuem função de ativação linear. Já o N4 possui função de ativação tangente hiperbólica (pesquise a fórmula e aplique).

Dada a entrada x1=-3, x2=1, dê:

a) $\Sigma xi + wi = -3 * 0.2 + 1 * 0.8 + 1 * 0.1 = 0.3$

$$u = 0.3$$

$$linear(u) = u = 0.3$$

$$Saída\ N1 = 0.3$$

b) $\Sigma xi + wi = -3 * 0.1 + 1 * 0.2 + 1 * 0.4 = 0.3$

$$u = 0.3$$

$$linear(u) = u = 0.3$$

$$Saída\ N2 = 0.3$$

c) $\Sigma xi + wi = -3 * 0.9 + 1 * 0.5 + 1 * 0.2 = -2$

$$u = -2$$

$$linear(u) = u = -2$$

$$Saída\ N3 = -2$$

d) $\Sigma xi + wi = 0.3 * 0.9 + 0.3 * 0.3 + (-2) * 0.3 + 1 * 0.1 = -0.14$

$$u = -0.14$$

$$tanh(u) = \frac{e^{u} - e^{-u}}{e^{u} - e^{-u}} = \frac{e^{-0.14} - e^{-(-0.14)}}{e^{-0.14} - e^{-(-0.14)}} = -0.1391$$

$$Saída\ N4 = -0.1391$$

APÊNDICE 2 – 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) O resultado foi: (267542, 11), indicando que o conjunto de dados possui 267.542 linhas e 11 colunas.

b) O resultado indicou que as colunas year_of_reference, month_of_reference, fipe_code, authentication, brand, model, fuel, gear, engine_size, year_model e avg_price_brl possuem 65.245 valores ausentes (aproximadamente 24% dos dados). Tratamento de valores faltantes: Dado que essas linhas não continham informações úteis (completamente ausentes), optamos por removê-las da base de dados usando o método dropna(). Após a remoção, o novo tamanho da base de dados foi de 202.295 linhas e 11 colunas.

c) Foram encontradas 2 linhas duplicadas, que foram removidas com o comando drop_duplicates. Após a remoção das duplicatas, a base ficou com 202.295 linhas, sem duplicatas.

d) Colunas Numéricas:

QUADRO 1 – COLUNAS NUMÉRICAS

Estatística	year_of_reference	engine_size	year_model	avg_price_brl
count	202.295	202.295	202.295	202.295
mean	2.021.564.695	1.822.302	2.011.271.514	52.756,77
std	0.571904	0.734432	6.376.241	51.628,91
min	2.021.000.000	1.000.000	2.000.000.000	6.647.000.000
25%	2.021.000.000	1.400.000	2.006.000.000	22.855.000.000
50%	2.022.000.000	1.600.000	2.012.000.000	38.027.000.000
75%	2.022.000.000	2.000.000	2.016.000.000	64.064.000.000

max	2.023.000.000	6.200.000	2.023.000.000	979.358.000.000

FONTE: O autor (2025).

Colunas Categóricas:

QUADRO 2 – COLUNAS CATEGÓRICAS

Estatística	**month_of_reference**	**fipe_code**	**authentication**	**brand**	**model**	**fuel**	**gear**
count	202.295	202.295	202.295	202.295	202.295	202.295	202.295
unique	12	2.091	425	6	2.112	3	2
top	January	003281-6	cfzlctzfwrcp	Fiat	Palio Week. Adv/Adv	Gasoline	Manual
freq	24.260	24.260	16.884	42.260	16.183	168.684	161.883

FONTE: O autor (2025).

e) Marca:

QUADRO 3 – MARCA DOS CARROS

Marca	**Valor**
Fiat	0.222260
VW - VolksWagen	0.219046
GM - Chevrolet	0.190761

Ford	0.163870
Renault	0.144299
Nissan	0.059764

FONTE: O autor (2025).

Modelo:

QUADRO 4 – MODELO DOS CARROS

Modelo	Valor
Palio Week. Adv/Adv TRYON 1.8 mpi Flex	0.002101
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p	0.002101
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.	0.001977
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V	0.001977
Corvette 5.7/ 6.0, 6.2 Targa/Stingray	0.001854
STEPWAY Zen Flex 1.0 12V Mec.	0.000010
Saveiro Robust 1.6 Total Flex 16V CD	0.000010
Saveiro Robust 1.6 Total Flex 16V	0.000010
Gol Last Edition 1.0 Flex 12V 5p	0.000010
Polo Track 1.0 Flex 12V 5p	0.000010

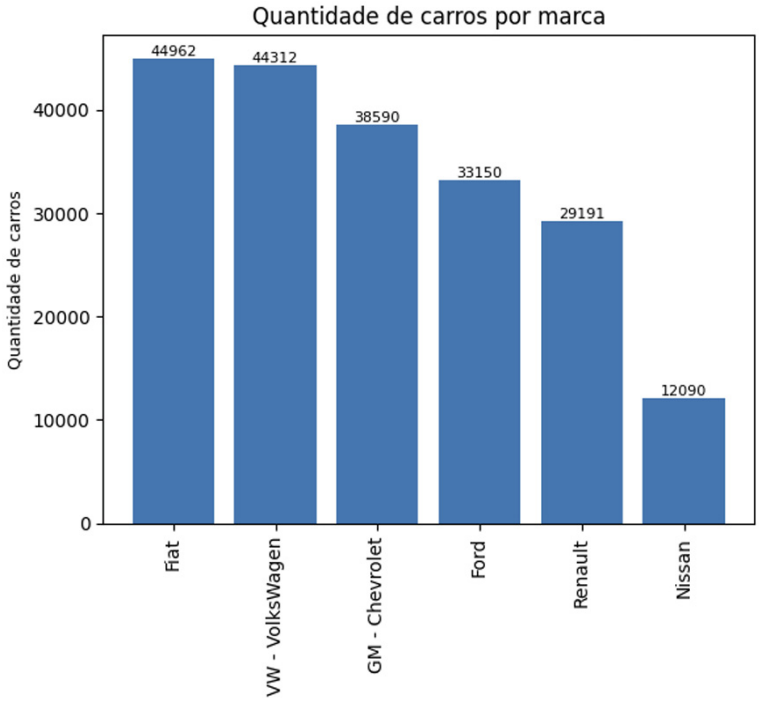
FONTE: O autor (2025).

Os principais resultados observados foram:
As marcas mais frequentes foram Fiat, VW - VolksWagen, GM - Chevrolet e Ford.
O modelo de carro mais comum foi o "Palio Week. Adv/Adv TRYON 1.8 mpi Flex", seguido por outros modelos da marca Fiat e Ford.

f) Através da análise exploratória, observou-se que a base de dados analisada é composta por 2112 modelos de carros distintos, com ano de fabricação entre 2000 e 2023. A mediana do preço médio dos carros foi de R$38 mil reais, e o modelo mais barato e mais caro custavam respectivamente, R$6,6 mil e R$979 mil. A marca mais frequente dos carros cadastrados foi Fiat, 83% dos automóveis são movidos à gasolina e 80% do tipo manual.

2- a) Gráfico de quantidade de carros por marca.

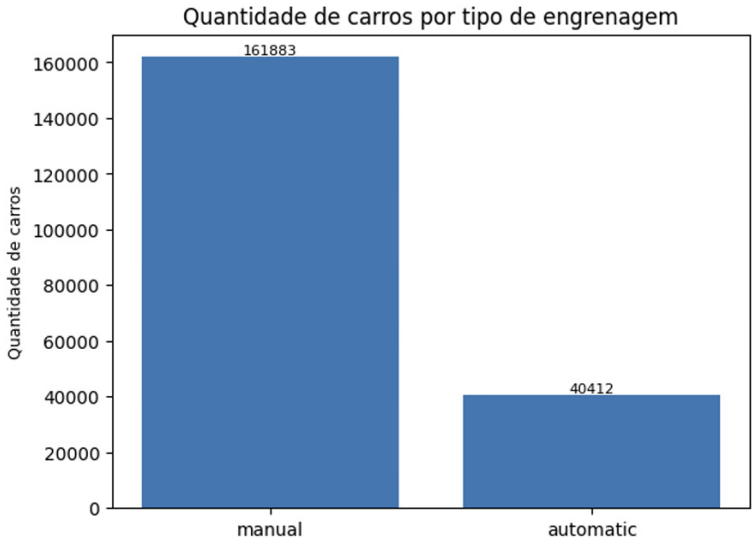
FIGURA 2 – GRÁFICO DE QUANTIDADE DE CARROS POR MARCA



FONTE: O autor (2025).

b) Gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro.

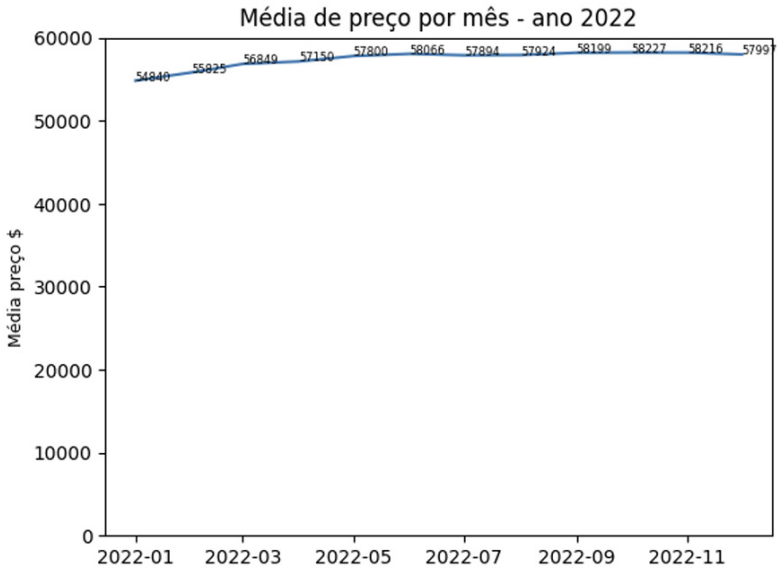
FIGURA 3 – GRÁFICO DE QUANTIDADE DE CARROS POR TIPO ENGRENAGEM



FONTE: O autor (2025).

c) Gráfico da evolução da média de preço dos carros ao longo dos meses de 2022 (variável de tempo no eixo X).

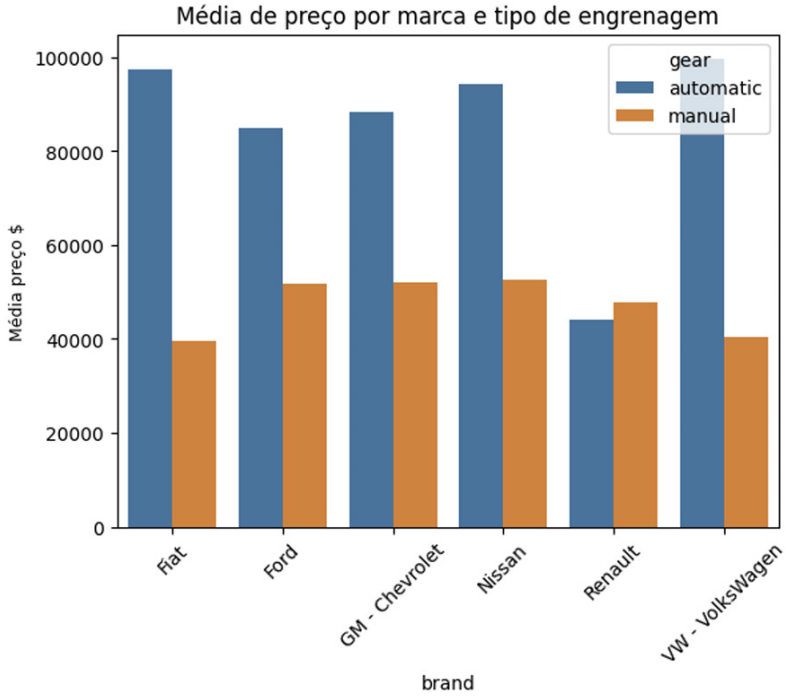
FIGURA 4 – MÉDIA DE PREÇOS DOS CARROS



FONTE: O autor (2025).

d) Gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem.

FIGURA 5 – MÉDIA DE PREÇOS POR MARCA E TIPO DE ENGRENAGEM

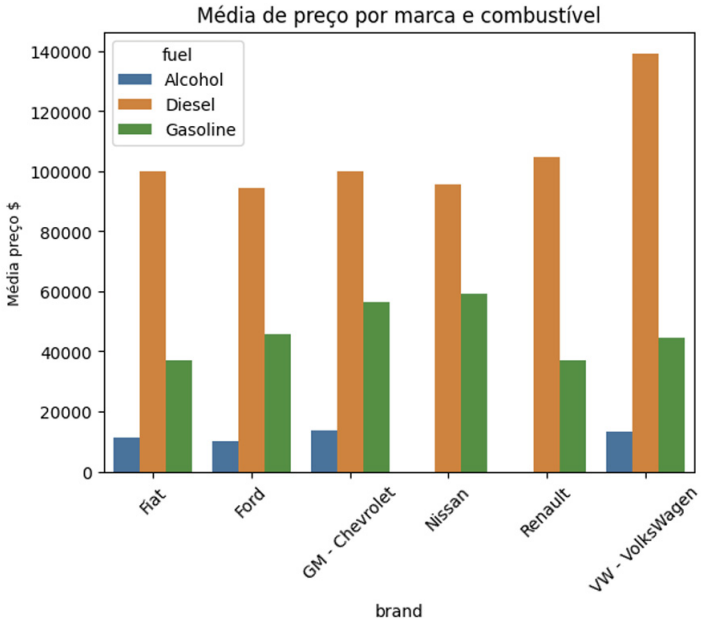


FONTE: O autor (2025).

e) Veículos com transmissão manual são, em média, mais baratos que os veículos com transmissão automática. Porém, há uma exceção nos veículos da marca Renault, que o valor médio dos veículos com transmissão automática são menores, isso pode ser explicado pelo fato de serem carros mais antigos, conforme foi observado quando a mediana do ano dos carros foi consultada. Observou-se também que a média de preço dos carros manuais da Fiat é mais baixa do que das demais marcas, seguidos pelos carros manuais da VW. Veículos com transmissão manual são, em média, mais baratos que os veículos com transmissão automática. Porém, há uma exceção nos veículos da marca Renault, que o valor médio dos veículos com transmissão automática são menores, isso pode ser explicado pelo fato de serem carros mais antigos, conforme foi observado quando a mediana do ano dos carros foi consultada. Observou-se também que a média de preço dos carros manuais da Fiat é mais baixa do que das demais marcas, seguidos pelos carros manuais da VW.

f) Gráfico da distribuição da média de preço dos carros por marca e tipo de combustível

FIGURA 6 – MÉDIA DE PREÇOS POR MARCA E COMBUSTÍVEL



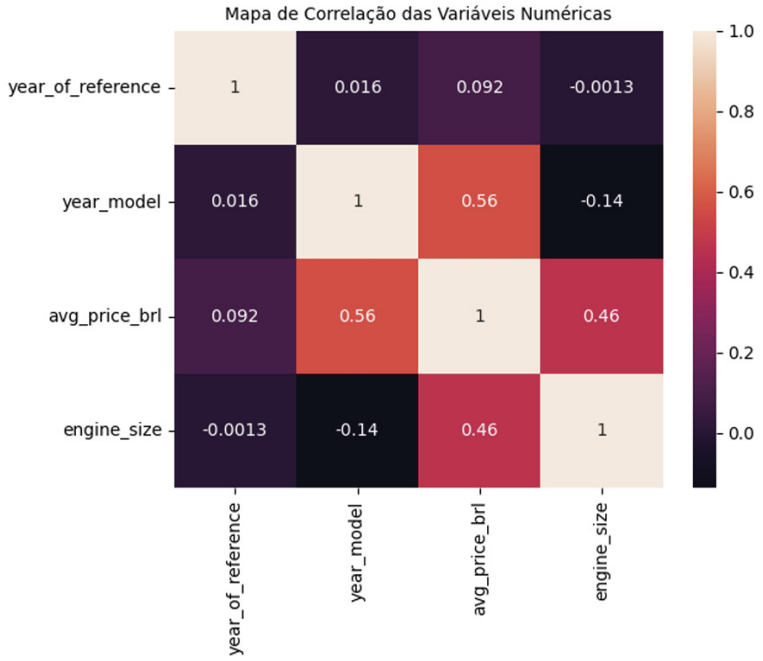
FONTE: O autor (2025).

g) Veículos a diesel são, em média, mais caros. Isso pode estar associado ao fato de que esses veículos são geralmente de grande porte e que são, intrinsecamente, mais caros. Também observa-se que algumas marcas não possuem veículos movidos exclusivamente a álcool, e isso pode ser pelo fato de que esse tipo de veículo é mais antigo e que veículos flex na FIPE são contabilizados como movidos a gasolina.

3- A partir da base de dados precos_carros_brasil.csv, foram executadas as seguintes tarefas para prever o preço médio dos carros:

a) Foi feita a matriz de correlação entre as variáveis numéricas e observada correlação de 0,56 entre ano do modelo do carro e preço médio do veículo, indicando correlação direta e moderada. Também foi identificada correlação positiva, porém fraca, phi = 0,46 entre preço médio e tamanho do motor. A correlação entre ano de referência e preço médio do veículo foi muito próxima de 0, não demonstrando haver correlação entre essas duas variáveis, e por isso não será utilizada no modelo. Também não será considerada a variável mês de referência, uma vez que o ano de referência não entra no modelo. Para as variáveis categóricas, foi feita a análise gráfica através dos gráficos da parte 2 e dos boxplots da variável resposta x variável independente para a seleção.

FIGURA 7 – MAPA DE CORRELAÇÃO DAS VARIÁVEIS NUMÉRICAS



FONTE: O autor (2025).

b) Abaixo está o código-fonte feito em python:

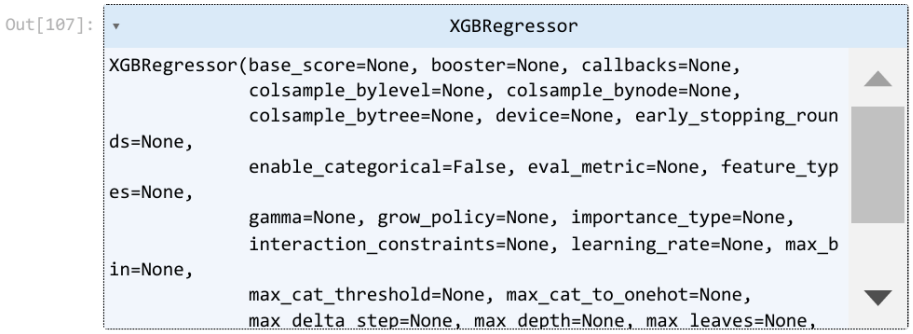
```
# Importando a função para dividir os dados
from sklearn.model_selection import train_test_split

# Seleção das variáveis independentes (X) e dependente (Y)
X = carros_modelo[['brand', 'model', 'fuel', 'gear', 'engine_size', 'year_model']]
Y = carros_modelo[['avg_price_brl']]

# Divisão dos dados em treino (75%) e teste (25%)
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.25,
random_state=42)
```

c) Abaixo está o código-fonte feito em python:

```
# Importando os modelos
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor

# Modelo Random Forest sem parâmetros
RF_semParametros = RandomForestRegressor()
RF_semParametros.fit(X_train, Y_train)
```

```
# Modelo XGBoost sem parâmetros
XGBoost_semParametro = XGBRegressor()
XGBoost_semParametro.fit(X_train, Y_train)
```

FIGURA 8 – XGBREGRESSOR



FONTE: O autor (2025).

O RandomForestRegressor foi treinado com seus parâmetros padrão, enquanto o XGBRegressor também foi treinado sem alteração de parâmetros.

d) Abaixo está o código-fonte feito em python:

```
# Previsões dos modelos
Yhat_RF_semParametros = RF_semParametros.predict(X_test)
Yhat_XG = XGBoost_semParametro.predict(X_test)
```

As previsões realizadas por ambos os modelos foram armazenadas nas variáveis Yhat_RF_semParametros e Yhat_XG, respectivamente.

e) Foram geradas as tabelas de importância das variáveis para ambos os modelos, mostrando quais variáveis mais influenciam na previsão do preço médio do carro.

QUADRO 5 – VARIÁVEIS E IMPORTÂNCIA

Variable	**Importance**
engine_size	0.439257
fuel	0.205469

year_model	0.170392
gear	0.123404
brand	0.043103
model	0.018376

FONTE: O autor (2025).

f) Nos três modelos a variável engine_size teve maior importância (sempre maior que 0,4). Nos modelos de Random Forest, duas variáveis, engine_size e year_model eram as responsáveis por quase 90% desse índice em cada ajuste. Apenas no XGBoost year_model não assumiu a segunda posição quanto à importância, perdendo para a variável fuel .

g) Abaixo está o código-fonte feito em python:

```
#Avaliação do modelo - MSE mse_md2 = mean_squared_error(Y_test, Yhat_RF_comParametros) mse_md2
```

Resultado da Saída: 134879179.34676224

```
#Avaliação do modelo - MAE mae_md2 = mean_absolute_error(Y_test, Yhat_RF_comParametros) mae_md2
```

Resultado da Saída: 5197.4533381621895

```
#Avaliação do modelo - R2 r2_score(Y_test, Yhat_RF_comParametros)
```

Resultado da Saída: 0.9483897141128078

h) Os três modelos tiveram bom ajuste nos dados, com R2 sempre superior a 0,9. No entanto, a Random Forest, sem parâmetros pré-definidos, apresentou MSE e MAE menores do que os demais modelos, e teve o melhor valor de R2, 0,98. Com isso, esse modelo, que teve como variáveis importantes engine_size (tamanho do motor) e year_model (ano do modelo) foi escolhido como o melhor.

APÊNDICE 3 – 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

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

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

- Erro padrão da estimativa: S_{yx}

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

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- Código fonte feito em R:

```
# Definir o mirror do CRAN
mirror <- "https://cran-r.c3sl.ufpr.br"
options(repos = mirror)

# Instale o pacote mlbench se ainda não estiver instalado
install.packages("mlbench")

# Carregue o pacote
library(mlbench)

# Carregue a base de dados Satellite
data(Satellite)

# Visualize a estrutura da base de dados
str(Satellite)
```

Resultado da Saída: *'data.frame': $ x.1 : num 92 84 84 80 84 80 76 76 76 76 ... $ x.2 $ x.3 $ x.4 $ x.5 $ x.6 $ x.7 $ x.8 $ x.9 : num 115 102 102 102 94 94 102 102 89 94*

```
... : num 120 106 102 102 102 98 106 106 98 98 ... : num 94 79 83 79 79 76 83 87 76
76 ... : num 84 84 80 84 80 80 76 80 76 76 ... : num 102 102 102 94 94 102 102 98
94 98 ... : num 106 102 102 102 98 102 106 106 98 102 ... : num 79 83 79 79 76 79
87 79 76 72 ... : num 84 80 84 80 80 76 80 76 76 76 ... $ x.10 : num 102 102 94 94
102 102 98 94 98 94 ... $ x.11 : num 102 102 102 98 102 102 106 102 102 90 ... $
x.12 : num 83 79 79 76 79 79 79 76 72 76 ... $ x.13 : num 101 92 84 84 84 76 80 80
80 76 ... $ x.14 : num 126 112 103 99 99 99 107 112 95 91 ... $ x.15 : num 133 118
104 104 104 104 118 118 104 104 ... $ x.16 : num 103 85 81 78 81 81 88 88 74 74 ...
$ x.17 : num 92 84 84 84 76 76 80 80 76 76 ... $ x.18 : num 112 103 99 99 99 99 112
107 91 95 ... $ x.19 : num 118 104 104 104 104 108 118 113 104 100 ... $ x.20 : num
85 81 78 81 81 85 88 85 74 78 ... $ x.21 : num 84 84 84 76 76 76 80 80 76 76 ... $
x.22 : num 103 99 99 99 99 103 107 95 95 91 ... $ x.23 : num 104 104 104 104 108
118 113 100 100 100 ... $ x.24 : num 81 78 81 81 85 88 85 78 78 74 ... $ x.25 : num
102 88 84 84 84 84 79 79 75 75 ... $ x.26 : num 126 121 107 99 99 103 107 103 91 91
... $ x.27 : num 134 128 113 104 104 104 113 104 96 96 ... $ x.28 : num 104 100 87
79 79 79 87 83 75 71 ... $ x.29 : num 88 84 84 84 84 79 79 79 75 79 ... $ x.30 :
num 121 107 99 99 103 107 103 103 91 87 ... $ x.31 : num 128 113 104 104 104 109
104 104 96 93 ... $ x.32 : num 100 87 79 79 79 87 83 79 71 71 ... $ x.33 : num 84
84 84 84 79 79 79 79 79 79 ... $ x.34 : num 107 99 99 103 107 107 103 95 87 87 ...
$ x.35 : num 113 104 104 104 109 109 104 100 93 93 ... $ x.36 : num 87 79 79 79 87
87 79 79 71 67 ... $ classes: Factor w/ 6 levels "red soil","cotton crop",..: 3 3 3
3 3 3 3 3 4 4 …
```

1.2 - Código fonte feito em python:

```
# Instale o pacote 'caret' se ainda não o tiver instalado
install.packages("caret")

# Carregue o pacote
library(caret)

# Defina a semente para reprodução dos resultados
set.seed(123)

# Crie a partição dos dados (80% treino, 20% teste)
particao <- createDataPartition(Satellite$classes, p = 0.8, list = FALSE)

# Separe os dados de treinamento e teste
dados_treino <- Satellite[particao, ]
dados_teste <- Satellite[-particao, ]
```

```
# Verifique o tamanho dos dados de treinamento e teste
print(paste("Tamanho dos dados de treinamento:", nrow(dados_treino)))
print(paste("Tamanho dos dados de teste:", nrow(dados_teste)))
```

Resultado da Saída: *[1] "Tamanho dos dados de treinamento: 5151"*
[1] "Tamanho dos dados de teste: 1284"

1.3 Código fonte feito em R:

```
# 3- Treine modelos RandomForest, SVM e RNA para predição destes dados.
install.packages("neuralnet")

# Carregue os pacotes necessários
library(randomForest)
library(e1071)
library(neuralnet)

# 3.1 Treinamento do modelo Random Forest
modelo_rf <- randomForest(classes ~ ., data = dados_treino)

# 3.2 Treinamento do modelo SVM
modelo_svm <- svm(classes ~ ., data = dados_treino)

# 3.3 Treinamento do modelo RNA
modelo_rna <- neuralnet(classes ~ ., data = dados_treino, hidden = c(5, 2),
linear.output = FALSE)

# Exiba os modelos treinados
print(modelo_rf)
print(modelo_svm)
print(modelo_rna)
```

Resultado da Saída:

QUADRO 6 – RESULTADO DOS MODELOS TREINADOS

	red soil	**cotton crop**	**grey soil**	**damp grey soil**	**vegetation stubble**	**very damp grey soil**	**class.error**
red soil	1206	4	13	0	4	0	0.0171
cotton crop	0	545	1	5	7	5	0.0320
grey soil	8	1	1044	23	2	9	0.0396
damp grey soil	5	4	94	293	4	101	0.4152
vegetation stubble	22	5	2	3	509	25	0.1007
very damp grey soil	0	0	20	55	24	1108	0.0820

FONTE: O autor (2025).

.A saída ao executar o print do modelo_rna é extremamente longa. Portanto, somente o começo da saída foi adicionada a este documento, conforme a tabela abaixo.

QUADRO 7 – RESULTADO DO MODELO RNA

	grey soil	**damp grey soil**	**vegetation stubble**	**very damp grey soil**
1	FALSE	FALSE	TRUE	FALSE
2	FALSE	FALSE	TRUE	FALSE
3	FALSE	FALSE	TRUE	FALSE

4	FALSE	FALSE	TRUE	FALSE
5	FALSE	FALSE	TRUE	FALSE
6	FALSE	FALSE	TRUE	FALSE
7	FALSE	FALSE	TRUE	FALSE
8	FALSE	TRUE	FALSE	FALSE
9	FALSE	TRUE	FALSE	FALSE
10	FALSE	TRUE	FALSE	FALSE
11	FALSE	TRUE	FALSE	FALSE
12	FALSE	TRUE	FALSE	FALSE
13	FALSE	TRUE	FALSE	FALSE
14	FALSE	TRUE	FALSE	FALSE
15	FALSE	FALSE	TRUE	FALSE
16	FALSE	FALSE	TRUE	FALSE

FONTE: O autor (2025).

1.4 Código fonte feito em R:

```
# 4. Escolha o melhor modelo com base em suas matrizes de confusão.
# Carregue o pacote 'caret' para calcular a matriz de confusão
library(caret)

# Função para calcular métricas de desempenho
calcular_metricas <- function(matriz_confusao) {
  # Precisão (precision)
  precisao <- diag(matriz_confusao) / colSums(matriz_confusao)
```

```
  # Recall
  recall <- diag(matriz_confusao) / rowSums(matriz_confusao)

  # F1-score
  f1_score <- 2 * (precisao * recall) / (precisao + recall)

  # Retornar as métricas
  return(data.frame(precisao = precisao, recall = recall, f1_score = f1_score))
}

# Função para imprimir as métricas
imprimir_metricas <- function(nome_modelo, matriz_confusao) {
  cat("\nModelo:", nome_modelo, "\n")
  print(calcular_metricas(matriz_confusao))
}

# Função para plotar a matriz de confusão
plotar_matriz_confusao <- function(nome_modelo, matriz_confusao) {
  confusionMatrix(matriz_confusao, main = nome_modelo)
}

# Prever os rótulos usando cada modelo
predicoes_rf <- predict(modelo_rf, newdata = dados_teste)
predicoes_svm <- predict(modelo_svm, newdata = dados_teste)
predicoes_rna <- predict(modelo_rna, newdata = dados_teste)

# Prever as probabilidades usando a RNA
probabilidades_rna <- predict(modelo_rna, newdata = dados_teste)

# Obter os nomes das classes
nomes_classes <- levels(dados_teste$classes)

# Transformar probabilidades em rótulos de classe
predicoes_rna <- apply(probabilidades_rna, 1, function(x)
nomes_classes[which.max(x)])

# Converter as predições em fatores
predicoes_rna <- factor(predicoes_rna, levels = nomes_classes)
```

```
# Calcular a matriz de confusão para cada modelo
matriz_confusao_rf <- confusionMatrix(predicoes_rf, dados_teste$classes)
matriz_confusao_svm <- confusionMatrix(predicoes_svm, dados_teste$classes)
matriz_confusao_rna <- confusionMatrix(predicoes_rna, dados_teste$classes)

# Imprimir as métricas de desempenho para cada modelo
imprimir_metricas("Random Forest", matriz_confusao_rf$table)
imprimir_metricas("SVM", matriz_confusao_svm$table)
imprimir_metricas("RNA", matriz_confusao_rna$table)

# Plotar as matrizes de confusão
plotar_matriz_confusao("Random Forest", matriz_confusao_rf$table)
plotar_matriz_confusao("SVM", matriz_confusao_svm$table)
plotar_matriz_confusao("RNA", matriz_confusao_rna$table)
```

Abaixo mostra os resultados de cada modelo começando pelo Random Forest.

QUADRO 8 – RESULTADO DO MODELO RANDOM FOREST

Class	**Precisão**	**Recall**	**F1-Score**
red soil	0.9934641	0.9712460	0.9822294
cotton crop	0.9857143	0.9857143	0.9857143
grey soil	0.9667897	0.9065744	0.9357143
damp grey soil	0.7200000	0.8181818	0.7659574
vegetation stubble	0.8510638	0.9600000	0.9022556
very damp grey soil	0.9169435	0.8990228	0.9078947

FONTE: O autor (2025).

A seguir, apresenta o resultado do modelo SVM.

QUADRO 9 – RESULTADO DO MODELO SVM

Class	**Precisão**	**Recall**	**F1-Score**
red soil	0.9934641	0.9589905	0.9759230
cotton crop	0.9785714	0.9785714	0.9785714
grey soil	0.9667897	0.8704319	0.9160839
damp grey soil	0.6320000	0.7117117	0.6694915
vegetation stubble	0.8085106	0.9500000	0.8735632
very damp grey soil	0.8704319	0.8881356	0.8791946

FONTE: O autor (2025).

Por último, o resultado do modelo RNA.

QUADRO 9 – RESULTADO DO MODELO RNA

Class	**Precisão**	**Recall**	**F1-Score**
red soil	0	NaN	NaN
cotton crop	0	NaN	NaN
grey soil	0	NaN	NaN

damp grey soil	1	0.09735202	0.1774308
vegetation stubble	0	NaN	NaN
very damp grey soil	0	NaN	NaN

FONTE: O autor (2025).

1.5 - Código fonte feito em R:

```
# 5 - Indique qual modelo dá o melhor resultado e a métrica utilizada

# Extrair os valores de F1-score para cada modelo
f1_rf <- calcular_metricas(matriz_confusao_rf$table)$f1_score
f1_svm <- calcular_metricas(matriz_confusao_svm$table)$f1_score
f1_rna <- calcular_metricas(matriz_confusao_rna$table)$f1_score

# Criar um data frame com os valores de F1-score
df_f1 <- data.frame(
  Modelo = c("Random Forest", "SVM", "RNA"),
  F1_Score = c(f1_rf, f1_svm, f1_rna)
)

# Ordenar o data frame pelo F1-score
df_f1 <- df_f1[order(df_f1$F1_Score, decreasing = TRUE), ]

# Imprimir o data frame
print(df_f1)

# Identificar o melhor modelo
melhor_modelo <- df_f1[1, "Modelo"]
cat("\nO melhor modelo é:", melhor_modelo, "\n")
```

Resultado da Saída:

```
 Modelo  F1_Score
 2            SVM 0.9857143
 1  Random Forest 0.9822294
```

```
 8            SVM 0.9785714
 7  Random Forest 0.9759230
 3            RNA 0.9357143
 9            RNA 0.9160839
 6            RNA 0.9078947
 5            SVM 0.9022556
 12           RNA 0.8791946
 11           SVM 0.8735632
 4  Random Forest 0.7659574
 10 Random Forest 0.6694915
 16 Random Forest 0.1774308
 13 Random Forest       NaN
 14           SVM       NaN
 15           RNA       NaN
 17           SVM       NaN
 18           RNA       NaN
 O melhor modelo é: SVM
```

2.1- Código fonte feito em R:

```
# --- 01 Carregar o arquivo Volumes.csv (http://www.razer.net.br/datasets/Volumes.csv) --
url_dataset <- "http://www.razer.net.br/datasets/Volumes.csv"
# Carregando a base de dados (ex 1)
log(paste("Carregando base de dados de volumes de árvores. URL:", url_dataset))
dataset <- read.csv2(url_dataset, header = TRUE, sep = ";")
```

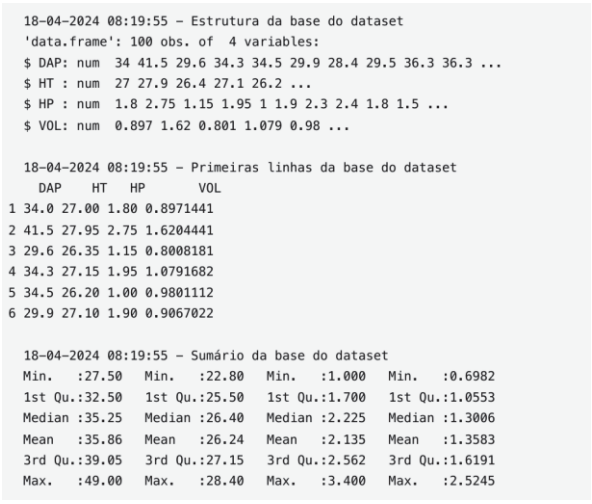
Resultado da Saída: *18-04-2024 08:19:54 - Carregando base de dados de volumes de árvores. URL: http://www.razer.net.br/datasets/Volumes.csv*

2.2- Código fonte feito em R:

```
# --- 02 Eliminar a coluna NR, que só apresenta um número sequencial --
dataset <- dataset[, !names(dataset) %in% "NR"]
# Visualizando a base de dados
log("Estrutura da base do dataset")
str(dataset)
log("Primeiras linhas da base do dataset")
head(dataset)
log("Sumário da base do dataset")
```

```
summary(dataset)
```

FIGURA 9 – VISUALIZAÇÃO DOS DADOS



```
18-04-2024 08:19:55 - Estrutura da base do dataset
'data.frame': 100 obs. of  4 variables:
$ DAP: num  34 41.5 29.6 34.3 34.5 29.9 28.4 29.5 36.3 36.3 ...
$ HT : num  27 27.9 26.4 27.1 26.2 ...
$ HP : num  1.8 2.75 1.15 1.95 1 1.9 2.3 2.4 1.8 1.5 ...
$ VOL: num  0.897 1.62 0.801 1.079 0.98 ...

18-04-2024 08:19:55 - Primeiras linhas da base do dataset
  DAP    HT   HP       VOL
1 34.0 27.00 1.80 0.8971441
2 41.5 27.95 2.75 1.6204441
3 29.6 26.35 1.15 0.8008181
4 34.3 27.15 1.95 1.0791682
5 34.5 26.20 1.00 0.9801112
6 29.9 27.10 1.90 0.9067022

18-04-2024 08:19:55 - Sumário da base do dataset
Min.   :27.50   Min.   :22.80   Min.   :1.000   Min.   :0.6982
1st Qu.:32.50   1st Qu.:25.50   1st Qu.:1.700   1st Qu.:1.0553
Median :35.25   Median :26.40   Median :2.225   Median :1.3006
Mean   :35.86   Mean   :26.24   Mean   :2.135   Mean   :1.3583
3rd Qu.:39.05   3rd Qu.:27.15   3rd Qu.:2.562   3rd Qu.:1.6191
Max.   :49.00   Max.   :28.40   Max.   :3.400   Max.   :2.5245
```

FONTE: O autor (2025).

2.3- Código fonte em R:

```
# --- 03 Criar partição de dados: treinamento 80%, teste 20% ---

# Setando uma semente de aleatoriedade
set.seed(123)

# Criando índices para o treino
cat("Particionando dados em treino e teste\n")
indices <- createDataPartition(dataset$VOL, p = 0.8, list = FALSE)

# Separando dados em treino e teste
dados_treino <- dataset[indices, ]
dados_teste <- dataset[-indices, ]

# Verificando quantidade de observações em cada partição
cat("Observações nos dados de treinamento:", nrow(dados_treino), "\n")
cat("Observações nos dados de teste:", nrow(dados_teste), "\n")
```

Resultado da Saída:

```
18-04-2024 08:19:55 - Particionando dados em treino e teste
[1] "Observações nos dados de treinamento: 80"
[1] "Observações nos dados de teste: 20"
```

2.4- Código fonte feito em R:

```
# --- 04 Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM
(svmRadial),
# O modelo alométrico é dado por: Volume = b0 + b1 * dap ^ 2 * Ht

# Treinando os modelos
cat("Treinando modelo Random Forest\n")
rf <- train(VOL ~ ., data = dados_treino, method = "rf")

cat("Treinando modelo SVM\n")
svm <- train(VOL ~ ., data = dados_treino, method = "svmRadial")

cat("Treinando modelo Neural Network\n")
rna <- train(VOL ~ ., data = dados_treino, method = "neuralnet")

cat("Treinando modelo Alométrico de SPURR\n")
alom <- nls(
  VOL ~ b0 + b1 * (DAP ^ 2) * HT,
  data = dados_treino,
  start = list(b0 = 0.5, b1 = 0.01)
)
```

Resultado da Saída:

```
18-04-2024 08:19:55 - Treinando modelo Random Forest
note: only 2 unique complexity parameters in default grid. Truncating the grid to 2
.
18-04-2024 08:19:57 - Treinando modelo SVM
18-04-2024 08:19:57 - Treinando modelo Neural Network
There were 37 warnings (use warnings() to see them)
18-04-2024 08:21:41 - Treinando modelo Alométrico de SPURR
```

2.5- Código fonte feito em R:

```
# --- 05 Efetue as predições nos dados de teste
log("Realizando predições")
predicoes_rf <- predict(rf, dados_teste)
predicoes_svm <- predict(svm, dados_teste)
```

```
predicoes_rna <- predict(rna, dados_teste)
predicoes_alom <- predict(alom, dados_teste)
```

Resultado da Saída: *18-04-2024 08:21:41 - Realizando predições*

2.6- Código fonte feito em R:

```
# Função para cálculo do coeficiente de determinação R²
calcular_coef_r2 <- function(observacoes, predicoes) {
  return(1 - sum((observacoes - predicoes) ^ 2) / sum((observacoes - mean(observacoes)) ^ 2))
}

# Função para erro padrão de estimativa: Syx
calcular_erro_syx <- function(observacoes, predicoes) {
  return(sqrt(sum((observacoes - predicoes) ^ 2) / (length(observacoes) - 2)))
}

# Função para o cálculo da porcentagem de erro Syx
calcular_erro_syx_percent <- function(observacoes, predicoes) {
  return((calcular_erro_syx(observacoes, predicoes) / mean(observacoes)) * 100)
}

# Função para calcular um score com base no valor de R² e Syx
calcular_score <- function(r2, syx) {
  return((r2 + (1 - syx)) / 2)
}

# Função para retornar as métricas de avaliação
calcular_metricas <- function(observacoes, predicoes, nome_modelo) {
  r2 <- calcular_coef_r2(observacoes, predicoes)
  syx <- calcular_erro_syx(observacoes, predicoes)
  syx_percent <- calcular_erro_syx_percent(observacoes, predicoes)
  score <- calcular_score(r2, syx)

  return(data.frame(model = nome_modelo, r2 = r2, syx = syx, syxPercentage = syx_percent, score = score))
}
```

Resultado da Saída: *N/A*

2.7- Código fonte feito em R:

```
# --- 07 Escolha o melhor modelo ---

cat("Calculando métricas para os modelos\n")

# Calculando as métricas para cada modelo
metricas_df <- calcular_metricas(dados_teste$VOL, predicoes_rf, "rf")
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL,
predicoes_svm, "svm"))
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL,
predicoes_rna, "rna"))
metricas_df <- rbind(metricas_df, calcular_metricas(dados_teste$VOL,
predicoes_alom, "alom"))

# Ordenando as métricas pelo score (do melhor para o pior)
metricas_df <- metricas_df[order(metricas_df$score, decreasing = TRUE), ]

cat("Métricas com base no score (melhor para o pior):\n")
print(metricas_df)
```

Na atividade foram treinados quatro modelos: Random Forest , SVM, Redes Neurais e modelo alométrico de SPURR. Após os modelos terem sido treinados, foram realizadas as predições com os dados para teste e comparado com os valores observados. Com esses resultados, foram calculadas três métricas:

- Coeficiente de determinação (R^2)
- Erro padrão da estimativa (Syx)
- Porcentagem do erro padrão da estimativa (Syx %)

Para o primeiro valor, quanto mais perto de 1, melhor. Já para o segundo, quanto mais perto de 0, melhor. A terceira métrica é derivada da segunda. Por fim foi calculado um score que considera o valor de R 2 e o S yx % (considerando o range de valores entre 0 e 1) com a seguinte fórmula:

$$score = (R2 + (1 - Syx)) / 2$$

Com esse score foi possível definir qual dos quatro modelos performou melhor, sendo que o resultado foi o seguinte (já ordenados do melhor para o pior):

QUADRO 10 – MODELOS COM MELHOR SCORE

Modelo	**R²**	**Syx**	**Syx %**	**Score**

RNA	0.8867930	0.1354473	1.006.305	0.8930813
Alométrico	0.8694429	0.1454567	1.080.670	0.8806879
Random Forest	0.8486654	0.1566040	1.163.489	0.8661582
SVM	0.7900779	0.1844433	1.370.321	0.8265229

FONTE: O autor (2025).

Portanto, pode-se concluir que o melhor modelo nesse caso é o modelo de Redes Neurais.

APÊNDICE 4 – 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.

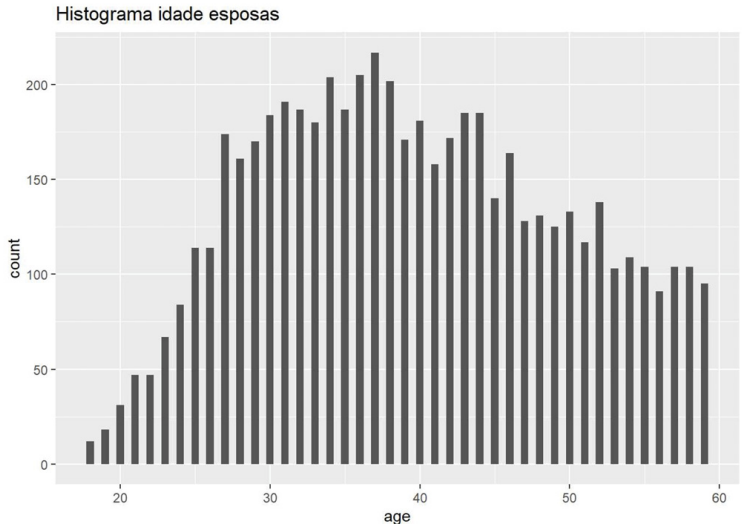
B – RESOLUÇÃO

1- Código fonte feito em R:

```
library(ggplot2)
library(dgof)

# Carregar os dados
load("C:/Users/livia/OneDrive/Documentos/salarios.Rdata")
dados <- salarios
```

```
# Criar dataset de teste com idades de esposas e maridos
dados_teste <- rbind.data.frame(
  cbind.data.frame(idade = dados$age, gender = rep("esposas", 5634)),
  cbind.data.frame(idade = dados$husage, gender = rep("maridos", 5634))
)

# Plotar histograma da idade das esposas
ggplot(dados, aes(x = age)) +
  geom_histogram(binwidth = 0.5) +
  labs(title = "Histograma idade esposas", x = "Idade", y = "Frequência")
```

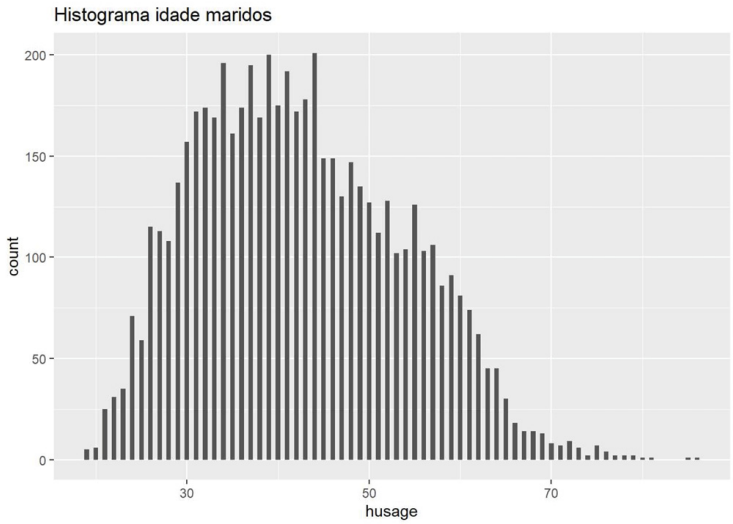
FIGURA 10 – HISTOGRAMA IDADE DAS ESPOSAS



FONTE: O autor (2025).

```
ggplot(dados, aes(husage)) +
geom_histogram(binwidth = 0.5)+
labs(title = "Histograma idade maridos")
```

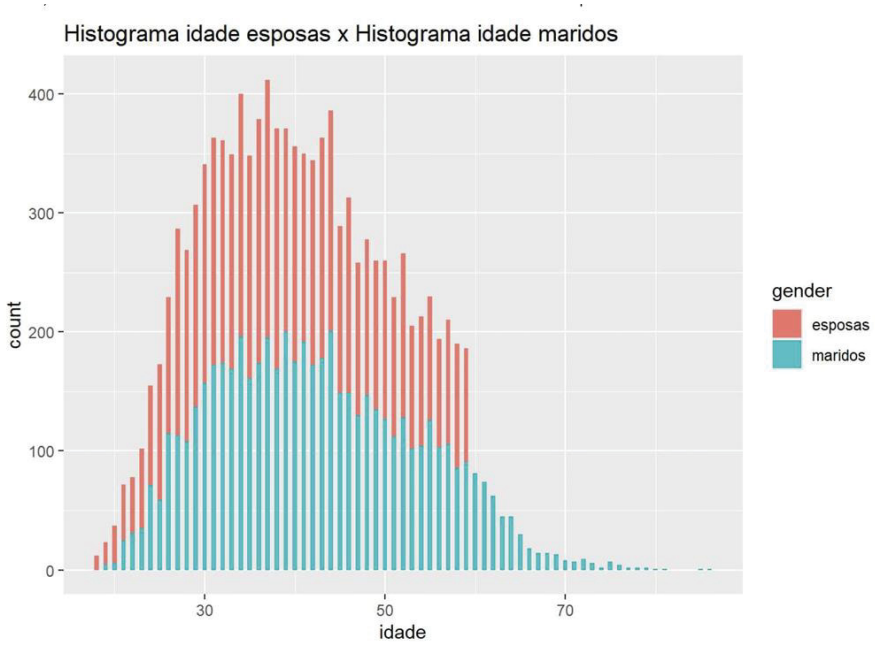
FIGURA 11 – HISTOGRAMA IDADE DOS MARIDOS



FONTE: O autor (2025).

```
ggplot(dados_teste, aes(idade, fill = gender)) +
geom_histogram(binwidth = 0.5)+
labs(title = "Histograma idade esposas x Histograma idade maridos")
```

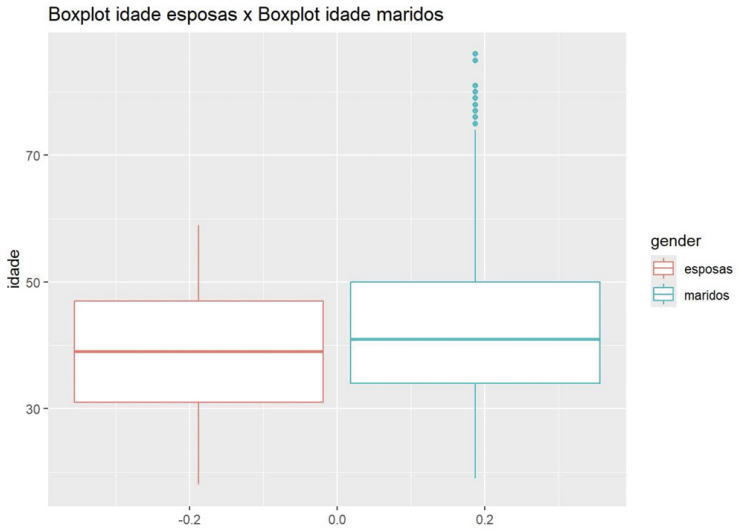
FIGURA 12 – HISTOGRAMA IDADE DAS ESPOSAS E DOS MARIDOS



FONTE: O autor (2025).

```
ggplot(dados_teste) +
geom_boxplot(aes(colour = gender, y = idade))+
labs(title = "Boxplot idade esposas x Boxplot idade maridos")
```

FIGURA 13 – BOXPLOT IDADE DAS ESPOSAS E DOS MARIDOS



FONTE: O autor (2025).

Observamos que a distribuição das idades das esposas está concentrada entre 30 e 50 anos, com idade máxima de 59 anos. Já para as idades dos maridos, observamos nos gráficos, que essa medida atinge valores mais altos, mesmo que tenha concentração também em torno de 30 e 50 anos e que homens e mulheres tenham mediana de idade similar, observamos que existem maridos com idade superior até a 70 anos, que aparecem como outliers no gráfico.

1.2- *table(dados$age)*

FIGURA 14 – DADOS DAS IDADES

```
##
##  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37
##  12  18  31  47  47  67  84 114 114 174 161 170 184 191 187 180 204 187 205 217
##  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57
## 202 171 181 158 172 185 185 140 164 128 131 125 133 117 138 103 109 104  91 104
##  58  59
## 104  95
```

FONTE: O autor (2025).

```
prop.table(table(dados$age))
```

FIGURA 15 – PROPORÇÃO DAS IDADES

```
##
##          18          19          20          21          22          23
## 0.002129925 0.003194888 0.005502307 0.008342208 0.008342208 0.011892084
##          24          25          26          27          28          29
## 0.014909478 0.020234292 0.020234292 0.030883919 0.028576500 0.030173944
##          30          31          32          33          34          35
## 0.032658857 0.033901313 0.033191338 0.031948882 0.036208733 0.033191338
##          36          37          38          39          40          41
## 0.036386226 0.038516152 0.035853745 0.030351438 0.032126376 0.028044018
##          42          43          44          45          46          47
## 0.030528931 0.032836351 0.032836351 0.024849130 0.029108981 0.022719205
##          48          49          50          51          52          53
## 0.023251686 0.022186723 0.023606674 0.020766773 0.024494143 0.018281860
##          54          55          56          57          58          59
## 0.019346823 0.018459354 0.016151935 0.018459354 0.018459354 0.016861910
```

FONTE: O autor (2025).

```
table(dados$husage)
```

FIGURA 16 – DADOS HUSAGE

```
##
##  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38
##   5   6  25  31  35  71  59 115 113 108 137 157 172 174 169 196 161 174 195 169
##  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58
## 200 175 192 172 178 201 149 149 130 147 135 127 112 128 102 104 126 103 106  86
##  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78
##  91  81  74  62  45  45  30  18  14  14  13   8   7   9   6   2   7   4   2   2
##  79  80  81  85  86
##   2   1   1   1   1
```

FONTE: O autor (2025).

```
prop.table(table(dados$husage))
```

FIGURA 17 – PROPORÇÃO DOS DADOS HUSAGE

```
##
##           19           20           21           22           23           24
## 0.0008874689 0.0010649627 0.0044373447 0.0055023074 0.0062122826 0.0126020589
##           25           26           27           28           29           30
## 0.0104721335 0.0204117856 0.0200567980 0.0191693291 0.0243166489 0.0278665247
##           31           32           33           34           35           36
## 0.0305289315 0.0308839191 0.0299964501 0.0347887824 0.0285764998 0.0308839191
##           37           38           39           40           41           42
## 0.0346112886 0.0299964501 0.0354987575 0.0310614129 0.0340788072 0.0305289315
##           43           44           45           46           47           48
## 0.0315938942 0.0356762513 0.0264465744 0.0264465744 0.0230741924 0.0260915868
##           49           50           51           52           53           54
## 0.0239616613 0.0225417110 0.0198793042 0.0227192048 0.0181043663 0.0184593539
##           55           56           57           58           59           60
## 0.0223642173 0.0182818601 0.0188143415 0.0152644657 0.0161519347 0.0143769968
##           61           62           63           64           65           66
## 0.0131345403 0.0110046148 0.0079872204 0.0079872204 0.0053248136 0.0031948882
##           67           68           69           70           71           72
## 0.0024849130 0.0024849130 0.0023074192 0.0014199503 0.0012424565 0.0015974441
##           73           74           75           76           77           78
## 0.0010649627 0.0003549876 0.0012424565 0.0007099752 0.0003549876 0.0003549876
##           79           80           81           85           86
## 0.0003549876 0.0001774938 0.0001774938 0.0001774938 0.0001774938
```

FONTE: O autor (2025).

As tabelas de frequência podem ser analisadas como um complemento aos gráficos anteriores, a partir delas, observamos que a idade mínima das esposas é 18 anos (0,2% das mulheres), enquanto a idade mínima dos maridos é de 19 anos (0,09% dos homens). As esposas com mais idade na base analisada tinham 59 anos (95 mulheres, que corrrespondem a 1,7% da base), enquanto o marido mais velho 86 anos, representando 0,01% da base.

2.1- *summary(dados$age)*

FIGURA 18 – DADOS DE IDADES

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.00   31.00   39.00   39.43   47.00   59.00
```

FONTE: O autor (2025).

```
moda <- sort(table(dados$age), decreasing = T) [1]
moda
```

Resultado da Saída:

```
# 37
# 217
```

```
summary (dados$husage)
```

FIGURA 19 – DADOS DE HUSAGE

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   34.00   41.00   42.45   50.00   86.00
```

FONTE: O autor (2025).

```
modah <- sort(table(dados$husage), decreasing = T) [1]
modah
```

Resultado da Saída:

```
# 44
# 201
```

Indo ao encontro do boxplot analisado, observamos que a mediana da idade das mulheres foi 39 anos e da idade dos homens, 41 anos. Essas medianas são um pouco mais baixas que as médias, e isso deve-se por valores mais altos que inflam um pouco a média. Os valores de primeiros quartis de idades de homens e mulheres são próximos, assim como os terceiros quartis. No entanto, observamos outliers na distribuição da idade dos homens, pelo boxplot analisado acima.

A moda se dá pelos valores mais frequentes, a moda das idades das mulheres foi de 37 anos, enquanto a moda das idades dos homens, 44 anos.

2.2- `var(dados$age)`

Resultado da Saída: `## [1] 99.75234`

`sd(dados$age)`

Resultado da Saída: `## [1] 9.98761`

`sd(dados$age)/mean(dados$age)*100`

Resultado da Saída: `## [1] 25.33153`

`var(dados$husage)`

Resultado da Saída: `## [1] 126.0717`

`sd(dados$husage)`

Resultado da Saída: *## [1] 11.22817*

```
sd(dados$husage)/mean(dados$husage)*100
```

Resultado da Saída: *## [1] 26.44849*

Observamos que o desvio padrão da idade das mulheres é inferior ao desvio padrão das idades dos homens, 9,9 e 11,2 respectivamente, assim como o desvio padrão, 25,3% e 26,4% para mulheres e homens respectivamente.

3- Teste de normalidade das variáveis age e husage através do teste de kolmogorov.

```
ks.test(dados$age,"pnorm",mean(dados$age),sd(dados$age))
```

Resultado da Saída:

```
##
## One-sample Kolmogorov-Smirnov test
##
## data: dados$age
## D = 0.058909, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks. test(dados$husage,"pnorm", mean(dados$husage), sd(dados$husage))
```

Resultado da Saída:

```
##
## One-sample Kolmogorov-Smirnov test
##
## data: dados$husage
## D = 0.059662, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Analisando o teste de kolmogorov para normalidade dos dados, observamos p-valor abaixo do nível de significância de 5%, então rejeitamos a hipótese nula de normalidade dos dados para as variáveis age e husage. Como as variáveis age e husage não têm distribuição normal, vamos precisar usar um teste não paramétrico. O teste apropriado para testar as medianas de duas amostras, é o teste de Mann Whitney.

Teste de Mann Whitney para comparar as medianas:
H0 : Não existe diferença entre os grupos

H1 : Há diferença entre os grupos

```
teste <- wilcox. test(idade~gender, data = dados_teste, exact=F, conf.int=T)
teste
```

FIGURA 20 – TESTE IDADE POR GÊNERO

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  idade by gender
## W = 13619912, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
##  -3.000024 -2.000033
## sample estimates:
## difference in location
##              -2.999966
```

FONTE: O autor (2025).

O p-valor do teste realizado foi inferior a 5%, com isso, rejeitamos a hipótese nula de que a mediana das idades de homens e mulheres são iguais. O intervalo de confiança da diferença entre as medianas está entre 2 e 3, com mediana da diferença igual a 3.

APÊNDICE 5 – 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

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 R2) e concluir qual o melhor modelo entre os três, e mostrar o resultado da predição com intervalos de confiança para os seguintes valores:

Processamento dos dados.

```
#install.packages("carData")
#install.packages("car")
#install.packages("RcmdrMisc")
#install.packages("zoo")
#install.packages("lmtest")
#install.packages("nortest")
#install.packages("lmtest")
#install.packages("sandwich")
#install.packages("caret")
#install.packages("glmnet")
library(carData)
library(car)
library(RcmdrMisc)
library(zoo)
library(lmtest)
library(nortest)
library(lmtest)
library(sandwich)
library(caret)
library(glmnet)
```

Leitura dos dados.

```
load("C:/Users/livia/Downloads/Bases de Dados Usadas nas Aulas Praticas
(2)/trabalhosalarios.
RData")
data_salarios <- trabalhosalarios
data_salarios
```

FIGURA 21 – DADOS DOS SALÁRIOS

	husage <dbl>	husunion <dbl>	husearns <dbl>	huseduc <dbl>	husblck <dbl>	hushisp <dbl>	hushrs <dbl>	kidge6 <dbl>	earns <dbl>
3	56	0	1500	14	0	0	40	1	100.0
13	31	0	800	17	0	0	40	0	480.0
20	33	0	950	13	0	0	60	0	455.0
21	34	0	1000	14	0	0	50	1	102.0
22	42	0	730	14	0	0	40	1	300.0
25	45	0	1154	16	0	0	38	1	425.0
26	33	0	1350	16	0	0	40	0	770.0
27	31	0	769	18	0	0	55	0	125.0
29	31	0	340	12	0	0	40	0	245.0
31	44	0	750	12	0	0	40	1	539.0

1-10 of 2,574 rows | 1-10 of 17 columns Previous 1 2 3 4 5 6 ... 100 Next

FONTE: O autor (2025).

Foi selecionado 80% dos dados de forma aleatória para fazer o treinamento dos modelos.

```
set.seed(123)
indice_treino <- sample(1:nrow(data_salarios), 0.8 * nrow(data_salarios))
dados_treino <- data_salarios[indice_treino, ]
dados_teste <- data_salarios[-indice_treino, ]
```

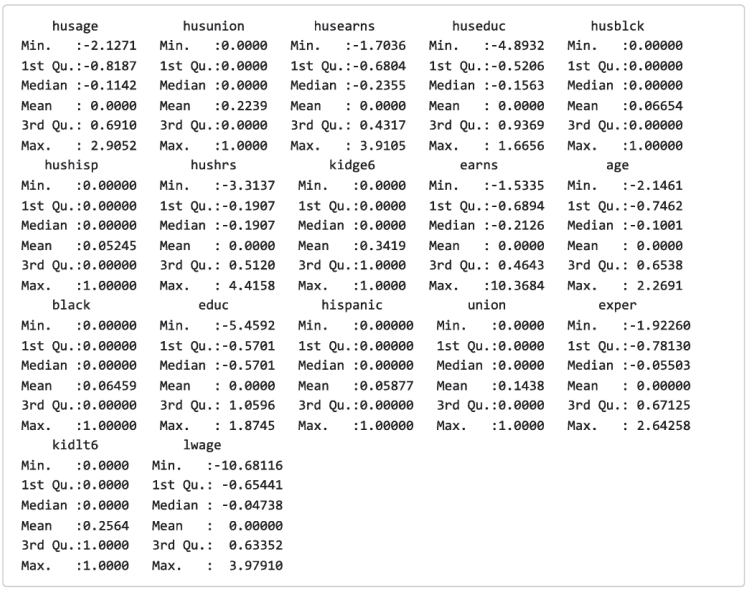
Padronização das variáveis numéricas.

```
cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'earns' , 'age' , 'educ' ,
'exper' , 'lwage' )
# Padronizando a base de treinamento e teste
pre_proc_val <- preProcess(dados_treino[,cols], method = c("center", "scale"))
dados_treino[, cols] = predict(pre_proc_val, dados_treino[,cols])
dados_teste[, cols] = predict(pre_proc_val, dados_teste[,cols])
```

Análise do Sumário das variáveis.

```
summary(dados_treino)
```

FIGURA 22 – DADOS DE TREINO

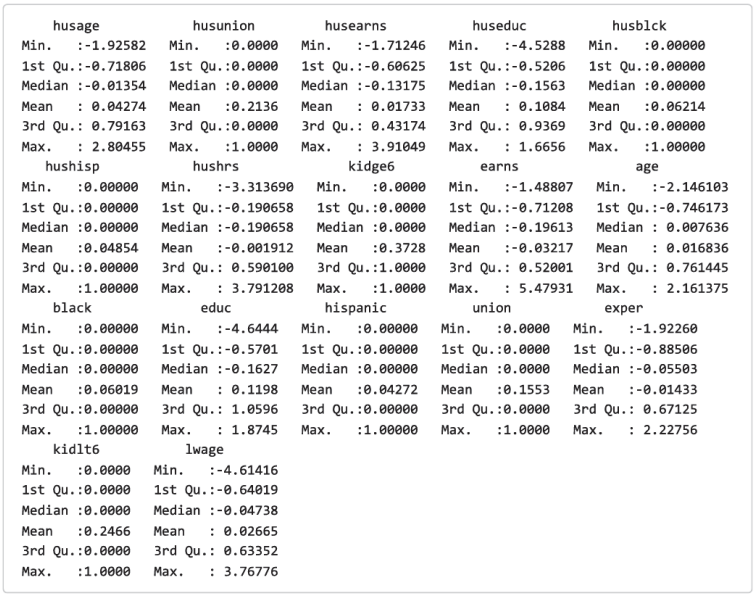


```
     husage           husunion         husearns          huseduc           husblck
 Min.   :-2.1271   Min.   :0.0000   Min.   :-1.7036   Min.   :-4.8932   Min.   :0.00000
 1st Qu.:-0.8187   1st Qu.:0.0000   1st Qu.:-0.6804   1st Qu.:-0.5206   1st Qu.:0.00000
 Median :-0.1142   Median :0.0000   Median :-0.2355   Median :-0.1563   Median :0.00000
 Mean   : 0.0000   Mean   :0.2239   Mean   : 0.0000   Mean   : 0.0000   Mean   :0.06654
 3rd Qu.: 0.6910   3rd Qu.:0.0000   3rd Qu.: 0.4317   3rd Qu.: 0.9369   3rd Qu.:0.00000
 Max.   : 2.9052   Max.   :1.0000   Max.   : 3.9105   Max.   : 1.6656   Max.   :1.00000
    hushisp            hushrs            kidge6            earns              age
 Min.   :0.00000   Min.   :-3.3137   Min.   :0.0000   Min.   :-1.5335   Min.   :-2.1461
 1st Qu.:0.00000   1st Qu.:-0.1907   1st Qu.:0.0000   1st Qu.:-0.6894   1st Qu.:-0.7462
 Median :0.00000   Median :-0.1907   Median :0.0000   Median :-0.2126   Median :-0.1001
 Mean   :0.05245   Mean   : 0.0000   Mean   :0.3419   Mean   : 0.0000   Mean   : 0.0000
 3rd Qu.:0.00000   3rd Qu.: 0.5120   3rd Qu.:1.0000   3rd Qu.: 0.4643   3rd Qu.: 0.6538
 Max.   :1.00000   Max.   : 4.4158   Max.   :1.0000   Max.   :10.3684   Max.   : 2.2691
     black              educ            hispanic           union             exper
 Min.   :0.00000   Min.   :-5.4592   Min.   :0.00000   Min.   :0.0000   Min.   :-1.92260
 1st Qu.:0.00000   1st Qu.:-0.5701   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:-0.78130
 Median :0.00000   Median :-0.5701   Median :0.00000   Median :0.0000   Median :-0.05503
 Mean   :0.06459   Mean   : 0.0000   Mean   :0.05877   Mean   :0.1438   Mean   : 0.00000
 3rd Qu.:0.00000   3rd Qu.: 1.0596   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.: 0.67125
 Max.   :1.00000   Max.   : 1.8745   Max.   :1.00000   Max.   :1.0000   Max.   : 2.64258
     kidlt6            lwage
 Min.   :0.0000   Min.   :-10.68116
 1st Qu.:0.0000   1st Qu.: -0.65441
 Median :0.0000   Median : -0.04738
 Mean   :0.2564   Mean   :  0.00000
 3rd Qu.:1.0000   3rd Qu.:  0.63352
 Max.   :1.0000   Max.   :  3.97910
```

FONTE: O autor (2025).

A seguir podemos visualizar a base de dados para o teste.

FIGURA 23 – DADOS DE TESTE



```
     husage           husunion         husearns          huseduc           husblck
 Min.   :-1.92582   Min.   :0.0000   Min.   :-1.71246   Min.   :-4.5288   Min.   :0.00000
 1st Qu.:-0.71806   1st Qu.:0.0000   1st Qu.:-0.60625   1st Qu.:-0.5206   1st Qu.:0.00000
 Median :-0.01354   Median :0.0000   Median :-0.13175   Median :-0.1563   Median :0.00000
 Mean   : 0.04274   Mean   :0.2136   Mean   : 0.01733   Mean   : 0.1084   Mean   :0.06214
 3rd Qu.: 0.79163   3rd Qu.:0.0000   3rd Qu.: 0.43174   3rd Qu.: 0.9369   3rd Qu.:0.00000
 Max.   : 2.80455   Max.   :1.0000   Max.   : 3.91049   Max.   : 1.6656   Max.   :1.00000
    hushisp            hushrs            kidge6            earns              age
 Min.   :0.00000   Min.   :-3.313690   Min.   :0.0000   Min.   :-1.48807   Min.   :-2.146103
 1st Qu.:0.00000   1st Qu.:-0.190658   1st Qu.:0.0000   1st Qu.:-0.71208   1st Qu.:-0.746173
 Median :0.00000   Median :-0.190658   Median :0.0000   Median :-0.19613   Median : 0.007636
 Mean   :0.04854   Mean   :-0.001912   Mean   :0.3728   Mean   :-0.03217   Mean   : 0.016836
 3rd Qu.:0.00000   3rd Qu.: 0.590100   3rd Qu.:1.0000   3rd Qu.: 0.52001   3rd Qu.: 0.761445
 Max.   :1.00000   Max.   : 3.791208   Max.   :1.0000   Max.   : 5.47931   Max.   : 2.161375
     black              educ            hispanic           union             exper
 Min.   :0.00000   Min.   :-4.6444   Min.   :0.00000   Min.   :0.0000   Min.   :-1.92260
 1st Qu.:0.00000   1st Qu.:-0.5701   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:-0.88506
 Median :0.00000   Median :-0.1627   Median :0.00000   Median :0.0000   Median :-0.05503
 Mean   :0.06019   Mean   : 0.1198   Mean   :0.04272   Mean   :0.1553   Mean   :-0.01433
 3rd Qu.:0.00000   3rd Qu.: 1.0596   3rd Qu.:0.00000   3rd Qu.:0.0000   3rd Qu.: 0.67125
 Max.   :1.00000   Max.   : 1.8745   Max.   :1.00000   Max.   :1.0000   Max.   : 2.22756
     kidlt6            lwage
 Min.   :0.0000   Min.   :-4.61416
 1st Qu.:0.0000   1st Qu.:-0.64019
 Median :0.0000   Median :-0.04738
 Mean   :0.2466   Mean   : 0.02665
 3rd Qu.:0.0000   3rd Qu.: 0.63352
 Max.   :1.0000   Max.   : 3.76776
```

FONTE: O autor (2025).

Foi feita a padronização dos dados de treino, as médias das variáveis numéricas são iguais a 0.

```
# Transformar os dados de treinamento e teste usando variáveis fictícias
train_dummies = predict(dummies, newdata = dados_treino[,cols_reg])
test_dummies = predict(dummies, newdata = dados_teste[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))
```

Resultado da Saída:

```
[1] 2059 16
[1] 515 16
```

```
# Vamos guardar a matriz de dados de treinamento das
# variáveis explicativas para o modelo em um objeto
# chamado "x"
x = as.matrix(train_dummies)

# Vamos guardar o vetor de dados de treinamento da
# variável dependente para o modelo em um objeto
# chamado "y_train"
y_train = dados_treino$lwage

# Vamos guardar o vetor de dados de teste da variável
# dependente para o modelo em um objeto chamado "y_test"
y_test = dados_teste$lwage
```

Métricas de avaliação para os futuros modelos.

```
# Vamos calcular o R^2 dos valores verdadeiros e
# preditos conforme a seguinte função:
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE / nrow(df))

  # As métricas de performance do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}
```

Modelo RIDGE

Cálculo do valor ótimo de lambda.

```
lambdas <- 10^seq(2, -3, by = -0.1)

# Calculando o lambda por validação cruzada:
```

```
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)

# Vamos ver qual o lambda ótimo
best_lambda_ridge <- ridge_lamb$lambda.min

# Imprimir o valor ótimo de lambda
cat('O valor ótimo de lambda foi: \n')
print(best_lambda_ridge)
```

Resultado da Saída: *[1] 0.02511886*

Estimando o modelo: coeficientes.

```
# Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
                   family = 'gaussian', lambda = best_lambda_ridge)

# Exibindo os coeficientes do modelo Ridge
ridge_reg[["beta"]]
```

FIGURA 24 – MODELO RIDGE

```
16 x 1 sparse Matrix of class "dgCMatrix"
                    s0
husage    0.012320360
husearns  0.074994288
huseduc   0.026346992
hushrs   -0.038692294
earns     0.705154984
age       0.029864158
educ      0.112606137
exper    -0.005222281
husunion  0.010018714
husblck  -0.004937410
hushisp   0.025557764
kidge6    0.046700360
black    -0.035976462
hispanic -0.100055093
union     0.136696857
kidlt6    0.139360366
```

FONTE: O autor (2025).

No modelo ridge, nenhum coeficiente estimado foi igual a zero.

Predição dos dados de treino (modelo ridge).

```
# Realizando previsões com o modelo Ridge para os dados de treinamento
```

```
predictions_train <- predict(ridge_reg, S = best_lambda_ridge, newx = x)

# Calculando as métricas para a base de treinamento
metricas_ridge_treino <- eval_results(y_train, predictions_train, dados_treino)

# Exibindo as métricas do modelo Ridge na base de treinamento
metricas_ridge_treino

RMSE                Rsquare
<dbl>               <dbl>
0.5595395           0.6867634
```

Predição dos dados de teste (modelo ridge).

```
# Predição e avaliação nos dados de teste
predictions_test <- predict(ridge_reg, s = best_lambda_ridge, newx = x_test)

# Calculando as métricas para a base de teste
metricas_ridge_teste <- eval_results(y_test, predictions_test, dados_teste)

# Exibindo as métricas do modelo Ridge na base de teste
metricas_ridge_teste

RMSE                Rsquare
<dbl>               <dbl>
0.5375854           0.7007985
```

Modelo Lasso.

Cálculo do valor ótimo de lambda.

```
# Estimando o lambda por validação cruzada para Lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1, lambda = lambdas,
                        nfolds = 10,
                        standardize = TRUE)

# Vamos ver qual o lambda ótimo
best_lambda_lasso <- lasso_lamb$lambda.min

# Exibindo o valor do melhor lambda
best_lambda_lasso  # 0.003162278
```

Resultado da Saída:

[1] 0.006309573

Estimando o modelo : coeficientes.

```
lasso_model <- glmnet(x, y_train, alpha = 1,
                      lambda = best_lambda_lasso,
                      standardize = TRUE)

lasso_model[["beta"]]
```

FIGURA 25 – MODELO DE LASSO

```
16 x 1 sparse Matrix of class "dgCMatrix"
                      s0
husage     0.0066304451
husearns   0.0692790074
huseduc    0.0210733446
hushrs    -0.0313862324
earns      0.7233054274
age        0.0201570373
educ       0.1087897978
exper      .
husunion   0.0007791477
husblck   -0.0026596447
hushisp    .
kidge6     0.0288263651
black     -0.0109971655
hispanic -0.0618425974
union      0.1194218725
kidlt6     0.1103154935
```

FONTE: O autor (2025).

Diferente do modelo Ridge, no modelo Lasso já identificamos alguns parâmetros zerados, das variáveis exper e huship.

Predição dos dados de treino (modelo Lasso).

```
predictions_train_lasso <- predict(lasso_model,
                                   S = best_lambda_lasso, newx = x)

metricas_lasso_treino <- eval_results(y_train, predictions_train_lasso,
dados_treino)

metricas_lasso_treino
RMSE                Rsquare
<dbl>               <dbl>
0.5596622           0.6866261
```

Predição dos dados de teste (modelo Lasso).

```
# Vamos fazer as predições na base de teste
predictions_test_lasso <- predict(lasso_model,
                                  s = best_lambda_lasso,
                                  newx = x_test)

# As métricas da base de teste são:
metricas_lasso_teste <- eval_results(y_test, predictions_test_lasso, dados_teste)

metricas_lasso_teste

RMSE               Rsquare
<dbl>              <dbl>
0.5340347          0.7047378
```

Modelo ELASTIC NET

Cálculo do valor ótimo de lambda e alpha e treinamento do modelo.

```
train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = FALSE)

elastic_reg <- train(lwage ~ husage + husearns + huseduc + hushrs + earns + age +
educ + expe +
                       husunion + husblck + hushisp + kidge6 + black + hispanic +
union + kidlt6,
                     data = dados_treino,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)

print('Valores ótimos para alpha e lambda:')
print(elastic_reg$bestTune)

alpha              lambda
<dbl>              <dbl>
0.8528671          0.005336688
```

Predição dos dados de treino (modelo ElasticNet).

```
predictions_train_elastic <- predict(elastic_reg, x)

# As métricas de performance na base de treinamento
# são:
metricas_elastic_treino <- eval_results(y_train, predictions_train_elastic, dados_treino)
metricas_elastic_treino
```

RMSE	Rsquare
<dbl>	<dbl>
0.5594508	0.6868627

Predição dos dados de teste (modelo ElasticNet).

```
# Vamos fazer as previsões na base de teste
predictions_test_elastic <- predict(elastic_reg, x_test)

# As métricas de performance na base de teste são:
metricas_elastic_teste <- eval_results(y_test, predictions_test_elastic,
dados_teste)
metricas_elastic_teste
```

RMSE	Rsquare
<dbl>	<dbl>
0.5341944	0.7045612

Métricas de qualidade dos modelos.

```
metricas_unificadas <- rbind.data.frame(
  metricas_ridge_treino,
  metricas_lasso_treino,
  metricas_elastic_treino,
  metricas_ridge_teste,
  metricas_lasso_teste,
  metricas_elastic_teste
)

row.names(metricas_unificadas) <- c(
  'RIDGE - TREINO',
  'LASSO - TREINO',
```

```
  'ELASTICNET - TREINO',
  'RIDGE - TESTE',
  'LASSO - TESTE',
  'ELASTICNET - TESTE'
)

metricas_unificadas
```

QUADRO 11 – MÉTRICAS DE QUALIDADE DOS MODELOS

Modelo	RMSE - TREINO	R² - TREINO	RMSE - TESTE	R² - TESTE
RIDGE	0.5595395	0.6867634	0.5375854	0.7007985
LASSO	0.5596622	0.6866261	0.5340347	0.7047378
ELASTICNET	0.5594508	0.6868627	0.5341944	0.7045612

FONTE: O autor (2025).

Predição caso proposto. Entrando com os dados.

```
# Vamos fazer uma predicao para:
# 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)

husage = (40 - pre_proc_val["mean"]]["husage"]]) /
pre_proc_val["std"]]["husage"]]
husunion = 0
# husearns = 600 (rendimento do marido em US$)
husearns = (600 - pre_proc_val["mean"]]["husearns"]]) /
pre_proc_val["std"]]["husearns"]]
# huseduc = 13 (anos de estudo do marido)
huseduc = (13 - pre_proc_val["mean"]]["huseduc"]]) /
pre_proc_val["std"]]["huseduc"]]
# husblck = 1 (o marido não é preto)
husblck = 1
# hushisp = 0 (o marido não é hispânico)
hushisp = 0
# hushrs = 40 (o marido trabalha 40 horas semanais)
hushrs = (40 - pre_proc_val["mean"]]["hushrs"]]) /
pre_proc_val["std"]]["hushrs"]]
# kidge6 = 1 (não tem filhos maiores de 6 anos)
kidge6 = 1
# earns = 355.5 (rendimento da esposa em US$)
earns = (355.5 - pre_proc_val["mean"]]["earns"]]) /
pre_proc_val["std"]]["earns"]]
# age = 38 anos (idade da esposa)
age = (38 - pre_proc_val["mean"]]["age"]]) / pre_proc_val["std"]]["age"]]
# black = 0 (esposa não é preta)
black = 0
# educ = 13 (esposa possui 13 anos de estudo)
educ = (13 - pre_proc_val["mean"]]["educ"]]) / pre_proc_val["std"]]["educ"]]
# hispanic = 1 (esposa é hispânica)
hispanic = 1
# union = 0 (o casal não possui união registrada)
union = 0
# exper = 18 (anos de experiência de trabalho da esposa)
exper = (18 - pre_proc_val["mean"]]["exper"]]) /
pre_proc_val["std"]]["exper"]]
# kidlt6 = 1 (não possui filhos com menos de 6 anos)
kidlt6 = 1

# Vamos construir uma matriz de dados para a predição
```

```
our_pred = as.matrix(data.frame(
  husage = husage,
  husunion = husunion,
  husearns = husearns,
  huseduc = huseduc,
  husblck = husblck,
  hushisp = hushisp,
  hushrs = hushrs,
  kidge6 = kidge6,
  earns = earns,
  age = age,
  black = black,
  educ = educ,
  hispanic = hispanic,
  union = union,
  exper = exper,
  kidlt6 = kidlt6
))

n <- nrow(dados_treino) # tamanho da amostra 2059
S <- pre_proc_val[["std"]][["lwage"]] # desvio padrão
dam <- S / sqrt(n) # distribuição da amostragem da média
```

RIDGE

```
# Predição com o modelo Ridge
predict_our_ridge <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = our_pred)
predict_our_ridge

# Resultado: s1
# [1,] 0.7051145

# O resultado é um valor padronizado, vamos convertê-lo
# para o valor nominal, consistente com o dataset original
wage_pred_ridge = (predict_our_ridge *
                    pre_proc_val[["std"]][["lwage"]]) +
                   pre_proc_val[["mean"]][["lwage"]]
# Resultado: s1
# [1,] 2.56323
```

```
# Intervalo de confiança RIDGE
m <- wage_pred_ridge # valor médio predito
CIlwr_ridge <- m + (qnorm(0.025)) * dam # intervalo inferior
CIupr_ridge <- m - (qnorm(0.025)) * dam # intervalo superior

# Os valores são:
CIlwr_ridge # 2.540604
```

Resultado da Saída: *s1*

```
[1,] 2.540604
CIupr_ridge # 2.585856
s1
[1,] 2.585856

LS_RIDGE <- exp(CIupr_ridge)
LI_RIDGE <- exp(CIlwr_ridge)
ESTIMAD_RIDGE <- exp(wage_pred_ridge)
```

LASSO

```
# Predição com o modelo Lasso
predict_our_lasso <- predict(lasso_model,
                             s = best_lambda_lasso,
                             newx = our_pred)
predict_our_lasso
```

Resultado da Saída: *s1*

```
[1,] 0.7441195

# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
wage_pred_lasso=(predict_our_lasso*
pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][ ["lwage" ]]

# 0 resultado eh:
wage_pred_lasso
```

Resultado da Saída: *s1*

```
[1,] 2.583662
```

```
# Este é o valor predito do salário por hora (US$),
# segundo as características que atribuímos
# s1
# 2.584648
```

Intervalo de confiança LASSO.

```
# O intervalo de confianca para o nosso exemplo eh:
m_lasso <- wage_pred_lasso # valor medio predito

CIlwr_lasso <- m_lasso + (qnorm(0.025))*dam # intervalo inferior
CIupr_lasso <- m_lasso - (qnorm(0.025))*dam # intervalo superior

# Os valores sao:
CIlwr_lasso # 2.56
```

Resultado da Saída: *s1*

```
[1,] 2.561036

CIupr_lasso # 2.61
```

Resultado da Saída: *s1*

```
[1,] 2.606288

LI_LASSO <- exp(CIlwr_lasso)
LS_LASSO <- exp(CIupr_lasso)
ESTIMAD_LASSO <- exp(m_lasso)
```

ELASTIC NET

```
# Fazendo a predicao do ELASTICNET:
predict_our_elastic <- predict(elastic_reg, our_pred)
predict_our_elastic
Resultado: [1] -0.03492759

# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
wage_pred_elastic=(predict_our_elastic*
pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][["lwage"]]
```

```
# O resultado eh:
wage_pred_elastic
```

Resultado da Saída: *[1] 2.175577*

Intervalo de confiança ELASTIC NET.

```
# O intervalo de confianca para o nosso exemplo eh:
m_elastic <- wage_pred_elastic # valor medio predito

CIlwr_elastic <- m_elastic + (qnorm(0.025))*dam # intervalo inferior
CIupr_elastic <- m_elastic - (qnorm(0.025))*dam # intervalo superior
# Os valores sao:
CIlwr_elastic # 2.56
```

Resultado da Saída: *[1] 2.152951*

```
CIupr_elastic # 2.61
```
Resultado da Saída: *[1] 2.198203*

```
LI_ELASTIC <- exp(CIlwr_elastic)
LS_ELASTIC <- exp(CIupr_elastic)
ESTIMAD_ELASTIC <- exp(m_elastic)
```

Valores preditos para pessoa x e intervalos de confiança

```
# Criando os dataframes para armazenar os resultados de cada modelo
estimacao_ridge <- cbind.data.frame(LI_RIDGE, ESTIMAD_RIDGE, LS_RIDGE)
estimacao_lasso <- cbind.data.frame(LI_LASSO, ESTIMAD_LASSO, LS_LASSO)
estimacao_elastic <- cbind.data.frame(s1 = LI_ELASTIC, s1 = ESTIMAD_ELASTIC, s1 =
LS_ELASTIC)

# Combinando os resultados de todos os modelos em um único dataframe
pessoa_x <- rbind(estimacao_ridge, estimacao_lasso, estimacao_elastic)

# Nomeando as colunas
colnames(pessoa_x) <- c('Limite Inferior IC', 'Valor Estimado', 'Limite Superior
IC')

# Nomeando as linhas
```

```
rownames(pessoa_x) <- c('Ridge', 'Lasso', 'ElasticNet')

# Exibindo o resultado final
pessoa_x
```

FIGURA 26 – VALORES PREDITIVOS

	Limite Inferior IC <dbl>	Valor Estimado <dbl>	Limite Superior IC <dbl>
Ridge	12.687332	12.977667	13.274646
Lasso	12.949224	13.245552	13.548661
ElasticNet	8.610227	8.807263	9.008807

3 rows

FONTE: O autor (2025).

Foram separados 80% dos dados para treino do modelo e 20% para teste. As bases de treino e teste foram as mesmas para os três modelos ajustados.

Verificamos que os três modelos tiveram métricas muito similares, com R2 em torno de 69% para os dados de treino, e 70% para os dados de teste, indicando ausência de overfitting nos modelos. Os RMSEs calculados também foram muito similares em torno de 0,56 para os dados de treino e 0,53 para os dados de teste.

Analisando os valores de R2 e RMSE, apesar de muito próximos nos três modelos, o Lasso tem métricas ligeiramente melhores. O valor do salário por hora estimado para a pessoa simulada foi de $13,25 com intervalo de confiança entre $12,95 e $13,55.

APÊNDICE 6 – 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

Identificador automático de idioma

Problema: Dados um texto de entrada, é possível identificar em qual língua o texto está escrito?

Entrada: "texto qualquer" Saída: português ou inglês ou francês ou italiano ou..

O processo de Reconhecimento de Padrões

O objetivo deste trabalho é demonstrar o processo de "construção de atributos" e como ele é fundamental para o **Reconhecimento de Padrões (RP).**

Primeiro um conjunto de "amostras" previamente conhecido (classificado)

```
SEED = 42  # Seed para poder debugar os dados. Se colocar None, usa tudo no modo aleatório

# Amostras de texto em diferentes línguas
ingles = [
    "Hello, how are you?",
    "I love to read books.",
    "The weather is nice today.",
    "Where is the nearest restaurant?",
    "What time is it?",
    "I enjoy playing soccer.",
    "Can you help me with this?",
    "I'm going to the movies tonight.",
    "This is a beautiful place.",
    "I like listening to music.",
    "Do you speak English?",
    "What is your favorite color?",
    "I'm learning to play the guitar.",
    "Have a great day!",
    "I need to buy some groceries.",
    "Let's go for a walk.",
    "How was your weekend?",
    "I'm excited for the concert.",
    "Could you pass me the salt, please?",
    "I have a meeting at 2 PM.",
    "I'm planning a vacation.",
    "She sings beautifully.",
    "The cat is sleeping.",
    "I want to learn French.",
    "I enjoy going to the beach.",
```

```
    "Where can I find a taxi?",
    "I'm sorry for the inconvenience.",
    "I'm studying for my exams.",
    "I like to cook dinner at home.",
    "Do you have any recommendations for restaurants?",
]

espanhol = [
    "Hola, ¿cómo estás?",
    "Me encanta leer libros.",
    "El clima está agradable hoy.",
    "¿Dónde está el restaurante más cercano?",
    "¿Qué hora es?",
    "Voy al parque todos los días.",
    "¿Puedes ayudarme con esto?",
    "Me gustaría ir de vacaciones.",
    "Este es mi libro favorito.",
    "Me gusta bailar salsa.",
    "¿Hablas español?",
    "¿Cuál es tu comida favorita?",
    "Estoy aprendiendo a tocar el piano.",
    "¡Que tengas un buen día!",
    "Necesito comprar algunas frutas.",
    "Vamos a dar un paseo.",
    "¿Cómo estuvo tu fin de semana?",
    "Estoy emocionado por el concierto.",
    "¿Me pasas la sal, por favor?",
    "Tengo una reunión a las 2 PM.",
    "Estoy planeando unas vacaciones.",
    "Ella canta hermosamente.",
    "El perro está jugando.",
    "Quiero aprender italiano.",
    "Disfruto ir a la playa.",
    "¿Dónde puedo encontrar un taxi?",
    "Lamento las molestias.",
    "Estoy estudiando para mis exámenes.",
    "Me gusta cocinar la cena en casa.",
    "¿Tienes alguna recomendación de restaurantes?",
]
```

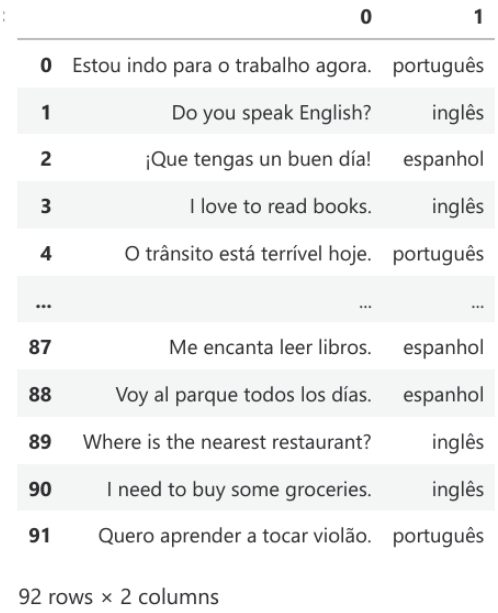
```
portugues = [
    "Estou indo para o trabalho agora.",
    "Adoro passar tempo com minha família.",
    "Preciso comprar leite e pão.",
    "Vamos ao cinema no sábado.",
    "Gosto de praticar esportes ao ar livre.",
    "O trânsito está terrível hoje.",
    "A comida estava deliciosa!",
    "Você já visitou o Rio de Janeiro?",
    "Tenho uma reunião importante amanhã.",
    "A festa começa às 20h.",
    "Estou cansado depois de um longo dia de trabalho.",
    "Vamos fazer um churrasco no final de semana.",
    "O livro que estou lendo é muito interessante.",
    "Estou aprendendo a cozinhar pratos novos.",
    "Preciso fazer exercícios físicos regularmente.",
    "Vou viajar para o exterior nas férias.",
    "Você gosta de dançar?",
    "Hoje é meu aniversário!",
    "Gosto de ouvir música clássica.",
    "Estou estudando para o vestibular.",
    "Meu time de futebol favorito ganhou o jogo.",
    "Quero aprender a tocar violão.",
    "Vamos fazer uma viagem de carro.",
    "O parque fica cheio aos finais de semana.",
    "O filme que assisti ontem foi ótimo.",
    "Preciso resolver esse problema o mais rápido possível.",
    "Adoro explorar novos lugares.",
    "Vou visitar meus avós no domingo.",
    "Estou ansioso para as férias de verão.",
    "Gosto de fazer caminhadas na natureza.",
    "O restaurante tem uma vista incrível.",
    "Vamos sair para jantar no sábado.",
                                        ]
```

A "amostras" de texto precisa ser "transformada" em padrões Um padrão é um conjunto de características, geralmente representado por um vetor e um conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as características e, geralmente, na última coluna a classe.

```
import random
```

```
# Verifica se a semente está definida
if SEED is not None:
    random.seed(SEED)

# Inicializa a lista de frases com as línguas correspondentes
pre_padroes = []

# Adiciona frases em inglês
for frase in ingles:
    pre_padroes.append([frase, 'inglês'])

# Adiciona frases em espanhol
for frase in espanhol:
    pre_padroes.append([frase, 'espanhol'])

# Adiciona frases em português
for frase in portugues:
    pre_padroes.append([frase, 'português'])

# Embaralha a lista
random.shuffle(pre_padroes)

# Importa o pandas para facilitar a visualização
import pandas as pd

# Cria o DataFrame com os dados
dados = pd.DataFrame(pre_padroes)

# Exibe o DataFrame
dados
```

FIGURA 27 – DADOS DE AMOSTRAS



	0	1
0	Estou indo para o trabalho agora.	português
1	Do you speak English?	inglês
2	¡Que tengas un buen día!	espanhol
3	I love to read books.	inglês
4	O trânsito está terrível hoje.	português
...	...	...
87	Me encanta leer libros.	espanhol
88	Voy al parque todos los días.	espanhol
89	Where is the nearest restaurant?	inglês
90	I need to buy some groceries.	inglês
91	Quero aprender a tocar violão.	português

92 rows × 2 columns

FONTE: O autor (2025).

Algoritmo para gerar Stop Words de maneira dinâmica (Não utilizado no resultado final).

Foram feitos vários destes extratores de features, e um dos que testamos foi utilizar Stop Words e Bag of Words. Ele funcionou bem e gerou resultados acima de 90%, porém notamos que no jeito que este trabalho está estruturado ocorre overfitting. Logo, decidimos não utilizar nem Stop Words nem Bag of Words no resultado final deste trabalho. Deixamos no entanto o código comentado para demonstrar a ideia que tentamos e também deixar documentado para possíveis discussões com o grupo ou com o professor.

```
import re

# Função para calcular a frequência de palavras em um dataset
def calcula_frequencia_no_dataset(elem_list, dataset):
    for elem in elem_list:
        if elem not in dataset:
            dataset[elem] = 1
        else:
            dataset[elem] += 1
    return dataset

# Função para criar o bag of words (BoW) por língua
def bagOfWords(pre_padroes):
    bow_por_lingua = {}
    for texto, lingua in pre_padroes:
```

```
        pattern_regex = re.compile('[^\w+]', re.UNICODE)  # Regex para identificar
caracteres não alfanuméricos
        texto = re.sub(pattern_regex, ' ', texto)  # Substitui todos os caracteres
não alfanuméricos por espaço
        texto = texto.lower()  # Converte o texto para minúsculas
        bow = re.findall(r'\b\w+\b', texto)  # Cria lista de palavras

        # Cria dataset para a língua, se não existir
        if lingua not in bow_por_lingua:
            bow_por_lingua[lingua] = {}

        # Atualiza o dataset da língua com a frequência das palavras
        bow_por_lingua[lingua] = calcula_frequencia_no_dataset(bow,
bow_por_lingua[lingua])

    return bow_por_lingua

# Função para encontrar as stopwords por língua
def stopWords(lista_de_linguas):
    stopWords_por_lingua = {}
    for lingua in lista_de_linguas:
        sorted_dict = sorted(bow_por_lingua[lingua].items(), key=lambda x: x[1],
reverse=True)
        stopWords_por_lingua[lingua] = dict(sorted_dict[:5])  # Seleciona as 5
palavras mais frequentes como stopwords
    return stopWords_por_lingua

# Gera a lista de línguas presentes em pre_padroes
lista_de_linguas = set(item[1] for item in pre_padroes)

# Cria o bag of words
bow_por_lingua = bagOfWords(pre_padroes)

# Encontra as stopwords por língua
stopWords_por_lingua = stopWords(lista_de_linguas)

# Exibe o Bag of Words
print('Bag of Words')
for l in lista_de_linguas:
```

```
    print(l, list(bow_por_lingua[l].keys())[:20])  # Exibe as 20 palavras mais
frequentes

# Exibe as stopwords
print('\nStop Words')
for l in lista_de_linguas:
    print(l, list(stopWords_por_lingua[l].keys()))  # Exibe as stopwords
```

Resultado de Saída: *Bag of Words português ['estou', 'indo', 'para', 'o', 'trabalho', 'agora', 'trânsito', 'está', 'terrível', 'hoj e', 'tenho', 'uma', 'reunião', 'importante', 'amanhã', 'adoro', 'passar', 'tempo', 'com', 'minha'] inglês ['do', 'you', 'speak', 'english', 'i', 'love', 'to', 'read', 'books', 'let', 's', 'go', 'fo r', 'a', 'walk', 'm', 'sorry', 'the', 'inconvenience', 'have'] espanhol ['que', 'tengas', 'un', 'buen', 'día', 'tienes', 'alguna', 'recomendación', 'de', 'restaura ntes', 'cuál', 'es', 'tu', 'comida', 'favorita', 'el', 'clima', 'está', 'agradable', 'hoy'] Stop Words português ['de', 'o', 'estou', 'para', 'no'] inglês ['i', 'the', 'to', 'a', 'm'] espanhol ['el', 'me', 'estoy', 'a', 'un']*

Construção dos atributos.

Esse é o coração desse trabalho e que deverá ser desenvolvido por vocês. Pensem em como podemos "medir" cada frase/sentença e extrair características que melhorem o resultado do processo de identificação.

Após a criação de cada novo atributo, execute as etapas seguintes e registre as métricas da matriz de confusão. Principalmente acurácia e a precisão.

Extratores de Features implementados:

- tamanhoMedioFrases: Calcula o tamanho médio das palavras na frase.
- frequenciaCaracteres: Calcula a frequência de ocorrência de cada caractere no texto.
- frequenciaBigramas: Calcula a frequência de ocorrência de bigramas (pares de caracteres consecutivos).
- frequenciaTrigramas: Calcula a frequência de ocorrência de trigramas (trios de caracteres consecutivos).
- frequenciaAcentuacoes: Calcula a frequência de ocorrência de caracteres acentuados.
- quantidadeAcentuacoes: Conta o número total de caracteres acentuados
- bagOfWords: Cria uma lista de palavras conhecidas para cada língua de maneira dinâmica (Acabamos não utilizando devido ao overfitting).
- stopWords: Cria uma lista de ocorrências mais frequentes de palavras nas frases. Elas acabam sendo
- palavras do tipo preposção, artigo, etc (Acabamos não utilizando devido ao overfitting)

```
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser "padrões"
import re
```

```
import numpy as np
import unicodedata

if SEED is not None:
    np.random.seed(SEED)

def tamanhoMedioFrases(texto):
    palavras = re.split("\s", texto)
    palvras = palavras.remove('')  # Remove palavras vazias
    # print(palavras)
    tamanhos = [len(s) for s in palavras if len(s) > 0]
    # print(tamanhos)
    soma = 0
    for t in tamanhos:
        soma = soma + t
    return soma / len(tamanhos)

def calcula_frequencia(elem_list):
    contagem_elementos = {}
    for elem in elem_list:
        if elem not in contagem_elementos:
            contagem_elementos[elem] = 1
        else:
            contagem_elementos[elem] += 1
    total = sum(contagem_elementos.values())
    # print(total)
    # print(contagem_elementos)
    frequencia_elementos = {}
    for elem, contagem in contagem_elementos.items():
        frequencia_elementos[elem] = contagem / total
    # print(frequencia_elementos)
    return frequencia_elementos

def conta_ocorrencia(elem_list):
    contagem_elementos = {}
    for elem in elem_list:
        if elem not in contagem_elementos:
            contagem_elementos[elem] = 1
        else:
            contagem_elementos[elem] += 1
```

```
    return contagem_elementos

def frequenciaCaracteres(texto):
    texto = texto.lower()
    lista_caracteres = [c for c in texto]
    frequencia_caracteres = conta_ocorrencia(lista_caracteres)
    sorted_dict = sorted(frequencia_caracteres.items(), key=lambda x: x[1],
reverse=True)
    frequencia_caracteres = dict(sorted_dict[:1])  # o mais frequente
    return frequencia_caracteres

def frequenciaBigramas(texto):
    texto = texto.lower()
    texto = re.sub(r'\s+', '', texto)  # Regex para remover todos os espaços do
texto
    bigramas = []
    for i in range(len(texto)-1):
        bigramas.append(texto[i] + texto[i+1])
    # print(bigramas)
    frequencia_bigramas = conta_ocorrencia(bigramas)
    sorted_dict = sorted(frequencia_bigramas.items(), key=lambda x: x[1],
reverse=True)
    frequencia_bigramas = dict(sorted_dict[:2])  # o mais frequente
    # print(frequencia_bigramas)
    return frequencia_bigramas

def frequenciaTrigramas(texto):
    texto = texto.lower()
    texto = re.sub(r'\s+', '', texto)  # Regex para remover todos os espaços do
texto
    trigramas = []
    for i in range(len(texto)-2):
        trigramas.append(texto[i:i+3])
    # print(trigramas)
    frequencia_trigramas = conta_ocorrencia(trigramas)
    sorted_dict = sorted(frequencia_trigramas.items(), key=lambda x: x[1],
reverse=True)
    frequencia_trigramas = dict(sorted_dict[:1])  # o mais frequente
    # print(frequencia_trigramas)
    return frequencia_trigramas
```

```
def frequenciaAcentuacoes(texto):
    texto = texto.lower()
    lista_caracteres_acentuados = []
    for c in texto:
        if c != unicodedata.normalize('NFKD', c):
            lista_caracteres_acentuados.append(c)
    # print(lista_caracteres_acentuados)
    frequencia_caracteres_acentuados =
conta_ocorrencia(lista_caracteres_acentuados)
    sorted_dict = sorted(frequencia_caracteres_acentuados.items(), key=lambda x:
x[1], reverse=True)
    frequencia_caracteres_acentuados = dict(sorted_dict[:1])  # o mais frequente
    return frequencia_caracteres_acentuados

def quantidadeAcentuacoes(texto):
    texto = texto.lower()
    qnt = 0
    for c in texto:
        if c != unicodedata.normalize('NFKD', c):
            qnt += 1
    return qnt

# def bagOfWords(texto, lingua):
#     texto = texto.lower()
#     palavras = re.findall(r'\b\w+\b', texto)
#     qnt = 0
#     for p in palavras:
#         if p in list(bow_por_lingua[lingua].keys()):
#             qnt += 1
#     return {f'bw_{lingua}': qnt}

# def stopWords(texto, lingua):
#     texto = texto.lower()
#     palavras = re.findall(r'\b\w+\b', texto)
#     qnt = 0
#     for p in palavras:
#         if p in list(stopWords_por_lingua[lingua].keys()):
#             qnt += 1
#     return {f'sw_{lingua}': qnt}
```

```
def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto, lingua = frase
    pattern_regex = re.compile('[^\w+]', re.UNICODE)  # Regex para identificar
caracteres que NÃO são
    texto = re.sub(pattern_regex, ' ', texto)  # Substitui todos os caracteres que
não são alfanuméricos
    # print(texto)
    caracteristica1 = tamanhoMedioFrases(texto)
    caracteristica2 = frequenciaCaracteres(texto)
    caracteristica3 = frequenciaBigramas(texto)
    caracteristica4 = frequenciaTrigramas(texto)
    caracteristica5 = frequenciaAcentuacoes(texto)
    caracteristica6 = quantidadeAcentuacoes(texto)
    # caracteristicaBagOfWords = bagOfWords(texto, lingua)
    # caracteristicaStopWords = stopWords(texto, lingua)
    # acrescente as suas funções no vetor padrão
    padrao = {
        'tamanhoMedioFrases': caracteristica1,
        **caracteristica2,  # O ** é um operador "Spread" de dicionários. ele
retorna todos os itens
        *caracteristica3,
        **caracteristica4,
        **caracteristica5,
        'qntAcentuacoes': caracteristica6,
        # **caracteristicaBagOfWords,
        # **caracteristicaStopWords,
        'lingua': frase[1]
    }
    return padrao

def geraPadroes(frases):
    padroes = []
    for frase in frases:
        padrao = extraiCaracteristicas(frase)
        padroes.append(padrao)
    return padroes

# converte o formato [frase classe] em
```

```
# [caracteristica_1, caracteristica_2,... caracteristica n, classe]
padroes = geraPadroes(pre_padroes)

# apenas para visualização
# print(padroes)
dados = pd.DataFrame(padroes)
dados.fillna(0, inplace=True)  # Substitui o que está com NaN para 0
dados.drop(' ', axis=1, inplace=True)  # Remove algum espaço que tenha ficado
# print(dict(dados.iloc[0]))
print(dados.shape)
Dados
```

FIGURA 28 – TAMANHO MÉDIO DAS FRASES

	tamanhoMedioFrases	ra	es	est	qntAcentuacoes	lingua	do	oy	doy	ue	...	ea	hav	rt
0	4.500000	3.0	1.0	1.0	0	português	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
1	4.250000	0.0	0.0	0.0	0	inglês	1.0	1.0	1.0	0.0	...	0.0	0.0	0.0
2	3.600000	0.0	0.0	0.0	1	espanhol	0.0	0.0	0.0	2.0	...	0.0	0.0	0.0
3	3.200000	0.0	0.0	0.0	0	inglês	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
4	5.000000	0.0	0.0	0.0	3	português	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
87	4.750000	0.0	0.0	0.0	0	espanhol	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
88	3.833333	0.0	0.0	0.0	1	espanhol	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
89	5.400000	0.0	0.0	0.0	0	inglês	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
90	3.833333	0.0	0.0	0.0	0	inglês	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0
91	5.000000	0.0	0.0	0.0	1	português	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0

92 rows × 170 columns

FONTE: O autor (2025).

Treinando o modelo com SVM Separando o conjunto de treinamento do conjunto de testes.

```
from sklearn.model_selection import train_test_split

if SEED is not None:  # Reseta o seed para evitar que de algum valor diferente
durante os testes
    np.random.seed(SEED)

# from sklearn.metrics import confusion_matrix
# vet = np.array(padroes)

classes = np.array(dados['lingua'])  # classes = [p[-1] for p in padroes]
# print(len(classes), classes)
```

```
padroes_sem_classe = np.array(dados.drop('lingua', axis=1))  # vet[:,0:-1]
# print(padroes_sem_classe)

X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe, classes,
test_size=0.25)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

Resultado da Saída: *(69, 169) (23, 169) (69,) (23,)*

Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.

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

treinador = svm.SVC()  # algoritmo escolhido
modelo = treinador.fit(X_train, y_train)

# score com os dados de treinamento
acuracia = modelo.score(X_train, y_train)
print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia * 100))

# melhor avaliar com a matriz de confusão
y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print(cm)
print(classification_report(y_train, y_pred))

# com dados de teste que não foram usados no treinamento
print('Métricas mais confiáveis')
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
print(cm)
print(classification_report(y_test, y_pred2))
```

Acurácia nos dados de treinamento: 82.61%

```
 [[11  2  9]
 [ 0 23  0]
 [ 0  1 23]]
```

QUADRO 12 – RESULTADO DAS CLASSIFICAÇÕES

Língua	Precisão (Precision)	Revocação (Recall)	F1-Score	Suporte (Support)
Espanhol	1.00	0.50	0.67	22
Inglês	0.88	1.00	0.94	23
Português	0.72	0.96	0.82	24
Acurácia			0.83	69
Média Macro	0.87	0.82	0.81	69
Média Ponderada	0.86	0.83	0.81	69

FONTE: O autor (2025).

Métricas mais confiáveis.

```
[[2 2 4]
[1 5 1]
[1 1 6]]
```

QUADRO 13 – RESULTADO DAS MÉTRICAS MAIS CONFIÁVEIS

Língua	Precisão (Precision)	Revocação (Recall)	F1-Score	Suporte (Support)
Espanhol	0.50	0.25	0.33	8
Inglês	0.62	0.71	0.67	7
Português	0.55	0.75	0.63	8
Acurácia			0.57	23
Média Macro	0.56	0.57	0.54	23
Média Ponderada	0.55	0.57	0.54	23

FONTE: O autor (2025).

Teste de Validação utilizando valores completamente fora do dataset de teste e de treino.

Dados criados por nós para ver se o modelo realmente acerta e também para simular uma aplicação real onde teríamos uma entrada fornecida por uma aplicação para que fosse inferido pelo nosso modelo treinado.

```
testes = [
    ['Cuando crezca, ganaré más dinero que mi papá.', 'espanhol'],
    ['When I grow up I will make more money than my dad.', 'inglês'],
    ['Quando crescer, quero ter mais dinheiro que meu pai.', 'português'],
    ["Learning a new language opens doors to new cultures and perspectives.",
'inglês'],
    ["Aprender um novo idioma abre portas para novas culturas e perspectivas.",
'português'],
    ["Aprender un nuevo idioma abre puertas a nuevas culturas y perspectivas.",
'espanhol'],
    ["Waking up early allows you to enjoy the quiet moments of the morning.",
'inglês'],
```

```
    ["Acordar cedo permite que você desfrute dos momentos tranquilos da manhã.",
'português'],
    ["Levantarse temprano te permite disfrutar de los momentos tranquilos de la
mañana.", 'espanhol']
]

for test in testes:
    features = extraiCaracteristicas(test)
    dados.loc[0] = features
    X_test = dados.drop('lingua', axis=1).loc[0].fillna(0)
    # dados.fillna(0, inplace=True)
    X_test = np.array(X_test)
    X_test = X_test.reshape(1, -1)
    # print(X_test.shape)
    y_pred = modelo.predict(X_test)
    print(f"Predição: {y_pred[0]}->{test[1]} ({'Correto' if y_pred[0] == test[1]
else 'Incorreto'})")
```

Resultado da Saída:
```
Predição: português->espanhol (Incorreto)
Predição: inglês->inglês (Correto)
Predição: português->português (Correto)
Predição: inglês->inglês (Correto)
Predição: português->português (Correto)
Predição: espanhol->espanhol (Correto)
Predição: inglês->inglês (Correto)
Predição: português->português (Correto)
Predição: português->espanhol (Incorreto)
```

Resultados e considerações finais

Com os extratores de features implementados, conseguimos 82,61% de acurácia no treinamento com a SEED fixa em 42. Com outros valores de SEED a acurácia varia de 75% a 85%. Cremos que cumprimos com a meta do trabalho de obter um resultado acima de 70%.

Observamos também que a acurácia no conjunto de teste foi consideravelmente menor, de 57%, sugerindo que o modelo pode estar com overfitting devido a pequena base de dados. Também, a precisão, recall e f1 score variam significativamente entre as classes, especialmente para o espanhol, onde o recall foi mais baixo no conjunto de teste, demonstrando que o modelo pode vir a ter dificuldades identificando espanhol. Por outro lado, o desempenho para inglês foi relativamente melhor, com uma precisão e recall mais equilibrados. Para testar isso criamos um pequeno set de validação (na célula acima), para podermos colocar dados totalmente diferentes do dataset inicial e ter uma validação a mais.

Uma solução para esses problemas de acurácia, e overfitting seria usar uma base de dados maior, ou reimplementar o código utilizando a nossa proposta de ter um stopwords a bag of words dinâmico.

APÊNDICE 7 – 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

CLASSIFICAÇÃO

Para o experimento de Classificação:

- Ordenar pela Acurácia (descendente), ou seja, a técnica de melhor acurácia ficará em primeiro na tabela.
- Após o quadro colocar:
    - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior Acurácia (criar um arquivo com novos casos à sua escolha)
    - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

FIGURA 29 – CLASSIFICAÇÃO PARA VEÍCULO

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	Reference Prediction bus opel saab van bus 43 0 0 0 opel 0 42 0 0 saab 0 0 43 0 van 0 0 0 39
RNA – Hold-out	size=5 decay=0.1	85% (0.8503)	Reference Prediction bus opel saab van bus 41 0 1 2 opel 0 30 7 1 saab 1 9 35 0 van 1 3 0 36
SVM – Hold-out	C=1 Sigma= 0.07189928	85% (0.8502)	Reference Prediction bus opel saab van bus 43 0 0 1 opel 0 26 7 0 saab 0 12 35 0 van 0 4 1 38
SVM – CV	C=100 Sigma= 0.015	84% (0.8443)	Reference Prediction bus opel saab van bus 40 0 0 1 opel 1 34 12 2 saab 0 8 31 0 van 2 0 0 36
RNA – CV	size=11 decay=0.4	81% (0.8084)	Reference Prediction bus opel saab van bus 39 0 1 1 opel 0 27 9 1 saab 1 11 32 0 van 3 4 1 37
RF – CV	mtry=10	74% (0.7365)	Reference Prediction bus opel saab van bus 42 0 1 0 opel 0 18 13 2 saab 0 21 27 1 van 1 3 2 36
KNN	k=1	68% (0.6766)	Reference Prediction bus opel saab van bus 39 1 1 3 opel 1 17 16 1 saab 1 20 22 0 van 2 4 4 35

FONTE: O autor (2025).

A seguir, veremos os novos casos para a classificação de veículos.

FIGURA 30 – NOVOS CASOS DE VEÍCULOS

Novos casos:

a	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVaraxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
1	75	55	89	105	133	36	117	60	24	140	130	806	204	70	3	28	192	190	**van**
2	89	42	56	241	124	7	200	34	28	148	200	450	261	92	1	29	184	208	**bus**
3	115	37	107	315	55	48	251	28	20	125	320	210	139	100	9	37	181	201	**opel**

FONTE: O autor (2025).

Comandos executados no RStudio:

```
# 1.a veículos (classificação) - Random Forest Hold Out
install.packages("e1071")
install.packages("caret")
library("caret")

setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-aprendizado-maquina/bases-de-dados/06veículos")

data <- read.csv("6-veiculos.csv")
View(data)

data$a <- NULL

any(is.na(data))
# FALSE

preproc_center_scale <- preProcess(data, method = c("center", "scale"))
normalized_data <- predict(preproc_center_scale, data)
# Dados normalizados com média centralizada em 0

View(normalized_data)

set.seed(202493)
ind <- createDataPartition(normalized_data$tipo, p = 0.8, list = F)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]
# --- Hold out ---

set.seed(202493)
rf <- train(tipo ~ ., data = normalized_data, method = "rf")
```

```
rf
# mtry = 2

predict.rf <- predict(rf, test)
confusionMatrix(predict.rf, as.factor(test$tipo))
# Accuracy: 1

# --- Novos casos (usando Hold out) ----

new_data <- read.csv("6-veiculos-novos-dados.csv")
View(new_data)

new_data$a <- NULL

any(is.na(new_data))
# FALSE

preproc_center_scale <- preProcess(new_data, method = c("center", "scale"))
normalized_new_data <- predict(preproc_center_scale, new_data)
# Dados normalizados com média centralizada em 0

View(normalized_new_data)

predict.rf_new_data <- predict(rf, normalized_new_data)
# van bus opel
# Levels: bus opel saab van

new_data$tipo <- NULL
result <- cbind(new_data, predict.rf_new_data)
names(result)[names(result) == "predict.rf_new_data"] <- "tipo"
View(result)
# Visualização do DF com os novos dados e a predição
```

FIGURA 31 – CLASSIFICAÇÃO PARA DIABETES

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	Reference Prediction neg pos neg 100 0 pos 0 53
SVM – Hold-out	C=0.25 Sigma= 0.1258432	78% (0.7843)	Reference Prediction neg pos neg 93 26 pos 7 27
RNA – Hold-out	size=1 decay=0.1	78% (0.7778)	Reference Prediction neg pos neg 90 24 pos 10 29
RF – CV	mtry=2	77% (0.7712)	Reference Prediction neg pos neg 89 24 pos 11 29
SVM – CV	C=2 Sigma= 0.015	76% (0.7582)	Reference Prediction neg pos neg 92 29 pos 8 24
RNA – CV	size=3 decay=0.1	76% (0.7581)	Reference Prediction neg pos neg 88 25 pos 12 28
KNN	k=9	73% (0.7255)	Reference Prediction neg pos neg 84 26 pos 16 27

FONTE: O autor (2025).

A seguir veremos novos casos para diabetes.

FIGURA 32 – NOVOS CASOS PARA DIABETES

Novos casos:

num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	2	182	97	52	88	44	2001	48	**pos**
2	8	99	114	24	249	28	1588	31	**neg**
3	14	48	68	87	659	21	1263	61	**neg**

FONTE: O autor (2025).

Comandos executados no RStudio:

```
# 1.b diabetes (classificação) - Random Forest – Hold Out
install.packages("e1071")
install.packages("caret")
library("caret")

setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-aprendizado-maquina/bases-de-dados/10diabetes")

data <- read.csv("10-diabetes.csv")
View(data)

data$num <- NULL

any(is.na(data))
# FALSE

preproc_center_scale <- preProcess(data, method=c("center", "scale"))
normalized_data <- predict(preproc_center_scale, data)
# Dados normalizados com média centralizada em 0

View(normalized_data)

set.seed(202493)
ind <- createDataPartition(normalized_data$diabetes, p = 0.8, list = FALSE)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]

# --- Hold out ---

set.seed(202493)
rf <- train(diabetes ~ ., data = normalized_data, method = "rf")
rf
# mtry = 2

predict.rf <- predict(rf, test)
confusionMatrix(predict.rf, as.factor(test$diabetes))
# Accuracy: 1

# --- Novos casos (usando Hold out) ----
```

```
new_data <- read.csv("10-diabetes-novos-dados.csv")
View(new_data)

new_data$num <- NULL

any(is.na(new_data))
# FALSE

preproc_center_scale <- preProcess(new_data, method = c("center", "scale"))
normalized_new_data <- predict(preproc_center_scale, new_data)
# Dados normalizados com média centralizada em 0

View(normalized_new_data)

predict.rf_new_data <- predict(rf, normalized_new_data)
predict.rf_new_data
# pos neg neg
# Levels: neg pos

new_data$diabetes <- NULL
result <- cbind(new_data, predict.rf_new_data)
names(result)[names(result) == "predict.rf_new_data"] <- "diabetes"
View(result)
# Visualização do DF com os novos dados e a predição
```

REGRESSÃO

Para o experimento de Regressão:

- Ordenar por R2 descendente, ou seja, a técnica de melhor R2 ficará em primeiro na tabela.
- Após o quadro, colocar:
    - Um resultado com 3 linhas com a predição de novos casos para a técnica/parâmetro de maior R2 (criar um arquivo com novos casos à sua escolha)
    - O Gráfico de Resíduos para a técnica/parâmetro de maior R2 o A lista de comandos emitidos no RStudio para conseguir os resultados obtidos

FIGURA 33 – DADOS DE REGRESSÃO

Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.9458	0.03368	0.97462	0.03334	0.02296
RNA – Hold-out	size=41 decay=0.1	0.8341	0.05895	0.91351	0.058348	0.044075
RNA – CV	size=16 decay=0.1	0.8218	0.06110	0.90938	0.060474	0.048163
SVM – CV	C=50 Sigma=0.015	0.8209	0.61251	0.90913	0.60622	0.04388
RF – CV	mtry=2	0.8046	0.06398	0.89732	0.06332	0.04513
SVM – Hold-out	C=0.5 Sigma= 0.1769097	0.8026	0.06430	0.89797	0.063643	0.045819
KNN	K=9	0.7883	0.06659	0.89068	0.065908	0.04751

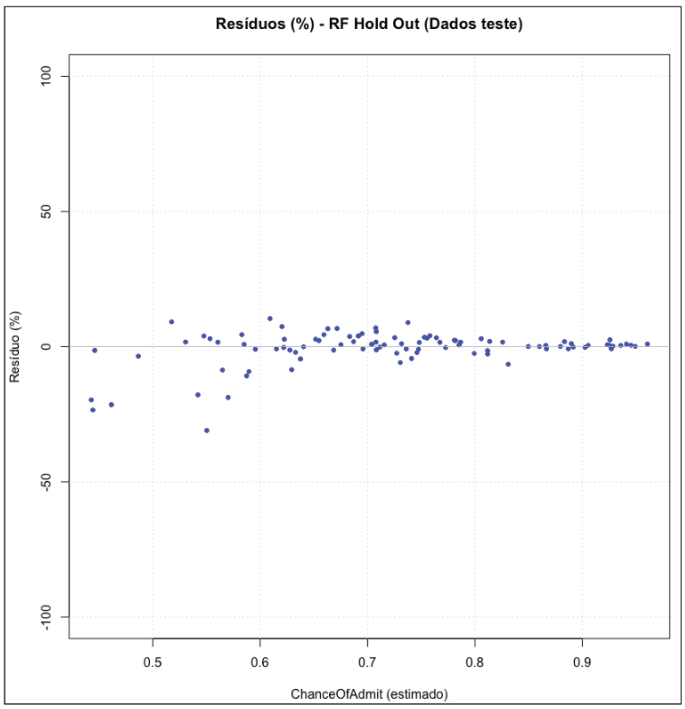
Novos casos:

num	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	ChanceOfAdmit
1	299	114	3	4	2	8.4	1	**0.6088426**
2	318	103	2	5	3	8.8	0	**0.7209769**
3	327	98	5	1	3	8.9	1	**0.7601318**

FONTE: O autor (2025).

Gráfico de resíduos:

FIGURA 34 – GRÁFICO DE RESÍDUOS



FONTE: O autor (2025).

Comandos executados no RStudio:

```
# 2.a admissão (regressão) - Random Forest – Hold Out
install.packages("e1071")
```

```
install.packages("kernlab")
install.packages("caret")
install.packages("mice")
library("caret")
library(Metrics)
library(stats)
library(mice)

setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-aprendizado-maquina/bases-de-
dados/09admissão")

data <- read.csv("9-admissao.csv")
View(data)

data$num <- NULL

any(is.na(data))
# FALSE

target_data <- data[["ChanceOfAdmit"]]
predictors <- data[, colnames(data) != "ChanceOfAdmit"]

preproc_center_scale <- preProcess(predictors, method=c("center", "scale"))
normalized_predictors <- predict(preproc_center_scale, predictors)

normalized_data <- cbind(normalized_predictors, target_data)

names(normalized_data)[names(normalized_data) == "target_data"] <- "ChanceOfAdmit"

View(normalized_data)

set.seed(202493)
ind <- createDataPartition(normalized_data$ChanceOfAdmit, p = 0.8, list = FALSE)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]

# --- Hold out ---

set.seed(202493)
rf_ho <- train(ChanceOfAdmit ~ ., data = normalized_data, method = "rf")
```

```
rf_ho
# mtry = 2

predict.rf_ho <- predict(rf_ho, test)

r2 <- function(predicted, observed) {
  return (1 - (sum((predicted - observed) ^ 2) / sum((observed - mean(observed)) ^
2)))
}

syx <- function(predicted, observed) {
  n <- length(observed)
  syx <- sqrt(sum((observed - predicted)^2) / (n - 2))
  return(syx)
}

rmse(test$ChanceOfAdmit, predict.rf_ho)
# 0.0333386

r2(predict.rf_ho, test$ChanceOfAdmit)
# 0.9458273

syx(predict.rf_ho, test$ChanceOfAdmit)
# 0.03368409

cor(test$ChanceOfAdmit, predict.rf_ho) # Pearson (library stats)
# 0.9746234

mae(test$ChanceOfAdmit, predict.rf_ho)
# 0.02295854

# --- Novos casos (usando Hold out) ----

new_data <- read.csv("9-admissao-novos-dados.csv")
View(new_data)

new_data$num <- NULL

any(is.na(new_data))
# FALSE
```

```
new_target_data <- new_data[["ChanceOfAdmit"]]
new_predictors <- new_data[, colnames(new_data) != "ChanceOfAdmit"]

preproc_center_scale <- preProcess(new_predictors, method=c("center", "scale"))
normalized_new_predictors <- predict(preproc_center_scale, new_predictors)

normalized_new_data <- cbind(normalized_new_predictors, new_target_data)
names(normalized_new_data)[names(normalized_new_data) == "new_target_data"] <-
  "ChanceOfAdmit"
# Dados normalizados com média centralizada em 0

View(normalized_new_data)

predict.rf_ho_new_data <- predict(rf_ho, normalized_new_data)
predict.rf_ho_new_data
# 1 2 3
# 0.6088426 0.7209769 0.7601318

new_data$ChanceOfAdmit <- NULL
result <- cbind(new_data, predict.rf_ho_new_data)
names(result)[names(result) == "predict.rf_ho_new_data"] <- "ChanceOfAdmit"
View(result)
# Visualização do DF com os novos dados e a predição

# --- Geração do Gráfico de Resíduos com RF Hold Out e Dados de teste ---

test_residuals <- ((test$ChanceOfAdmit - predict.rf_ho) / test$ChanceOfAdmit) *
100

plot(
  predict.rf_ho,
  test_residuals,
  col = "blue",
  pch = 20,
  main = "Resíduos (%) - RF Hold Out (Dados teste)",
  xlab = "ChanceOfAdmit (estimado)",
  ylab = "Resíduo (%)",
  ylim=c(-100, 100)
)
```

```
abline(h = 0, col = "gray")
grid()
```

FIGURA 35 – DADOS DE BIOMASSA

Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.98	125.5789	0.9964	123.4682	41.8887
RF – CV	mtry=3	0.9702	153.4874	0.99179	150.9076	67.3541
RNA – CV	Size=36 decay=0.1	0.9397	218.1783	0.97679	214.5112	106.5668
KNN	K=1	0.9181	254.4146	0.95862	250.1385	101.6077
SVM – CV	C=100 Sigma=0.01	0.8987	282.9218	0.98783	278.1665	124.8422
RNA – Hold-out	size=1 decay=0.4	0.8645	327.1441	0.97536	321.6455	165.1712
SVM – Hold-out	C=1 Sigma=1.443104	0.8004	397.1035	0.92895	390.429	169.621

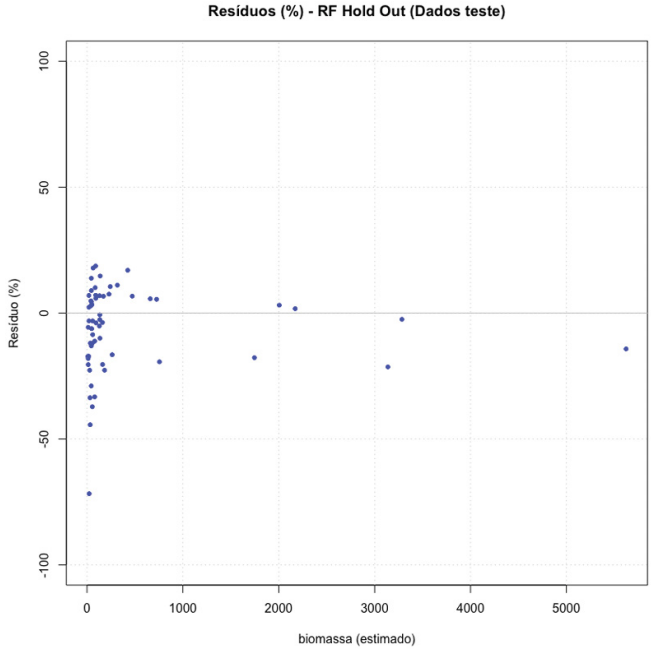
Novos casos:

dap	h	Me	biomassa
122.8	40.9	0.7	**568.6312**
56	16.1	0.83	**73.9615**
13.2	27.7	10.1	**45.1129**

FONTE: O autor (2025).

Abaixo temos o gráfico de resíduos e dados de teste.

FIGURA 36 – GRÁFICO DE RESÍDUOS E DADOS DE TESTE



FONTE: O autor (2025).

```
# 2.b biomassa (regressão) - Random Forest – Hold Out
install.packages("e1071")
```

```
install.packages("kernlab")
install.packages("caret")
install.packages("mice")
library("caret")
library(Metrics)
library(stats)
library(mice)

setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-aprendizado-maquina/bases-de-dados/05biomassa")

data <- read.csv("5-biomassa.csv")
View(data)

any(is.na(data))
# FALSE

target_data <- data[["biomassa"]]
predictors <- data[, colnames(data) != "biomassa"]

preproc_center_scale <- preProcess(predictors, method=c("center", "scale"))
normalized_predictors <- predict(preproc_center_scale, predictors)

normalized_data <- cbind(normalized_predictors, target_data)

names(normalized_data)[names(normalized_data) == "target_data"] <- "biomassa"

View(normalized_data)

set.seed(202493)
ind <- createDataPartition(normalized_data$biomassa, p=0.8, list=FALSE)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]

# --- Hold out ---

set.seed(202493)
rf_ho <- train(biomassa ~ ., data = normalized_data, method = "rf")
rf_ho
# mtry = 2
```

```
predict.rf_ho <- predict(rf_ho, test)

r2 <- function(predicted, observed) {
  return (1 - (sum((predicted - observed) ^ 2) / sum((observed - mean(observed)) ^
2)))
}

syx <- function(predicted, observed) {
  n <- length(observed)
  syx <- sqrt(sum((observed - predicted)^2) / (n - 2))
  return(syx)
}

rmse(test$biomassa, predict.rf_ho)
# 123.4682

r2(predict.rf_ho, test$biomassa)
# 0.9800358

syx(predict.rf_ho, test$biomassa)
# 125.5789

cor(test$biomassa, predict.rf_ho) # Pearson (library stats)
# 0.9963851

mae(test$biomassa, predict.rf_ho)
# 41.88866

# --- Novos casos (usando Hold out) ----

new_data <- read.csv("5-biomassa-novos-dados.csv")
View(new_data)

any(is.na(new_data))
# FALSE

new_target_data <- new_data[["biomassa"]]
new_predictors <- new_data[, colnames(new_data) != "biomassa"]
```

```
preproc_center_scale <- preProcess(new_predictors, method=c("center", "scale"))
normalized_new_predictors <- predict(preproc_center_scale, new_predictors)

normalized_new_data <- cbind(normalized_new_predictors, new_target_data)

names(normalized_new_data)[names(normalized_new_data) == "new_target_data"] <-
  "biomassa"

# Dados normalizados com média centralizada em 0

View(normalized_new_data)

predict.rf_ho_new_data <- predict(rf_ho, normalized_new_data)

predict.rf_ho_new_data
# 1 2 3
# 568.6312 73.9615 45.1129

new_data$biomassa <- NULL
result <- cbind(new_data, predict.rf_new_data)
names(result)[names(result) == "predict.rf_new_data"] <- "biomassa"
View(result)
# Visualização do DF com os novos dados e a predição

# --- Geração do Gráfico de Resíduos com RF Hold Out e Dados de teste ---

test_residuals <- ((test$biomassa - predict.rf_ho) / test$biomassa) * 100

plot(predict.rf_ho, test_residuals,
     col = "blue", pch = 20,
     main = "Resíduos (%) - RF Hold Out (Dados teste)",
     xlab = "biomassa (estimado)", ylab = "Resíduo (%)",
     ylim=c(-100, 100))

abline(h = 0, col = "gray")
grid()
```

AGRUPAMENTO

FIGURA 37 – LISTA DE CLUSTERS DE VEÍCULO

Veículo

Lista de Clusters gerados:

```
K-modes clustering with 10 clusters of sizes 75, 76, 109, 74, 72, 144, 62, 56, 83, 95
```

10 primeiras linhas do arquivo com o cluster correspondente.

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

Usa 10 clusters no experimento.

Cluster modes:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
1	85	39	66	116	56	6	122	57	17	135	137	209	121	64	7	7	181	183	van
2	89	36	75	169	64	7	161	41	20	131	189	430	139	66	1	2	195	202	saab
3	104	54	103	197	59	11	213	31	24	162	226	669	212	72	1	11	188	201	opel
4	89	44	72	141	56	7	152	44	19	145	170	314	177	72	0	14	187	187	opel
5	94	45	85	150	64	10	157	43	20	148	173	354	159	69	1	12	192	196	van
6	85	43	70	120	54	7	150	45	19	144	169	325	171	85	1	1	180	184	bus
7	89	46	85	133	56	11	159	43	20	160	176	349	186	74	7	16	183	192	van
8	100	43	96	197	63	6	186	35	22	143	202	485	198	68	4	6	195	199	bus
9	104	55	101	230	62	11	222	30	25	172	228	713	201	71	0	21	187	198	opel
10	101	48	104	209	62	10	208	32	23	158	214	635	214	73	5	11	187	197	saab

Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos

```r
# Veículos (agrupamento)
install.packages("e1071")
install.packages("caret")
install.packages("klaR")
library(klaR)
library("caret")

setwd("D:/Cursos/UFPR/IAA008 - Aprendizado de Máquina/0 - Apresentação da disciplina/praticas/06-Veiculos")
data <- read.csv("6-Veiculos-Dados.csv")
View(data)

data$a <- NULL
any(is.na(data))
# FALSE

set.seed(202493)

cluster.results <- kmodes(data, 10, iter.max = 10, weighted = FALSE )
cluster.results

resultado <- cbind(data, cluster.results$cluster)
resultado
```

FONTE: O autor (2025).

FIGURA 38 – VISUALIZAÇÃO DE DADOS DE MUSCULAÇÃO

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

Visualizando os dados:

```
> inspect(dados[1:10])
     items
[1]  {Afundo, Crucifixo, Gemeos, LegPress}
[2]  {Agachamento, Gemeos, LegPress}
[3]  {Afundo, Agachamento, Gemeos, LegPress}
[4]  {Adutor, Agachamento, LegPress}
[5]  {Afundo, Bicicleta, Gemeos, LegPress}
[6]  {Agachamento, Gemeos, LegPress}
[7]  {Afundo, Bicicleta, Gemeos, LegPress}
[8]  {Adutor, Agachamento, LegPress}
[9]  {AgachamentoSmith, Bicicleta, Esteira, Extensor, Gemeos}
[10] {AgachamentoSmith, Bicicleta, Esteira, Extensor, LegPress}
```

Obtendo a visão geral dos dados:

- O arquivo contém 26 linhas
- Itens mais frequentes (LegPress 21 vezes e Gemeos 17 vezes)

```
> summary(dados)
transactions as itemMatrix in sparse format with
 26 rows (elements/itemsets/transactions) and
 11 columns (items) and a density of 0.3881119

most frequent items:
 LegPress    Gemeos Bicicleta  Extensor   Esteira   (Other)
       21        17        14        13        12        34

element (itemset/transaction) length distribution:
sizes
 3  4  5
 6  7 13
```

Definindo Suporte=0,2 e Confianca=0,7:

- Foram criadas 44 regras

FONTE: O autor (2025).

FIGURA 39 – AVALIAÇÃO DAS REGRAS

- 1 regras com 1 itens
- 14 regras com 2 itens
- 23 regras com 3 itens
- 6 regras com 4 itens

```
> summary(rules)
set of 44 rules

rule length distribution (lhs + rhs):sizes
 1  2  3  4
 1 14 23  6

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.0     3.0     2.8     3.0     4.0

summary of quality measures:
    support         confidence          coverage            lift            count
 Min.   :0.23   Min.   :0.70   Min.   :0.23   Min.   :0.87   Min.   : 6.0
 1st Qu.:0.23   1st Qu.:0.77   1st Qu.:0.31   1st Qu.:1.52   1st Qu.: 6.0
 Median :0.27   Median :0.86   Median :0.31   Median :1.64   Median : 7.0
 Mean   :0.31   Mean   :0.85   Mean   :0.37   Mean   :1.59   Mean   : 8.0
 3rd Qu.:0.36   3rd Qu.:0.89   3rd Qu.:0.38   3rd Qu.:1.81   3rd Qu.: 9.2
 Max.   :0.81   Max.   :1.00   Max.   :1.00   Max.   :2.00   Max.   :21.0

mining info:
  data ntransactions support confidence                                                call
 dados            26     0.2        0.7 apriori(data = dados, parameter = list(supp = 0.2, conf = 0.7))
```

Avaliação de regras:

- As duas primeiras regras nos dizem que quem faz Agachamento tem 100% de chance fazer LegPress e quem faz Afundo tem também 100% de chance de fazer Gemeos
- Já a terceira regra diz que quem faz Esteira tem 92% de chance de fazer Extensor

```
     lhs                    rhs          support confidence coverage lift count
[1]  {Agachamento}       => {LegPress}   0.31    1.00       0.31     1.24  8
[2]  {Afundo}            => {Gemeos}     0.35    1.00       0.35     1.53  9
[3]  {Esteira}           => {Extensor}   0.42    0.92       0.46     1.83 11
[4]  {AgachamentoSmith}  => {Extensor}   0.35    0.90       0.38     1.80  9
[5]  {Extensor}          => {Esteira}    0.42    0.85       0.50     1.83 11
```

FONTE: O autor (2025).

Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

FIGURA 40 – COMANDOS EMITIDOS NO RSTUDIO

```
# musculação (associação)
install.packages("e1071")
install.packages("caret")
install.packages('arules', dep=T)
library(arules)
library(datasets)
library("caret")

setwd("D:/Cursos/UFPR/IAA008 - Aprendizado de Máquina/0 - Apresentação da disciplina/praticas/12-Regras_de_Associacao-Praticas/12-Regras_de_Associacao
dados <- read.transactions(file="2-Musculacao-Dados.csv",format="basket",sep=";")
inspect(dados[1:10])
summary(dados)

set.seed(202493)
#definindo o suporte=0,2 e confiança=0,7
rules <- apriori(dados, parameter = list(supp = 0.2, conf = 0.7))
summary(rules)
options(digits=2)
inspect(sort(rules[1:5], by="confidence"))
```

FONTE: O autor (2025).

APÊNDICE 8 – 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

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.

Link Google Colab: https://colab.research.google.com/drive/1_z-wM_G7r-ynSifDmfEYVPunT0JrOLGP?usp=sharing

```
# Importar as bibliotecas necessárias
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
# Carregar o conjunto de dados CIFAR-10
```

```
(train_images, train_labels), (test_images, test_labels) =
datasets.cifar10.load_data()

# Normalizar as imagens para o intervalo [0, 1]
train_images, test_images = train_images / 255.0, test_images / 255.0

train_labels, test_labels = train_labels.flatten(), test_labels.flatten()

print("train_images.shape: ", train_images.shape)
print("train_labels.shape: ", train_labels.shape)
print("test_images.shape: ", test_images.shape)
print("test_labels.shape: ", test_labels.shape)

# Mapear os rótulos para nomes das classes
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog',
'horse', 'ship', 'truck']
```

FIGURA 41 – DADOS DE TREINAMENTO E TESTE

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170498071/170498071 [==============================] - 13s 0us/step
train_images.shape:  (50000, 32, 32, 3)
train_labels.shape:  (50000,)
test_images.shape:  (10000, 32, 32, 3)
test_labels.shape:  (10000,)
```

FONTE: O autor (2025).

```
# Construir a arquitetura da CNN

# Quantidade de parâmetros de saída com base na quantidade de classes
k = len(set(train_labels))

model = models.Sequential([
    # Estágio 1
    layers.Input(shape=train_images[0].shape),

    # Camada convolucional 1
    layers.Conv2D(32, (3, 3), activation='relu', strides=2),

    # Camada convolucional 2
  layers.Conv2D(64, (3, 3), activation='relu', strides=2),
```

```
    # Camada convolucional 3
    layers.Conv2D(128, (3, 3), activation='relu', strides=2),

    # Camada de flatten e fully connected
    layers.Flatten(),

    # Estágio 2
    layers.Dropout(0.5),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(k, activation='softmax')
])

model.summary()
```

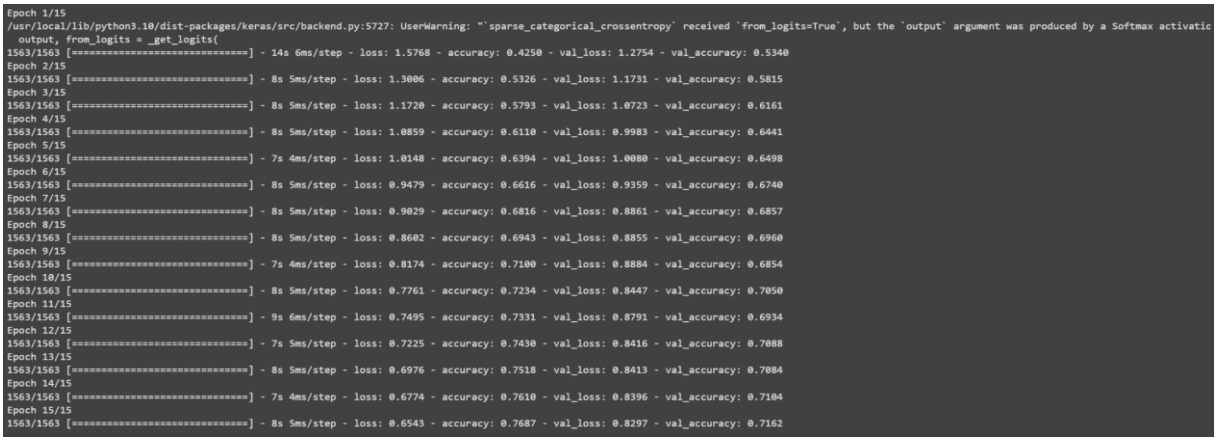
FIGURA 42 – MODELO SEQUENCIAL

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 15, 15, 32)        896

 conv2d_1 (Conv2D)           (None, 7, 7, 64)          18496

 conv2d_2 (Conv2D)           (None, 3, 3, 128)         73856

 flatten (Flatten)           (None, 1152)              0

 dropout (Dropout)           (None, 1152)              0

 dense (Dense)               (None, 1024)              1180672

 dropout_1 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 10)                10250

=================================================================
Total params: 1284170 (4.90 MB)
Trainable params: 1284170 (4.90 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

FONTE: O autor (2025).

```
# Compilar o modelo
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
r = model.fit(train_images, train_labels, epochs=15,
                    validation_data=(test_images, test_labels))
```

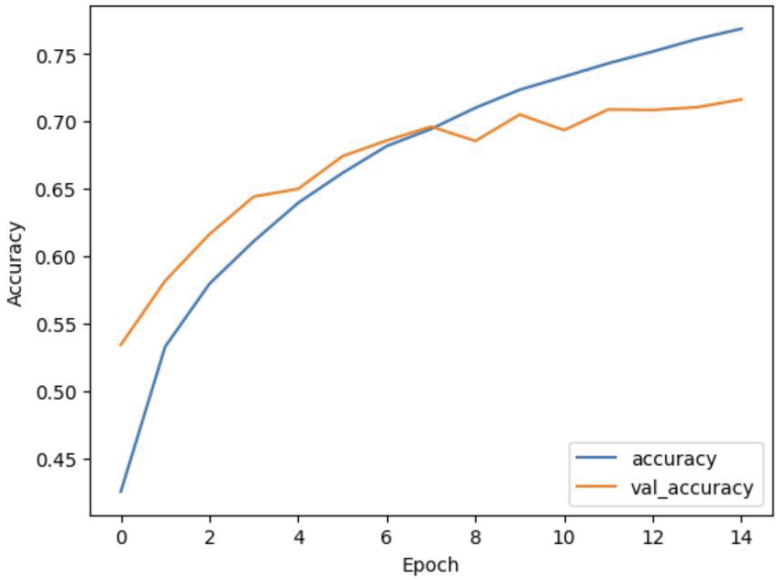
FIGURA 43 – CARREGAMENTO DAS IMAGENS



FONTE: O autor (2025).

```
# Visualizar o desempenho

# Gráfico de acurácia
plt.plot(r.history['accuracy'], label='accuracy')
plt.plot(r.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.show()

# Gráfico de perda
plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='lower right')
plt.show()
```

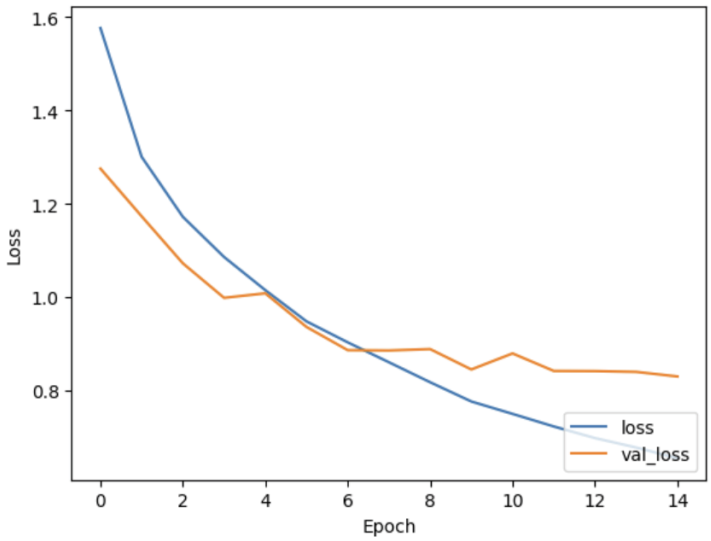
FIGURA 44 – GRÁFICO DE ACURÁCIA



FONTE: O autor (2025).

Abaixo os dados de perda.

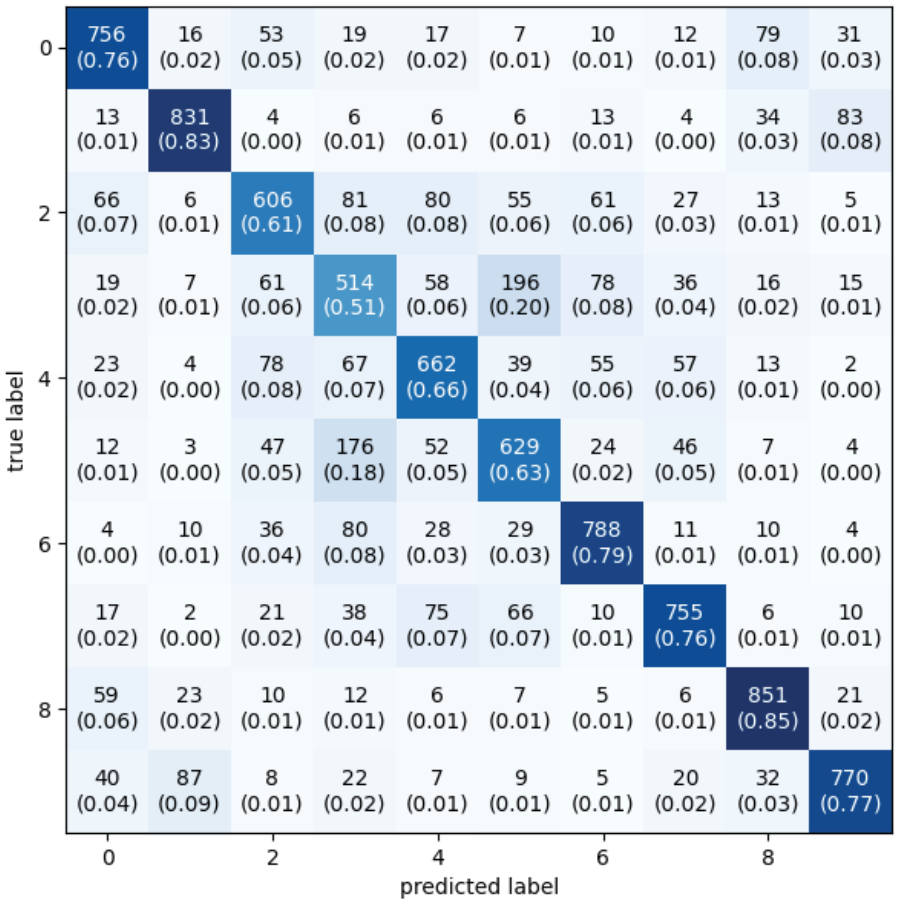
FIGURA 45 – GRÁFICO DE PERDA



FONTE: O autor (2025).

```
# Efetuando a predição

pred_labels = model.predict(test_images).argmax(axis=1)

# Matriz de confusão
cm = confusion_matrix(test_labels, pred_labels)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
```

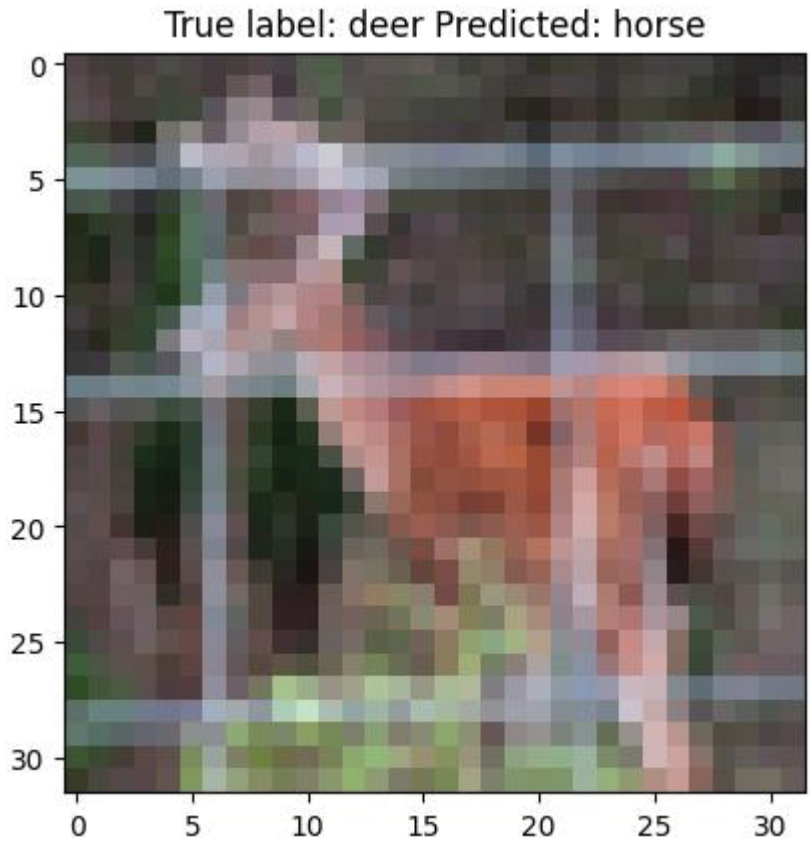
FIGURA 46 – MATRIZ DE CONFUSÃO



FONTE: O autor (2025).

```
# Exibindo algumas classificações erradas
misclassified = np.where(pred_labels != test_labels)[0]
i = np.random.choice(misclassified)

plt.imshow(test_images[i], cmap="gray")
plt.title(f"True label: {class_names[test_labels[i]]} Predicted:
{class_names[pred_labels[i]]}")
```

FIGURA 47 – CLASSIFICAÇÕES ERRADAS



FONTE: O autor (2025).

2. Detector de SPAM (RNN

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

Link Google Colab: https://colab.research.google.com/drive/10vDAJzRrQR4xkgujlOKuYvH4okbr8qc6?usp=sharing

```
# Importação

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.models import Model
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.preprocessing.text import Tokenizer
# Carregar base de dados

# Verificar o caminho do arquivo
df = pd.read_csv("/content/spam.csv", encoding="ISO-8859-1")
df.head()

# removendo algumas colunas
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
print(df.head())
```

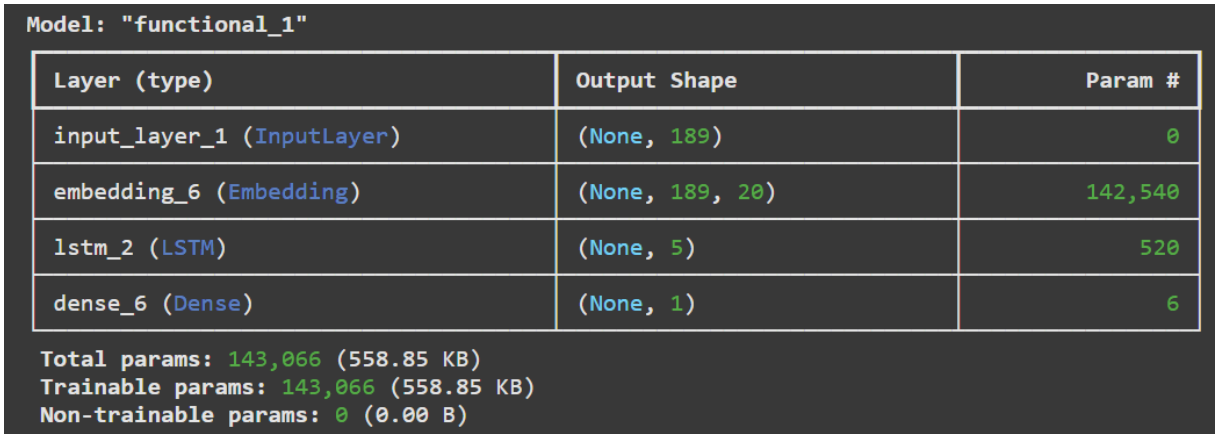
FIGURA 48 – REMOÇÃO DE ALGUMAS COLUNAS

```
  labels                                               data  b_labels
0    ham  Go until jurong point, crazy.. Available only ...         0
1    ham                      Ok lar... Joking wif u oni...         0
2   spam  Free entry in 2 a wkly comp to win FA Cup fina...         1
3    ham  U dun say so early hor... U c already then say...         0
4    ham  Nah I don't think he goes to usf, he lives aro...         0
```

FONTE: O autor (2025).

```
# Separação das bases para treinamento e teste
x_train, x_test, y_train, y_test = train_test_split(df["data"], y, test_size=0.33)
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(f"{V} tokens")

7126 tokens

# Acertar tamanho das sequências

data_train = pad_sequences(sequences_train)

T = data_train.shape[1]
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

data_train.shape:  (3733, 189)
data_test.shape:  (1839, 189)

# Definição do modelo

D = 20
M = 5

i = Input(shape=(T,))
x = Embedding(V + 1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x)
model = Model(i, x)

model.summary()
```

FIGURA 49 – MODELO FUNCIONAL



Model: "functional_1"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 189)	0
embedding_6 (Embedding)	(None, 189, 20)	142,540
lstm_2 (LSTM)	(None, 5)	520
dense_6 (Dense)	(None, 1)	6

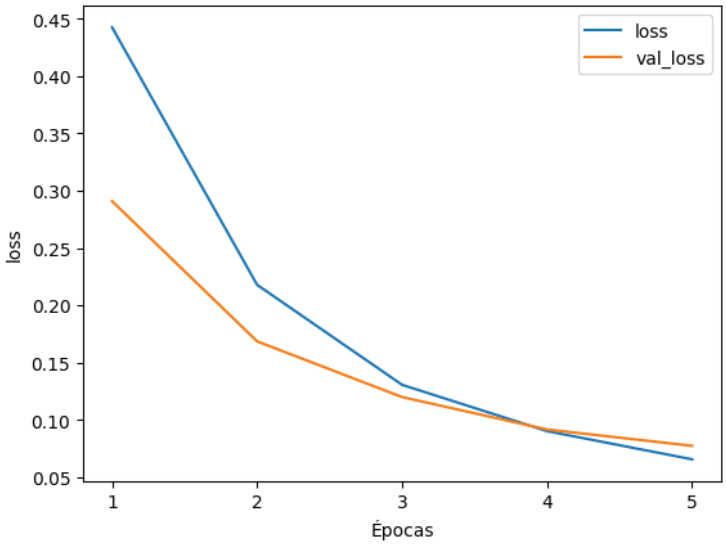
Total params: 143,066 (558.85 KB)
Trainable params: 143,066 (558.85 KB)
Non-trainable params: 0 (0.00 B)

FONTE: O autor (2025).

```
# Compilar 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)
)
# Visualizar accuracy e loss

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("accuracy")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs + 1))
plt.legend()
plt.show()
```

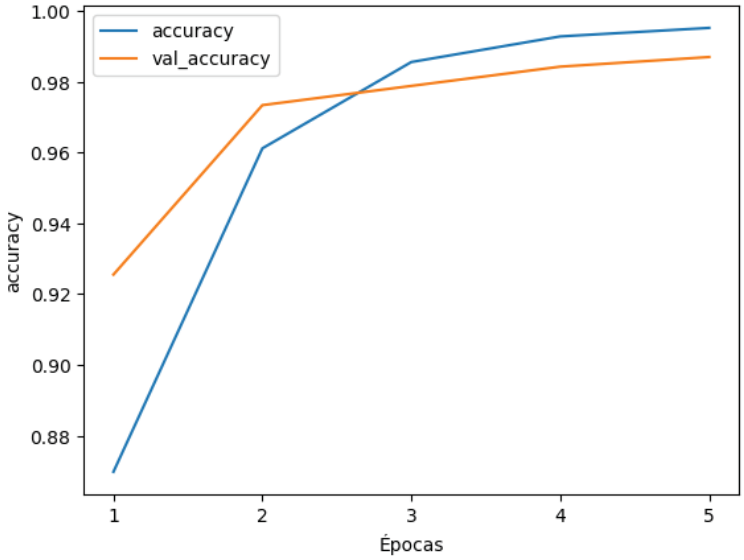
FIGURA 50 – ACURÁCIA E PERDA



FONTE: O autor (2025).

A seguir o gráfico de acurácia.

FIGURA 51 – GRÁFICO DE ACURÁCIA



FONTE: O autor (2025).

```
text = "Is your cellphone carrier bad? Check out our free for all plans!. Click the
link"

seq_texto = tokenizer.texts_to_sequences([text])
data_texto = pad_sequences(seq_texto, maxlen = T)

pred = model.predict(data_texto)
print(pred)
print("SPAM" if pred >= 0.5 else "OK")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 102ms/step
[[0.5963417]]
SPAM
```

3. Gerador de Dígitos Fake (GAN)

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

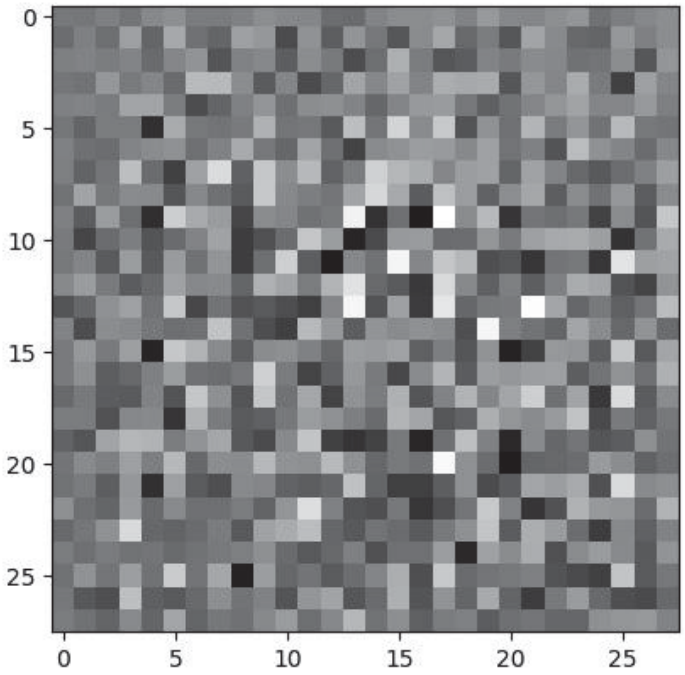
Link Google Colab:
https://colab.research.google.com/drive/1Qdws2GSR3Q1ePH-In27KZeujp5fvudXW?usp=sharing

```
# 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
# Carrega a base de dados MNIST
(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')
# Normaliza entre [-1, 1]
train_images = (train_images - 127.5) / 127.5
# 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).shuffle(BUFFER_SIZE).batch(BATCH_
SIZE)
# Cria o GERADOR
def make_generator_model():
  # Modelo sequencial
  # Camada de entrada
  # Camada de batch normalization
  # Camada de ativação
  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())

  # Camada de reshape
  model.add(layers.Reshape((7, 7, 256)))
  assert model.output_shape == (None, 7, 7, 256)
```

```
  # Camada de convolução transposta
  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'))
  # Camada de batch normalization
  assert model.output_shape == (None, 28, 28, 1)

  return model
# Teste do GERADOR, ainda não treinado
# Cria um modelo
generator = make_generator_model()

# Gera uma imagem
noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)

# Plota a imagem
plt.imshow(generated_image[0, :, :, 0], cmap='gray')
```

FIGURA 52 – GERANDO IMAGEM



FONTE: O autor (2025).

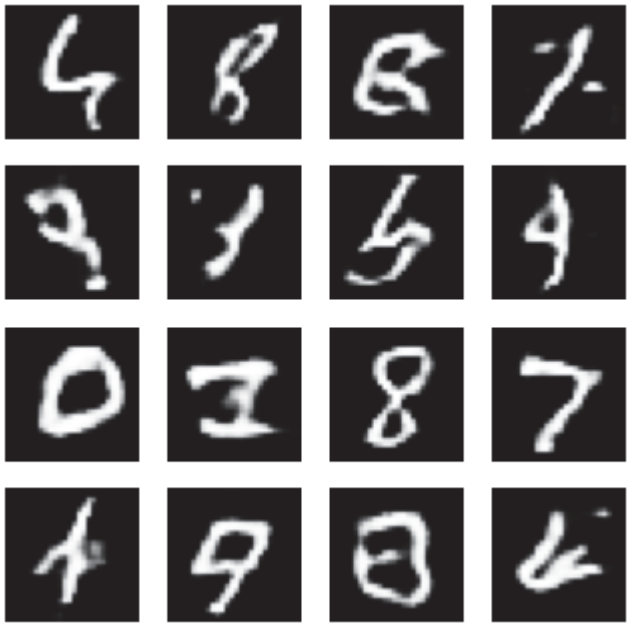
```
# Cria o DISCRIMADOR
def make_discriminator_model():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(64,    (5,    5),    strides=(2,    2),    padding='same',
input_shape=[28, 28, 1]))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))

  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)
# Perda binária cruzada
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
```

```
# Função que calcula a perda do discriminador
def discriminator_loss(real_output, fake_output):
  # Calcula a perda do real
  real_loss = cross_entropy(tf.ones_like(real_output), real_output)
  # Calcula a perda do fake
  fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
  # Calcula a perda total
  total_loss = real_loss + fake_loss
  return total_loss

# Função que calcula a perda do gerador
def generator_loss(fake_output):
  # Calcula a perda do fake
  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
# ou interrupção abrupta
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
# Parâmetros
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
# para um código mais rápido quando chamada
@tf.function
def train_step(images):
  # Gerar ruído
```

```
  noise = tf.random.normal([BATCH_SIZE, noise_dim])
  # Treinar
  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    generated_images = generator(noise, training=True)
    # Treinar o discriminador
    real_output = discriminator(images, training=True)
    fake_output = discriminator(generated_images, training=True)
    # Calcular a perda
    gen_loss = generator_loss(fake_output)
    disc_loss = discriminator_loss(real_output, fake_output)
  # Calcular os gradientes
  gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
  gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
  # Aplicar os gradientes
  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
# Função que treina o modelo
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))
```

FIGURA 53 – TREINANDO O MODELO

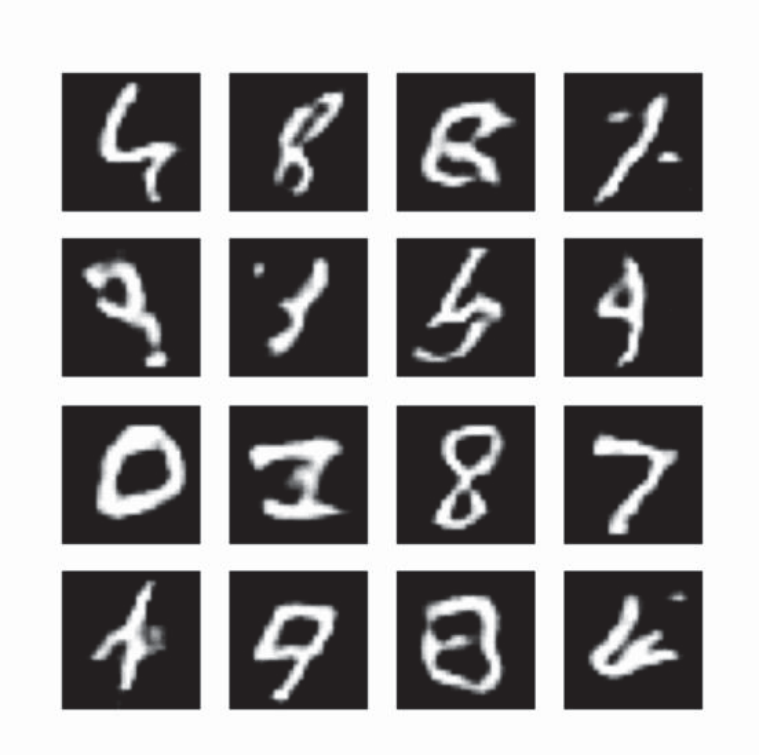


FONTE: O autor (2025).

```
# 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)
```

FIGURA 54 – CRIANDO UM GIF DE IMAGEM



FONTE: O autor (2025).

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

Link Google Colab: https://colab.research.google.com/drive/1nl2rBx2wwPHIomhOpX5wyd9-AyId4D0t?usp=sharing

```
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0
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
examples, metadata = tfds.load(
    'ted_hrlr_translate/pt_to_en',
    with_info=True,
    as_supervised=True
)
train_examples, val_examples = examples['train'], examples['validation']
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'))
e quando melhoramos a procura , tiramos a única vantagem da impressão , que é a serendipidade .
mas e se estes fatores fossem ativos ?
mas eles não tinham a curiosidade de me testar .

and when you improve searchability , you actually take away the one advantage of print , which is serendipity .
but what if it were active ?
but they did n't test for curiosity .

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)
```

```
def tokenize_pairs(pt, en):
  pt = tokenizers.pt.tokenize(pt)
  pt = pt.to_tensor()

  en = tokenizers.en.tokenize(en)
  en = en.to_tensor()
  return pt, en

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)
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
  )
  angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
  angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

  pos_encoding = angle_rads[np.newaxis, ...]
  return tf.cast(pos_encoding, dtype=tf.float32)

n, d = 2048, 512
```

```
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]

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))

# o plot a seguir não é necessário
# plt.pcolormesh(pos_encoding, cmap='RdBu')
# plt.ylabel('Depth')
# plt.xlabel('Position')
# plt.colorbar()
# plt.show()

def create_padding_mask(seq):
  seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
  return seq[:, tf.newaxis, tf.newaxis, :]

def create_look_ahead_mask(size):
  mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
  return mask
def scaled_dot_product_attention(q, k, v, mask):
  # Q * K ^ T
  matmul_qk = tf.matmul(q, k, transpose_b=True)
  dk = tf.cast(tf.shape(k)[-1], tf.float32)

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

  if mask is not None:
    scaled_attention_logits += (mask * -1e9)

  attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
  output = tf.matmul(attention_weights, v)
  return output, attention_weights

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

```
    super().__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):
    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)
    k = self.wk(k)
    v = self.wv(v)

    q = self.split_heads(q, batch_size)
    k = self.split_heads(k, batch_size)
    v = self.split_heads(v, batch_size)

    scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v,
mask)
    scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
    concat_attention = tf.reshape(scaled_attention, (batch_size, -1, self.d_model))

    output = self.dense(concat_attention)

    return output, attention_weights

def point_wise_feed_forward_network(d_model, dff):
  return tf.keras.Sequential([
      tf.keras.layers.Dense(dff, activation='relu'),
```

```
      tf.keras.layers.Dense(d_model)
  ])
class EncoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate = 0.1):
    super().__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)
    attn_output = self.dropout1(attn_output, training = training)
    out1 = self.layernorm1(x + attn_output)

    ffn_output = self.ffn(out1)
    ffn_output = self.dropout2(ffn_output, training = training)
    out2 = self.layernorm2(out1 + ffn_output)

    return out2
class DecoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate = 0.1):
    super().__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):
    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training = training)
    out1 = self.layernorm1(attn1 + x)

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

    ffn_output = self.ffn(out2)
    ffn_output = self.dropout3(ffn_output, training = training)
    out3 = self.layernorm3(ffn_output + out2)

    return out3, attn_weights_block1, attn_weights_block2

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().__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]
    x = self.embedding(x)
    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
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().__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)
    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

    return x, attention_weights
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):
    print('inputs:', inputs)
    print('training:', training)
    inp, tar = inputs
    enc_padding_mask, look_ahead_mask, dec_padding_mask = self.create_masks(inp,
tar)
    enc_output = self.encoder(inp, training, enc_padding_mask)
    dec_output, attention_weights = self.decoder(
        tar,
        enc_output,
        training,
        look_ahead_mask,
        dec_padding_mask
    )
    final_output = self.final_layer(dec_output)

    return final_output, attention_weights

  def create_masks(self, inp, tar):
    enc_padding_mask = create_padding_mask(inp)
    dec_padding_mask = create_padding_mask(inp)
    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

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

class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
  def __init__(self, d_model, warmup_steps = 4000):
```

```
    super().__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)
    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
)
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')

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
)
EPOCHS = 25
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()
  epoch_count = epoch + 1
```

```
  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_count % 5 == 0:
    ckpt_save_path = ckpt_manager.save()
    print(f"Saving checkpoint for epoch {epoch_count} at {ckpt_save_path}")

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

Epoch 1 Batch 0 Loss 8.8324 Accuracy 0.0022
Epoch 1 Batch 50 Loss 8.7743 Accuracy 0.0079
Epoch 1 Batch 100 Loss 8.6878 Accuracy 0.0256
Epoch 1 Batch 150 Loss 8.5781 Accuracy 0.0359
Epoch 1 Batch 200 Loss 8.4368 Accuracy 0.0428
Epoch 1 Batch 250 Loss 8.2663 Accuracy 0.0487
Epoch 1 Batch 300 Loss 8.0755 Accuracy 0.0582
Epoch 1 Batch 350 Loss 7.8788 Accuracy 0.0660
Epoch 1 Batch 400 Loss 7.6913 Accuracy 0.0730
Epoch 1 Batch 450 Loss 7.5252 Accuracy 0.0792
Epoch 1 Batch 500 Loss 7.3793 Accuracy 0.0851
Epoch 1 Batch 550 Loss 7.2447 Accuracy 0.0916
Epoch 1 Batch 600 Loss 7.1197 Accuracy 0.0985
Epoch 1 Batch 650 Loss 7.0035 Accuracy 0.1051
Epoch 1 Batch 700 Loss 6.8972 Accuracy 0.1112
Epoch 1 Batch 750 Loss 6.7963 Accuracy 0.1170
Epoch 1 Batch 800 Loss 6.7057 Accuracy 0.1223
Epoch 1 Loss 6.6897 Accuracy 0.1233
Time taken for epoch 1: 171.75 secs

Epoch 25 Batch 0 Loss 1.3378 Accuracy 0.6965
Epoch 25 Batch 50 Loss 1.2698 Accuracy 0.7064
Epoch 25 Batch 100 Loss 1.2818 Accuracy 0.7054
Epoch 25 Batch 150 Loss 1.2908 Accuracy 0.7041
```

```
Epoch 25 Batch 200 Loss 1.2908 Accuracy 0.7043
Epoch 25 Batch 250 Loss 1.2914 Accuracy 0.7042
Epoch 25 Batch 300 Loss 1.2943 Accuracy 0.7042
Epoch 25 Batch 350 Loss 1.2969 Accuracy 0.7037
Epoch 25 Batch 400 Loss 1.3001 Accuracy 0.7031
Epoch 25 Batch 450 Loss 1.3048 Accuracy 0.7024
Epoch 25 Batch 500 Loss 1.3072 Accuracy 0.7020
Epoch 25 Batch 550 Loss 1.3089 Accuracy 0.7016
Epoch 25 Batch 600 Loss 1.3096 Accuracy 0.7018
Epoch 25 Batch 650 Loss 1.3128 Accuracy 0.7012
Epoch 25 Batch 700 Loss 1.3153 Accuracy 0.7010
Epoch 25 Batch 750 Loss 1.3201 Accuracy 0.7002
Epoch 25 Batch 800 Loss 1.3243 Accuracy 0.6995
Saving checkpoint for epoch 25 at ./checkpoints/train/ckpt-5
Epoch 25 Loss 1.3248 Accuracy 0.6994
Time taken for epoch 25: 97.99 secs
```

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

  def __call__(self, sentence, max_length = 20):
    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

    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:, :]
      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())
    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

translator = Translator(tokenizers, transformer)

sentence = "vamos testar o tradutor."

translated_text, translated_tokens, attention_weights = translator(
    tf.constant(sentence)
)

print(f"{'Original':15s} {sentence}")
print(f"{'Prediction':15s} {translated_text}")
```

```
Original        vamos testar o tradutor.
Prediction      b"let ' s test the translator ."
```

APÊNDICE 9 – 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

Título: IMPLEMENTAÇÃO DE APLICAÇÃO DE BIG DATA: ESTUDO DE CASO COM NOSQL E NEWSQL

RESUMO: A crescente geração e complexidade dos dados impulsionam a necessidade de novas abordagens para o gerenciamento de Big Data. Este trabalho explora a implementação de uma aplicação de e-commerce utilizando tecnologias NoSQL e NewSQL para otimizar o armazenamento e processamento de grandes volumes de dados. O estudo de caso apresenta o uso de MongoDB para dados semi-estruturados e logs, Cassandra para dados de transações e CockroachDB para dados estruturados e transações ACID. A análise destaca as características de cada tecnologia, a modelagem necessária e a eficácia em diferentes cenários de dados. A combinação dessas ferramentas permite uma solução robusta, escalável e eficiente, adequando se às necessidades específicas da aplicação de e-commerce.
Palavras-chave: Big Data. NoSQL. NewSQL. MongoDB. Modelagem de Dados.

ABSTRACT: The growing generation and complexity of data drive the need for new approaches to Big Data management. This paper explores the implementation of an e-commerce application using NoSQL and NewSQL technologies to optimize the storage and processing of large data volumes. The case study presents the use of MongoDB for semi-structured data and logs, Cassandra for transaction data, and CockroachDB for structured data and ACID transactions. The analysis highlights the characteristics of each technology, the necessary modeling, and effectiveness in different data scenarios. The combination of these tools enables a robust, scalable, and efficient solution, tailored to the specific needs of the e-commerce application.
Keywords: Big Data. NoSQL. NewSQL. MongoDB. Data Modeling.

1 INTRODUÇÃO

Com o crescimento exponencial dos dados gerados por empresas e usuários, as soluções tradicionais de banco de dados relacional se tornaram insuficientes para atender às demandas de escalabilidade, desempenho e flexibilidade. Este documento explora a implementação de uma aplicação de Big Data, focando em como ferramentas NoSQL e NewSQL podem ser usadas para

gerenciar grandes volumes de dados. O estudo de caso apresentado envolve uma plataforma de e-commerce que utiliza essas tecnologias para melhorar sua eficiência e experiência do usuário.

2 CARACTERIZAÇÃO DOS DADOS

Na aplicação de e-commerce, os dados são variados e incluem:

- Dados de Transações: Informações sobre compras, pagamentos e devoluções.
- Dados de Usuários: Perfis de clientes, histórico de navegação e preferências.
- Dados de Produtos: Detalhes dos produtos, categorias e avaliações.
- Dados de Logs: Registros de atividades dos usuários e do sistema.

Esses dados têm diferentes características e exigem modelos de armazenamento e processamento específicos. Por exemplo, os dados de transações são estruturados e frequentemente consultados, enquanto os dados de logs são semiestruturados e precisam ser processados rapidamente para análise em tempo real.

3 FERRAMENTAS UTILIZADAS

1. NoSQL

1.1. MongoDB

Modelo de Dados: Documentos JSON

Características: Alta escalabilidade e flexibilidade. Ideal para dados semiestruturados e não-estruturados, como logs de atividades e perfis de usuários.

Modelagem: Os dados de usuários e produtos são armazenados em coleções de documentos. Isso permite consultas rápidas e escalabilidade horizontal.

1.2. Cassandra

Modelo de Dados: Colunas.

Características: Alta disponibilidade e desempenho para grandes volumes de dados. Adequado para dados de transações e logs, onde a escrita e leitura rápida são essenciais.

Modelagem: Os dados de transações são modelados como linhas em uma tabela de colunas, permitindo consultas rápidas e eficientes.

2. NewSQL

2.1. CockroachDB

Modelo de Dados: Relacional com suporte a SQL.

Características: Combina a escalabilidade horizontal dos bancos NoSQL com a consistência e a robustez dos bancos de dados relacionais.

Modelagem: Os dados de produtos e transações são armazenados em tabelas relacionais, garantindo consistência e integridade referencial, enquanto suportam grandes volumes e alta concorrência.

4 MODELAGEM DE DADOS

Para a implementação da aplicação, a modelagem de dados foi adaptada conforme o modelo de banco de dados escolhido:

1. NoSQL (MongoDB e Cassandra):

Modelagem de Documentos (MongoDB): Os dados são armazenados em documentos JSON, permitindo a inclusão de campos aninhados e flexíveis, o que é ideal para perfis de usuários e logs de atividades.

Modelagem de Colunas (Cassandra): As tabelas são desenhadas para suportar grandes volumes de dados com alta taxa de escrita, como as transações de e-commerce.

2. NewSQL (CockroachDB):

Modelagem Relacional: Dados estruturados são armazenados em tabelas com esquemas fixos, proporcionando consistência e suporte a transações ACID. Isso é ideal para dados críticos de produtos e transações financeiras.

5 CONSIDERAÇÕES FINAIS

A escolha entre NoSQL e NewSQL depende das necessidades específicas da aplicação. NoSQL é excelente para flexibilidade e escalabilidade em dados semi estruturados e não-estruturados, enquanto NewSQL oferece o melhor dos dois mundos com escalabilidade e consistência para dados estruturados. A combinação dessas tecnologias pode proporcionar uma solução robusta e eficiente para aplicações de Big Data.

REFERÊNCIAS

Documentação oficial do MongoDB, Cassandra e CockroachDB.
Artigos e estudos de caso sobre implementação de Big Data com NoSQL e NewSQL.

APÊNDICE 10 – 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

1- Extração de características

Métricas dos modelos:

Random Forest- Sensitivity: 0.91, Specificity: 0.39, F1-Score: 0.65

SVM-Sensitivity: 1.00, Specificity: 0.00, F1-Score: 0.52

RNA- Sensitivity: 1.00, Specificity: 0.00, F1-Score: 0.51

Conclusão:

Entre os três modelos, o Random Forest parece ser a melhor opção. Ele tem uma sensibilidade de 0.91, o que significa que consegue identificar bem as imagens da classe positiva, e uma especificidade de 0.39, mostrando que, mesmo que não seja perfeito, ainda consegue diferenciar as classes negativas. Já o SVM e o RNA têm uma sensibilidade de 1.00, o que parece muito bom, mas a especificidade deles é 0.00, ou seja, eles falham completamente em identificar corretamente as classes negativas. Além disso, o F1-Score do Random Forest é o mais alto (0.65), o que indica que ele oferece um melhor equilíbrio geral entre precisão e recall, sendo, no fim, o modelo mais confiável para essa tarefa.

2- Redes Neurais

Avaliação do VGG16 (sem augmentation):

Matriz de confusão:

	0	1	2	3
0	[29	13	34	25]
1	[27	16	33	14]
2	[30	6	30	24]
3	[37	12	24	17]

FIGURA 55 – AVALIAÇÃO DO VGG16 (SEM ARGUMENTAÇÃO)

Report de classificação:

	precision	recall	f1-score	support
0	0.24	0.29	0.26	101
1	0.34	0.18	0.23	90
2	0.25	0.33	0.28	90
3	0.21	0.19	0.20	90
accuracy			0.25	371
macro avg	0.26	0.25	0.24	371
weighted avg	0.26	0.25	0.24	371

Acurácia: 24.80%

FONTE: O autor (2025).

Avaliação do VGG16 (com augmentation):

Matriz de confusão:

	0	1	2	3
0	[33	15	28	25]
1	[35	10	29	16]
2	[35	12	29	14]
3	[20	10	35	25]

FIGURA 56 – AVALIAÇÃO DO VGG16 (COM ARGUMENTAÇÃO)

Report de classificação:

	precision	recall	f1-score	support
0	0.27	0.33	0.29	101
1	0.21	0.11	0.15	90
2	0.24	0.32	0.27	90
3	0.31	0.28	0.29	90
accuracy			0.26	371
macro avg	0.26	0.26	0.25	371
weighted avg	0.26	0.26	0.25	371

Acurácia: 26.15%

FONTE: O autor (2025).

Avaliação do ResNet50 (sem augmentation):

Matriz de confusão:

	0	1	2	3
0	[67	0	30	4]
1	[61	0	19	10]
2	[61	0	22	7]
3	[58	0	24	8]

FIGURA 57 – AVALIAÇÃO DO RESNET50 (SEM ARGUMENTAÇÃO)

Report de classificação:

	precision	recall	f1-score	support
0	0.27	0.66	0.39	101
1	0.00	0.00	0.00	90
2	0.23	0.24	0.24	90
3	0.28	0.09	0.13	90
accuracy			0.26	371
macro avg	0.19	0.25	0.19	371
weighted avg	0.20	0.26	0.20	371

Acurácia: 26.15%

FONTE: O autor (2025).

Avaliação do ResNet50 (com augmentation):

	0	1	2	3
0	[64	0	28	9]
1	[63	0	21	6]
2	[59	0	24	7]
3	[61	0	22	7]

FIGURA 58 – AVALIAÇÃO DO RESNET50 (COM ARGUMENTAÇÃO)

Report de classificação:

	precision	recall	f1-score	support
0	0.26	0.63	0.37	101
1	0.00	0.00	0.00	90
2	0.25	0.27	0.26	90
3	0.24	0.08	0.12	90
accuracy			0.26	371
macro avg	0.19	0.24	0.19	371
weighted avg	0.19	0.26	0.19	371

Acurácia: 25.61%

FONTE: O autor (2025).

Conclusão:

Após a avaliação dos modelos VGG16 e ResNet50, ambos apresentaram desempenhos semelhantes, com uma acurácia de 26.15% para o VGG16 e 25.61% para o ResNet50 com Data Augmentation. No entanto, o VGG16 teve um desempenho superior, especialmente na classe 0, com uma sensibilidade de 66%, enquanto a ResNet50 teve dificuldades em classificar corretamente a classe 1, resultando em um recall de 0%.

Logo, considerando as métricas de precisão, recall e F1-score, o VGG16 se destaca como a melhor escolha devido à sua capacidade mais consistente de identificar as classes, especialmente a classe 0. Assim, o VGG16 é o modelo recomendado para a classificação das imagens.

APÊNDICE 11 – 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

1 INTRODUÇÃO

Com o avanço da inteligência artificial (IA), assistentes virtuais como o ChatGPT têm sido cada vez mais integrados em diferentes setores, incluindo o campo do aconselhamento psicológico online. Esta aplicação levanta questões profundas sobre ética, especialmente relacionadas à privacidade dos dados dos usuários, qualidade do aconselhamento oferecido, viés algorítmico e responsabilidade ética. Este estudo de caso explora essas implicações éticas específicas, oferecendo uma análise crítica do uso do ChatGPT em um contexto sensível como o suporte emocional e aconselhamento psicológico online.

2 PRIVACIDADE E CONFIDENCIALIDADE

A privacidade e a confidencialidade são preocupações centrais no uso de assistentes virtuais como o ChatGPT para aconselhamento psicológico. A natureza sensível das informações compartilhadas pelos usuários exige medidas rigorosas para proteger seus dados pessoais contra acesso não autorizado e violações de privacidade. Floridi (2020) discute que a proteção de dados é essencial para manter a confiança dos usuários e garantir o cumprimento de regulamentações de privacidade, como o GDPR.

Plataformas que implementam ChatGPT devem adotar políticas claras de privacidade e segurança de dados, incluindo criptografia robusta, armazenamento seguro e protocolos de acesso restrito. É fundamental que os usuários sejam informados de maneira transparente sobre como seus dados serão usados e protegidos ao interagir com o assistente virtual.

3 QUALIDADE DO ACONSELHAMENTO E RESPONSABILIDADE

Um aspecto crítico do uso do ChatGPT em aconselhamento psicológico é a avaliação da qualidade do serviço oferecido em comparação com o fornecido por profissionais humanos. Bostrom e Yudkowsky (2014) destacam a importância de avaliar a competência da IA em lidar com questões complexas e sensíveis, como as encontradas na psicologia clínica.

Embora o ChatGPT possa oferecer respostas rápidas e acessíveis, há limitações significativas em sua capacidade de compreender nuances emocionais, contexto individual e dinâmicas interativas que são essenciais para o aconselhamento eficaz. Isso levanta questões sobre a responsabilidade ética das plataformas que oferecem serviços de aconselhamento baseados em IA. Os desenvolvedores e os provedores de serviços devem estabelecer diretrizes claras para o uso responsável do ChatGPT em contextos terapêuticos, garantindo que o bem-estar dos usuários seja priorizado acima de considerações comerciais.

4 VIÉS ALGORÍTMICO E DISCRIMINAÇÃO

A questão do viés algorítmico é um desafio significativo em qualquer aplicação de IA, incluindo o aconselhamento psicológico. Mittelstadt et al. (2016) discutem como algoritmos de IA podem inadvertidamente perpetuar vieses culturais, raciais ou de gênero, impactando negativamente certos grupos demográficos.

No contexto do ChatGPT, é fundamental implementar técnicas avançadas de mitigação de viés algorítmico, como a diversificação dos conjuntos de dados de treinamento, a revisão humana de interações críticas e o monitoramento contínuo das respostas geradas pelo assistente virtual. Além disso, políticas de inclusão e diversidade devem orientar o desenvolvimento e a implementação de algoritmos para evitar discriminações injustas ou prejudiciais.

5 TOMADA DE DECISÃO ÉTICA

A tomada de decisão ética envolve determinar quando e como o ChatGPT pode ser utilizado de maneira ética no aconselhamento psicológico. Jobin et al. (2019) destacam a importância de diretrizes éticas robustas que orientem o uso responsável da IA em contextos sensíveis, como saúde mental.

É essencial que as plataformas que oferecem aconselhamento baseado em ChatGPT forneçam transparência aos usuários sobre os limites e as capacidades do assistente virtual. Isso inclui educar os usuários sobre a natureza da IA, seus propósitos e as expectativas realistas quanto ao tipo de suporte emocional que pode ser oferecido. Além disso, é necessário estabelecer procedimentos claros para encaminhar usuários para serviços profissionais de saúde mental sempre que necessário, garantindo uma abordagem integrada e ética ao cuidado psicológico.

6 PROPOSTA E SOLUÇÕES RESPONSÁVEIS

Para enfrentar esses desafios éticos, é fundamental implementar soluções responsáveis que promovam o uso ético do ChatGPT no aconselhamento psicológico online:

1. Políticas Claras de Privacidade e Segurança de Dados: Desenvolver e aplicar políticas robustas de privacidade que garantam a proteção adequada dos dados dos usuários.

2. Diretrizes Éticas Específicas: Estabelecer diretrizes éticas específicas para o uso de IA em aconselhamento psicológico, com ênfase na transparência, responsabilidade e respeito aos direitos dos usuários.

3. Mitigação de Viés Algorítmico: Implementar medidas eficazes para identificar e mitigar viéses algorítmicos, incluindo revisão humana e diversificação dos conjuntos de dados de treinamento.

4. Educação e Conscientização dos Usuários: Educar os usuários sobre as capacidades e limitações do ChatGPT, promovendo uma compreensão informada do uso de IA no suporte emocional.

5. Integração de Supervisão Humana: Integrar supervisão humana qualificada para monitorar e revisar interações críticas, garantindo uma abordagem ética ao aconselhamento psicológico.

7 CONSIDERAÇÕES FINAIS

Em resumo, o uso do ChatGPT em aconselhamento psicológico online apresenta benefícios potenciais significativos, como a expansão do acesso a serviços de suporte emocional. No entanto, também levanta desafios éticos complexos que exigem uma abordagem cuidadosa e responsável. Ao enfrentar questões de privacidade dos dados, qualidade do serviço, viés algorítmico e tomada de decisão ética, é possível desenvolver práticas que promovam o uso ético da IA no cuidado psicológico.

As propostas de soluções responsáveis destacadas neste estudo de caso são essenciais para orientar o desenvolvimento e a implementação de sistemas de IA que respeitem os princípios éticos fundamentais, protegendo o bem-estar dos usuários e promovendo uma sociedade digital mais justa e inclusiva.

REFERÊNCIAS

Floridi, L. (2020). "Soft Ethics, the Governance of the Digital and the General Data Protection Regulation: Developing a Data Ethics Framework." Philosophy & Technology, 33(2), 179-185.

Turilli, M., & Floridi, L. (2009). "The Ethics of Information Transparency." Ethics and Information Technology, 11(2), 105-112.

Bostrom, N., & Yudkowsky, E. (2014). "The Ethics of Artificial Intelligence." In E. Frankish & W. M. Ramsey (Eds.), The Cambridge Handbook of Artificial Intelligence (pp. 316-334). Cambridge University Press.

Jobin, A., Ienca, M., & Vayena, E. (2019). "The Global Landscape of AI Ethics Guidelines." Nature Machine Intelligence, 1(9), 389-399.

Bryson, J. J. (2018). "Patient data and artificial intelligence." Nature Biomedical Engineering, 2(5), 293-293.

Taddeo, M., & Floridi, L. (2018). "How AI Can Be a Force for Good." Science, 361(6404), 751-752.

Mittelstadt, B. D., et al. (2016). "The Ethics of Algorithms: Mapping the Debate." Big Data & Society, 3(2), 2053951716679679.

Jobin, A., et al. (2019). "Artificial Intelligence: The Ambiguity of Ethics and Intelligence." Nature, 568(7750), 626-628.

APÊNDICE 12 – 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

Qual o objetivo do estudo descrito pelo artigo?

O objetivo principal do estudo descrito no artigo é abordar uma lacuna significativa na pesquisa sobre as práticas recomendadas e os desafios envolvidos no design de arquiteturas para sistemas habilitados para aprendizado de máquina (ML). A pesquisa busca compreender, em profundidade, os principais obstáculos enfrentados pelos profissionais durante o desenvolvimento desses sistemas, as práticas de design mais eficazes e as decisões arquiteturais críticas que impactam diretamente a performance e a adaptabilidade dos sistemas de ML.

Qual o problema/oportunidade/situação que levou à necessidade de realização desse estudo?

O aumento expressivo do uso de soluções de aprendizado de máquina (ML) em diversos campos, como defesa cibernética, biologia computacional, robótica e veículos autônomos, tem gerado uma demanda crescente e complexa por sistemas de software projetados especificamente para suportar ML. Contudo, persiste uma lacuna significativa na compreensão de como os profissionais da área percebem e aplicam as decisões de design na arquitetura desses sistemas, bem como nos critérios que influenciam essas escolhas. Esse cenário torna-se ainda mais desafiador quando se considera a complexidade intrínseca do design arquitetônico de sistemas de ML, que exige o equilíbrio de múltiplas qualidades, como desempenho, escalabilidade, segurança e manutenibilidade, além da necessidade de integrar perfeitamente os componentes de ML com outros sistemas operacionais e de software convencionais.

Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?

A metodologia do estudo foi cuidadosamente planejada para assegurar uma análise rigorosa e abrangente dos dados, combinando várias abordagens metodológicas para reforçar a robustez do processo. O estudo foi conduzido em três fases principais: planejamento, condução e documentação, cada uma projetada para garantir a validade e a confiabilidade dos resultados.

Quais os principais resultados obtidos pelo estudo?

Os principais resultados do estudo revelam correlações detalhadas entre desafios, melhores práticas e decisões de design em seis áreas cruciais: arquitetura, dados, evolução, garantia de qualidade (QA), modelo e ciclo de vida de desenvolvimento de software (SDLC). Estes achados oferecem uma visão aprofundada sobre as especificidades e complexidades do desenvolvimento de sistemas habilitados para aprendizado de máquina (ML). Arquitetura: A adoção de padrões e estilos arquitetônicos, como a arquitetura de microsserviços, mostrou-se vantajosa ao promover a manutenibilidade e a flexibilidade dos sistemas de ML. Dados: Nessa categoria, o estudo identificou nove desafios específicos, relacionados a aspectos como o gerenciamento, a visualização e a privacidade dos dados. Problemas de qualidade e precisão dos dados, fundamentais para o desempenho dos modelos de ML, emergem como áreas em que ainda faltam diretrizes bem definidas e práticas de mitigação robustas. Evolução: O estudo destaca a importância de práticas de evolução contínua em sistemas ML, enfatizando a necessidade de atualizações regulares dos modelos e do pipeline de dados para manter a acurácia e a relevância do sistema ao longo do tempo. Garantia de Qualidade (QA): A complexidade dos sistemas ML exige uma abordagem de QA abrangente que aborde não apenas a funcionalidade, mas também a confiabilidade e a segurança dos modelos. Modelo: A seleção e otimização de modelos de ML é um desafio central identificado no estudo. A escolha de algoritmos e frameworks, como TensorFlow e PyTorch, deve ser guiada não apenas pelos requisitos do domínio, mas também pelas necessidades específicas de escalabilidade e eficiência do sistema.

APÊNDICE 13 – 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

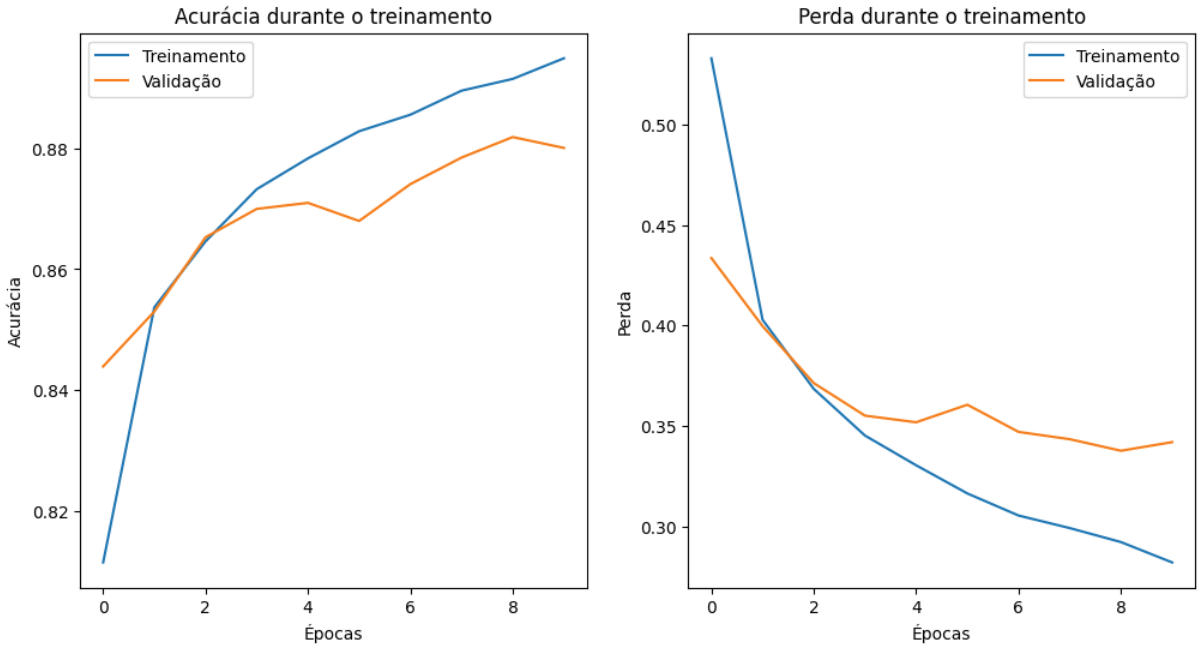
1- Classificação (RNA)

```
### 1 Classificação (RNA)

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

# Carregar a base de dados Fashion MNIST
data = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = data.load_data()

# Normalizar as imagens de 0-255 para 0-1
x_train, x_test = x_train / 255.0, x_test / 255.0

# Definir o modelo da rede neural
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),  # Flatten a imagem 28x28 para um vetor 1D
```

```
    tf.keras.layers.Dense(128, activation='relu'),  # Camada densa com 128 neurônios e ReLU
    tf.keras.layers.Dropout(0.2),  # Dropout para evitar overfitting
    tf.keras.layers.Dense(10, activation='softmax')   # Camada de saída com 10 classes (uma para cada categoria)
])

# Compilar o modelo
model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Treinar o modelo
history = model.fit(x_train, y_train, epochs=10, validation_data=(x_test, y_test))

# Avaliar o modelo no conjunto de teste
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('\nTest accuracy:', test_acc)

# Gráficos de perda e acurácia durante o treinamento
# Plotando a acurácia de treino e validação
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Treinamento')
plt.plot(history.history['val_accuracy'], label='Validação')
plt.title('Acurácia durante o treinamento')
plt.xlabel('Épocas')
plt.ylabel('Acurácia')
plt.legend()

# Plotando a perda de treino e validação
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Treinamento')
plt.plot(history.history['val_loss'], label='Validação')
plt.title('Perda durante o treinamento')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend()
```

```
plt.show()

# Mostrar algumas classificações erradas
predictions = model.predict(x_test)
incorrect_indices = np.where(np.argmax(predictions, axis=1) != y_test)[0]

# Exibir 5 classificações erradas
for i in range(5):
    index = incorrect_indices[i]
    plt.imshow(x_test[index], cmap=plt.cm.binary)
    plt.title(f"Predição:       {np.argmax(predictions[index])},       Verdadeiro:
{y_test[index]}")
    plt.show()
```

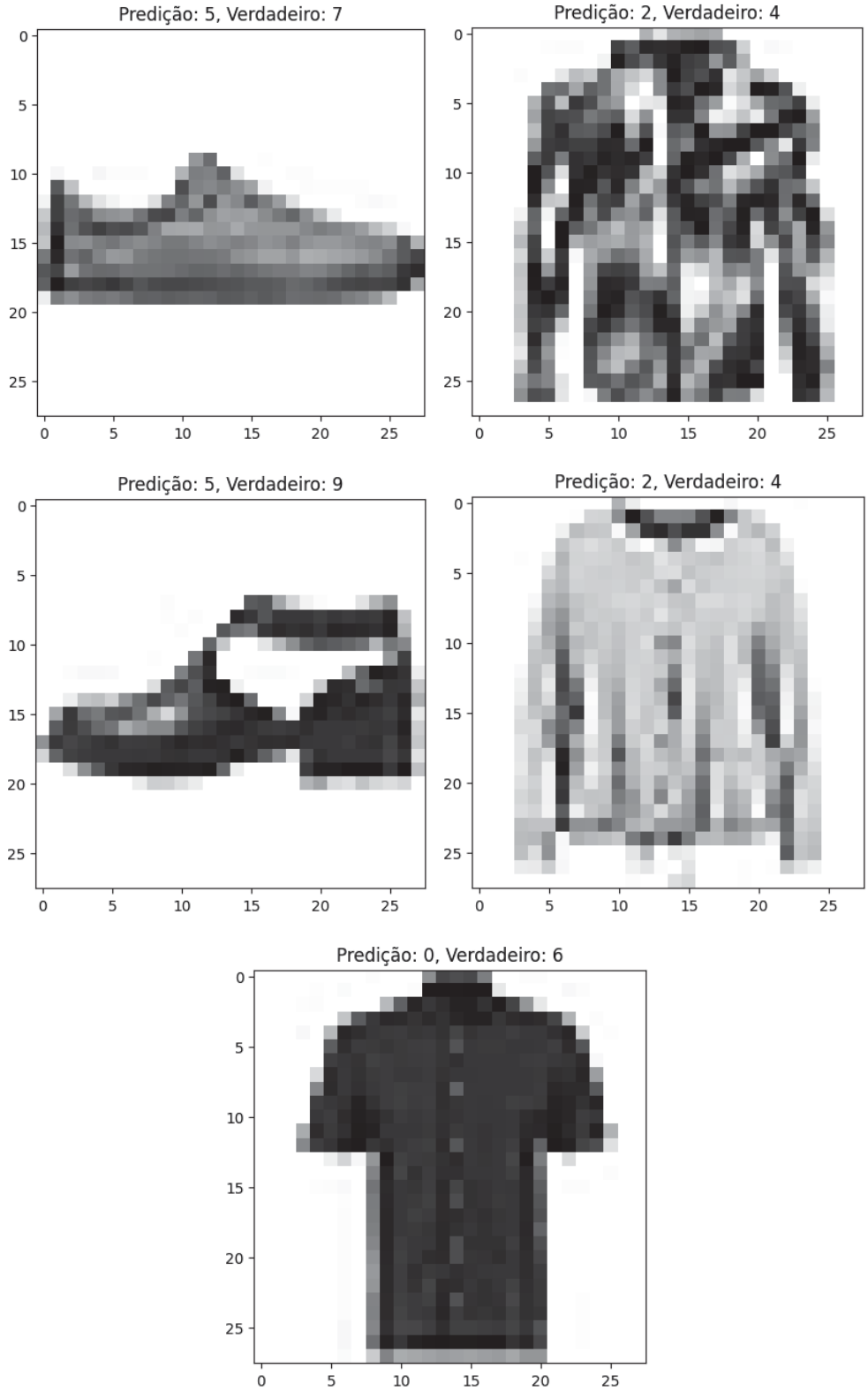
FIGURA 59 – ACURÁCIA E PERDA DURANTE O TREINAMENTO



FONTE: O autor (2025).

A seguir são exibidas 5 imagens com classificações erradas.

FIGURA 60 – IMAGENS ERRADAS



FONTE: O autor (2025).

Com base nos gráficos de acurácia e função de perda é possível verificar que o treinamento trouxe bons resultados para o modelo, sendo que a acurácia ficou em torno de 88% e a função de perda foi reduzida para um valor em torno de 0,34. Observando o gráfico da função de perda, é possível

ver que a queda no valor de perda dos dados de validação começa a reduzir, o que pode significar que, se o treino fosse realizado com mais épocas, possivelmente teríamos um cenário de overfitting.

Por fim, visto que o modelo, apesar de ter uma acurácia alta, ainda assim pode cometer erros, como é o caso das imagens que foram preditas erradas e estão sendo exibidas na útlima seção do caderno, no qual tem a classe que foi predita e a classe real da imagem.

2- Regressão (RNA)

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

#importação dos dados

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

data.head()
```

FIGURA 61 – DADOS DE REGRESSÃO (RNA)

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

FONTE: O autor (2025).

```
#mudando nome das colunas
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
]

data.head()
```

FIGURA 62 – ALTERAÇÃO DOS NOMES DAS COLUNAS

	acidez_fixa	acidez_volatil	acido_citrico	acucar_residual	cloretos	dioxido_de_enxofre_livre	dioxido_de_enxofre_total	densidade	pH	sulfatos	alcool	score_qualidade_vinho
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

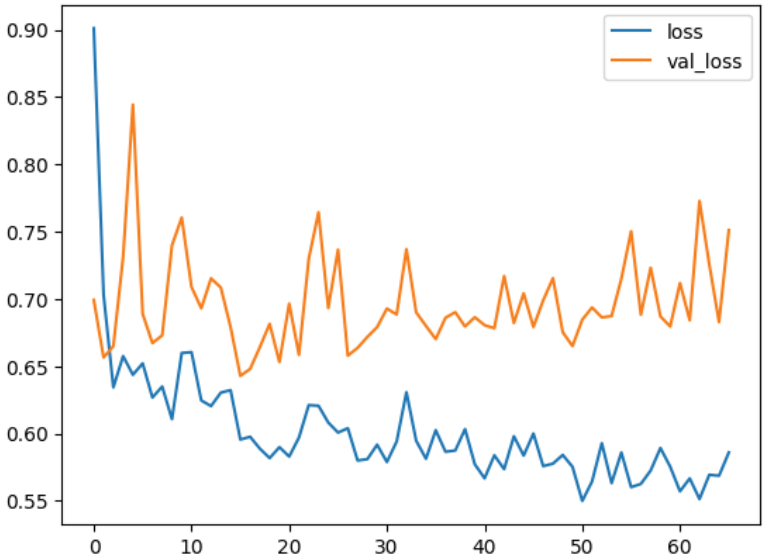
FONTE: O autor (2025).

```
print(data.shape)
(1599, 12)

#separa variáveis explicativas da variável resposta
x = data[['acidez_fixa',
'acidez_volatil',
'acido_citrico',
'acucar_residual',
'cloretos',
'dioxido_de_enxofre_livre',
'dioxido_de_enxofre_total',
'densidade',
'pH',
'sulfatos',
'alcool']].values.astype(float)

y = data['score_qualidade_vinho'].values.astype(float)

print(type(x))
print(type(y))

<class 'numpy.ndarray'>
<class 'numpy.ndarray'>
```

```
#verificando dados faltantes ou infinito

print(np.isnan(x).any(),np.isnan(y).any())
print(np.isinf(x).any(),np.isinf(y).any())

False False
False False

#normalização dos dados
from sklearn.preprocessing import StandardScaler

scaler_x = StandardScaler()
x = scaler_x.fit_transform(x)

scaler_y = StandardScaler()
y = scaler_y.fit_transform(y.reshape(-1, 1))  # Para regressão



#separando base de treino e teste

x_treino, x_teste, y_treino, y_teste = train_test_split(x, y, test_size=0.3,
random_state=308)

x_teste

array([[-0.29854743,  0.51495855, -1.13471997, ..., -0.13679827,
        -0.16611498, -1.05411336],
       [-1.04543701,  0.57082331,  0.30309297, ...,  0.3167512 ,
        -0.10710191, -0.30317536],
       [-1.27524919,  0.98980905, -0.87796766, ...,  1.87177795,
         0.01092425, -0.20930812],
       ...,
       [ 0.21852997, -0.65820153,  0.71389667, ..., -0.46076217,
         0.66006809,  2.04350586],
       [-1.6774205 , -0.60233677, -0.0050098 , ...,  3.03804801,
        -0.10710191,  1.76190411],
       [ 0.27598301,  0.12390519, -1.18607043, ..., -0.65514052,
        -0.34315421,  0.44776263]])
```

```
#criação do modelo
i = tf.keras.layers.Input(shape=(11,))
m = tf.keras.layers.Dense(70, activation='relu')(i)
m = tf.keras.layers.Dense(1)(m)

modelo_1 = tf.keras.models.Model(inputs=i, outputs=m)

from keras import backend

#funções para r2 e rmse
def rmse(y_true, y_pred):
    return
tf.keras.backend.sqrt(tf.keras.backend.mean(tf.keras.backend.square(y_pred        -
y_true)))

def r2(y_true, y_pred):
    media = tf.keras.backend.mean(y_true)
    ss_res = tf.keras.backend.sum(tf.keras.backend.square(y_true - y_pred))
    ss_tot = tf.keras.backend.sum(tf.keras.backend.square(y_true - media))
    return (1-ss_res/(ss_tot))

#ajuste do modelo

optimizer = tf.keras.optimizers.Adam(learning_rate=0.05)

modelo_1.compile(optimizer=optimizer, loss='mse', metrics=[rmse,r2])

#stops para epocas

early_stops  =  tf.keras.callbacks.EarlyStopping(monitor='val_loss',  patience=50,
restore_best_weights=True)

#treinamento do modelo
treino_modelo                                                                    =
modelo_1.fit(x_treino,y_treino,epochs=1000,validation_data=(x_teste,y_teste),callb
acks = [early_stops])
#avaliação do modelo
plt.plot(modelo_1.history.history['loss'], label='loss')
plt.plot(modelo_1.history.history['val_loss'], label='val_loss')
plt.legend()
```

FIGURA 63 – AVALIAÇÃO DO MODELO

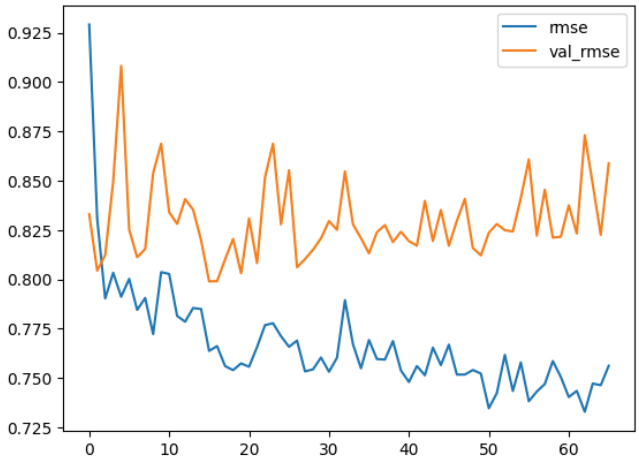


FONTE: O autor (2025).

O gráfico acima representa a evolução da função de perda conforme o número de épocas aumenta. Observamos queda na função de perda conforme aumenta o número de épocas, o que é esperado, indicando que o modelo está aprendendo a minimizar o erro.

```
#RMSE
PLT.PLOT(MODELO_1.HISTORY.HISTORY['RMSE'], LABEL='RMSE')
PLT.PLOT(MODELO_1.HISTORY.HISTORY['VAL_RMSE'], LABEL='VAL_RMSE')
PLT.LEGEND()
```

FIGURA 64 – AVALIAÇÃO DO MODELO RMSE

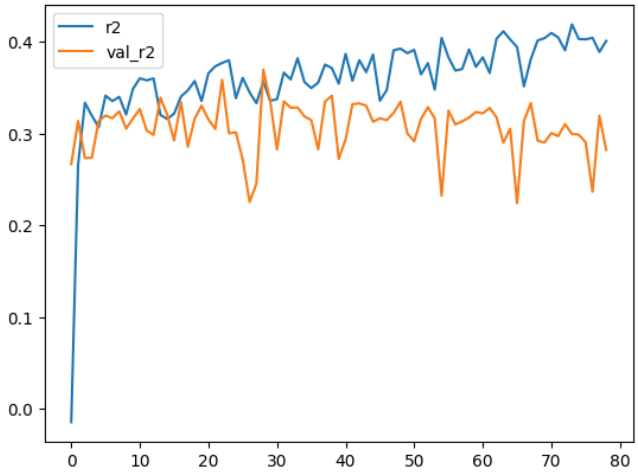


FONTE: O autor (2025).

O RMSE, também uma medida de erro, diminui a medida que a quantidade de épocas aumenta.

```
#plotando r2

plt.plot(modelo_1.history.history['r2'], label='r2')

plt.plot(modelo_1.history.history['val_r2'], label='val_r2')

plt.legend()
```

FIGURA 65 – AVALIAÇÃO DO MODELO R2



FONTE: O autor (2025).

O R2 é uma medida de acurácia do modelo, e quanto mais próximo de 1 melhor. Nas primeiras épocas ela é bem baixa e vai aumentando conforme a quantidade de épocas aumenta.

```
y_hat = modelo_1.predict(x_teste).flatten()

mse = mean_squared_error(y_teste,y_hat)

rmse = sqrt(mse)

r2 = r2_score(y_teste,y_hat)

print(f'MSE: {mse}')

print(f'RMSE: {rmse}')

print(f'R2: {r2}')

MSE: 0.6280485759170473
```

```
RMSE: 0.7924951582924954

R2: 0.3915046095974757
```

Utilizando os dados de teste, observamos um R2, técnica de acurácia, de 39%, próximo aos valores analisados no gráfico de R2 versus épocas para os valores preditos no ajuste do modelo. Tentamos ajustar um modelo com métricas de desempenho melhores, através do aumento de neurônios, mudança da função de ativação para linear, mudança no parâmetro de patience na técnica de early stopping, no entanto não obtivemos melhores resultados.

3- Sistema de recomendação

```
csv_path = '/content/base-livros.csv'
K = 25
epochs = 50
batch_size = 1024

# 1. Importação das bibliotecas
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding, Flatten, Concatenate
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD

from sklearn.utils import shuffle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# 2.1 Carregamento dos dados no dataframe
df = pd.read_csv(csv_path)

# 2.2 Visualização básica dos dados
print(df.dtypes)
print('-----')
print('Menor nota: ', df.Notas.min())
print('Maior nota: ', df.Notas.max())
print('-----')
print('Shape: ', df.shape)
df.head()
```

FIGURA 66 – VISUALIZAÇÃO BÁSICA DOS DADOS

```
ISBN          object
Titulo        object
Autor         object
Ano            int64
Editora       object
ID_usuario     int64
Notas          int64
dtype: object
-----
Menor nota:  0
Maior nota:  10
-----
Shape:  (128895, 7)
```

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas
0	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0
1	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2
2	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6
3	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1
4	399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	276729	9

FONTE: O autor (2025).

```
# 3.1 Conversão de tipos de valores para embeddings
# Converter o ISBN e o ID_usuario para valores categóricos (Embeddings)

df.ISBN = pd.Categorical(df.ISBN)
df['isbn_cat_codes'] = df.ISBN.cat.codes


df.ID_usuario = pd.Categorical(df.ID_usuario)
df['id_usuario_cat_codes'] = df.ID_usuario.cat.codes

df.Notas = df.Notas.astype(np.float32)

print(df.dtypes)
df.head()
```

FIGURA 67 – VALORES CATEGÓRICOS

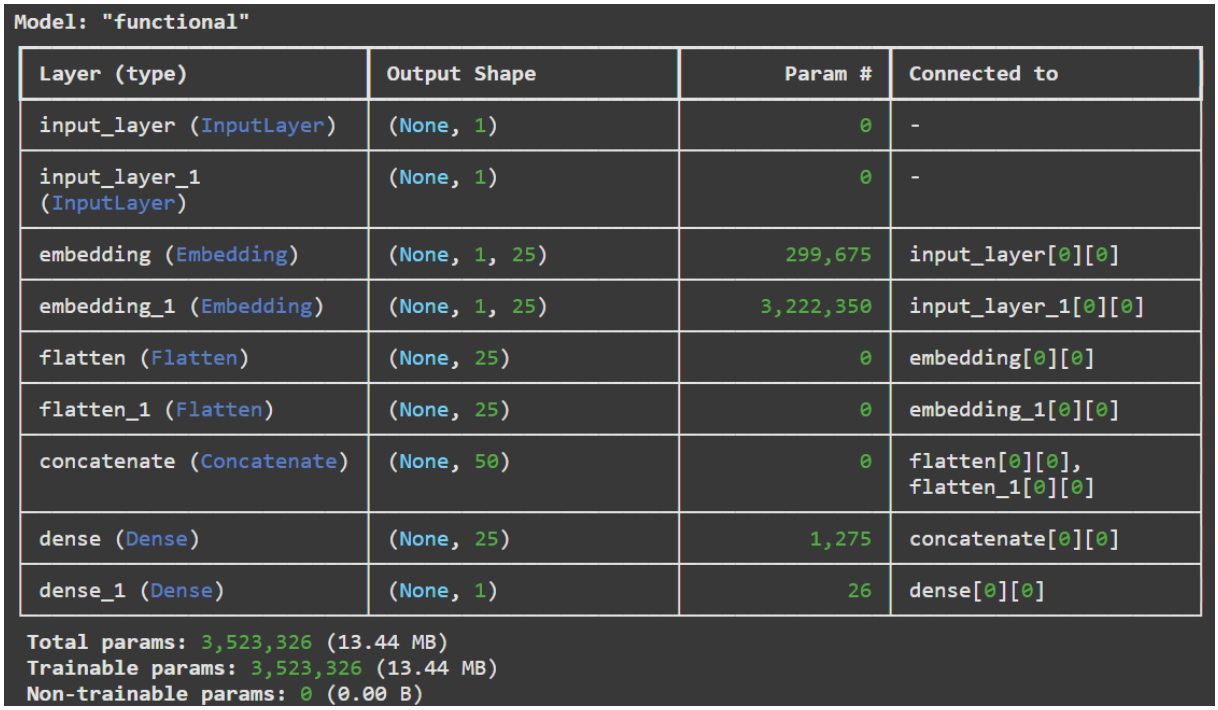
```
ISBN                category
Titulo                object
Autor                 object
Ano                    int64
Editora               object
ID_usuario          category
Notas                float32
isbn_cat_codes         int32
id_usuario_cat_codes   int16
dtype: object
```

	ISBN	Titulo	Autor	Ano	Editora	ID_usuario	Notas	isbn_cat_codes	id_usuario_cat_codes
0	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0.0	26377	11137
1	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2.0	87069	11138
2	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6.0	50383	11139
3	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1.0	56644	11140
4	399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	276729	9.0	59080	11140

FONTE: O autor (2025).

```
# 3.2 Conversão de dimensões
# Obter tamanho das listas de ISBN e ID_usuario únicos
N = len(set(df.id_usuario_cat_codes))
M = len(set(df.isbn_cat_codes))

print(f"Número de usuários únicos: {N}")
print(f"Número de livros únicos: {M}")

Número de usuários únicos: 11987
Número de livros únicos: 128894

# 4.1 Criação de camadas referentes ao usuario
u = Input(shape=(1, ))
u_emb = Embedding(N, K)(u)
u_emb = Flatten()(u_emb)

# 4.2 Criação das camadas referentes ao ISBN
i = Input(shape=(1, ))
i_emb = Embedding(M, K)(i)
i_emb = Flatten()(i_emb)

# Junção dos conjuntos de camadas
x = Concatenate()([u_emb, i_emb])
x = Dense(K, activation='relu')(x)
x = Dense(1)(x)

model = Model(inputs=[u, i], outputs=x)

# 5.1 Compilar o modelo
model.compile(
    loss='mse',
    optimizer=SGD(learning_rate=0.07, momentum=0.5)
)

# 5.2 Sumário do modelo
model.summary()
```

FIGURA 68 – SUMÁRIO DO MODELO



Model: "functional"

Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1)	0	-
input_layer_1 (InputLayer)	(None, 1)	0	-
embedding (Embedding)	(None, 1, 25)	299,675	input_layer[0][0]
embedding_1 (Embedding)	(None, 1, 25)	3,222,350	input_layer_1[0][0]
flatten (Flatten)	(None, 25)	0	embedding[0][0]
flatten_1 (Flatten)	(None, 25)	0	embedding_1[0][0]
concatenate (Concatenate)	(None, 50)	0	flatten[0][0], flatten_1[0][0]
dense (Dense)	(None, 25)	1,275	concatenate[0][0]
dense_1 (Dense)	(None, 1)	26	dense[0][0]

Total params: 3,523,326 (13.44 MB)
Trainable params: 3,523,326 (13.44 MB)
Non-trainable params: 0 (0.00 B)

FONTE: O autor (2025).

```
# 6.1 Separar os dados em treino e teste
id_usuarios, isbns, notas = shuffle(df.id_usuario_cat_codes, df.isbn_cat_codes,
df.Notas)
print("Usuarios: ", len(id_usuarios), " - ", id_usuarios[:10])
print("ISBNs: ", len(isbns), " - ", isbns[:10])
print("Notas: ", len(notas), " - ", notas[:10])

n_train = int(0.8 * len(notas))

train_usuarios = id_usuarios[:n_train]
train_isbns = isbns[:n_train]
train_notas = notas[:n_train]

test_usuarios = id_usuarios[n_train:]
test_isbns = isbns[n_train:]
test_notas = notas[n_train:]
```

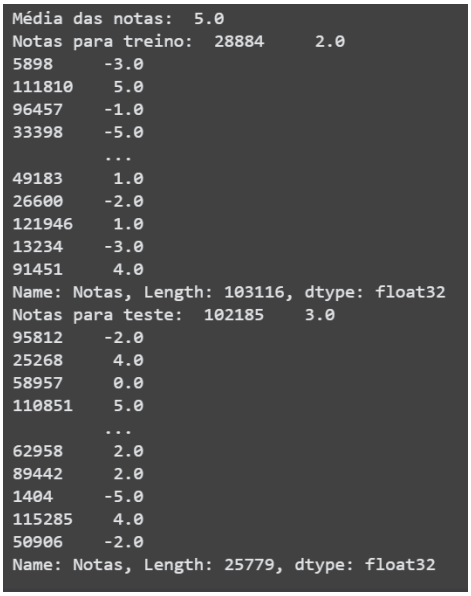
FIGURA 69 – SEPARAÇÃO DOS DADOS DE TREINO E TESTE

```
Usuarios:  128895  -  28884      2384
5898      11815
111810     9597
96457      7954
33398      2888
113646     9787
9701         54
113159     9698
112661     9638
26967      2303
Name: id_usuario_cat_codes, dtype: int16
ISBNs:  128895  -  28884       1266
5898       83465
111810     12108
96457      71499
33398     117161
113646     62764
9701       24647
113159     99485
112661    122670
26967      78506
Name: isbn_cat_codes, dtype: int32
Notas:  128895  -  28884      7.0
5898       2.0
111810    10.0
96457      4.0
33398      0.0
113646     0.0
9701       9.0
113159     1.0
112661     2.0
26967      3.0
Name: Notas, dtype: float32
```

FONTE: O autor (2025).

```
# 6.2 Pré-processamento dos dados de treino e teste
# Centralização das notas com base na média
avg_notas = round(notas.mean(), 1)
train_notas = train_notas - avg_notas
test_notas = test_notas - avg_notas

print("Média das notas: ", avg_notas)
print("Notas para treino: ", train_notas)
print("Notas para teste: ", test_notas)
```

FIGURA 70 – MÉDIA DAS NOTAS DE TREINO E TESTE

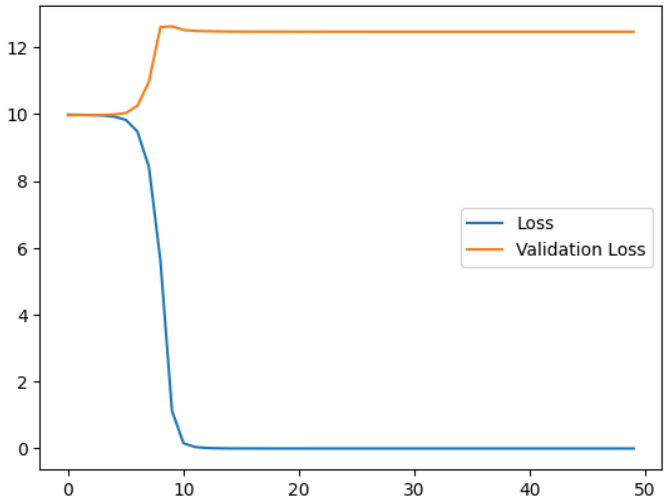


```
Média das notas:  5.0
Notas para treino:  28884     2.0
5898     -3.0
111810    5.0
96457    -1.0
33398    -5.0
         ...
49183     1.0
26600    -2.0
121946    1.0
13234    -3.0
91451     4.0
Name: Notas, Length: 103116, dtype: float32
Notas para teste:  102185    3.0
95812    -2.0
25268     4.0
58957     0.0
110851    5.0
         ...
62958     2.0
89442     2.0
1404     -5.0
115285    4.0
50906    -2.0
Name: Notas, Length: 25779, dtype: float32
```

FONTE: O autor (2025).

```
# 7. Treinar o modelo
x_train = [train_usuarios, train_isbns]
y_train = train_notas

x_test = [test_usuarios, test_isbns]
y_test = test_notas

r = model.fit(
    x=x_train,
    y=y_train,
    epochs=epochs,
    batch_size=batch_size,
    verbose=2,
    validation_data=(x_test, y_test)
)
# 8. Plotar a função de perda
plt.plot(r.history['loss'], label='Loss')
plt.plot(r.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

FIGURA 71 – GRÁFICO DA FUNÇÃO DE PERDA



FONTE: O autor (2025).

```
# 9.1 Gerar um array com usuário único
input_usuario = np.repeat(a=203, repeats=M)
books = np.array(list(set(isbns)))

print("input_usuario: ", input_usuario)
print("books: ", books)
print("len input_user: ", len(input_usuario))
print("len books: ", len(books))

input_usuario:  [203 203 203 ... 203 203 203]
books:  [     0      1      2 ... 128891 128892 128893]
len input_user:  128894
len books:  128894

# 9.3 Tratamento da predição
notas_finais = preds.flatten() + avg_notas

max_idx = np.argmax(notas_finais)
result = df[df.isbn_cat_codes == books[max_idx]]
print(f"Recomendação: {result.Titulo.values[0]} por {result.Autor.values[0]}. Nota:
{round(notas_finais[max_idx], 1)} ")

Recomendação: The Outside Dog (I Can Read Book 3) por Charlotte Pomerantz. Nota:
10.300000190734863
```

Conclusões:

Com base no gráfico de perda aparentemente pode estar ocorrendo um overfitting (por volta da epoch 10 em que a função de perda dos dados de treino começa a cair ao passo que a função de perda dos dados de validação começam a subir). Foram realizados testes alterando diversos parâmtros, como o tamanho do embedding, batch size, learning rate, momentum, função de ativação e atributo utilizado para o treino (título do livro ao invés de ISBN). Em geral o mesmo comportamento foi observado, sendo que em alguns casos acabava levando mais *epochs* para se notar uma redução significativa na função de perda.

4- Deepdream

```
## 1. Importação das bibliotecas

import tensorflow as tf
import numpy as np
import matplotlib as mpl
import IPython.display as display
import PIL.Image

## 2. Importação da imagem

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

# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
  name = url.split('/')[-1]
  image_path = tf.keras.utils.get_file(name, origin=url)
  img = PIL.Image.open(image_path)
  if max_dim:
    img.thumbnail((max_dim, max_dim))
  return np.array(img)

# Normalização da imagem
def deprocess(img):
  img = 255*(img + 1.0)/2.0
  return tf.cast(img, tf.uint8)

# Mostra a imagem
def show(img):
  display.display(PIL.Image.fromarray(np.array(img)))
```

```
# Redução do tamanho da imagem para facilitar o trabalho da RNN
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('Image                    cc-by:                    <a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-
cat_on_snow.jpg">Von.grzanka</a>'))
```

FIGURA 72 – DEEPDREAM

FONTE: O autor (2025).

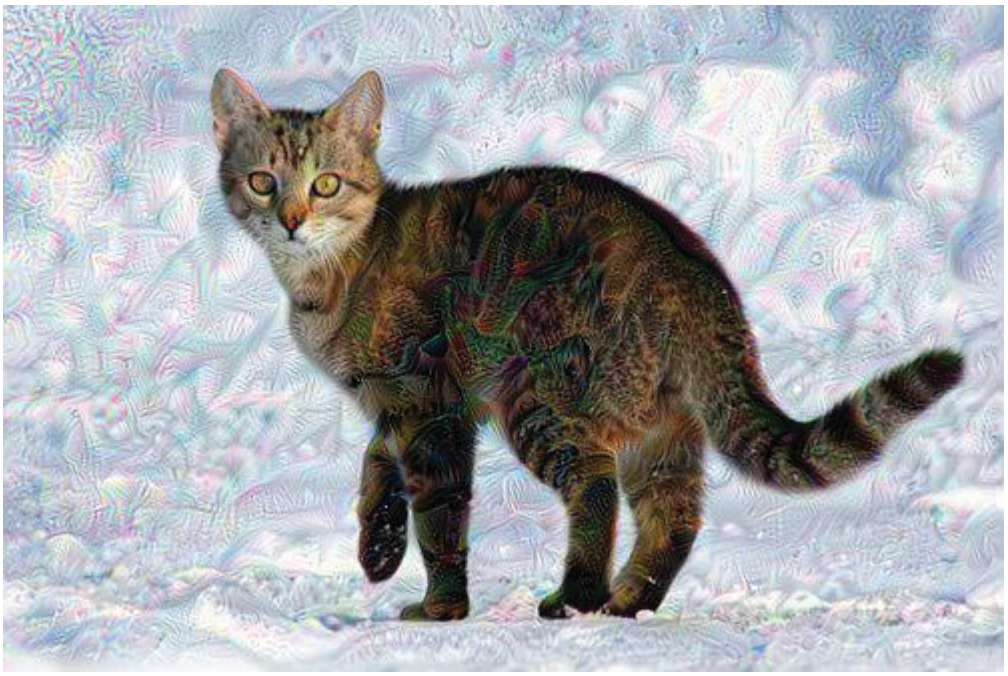
```
## 3. Preparar o modelo de extração de recursos
base_model          =          tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')

# Maximizando as ativações das camadas
names = ['mixed6', 'mixed8']
layers = [base_model.get_layer(name).output for name in names]

# Criação do modelo
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)

## 4. Cálculo da perda (loss)
```

```
def calc_loss(img, model):
  # Passe a imagem pelo modelo para recuperar as ativações.
  # Converte a imagem em um batch de tamanho 1.
  img_batch = tf.expand_dims(img, axis=0)
  layer_activations = model(img_batch)
  if len(layer_activations) == 1:
    layer_activations = [layer_activations]

  losses = []
  for act in layer_activations:
    loss = tf.math.reduce_mean(act)
    losses.append(loss)

  return  tf.reduce_sum(losses)
## 5. Subida de gradiente (Gradient ascent)

class DeepDream(tf.Module):
  def __init__(self, model):
    self.model = model

  @tf.function(
      input_signature=(
        tf.TensorSpec(shape=[None,None,3], dtype=tf.float32),
        tf.TensorSpec(shape=[], dtype=tf.int32),
        tf.TensorSpec(shape=[], dtype=tf.float32),)
  )
  def __call__(self, img, steps, step_size):
      print("Tracing")
      loss = tf.constant(0.0)

      for n in tf.range(steps):
        with tf.GradientTape() as tape:
          # Gradientes relativos a img
          tape.watch(img)
          loss = calc_loss(img, self.model)

        # Calculo do gradiente da perda em relação aos pixels da imagem de entrada.
        gradients = tape.gradient(loss, img)

        # Normalizacao dos gradintes
```

```
        gradients /= tf.math.reduce_std(gradients) + 1e-8

        # Na subida gradiente, a "perda" é maximizada.
        # Você pode atualizar a imagem adicionando diretamente os gradientes (porque
eles têm o mesmo formato!)
        img = img + gradients*step_size
        img = tf.clip_by_value(img, -1, 1)

      return loss, img

deepdream = DeepDream(dream_model)

## 6. Circuito princial (Main Loop)
def run_deep_dream_simple(img, steps=100, step_size=0.01):

  img = tf.keras.applications.inception_v3.preprocess_input(img)
  img = tf.convert_to_tensor(img)
  step_size = tf.convert_to_tensor(step_size)
  steps_remaining = steps
  step = 0
  while steps_remaining:
    if steps_remaining>100:
      run_steps = tf.constant(100)
    else:
      run_steps = tf.constant(steps_remaining)
    steps_remaining -= run_steps
    step += run_steps

    loss, img = deepdream(img, run_steps, tf.constant(step_size))

    display.clear_output(wait=True)
    show(deprocess(img))
    print ("Step {}, loss {}".format(step, loss))


  result = deprocess(img)
  display.clear_output(wait=True)
  show(result)

  return result
```

```
dream_img = run_deep_dream_simple(img=original_img,
                                  steps=100, step_size=0.01)
```

FIGURA 73 – RESULTADO DO DEEPDREAM



FONTE: O autor (2025).

```
## 7. Levando o modelo até um oitava
import time
start = time.time()

OCTAVE_SCALE = 1.30

img = tf.constant(np.array(original_img))
base_shape = tf.shape(img)[:-1]
float_base_shape = tf.cast(base_shape, tf.float32)

for n in range(-2, 3):
  new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32)

  img = tf.image.resize(img, new_shape).numpy()

  img = run_deep_dream_simple(img=img, steps=50, step_size=0.01)

display.clear_output(wait=True)
```

```
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```

FIGURA 74 – RESULTADO DO DEEPDREAM ATÉ UM OITAVA

FONTE: O autor (2025).

Imagem onírica obtida por Main Loop;

Após o processamento pelo Main Loop com camadas Mixed6 e Mixed8, que são partes da rede neural Inception (usada no treinamento de visão computacional), padrões visuais abstratos e psicodélicos surgem sobre a imagem. Esses padrões geralmente lembram estruturas orgânicas como olhos, espirais ou texturas semelhantes a folhas e animais.

A técnica funciona "exagerando" características que a rede neural detecta, criando esse efeito de sonho surrealista, como se a máquina estivesse projetando sua própria interpretação da imagem.

Imagem onírica obtida ao levar o modelo até uma oitava;

Após o processamento com a técnica de oitavas, como resultado a imagem original do felino em um cenário de neve foi transformada em uma versão onírica com padrões ainda mais visíveis e elaborados. Nessa versão, há a impressão de múltiplas texturas e formas, como olhos e detalhes geométricos, espalhados de maneira fractal sobre a pelagem do animal e o ambiente ao redor.

Esse efeito mais refinado e detalhado é característico do uso das oitavas, pois ele permite que a rede neural detecte e realce padrões tanto em níveis macro (grandes formas) quanto micro (detalhes finos), gerando uma aparência mais complexa e psicodélica.

Diferenças entre imagens oníricas obtidas com Main Loop e levando o modelo até a oitava.

Main Loop: Foca no processamento direto da imagem em uma única etapa ou em camadas específicas da rede neural (ex. Mixed6, Mixed8). Os padrões visuais oníricos surgem de forma mais sutil e menos detalhada. As formas, como olhos, espirais ou texturas, aparecem mais uniformemente distribuídas pela imagem, mas com menos refinamento em pequenas escalas.

Levando o modelo até a oitava: A imagem é processada em múltiplas resoluções (oitavas), começando em baixa resolução e refinando progressivamente até atingir a imagem completa. O resultado é mais detalhado e complexo, com padrões em múltiplas escalas (macro e micro), criando uma aparência mais fractal e elaborada. Formas como olhos e texturas são mais visíveis, sobrepostas e densamente distribuídas, dando um aspecto mais psicodélico.

Resumo: A técnica de oitavas gera imagens mais detalhadas e refinadas ao realçar padrões em várias escalas, enquanto o Main Loop tende a produzir um efeito mais sutil e homogêneo.

APÊNDICE 14 – 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 publico-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**)

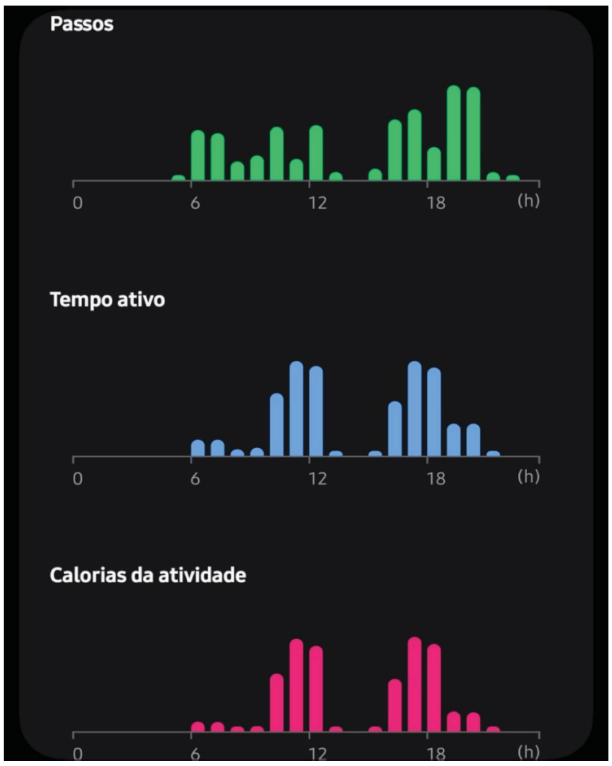
B – RESOLUÇÃO

.

O conjunto de dados brutos:

Visualização de dados a ser melhorada, gráfico apresentado no aplicativo de monitoramento de atividades físicas Samsung Health.

FIGURA 75 – GRÁFICO ATUAL MONITORAMENTO SAMSUNG HEALTH

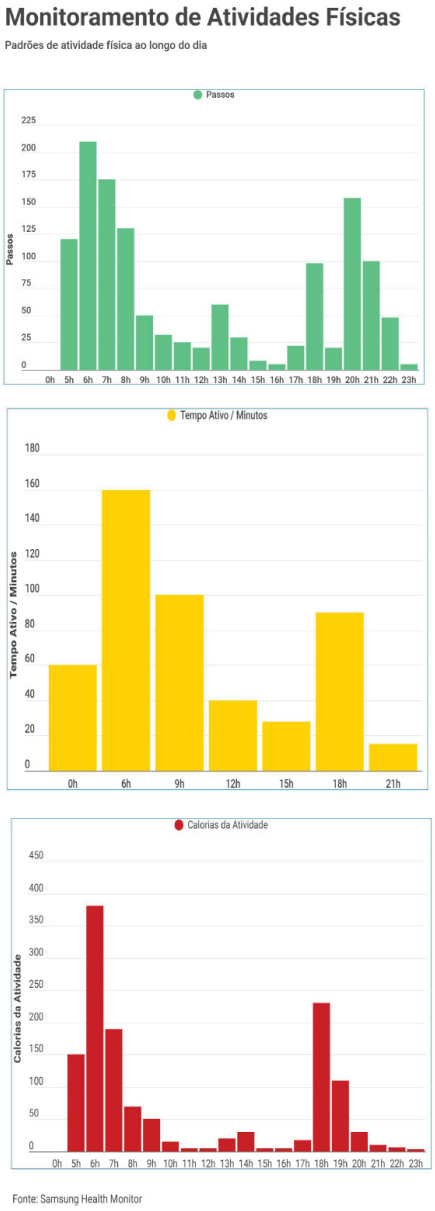


FONTE: O autor (2025).

Explicação do contexto:

Esta visualização é voltada para indivíduos interessados em saúde e bem-estar, pessoas que querem compreender melhor seus padrões diários e adotar mudanças positivas. Também é útil para treinadores, profissionais de saúde e pesquisadores em busca de dados para auxiliar em programas de saúde personalizados. A visualização de dados apresentada é uma ferramenta poderosa para interpretar padrões de atividade física ao longo de um dia típico. Com três gráficos de barras ilustrando os "Passos", o "Tempo ativo" e as "Calorias da atividade", essa narrativa oferece insights sobre comportamento, saúde e potencial de melhoria. De forma geral, essa narrativa destaca como pequenas mudanças baseadas na análise de dados podem levar a grandes transformações no estilo de vida e na saúde geral.

FIGURA 76 – NOVO GRÁFICO DE MONITORAMENTO PROPOSTO



FONTE: O autor (2025).

Escolha sobre o tipo de visualização:

A escolha pelo gráfico de barras se dá pelo fato da visualização ser clara, direta e intuitiva, sendo considerado uma das melhores opções para interpretar dados. Abaixo vou elencar alguns pontos que corroboram com essa afirmação:

Comparação simples: Os gráficos de barras permitem uma fácil comparação entre diferentes categorias ou conjuntos de dados. As alturas ou comprimentos das barras oferecem uma representação visual que é rapidamente compreendida.

Versatilidade: Eles são adequados tanto para dados quantitativos quanto qualitativos, tornando-os uma escolha flexível para uma ampla gama de cenários.

Leitura rápida: Nosso cérebro processa tamanhos e comprimentos com facilidade, permitindo que os dados sejam interpretados rapidamente sem a necessidade de cálculos adicionais.

Detectar tendências: Com gráficos de barras, é fácil identificar tendências, como a categoria com maior ou menor valor.

Clareza visual: Em comparação com outros tipos de gráficos, como os de pizza, que podem se tornar confusos com muitas categorias, os gráficos de barras mantêm sua clareza mesmo com mais dados.

Escolha sobre a ferramenta utilizada:

O Infogram é amplamente reconhecido como uma das melhores ferramentas para desenvolvimento de gráficos devido à sua combinação de simplicidade, versatilidade e recursos avançados. Ele combina uma interface intuitiva onde é projetado para ser fácil de usar, mesmo para quem não tem experiência em design ou análise de dados. Variedade de opções com uma ampla gama de tipos de gráficos, como barras, linhas, pizza, mapas de calor e nuvens de palavras. Interatividade onde encontramos a capacidade de criar gráficos interativos, que engajam o público e tornam a apresentação de dados mais dinâmica. Outros pontos fortes da ferramenta também combinam integração de dados, permitindo importar dados de várias fontes, como planilhas, bancos de dados e até mesmo atualizações em tempo real, facilitando a criação de visualizações precisas e atualizadas. Personalização, os usuários podem ajustar cores, fontes, rótulos e outros elementos visuais para alinhar os gráficos à identidade visual de sua marca ou projeto e o compartilhamento onde os gráficos criados podem ser incorporados em sites, compartilhados em redes sociais ou exportados em formatos como imagens ou vídeos.

Descrição da narrativa/storytelling dessa visualização de dados:

Objetivo da visualização:

O propósito central é entender a rotina física diária de uma pessoa, identificando momentos de maior e menor atividade. Isso serve como base para decisões informadas que promovam um estilo de vida mais saudável e equilibrado.

O que esses dados significam?

1. Passos: Este gráfico evidencia os períodos de maior locomoção. Os picos observados podem sugerir deslocamentos como caminhar para o trabalho, fazer exercícios ou realizar tarefas do dia a dia.

2. Tempo ativo / Minutos: Ele complementa o gráfico de passos ao demonstrar não apenas a movimentação, mas a duração dela. Este dado ressalta momentos de maior continuidade na atividade física.

3. Calorias da atividade: Aqui, o impacto do esforço físico é traduzido em energia consumida, mostrando o resultado direto dos movimentos do dia.

Possíveis ações com base nos dados:

 Planejamento otimizado de exercícios: Identificar os horários em que a pessoa já é mais ativa pode ajudar a criar rotinas de treino eficazes nesses períodos.

 Redução da inatividade: Se houver momentos de baixa atividade, isso pode ser um gatilho para adicionar pausas para alongamento ou pequenas caminhadas.

 Monitoramento de progresso: Com comparações ao longo do tempo, é possível avaliar melhorias e redirecionar metas de atividade física.

APÊNDICE 15 – 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

1- Algoritmo Genético

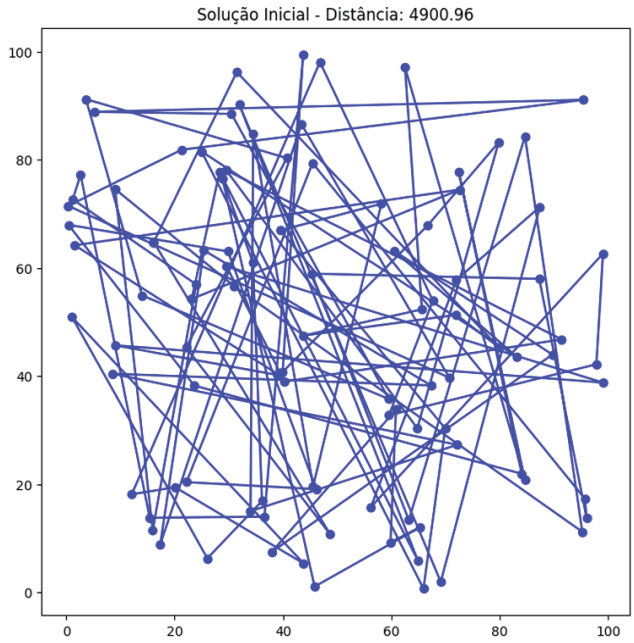
```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations
from sklearn.decomposition import PCA

# Definição dos parâmetros do problema
NUM_CIDADES = 100
ESPACO_LIMITE = 100
POPULACAO_SIZE = 100
GERACOES = 1000
MUTACAO_RATE = 0.01
CROSSOVER_RATE = 0.9

# Gerar coordenadas aleatórias para as cidades
cidades = np.random.rand(NUM_CIDADES, 2) * ESPACO_LIMITE

# Função de cálculo da distância euclidiana
def calcular_distancia(percurso):
    distancia = 0
    for i in range(len(percurso) - 1):
        distancia += np.linalg.norm(cidades[percurso[i]] - cidades[percurso[i + 1]])
    distancia += np.linalg.norm(cidades[percurso[-1]] - cidades[percurso[0]])  # Retorno à cidade inicial
    return distancia

# Inicializar população aleatória
```

```
def inicializar_populacao():
    populacao = []
    for _ in range(POPULACAO_SIZE):
        percurso = list(range(NUM_CIDADES))
        random.shuffle(percurso)
        populacao.append(percurso)
    return populacao

# Função de seleção por torneio
def selecao(populacao):
    candidatos = random.sample(populacao, 5)
    return min(candidatos, key=calcular_distancia)

# Operador de crossover OX (Order Crossover)
def crossover(pai1, pai2):
    tamanho = len(pai1)
    inicio, fim = sorted(random.sample(range(tamanho), 2))
    filho = [-1] * tamanho
    filho[inicio:fim] = pai1[inicio:fim]
    ptr = fim
    for gene in pai2:
        if gene not in filho:
            if ptr >= tamanho:
                ptr = 0
            filho[ptr] = gene
            ptr += 1
    return filho

# Operador de mutação (swap entre duas cidades)
def mutacao(percurso):
    if random.random() < MUTACAO_RATE:
        i, j = random.sample(range(len(percurso)), 2)
        percurso[i], percurso[j] = percurso[j], percurso[i]
    return percurso

# Algoritmo Genético
def algoritmo_genetico():
    populacao = inicializar_populacao()
    melhor_percurso = min(populacao, key=calcular_distancia)
    melhor_distancia = calcular_distancia(melhor_percurso)
```

```
    for _ in range(GERACOES):
        nova_populacao = []
        for _ in range(int(POPULACAO_SIZE * CROSSOVER_RATE)):
            pai1, pai2 = selecao(populacao), selecao(populacao)
            filho = crossover(pai1, pai2)
            filho = mutacao(filho)
            nova_populacao.append(filho)
        while len(nova_populacao) < POPULACAO_SIZE:
            nova_populacao.append(selecao(populacao))
        populacao = nova_populacao
        melhor_atual = min(populacao, key=calcular_distancia)
        melhor_atual_distancia = calcular_distancia(melhor_atual)
        if melhor_atual_distancia < melhor_distancia:
            melhor_percurso,        melhor_distancia        =        melhor_atual,
melhor_atual_distancia

    return melhor_percurso, melhor_distancia

# Função para plotar um percurso
def plotar_percurso(percurso, titulo):
    plt.figure(figsize=(8, 8))
    caminho = cidades[percurso + [percurso[0]]]
    plt.plot(caminho[:, 0], caminho[:, 1], 'bo-')
    plt.title(titulo)
    plt.show()

# Primeira solução aleatória
populacao = inicializar_populacao()
solucao_inicial = populacao[0]
distancia_inicial = calcular_distancia(solucao_inicial)
plotar_percurso(solucao_inicial,      f'Solução      Inicial      -      Distância:
{distancia_inicial:.2f}')

# Melhor solução após evolução
melhor_percurso, melhor_distancia = algoritmo_genetico()
plotar_percurso(melhor_percurso,      f'Melhor      Solução      -      Distância:
{melhor_distancia:.2f}')

# Aplicação da PCA em modelos vetoriais de um texto
```

```
def aplicar_pca(modelo1, modelo2):
    dados = np.vstack((modelo1, modelo2))
    pca = PCA(n_components=2)
    resultado_pca = pca.fit_transform(dados)
    plt.figure(figsize=(8, 6))
    plt.scatter(resultado_pca[:len(modelo1), 0], resultado_pca[:len(modelo1), 1],
label='Modelo 1', alpha=0.7)
    plt.scatter(resultado_pca[len(modelo1):, 0], resultado_pca[len(modelo1):, 1],
label='Modelo 2', alpha=0.7)
    plt.legend()
    plt.title("Visualização com PCA")
    plt.show()

# Exemplo de uso
modelo1 = np.random.rand(50, 300)  # Exemplo de embeddings de palavras
modelo2 = np.random.rand(50, 300)
aplicar_pca(modelo1, modelo2)
```

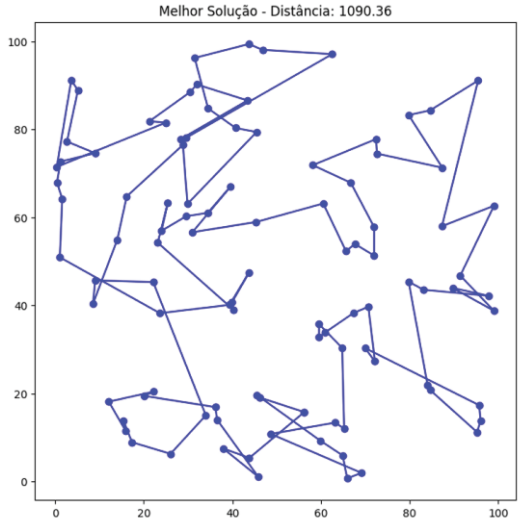
FIGURA 77 – SOLUÇÃO INICIAL



FONTE: O autor (2025).

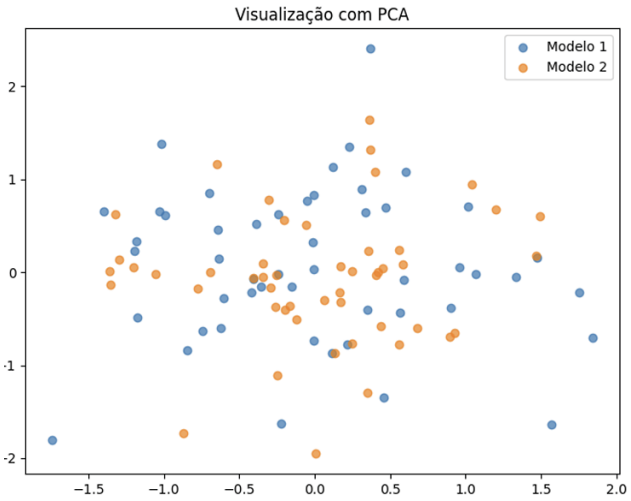
A seguir temos a melhor solução.

FIGURA 78 – MELHOR SOLUÇÃO



FONTE: O autor (2025).

FIGURA 79 – VISUALIZAÇÃO COM PCA



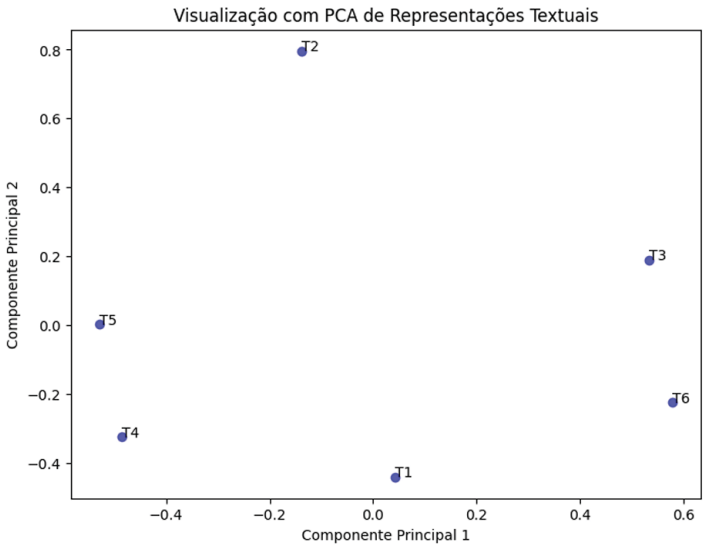
FONTE: O autor (2025).

2- Compare a representação de dois modelos vetoriais

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer

# Aplicação da PCA em representações vetoriais de textos
textos = [
    "O cachorro correu pelo parque e brincou com a bola.",
    "O gato dormiu no sofá durante a tarde inteira.",
    "As crianças brincaram no parque e correram felizes.",
    "O leão é um animal selvagem que vive na savana.",
    "O cachorro e o gato dormiram juntos na cama.",
```

```
    "O parque estava cheio de crianças brincando e correndo."
]

# Converter textos para vetores usando TF-IDF
vectorizer = TfidfVectorizer()
vetores_texto = vectorizer.fit_transform(textos).toarray()

# Aplicar PCA
pca = PCA(n_components=2)
resultado_pca = pca.fit_transform(vetores_texto)

# Plotar os vetores projetados
plt.figure(figsize=(8, 6))
plt.scatter(resultado_pca[:, 0], resultado_pca[:, 1], color='blue', alpha=0.7)
for i, txt in enumerate(textos):
    plt.annotate(f'T{i+1}', (resultado_pca[i, 0], resultado_pca[i, 1]))

plt.title("Visualização com PCA de Representações Textuais")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.show()
```

FIGURA 80 – VISUALIZAÇÃO COM PCA DE REPRESENTAÇÕES TEXTUAIS



FONTE: O autor (2025).