

UNIVERSIDADE FEDERAL DO PARANÁ

EDSON FARIA ALVES

MEMORIAL DE PROJETOS: CONVERGÊNCIA ENTRE AGENTES AUTÔNOMOS E
MICROSSERVIÇOS

CURITIBA

2025

EDSON FARIA ALVES

MEMORIAL DE PROJETOS: CONVERGÊNCIA ENTRE AGENTES AUTÔNOMOS E
MICROSSERVIÇOS

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 Montaña

CURITIBA

2025

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 **EDSON FARIA ALVES**, intitulada: **MEMORIAL DE PROJETOS: CONVERGÊNCIA ENTRE AGENTES AUTÔNOMOS E MICROSERVIÇOS**, 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, 20 de Outubro de 2025.



RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora



JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Este parecer técnico apresenta uma análise introdutória sobre a convergência entre agentes autônomos de Inteligência Artificial (IA) e a arquitetura de microserviços, destacando suas complementaridades e desafios. Agentes, especialmente os baseados em modelos de linguagem larga escala (LLMs), estruturam-se em módulos de percepção, raciocínio, memória e execução, possibilitando comportamento adaptativo, aprendizado contínuo e tomada de decisão pró-ativa. Microserviços, por sua vez, dividem sistemas complexos em componentes independentes, escaláveis e de fácil manutenção, promovendo modularidade e baixo acoplamento. Embora ambos compartilhem princípios de independência e interoperabilidade, diferem na autonomia decisória e nas exigências de controle e auditoria. A integração dessas abordagens permite criar ecossistemas inteligentes e distribuídos, nos quais agentes coordenam e otimizam serviços de forma dinâmica. São citados casos de uso que abrangem os setores de saúde, educação, finanças e indústria, além de desafios técnicos e éticos, como segurança, vieses e governança. Conclui-se que essa convergência representa um avanço significativo, mas requer práticas sólidas de governança, segurança e explicabilidade para garantir operação ética, confiável e alinhada a objetivos organizacionais.

Palavras Chaves: agentes autônomos; inteligência artificial; microserviços; arquiteturas distribuídas; sistemas multiagentes.

ABSTRACT

This technical report presents an introductory analysis of the convergence between autonomous Artificial Intelligence (AI) agents and microservices architecture, highlighting their complementarities and challenges. Agents, especially those based on large-scale language models (LLMs), are structured into perception, reasoning, memory, and execution modules, enabling adaptive behavior, continuous learning, and proactive decision-making. Microservices, in contrast, divide complex systems into independent, scalable, and easily maintainable components, promoting modularity and low coupling. Although both share principles of independence and interoperability, they differ in decision-making autonomy and requirements for control and auditing. The integration of these approaches enables the creation of intelligent distributed ecosystems in which agents dynamically coordinate and optimize services. The report highlights use cases in sectors such as healthcare, education, finance, and industry, and discusses technical and ethical challenges such as security, bias, and governance. It is concluded that this convergence represents a significant advancement but requires solid practices in governance, security, and explainability to ensure ethical, reliable operation aligned with organizational objectives.

Keywords: autonomous agents; artificial intelligence; microservices; distributed architectures; multi-Agent systems.

SUMÁRIO

1 PARECER TÉCNICO	7
1.1 INTRODUÇÃO	7
1.2 REFERENCIAL TEÓRICO	7
1.3 ANÁLISE COMPARATIVA.....	9
1.4 CASOS DE USO E DESAFIOS.....	9
1.5 CONCLUSÃO E PERSPECTIVAS FUTURAS	10
REFERÊNCIAS.....	11
APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL	13
APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA - PYTHON.....	19
APÊNDICE 3 – LINGUAGEM DE PROGRAMAÇÃO APLICADA - R.....	42
APÊNDICE 4 – ESTATÍSTICA APLICADA I	56
APÊNDICE 5 – ESTATÍSTICA APLICADA II	69
APÊNDICE 6 – ARQUITETURA DE DADOS	96
APÊNDICE 7 – VISUALIZAÇÃO DE DADOS E STORYTELLING	125
APÊNDICE 8 – APRENDIZADO DE MÁQUINA.....	135
APÊNDICE 9 – DEEP LEARNING	163
APÊNDICE 10 – BIG DATA	200
APÊNDICE 11 – VISÃO COMPUTACIONAL	207
APÊNDICE 12 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL	243
APÊNDICE 13 – GESTÃO DE PROJETOS DE IA.....	286
APÊNDICE 14 – TÓPICOS DE INTELIGÊNCIA ARTIFICIAL	289
APÊNDICE 15 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	309

1 PARECER TÉCNICO

1.1 INTRODUÇÃO

A arquitetura de microserviços consolidou-se como um dos principais paradigmas para desenvolvimento de sistemas distribuídos, oferecendo modularidade, escalabilidade e independência de implantação. Essa abordagem parte da premissa de que sistemas complexos podem ser decompostos em serviços menores e especializados, que se comunicam por interfaces padronizadas e possuem ciclo de vida independente (Velepucha; Flores, 2023).

Paralelamente, a evolução dos modelos de linguagem de grande porte (*Large Language Models – LLMs*) possibilitou o surgimento de agentes de Inteligência Artificial (IA) com autonomia para perceber o ambiente, raciocinar sobre informações recebidas e executar ações de forma adaptativa. Esses agentes, organizados em módulos funcionais como percepção, memória, raciocínio e execução, diferenciam-se de sistemas convencionais pela capacidade de agir proativamente com base em objetivos próprios, incorporando aprendizado contínuo (Zhao; Jin; Cheng, 2023).

O presente parecer técnico não constitui um estudo aprofundado, mas uma abordagem exploratória, cujo objetivo é discutir o paralelismo e a possível convergência entre arquiteturas de microserviços e agentes LLM, identificando oportunidades de integração e os desafios técnicos e éticos envolvidos.

1.2 REFERENCIAL TEÓRICO

Esta seção apresenta os conceitos fundamentais para este parecer, abordando a arquitetura de microserviços, os agentes de inteligência artificial baseados em LLM e a convergência entre esses dois paradigmas. São destacadas as características, os desafios e a integração dessas tecnologias para o desenvolvimento de sistemas inteligentes escaláveis e eficientes.

1.2.1 Arquitetura de Microserviços

Microserviços caracterizam-se por serem componentes autônomos que implementam funcionalidades específicas e podem ser implantados, escalados e versionados de forma independente. A comunicação ocorre geralmente por meio de APIs REST ou mensageria assíncrona, com troca de dados em formatos padronizados como JSON. Essa organização favorece a manutenção, a evolução incremental e a resiliência da aplicação, mas também impõe desafios relacionados à consistência de dados, orquestração e monitoramento distribuído (Velepucha; Flores, 2023).

1.2.2 Agentes de Inteligência Artificial Baseados em LLM

Agentes LLM seguem uma arquitetura modular, integrando componentes de percepção (interpretação de dados e contexto), raciocínio (planejamento e inferência), memória (armazenamento e recuperação de informações passadas) e execução (interação com o ambiente). Além disso, incorporam mecanismos de aprendizado contínuo, permitindo adaptar seu comportamento ao longo do tempo. Essa estrutura amplia significativamente o escopo de atuação em relação a sistemas reativos tradicionais, permitindo decisões baseadas não apenas em entradas imediatas, mas em objetivos e estratégias pré-definidas (Zhao; Jin; Cheng, 2023).

1.2.3 Convergência Arquitetural

A combinação de microserviços e agentes LLM propõe a criação de ecossistemas híbridos: serviços especializados provendo dados, funcionalidades ou ações, e agentes autônomos coordenando, interpretando e decidindo sobre o uso dessas capacidades. Estudos recentes discutem a utilização da modularidade de microserviços para hospedar e interligar agentes, promovendo interoperabilidade e escalabilidade (Goyal; Bhasin, 2025).

1.3 ANÁLISE COMPARATIVA

A modularidade é um ponto central nos dois paradigmas. Enquanto microserviços seguem o princípio do acoplamento fraco para facilitar evolução e manutenção, agentes LLM utilizam modularidade funcional para distribuir responsabilidades cognitivas (percepção, raciocínio, execução). Em ambos os casos, a comunicação é fundamental, mas os protocolos diferem: microserviços dependem de APIs e mensageria, enquanto agentes empregam protocolos semânticos próprios, como linguagens de comunicação entre agentes (ACLs) e padrões recentes como MCP (*Multi-Agent Communication Protocol*) (Goyal; Bhasin, 2025).

Outra diferença essencial está no modelo de decisão. Microserviços são, em sua maioria, reativos: respondem a eventos externos segundo regras pré-programadas. Agentes LLM, por outro lado, podem iniciar ações com base em objetivos internos e experiências passadas, apresentando comportamento pró-ativo. Essa característica aumenta o potencial de automação e autonomia, mas também amplia riscos de imprevisibilidade, exigindo mecanismos de auditoria e supervisão (Yu *et al.*, 2025).

No quesito escalabilidade, microserviços contam com práticas consolidadas, como autoscaling e balanceamento de carga. Agentes LLM, entretanto, trazem novos requisitos, incluindo pipelines de avaliação contínua, rollbacks adaptativos e integração com MLOps/AIOps para atualização segura de modelos e comportamentos (Goyal; Bhasin, 2025).

1.4 CASOS DE USO E DESAFIOS

Entre os casos de uso mais relevantes da integração entre microserviços e agentes LLM estão:

- Educação: tutores virtuais adaptativos, integrados a plataformas de ensino, capazes de personalizar conteúdo e ritmo de aprendizado;
- Saúde: sistemas de apoio ao diagnóstico, conectados a prontuários eletrônicos, oferecendo análises contextuais e recomendações clínicas;
- Indústria: coordenação de robôs autônomos em linhas de produção, com agentes monitorando métricas de eficiência e acionando manutenção preditiva.

Os desafios dessa integração incluem:

- Segurança e confiabilidade: necessidade de mecanismos de validação de decisões e mitigação de vulnerabilidades (Yu *et al.*, 2025);
- Explicabilidade: dificuldade em interpretar as decisões tomadas por modelos complexos, o que impacta a confiança do usuário;
- Governança: aplicação de frameworks como TRiSM (*Trust, Risk and Security Management*) para garantir conformidade e segurança (Raza *et al.*, 2025).

1.5 CONCLUSÃO E PERSPECTIVAS FUTURAS

A convergência entre microserviços e agentes LLM representa um avanço significativo na construção de sistemas distribuídos inteligentes, combinando a modularidade e resiliência dos microserviços com a adaptabilidade e proatividade dos agentes.

No entanto, para que essa integração seja viável em produção, será necessário desenvolver práticas robustas de governança, monitoramento e avaliação contínua, mitigando riscos associados à autonomia excessiva. Tecnologias emergentes como agentes neuro-simbólicos e protocolos avançados de coordenação multiagente despontam como caminhos promissores para ampliar a interpretabilidade e a escalabilidade dessas soluções (Raza *et al.*, 2025).

O potencial é elevado, mas a adoção deve ser acompanhada de responsabilidade técnica e ética, garantindo que sistemas autônomos e distribuídos operem de forma segura, transparente e alinhada a objetivos organizacionais e sociais.

REFERÊNCIAS

- BELLMAN, R. **An Introduction to Artificial Intelligence: Can Computers Think?** San Francisco, CA: Boyd & Fraser Publishing Company, 1978. ISBN 9780878350667.
- CUNHA, M. B.; ALMEIDA, J. P. **Inteligência Artificial e Direito: Regulação e Desafios Éticos.** Curitiba, PR: Editora Juruá, 2020.
- FLORIDI, L. **A Revolução da Informação: Como a Filosofia Pode Explicar a Revolução Digital.** Petrópolis, RJ: Editora Vozes, 2020.
- GOYAL, M.; BHASIN, P. **Moving From Monolithic To Microservices Architecture for Multi-Agent Systems.** arXiv preprint, 2025. ArXiv:2505.07838. Disponível em: <<https://arxiv.org/abs/2505.07838>>. Acesso em: 05 set. 2025.
- HAUGELAND, J. **Artificial Intelligence: The Very Idea.** Cambridge, MA: MIT Press, 1985. ISBN 9780262081535.
- HUBSPOT Brasil. **ChatGPT: o que é, como funciona e dicas para usar a ferramenta.** 2024. <<https://br.hubspot.com/blog/marketing/chatgpt>>. Acesso em: 15 fev. 2024.
- KRIZHEVSKY, A. **Learning Multiple Layers of Features from Tiny Images.** 2009. <<https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz>>. Acesso em: 16 set. 2025.
- MENEZES, C. A.; FERNANDES, L. V. **Algoritmos e Sociedade: Impactos da Inteligência Artificial no Cotidiano.** São Paulo, SP: Editora Senac, 2019.
- NASSIF, E. M.; MEIRELES, M. **Inteligência Artificial: Fundamentos, Aplicações e Tecnologias.** Rio de Janeiro, RJ: Editora LTC, 2019.
- OLIVEIRA, T. S.; CARDOSO, R. M. **Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável.** São Paulo, SP: Editora Atlas, 2020.
- PARASMO. **A linha do tempo da indústria automotiva no Brasil.** 2024. <<https://parasmo.com.br/a-linha-do-tempo-da-industria-automotiva-no-brasil-2/>>. Acesso em: 05 mar. 2025.
- PEREIRA, G. M.; SANTOS, A. R. **Segurança e Privacidade na Era da Inteligência Artificial.** São Paulo, SP: Editora Novatec, 2021.
- RAZA, S. *et al.* **TRiSM for Agentic AI: a review of trust, risk, and security management in llm-based agentic multi-agent systems.** arXiv preprint, 2025. ArXiv:2506.04133. Disponível em: <<https://arxiv.org/abs/2506.04133>>. Acesso em: 02 set. 2025.
- REVISTA Pegada - UNESP. **Análise da indústria automobilística brasileira.** 2021. <<https://revista.fct.unesp.br/index.php/pegada/article/download/8555/pdf/33093>>. Acesso em: 05 mar. 2025.
- RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach.** 4th. ed. London, UK: Pearson, 2021.
- SILVA, J. R.; LIMA, F. P. **Inteligência Artificial na Educação: Potencialidades e Limitações.** Porto Alegre, RS: Editora Penso, 2022.

SOUZA, A. C. M.; GONÇALVES, P. D. **Ética e Inteligência Artificial: Desafios e Oportunidades**. Curitiba, PR: Editora UFPR, 2021.

TG Poli. **História e evolução da indústria automotiva brasileira**. 2023. <<https://www.tgpoli.com.br/noticias/historia-e-evolucao-da-industria-automotiva-brasileira/>>. Acesso em: 05 mar. 2025.

VELEPUCHA, V.; FLORES, P. **A Survey on Microservices Architecture: principles, patterns and migration challenges**. IEEE Access, v. 11, p. 85937–85957, 2023. DOI:10.1109/ACCESS.2023.3305687. Disponível em: <<https://doi.org/10.1109/ACCESS.2023.3305687>>. Acesso em: 18 out. 2025.

XIN, D. *et al.* **Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows**. In: CHI Conference on Human Factors in Computing Systems (CHI'21). Yokohama, Japão: [s.n.], 2021. Acesso em: 05 mar. 2025. Disponível em: <<https://doi.org/10.1145/3411764.3445306>>.

YU, M. *et al.* **A Survey on Trustworthy LLM Agents: threats and countermeasures**. arXiv preprint, 2025. ArXiv:2503.09648. Disponível em: <<https://arxiv.org/abs/2503.09648>>. Acesso em: 02 set. 2025.

ZHAO, P.; JIN, Z.; CHENG, N. **An In-depth Survey of Large Language Model-based Artificial Intelligence Agents**. arXiv preprint, 2023. ArXiv:2309.14365. Disponível em: <<https://arxiv.org/abs/2309.14365>>. Acesso em: 25 ago. 2025.

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

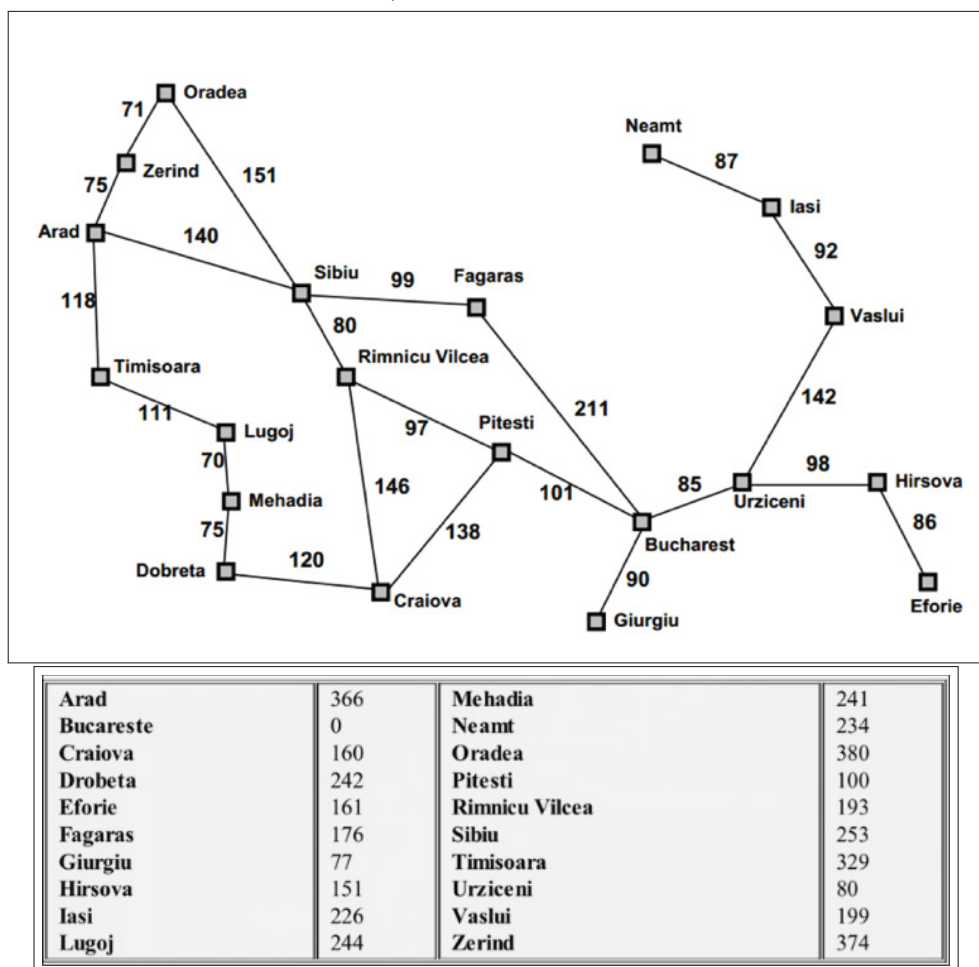
Realize uma busca utilizando o algoritmo A^* para encontrar o melhor caminho para chegar a Bucharest partindo de Lugoj. Construa a árvore de busca criada pela execução do algoritmo apresentando os valores de $f(n)$, $g(n)$ e $h(n)$ para cada nó. Utilize a heurística de distância em linha reta, que pode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

FIGURA 1 – 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. 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: $\alpha, \beta \vdash \alpha \wedge \beta$

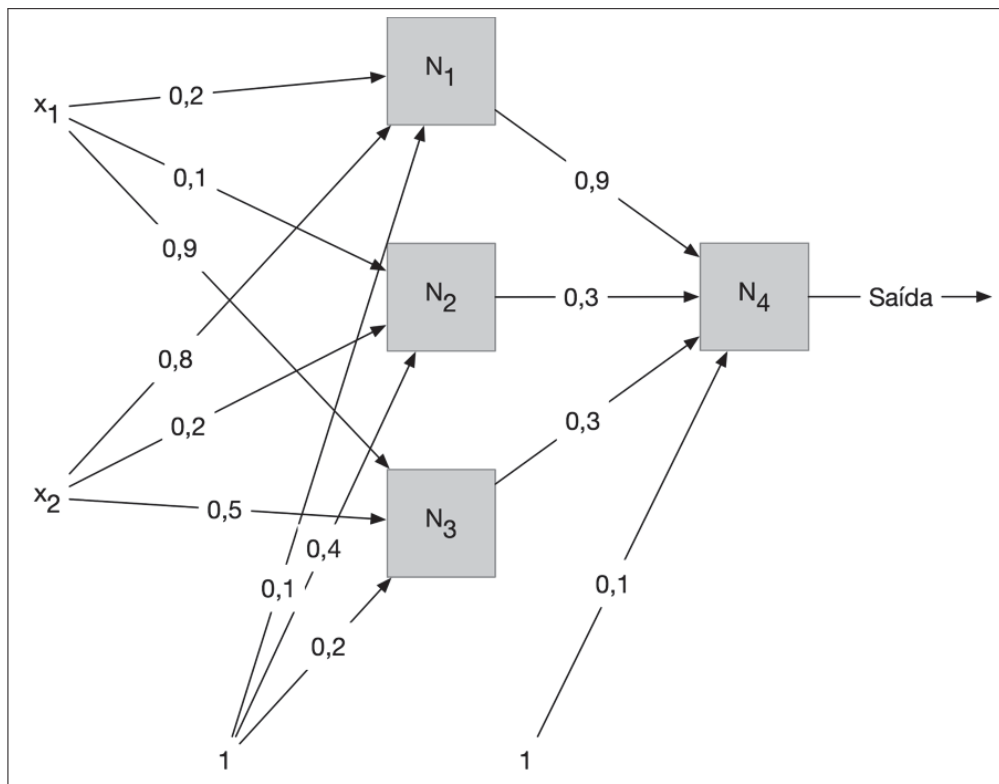
Equivalência Bicondicional: $(\alpha \leftrightarrow \beta)$ equivale a $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

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.

FIGURA 2 – REDE NEURAL ARTIFICIAL - INTRODUÇÃO A IA.



Os neurônios N1, N2 e N3 possuem função de ativação linear. Já N4 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. ChatGPT

- (a) ChatGPT:

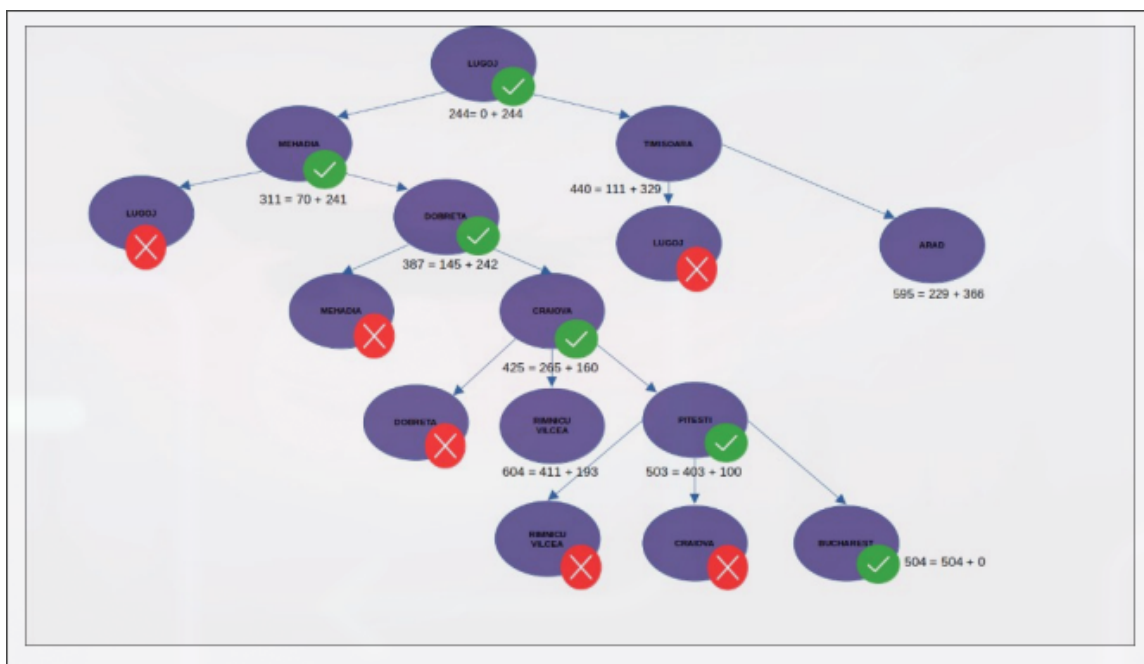
"Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e programas capazes de realizar tarefas que, quando realizadas por seres humanos, geralmente requerem inteligência. Essas tarefas incluem reconhecimento de padrões, resolução de problemas, aprendizado, tomada de decisões, compreensão da linguagem natural e interação com o ambiente. A IA visa simular processos cognitivos humanos em computadores, permitindo que máquinas ajam de forma inteligente em diversas situações. Os principais subcampos da IA incluem aprendizado de máquina, visão computacional, processamento de linguagem natural, robótica e planejamento de sistemas inteligentes."

- (b) A resposta do ChatGPT pode ser classificada principalmente na abordagem *Pensar como humano*, pois enfatiza a simulação dos processos cognitivos humanos em computadores, permitindo que máquinas atuem de forma inteligente. Conforme Haugeland (Haugeland, 1985), a IA é "o novo e interessante esforço para fazer os computadores pensarem, máquinas com mentes, no sentido total e literal". Bellman (Bellman, 1978) também ressalta que a IA automatiza atividades associadas ao pensamento humano, como tomada de decisões, resolução de problemas e aprendizado.
- (c) O ChatGPT é uma ferramenta conversacional online baseada em técnicas de modelagem de linguagem. Seu funcionamento depende de uma base massiva de dados composta por informações padronizadas, provenientes de diversas fontes na internet e das interações dos usuários. Por se basear em múltiplas fontes, as respostas podem conter imprecisões, o que requer análise crítica. O modelo também opera de forma colaborativa, permitindo que os usuários corrijam e complementem as informações. Diferentemente dos mecanismos tradicionais de busca, como o Google, o ChatGPT não recupera informações específicas, mas gera textos completos em tempo real (HubSpot Brasil, 2024) .
- (d) Analisando o funcionamento do ChatGPT, observa-se que ele se enquadra na abordagem *Agir racionalmente*. Embora pareça agir como humano pela vasta base de dados, seu

processo de treinamento contínuo permite que aprenda novos conhecimentos e estratégias para geração de respostas coerentes, configurando um raciocínio automatizado (Russell; Norvig, 2021).

2. Busca Heurística

FIGURA 3 – RESPOSTA PARA A BUSCA HERÍSTICA



FONTE: Elaborado pelo autor (2024).

3. Lógica

1º Aplicando a **Equivalência da Implicação** em $R3$ ($\neg r \vee p \equiv r \rightarrow p$):

$$R4 : r \rightarrow p$$

2º Aplicando o **Silogismo Hipotético** em $R1$ e $R2$:

$$R5 : p \rightarrow r$$

3º Aplicando a **Equivalência Bicondicional** entre $R4$ e $R5$:

$$R6 : r \leftrightarrow p \equiv (r \rightarrow p) \wedge (p \rightarrow r)$$

4º Aplicando o **Silogismo Hipotético** entre $R1 : p \rightarrow q$ e $R6 : r \leftrightarrow p$, podemos substituir r por p na implicação para concluir:

$$R7 : q \leftrightarrow p$$

Conclusão: A partir das inferências aplicando as regras indicadas, obtemos que:

$$q \leftrightarrow p,$$

ou seja, a raposa come as uvas se, e somente se, as uvas caem, confirmando a validade do argumento.

4. Redes Neurais Artificiais

FIGURA 4 – RESPOSTA REDE NEURAL ARTIFICIAL - INTRODUÇÃO IA

VARIÁVEIS	VALORES	PESOS			
		→ N1	→ N2	→ N3	→ N4
X1	-3,0000	0,2000	0,1000	0,9000	
X2	1,0000	0,8000	0,2000	0,5000	
BIAS	1,0000	0,1000	0,4000	0,2000	0,1000
N1	0,3000				0,9000
N2	0,3000				0,3000
N3	-2,0000				0,3000
N4	-0,1391				
RESPOSTAS DA QUESTÃO					
a) Valor de saída do neurônio N1		0,3000			
b) Valor de saída do neurônio N2		0,3000			
c) Valor de saída do neurônio N3		-2,0000			
d) Valor de saída do neurônio N4		-0,1391			

FONTE: Resultado obtido pelo autor (2024).

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA - PYTHON

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:

TABELA 1 – DESCRIÇÃO DOS DADOS DO ARQUIVO `precos_carros_brasil.csv`

Nome do campo	Descrição
<code>year_of_reference</code>	O preço médio corresponde a um mês de ano de referência
<code>month_of_reference</code>	O preço médio corresponde a um mês de referência, ou seja, a FIPE atualiza sua tabela mensalmente
<code>fipe_code</code>	Código único da FIPE
<code>authentication</code>	Código de autenticação único para consulta no site da FIPE
<code>brand</code>	Marca do carro
<code>model</code>	Modelo do carro
<code>fuel</code>	Tipo de combustível do carro
<code>gear</code>	Tipo de engrenagem do carro
<code>engine_size</code>	Tamanho do motor em centímetros cúbicos
<code>year_model</code>	Ano do modelo do carro. Pode não corresponder ao ano de fabricação
<code>avg_price</code>	Preço médio do carro, em reais

FONTE: <<https://www.kaggle.com/datasets/vagnerbessa/average-car-prices-brazil/data>> (2024).

1. Análise Exploratória dos dados

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

- Carregue a base de dados `media_precos_carros_brasil.csv`.
- 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 uma 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 R2.
- h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada.

B - RESOLUÇÃO

Importação das bibliotecas

Este trecho do código-fonte realiza a importação das bibliotecas necessárias para a análise de dados, visualização gráfica, modelagem preditiva e avaliação de desempenho. As bibliotecas são organizadas por função: manipulação de dados (pandas, numpy), gráficos (seaborn, matplotlib), aprendizado de máquina (scikit-learn, xgboost) e supressão de alertas (warnings).

```
1  ## Biblioteca para manipulacao e analise de dados
2  import pandas as pd
3
4  ## Bibliotecas para exibicao de graficos
5  import seaborn as sns
6  import matplotlib.pyplot as plt
7
8  ## Biblioteca que possibilita a computacao numerica com Python
9  import numpy as np
10
11  ## Desabilitar os warnings
12  import warnings
13  warnings.filterwarnings('ignore')
14
15  # Bibliotecas de machine learning
16  from sklearn.model_selection import train_test_split
17  from sklearn.ensemble import RandomForestRegressor
18  from xgboost import XGBRegressor
19  from sklearn.preprocessing import LabelEncoder
20
21  # Metricas de avaliacao dos modelos
22  from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
```

1. Análise Exploratória dos dados

a. Carregar base de dados

Este trecho realiza o carregamento de uma planilha CSV contendo dados de preços de carros no Brasil para um DataFrame do pandas. Em seguida, apresenta um resumo das informações disponíveis no conjunto de dados, como o número de linhas e colunas, tipos de dados e valores ausentes, auxiliando na etapa inicial de compreensão e inspeção da estrutura do dataset.

```

1  ## Carregamento da planilha de precos em um dataframe
2  precos_carros = pd.read_csv("../dados/precos_carros_brasil.csv", decimal=',')
3
4  Planilha carregada no dataframe "precos_carros".
5
6  #-----
7  ### Resumo da planilha
8  precos_carros.info()
9
10 <class 'pandas.core.frame.DataFrame'>
11 RangeIndex: 267542 entries, 0 to 267541
12 Data columns (total 11 columns):
13 #   Column                Non-Null Count  Dtype
14 ---  ---
15 0   year_of_reference      202297 non-null  float64
16 1   month_of_reference     202297 non-null  object
17 2   fiipe_code             202297 non-null  object
18 3   authentication         202297 non-null  object
19 4   brand                  202297 non-null  object
20 5   model                  202297 non-null  object
21 6   fuel                   202297 non-null  object
22 7   gear                   202297 non-null  object
23 8   engine_size            202297 non-null  float64
24 9   year_model             202297 non-null  float64
25 10  avg_price_brl          202297 non-null  float64
26 dtypes: float64(4), object(7)
27 memory usage: 22.5+ MB
28
29 #-----
30 ### Composicao inicial do dataset
31 precos_carros.shape
32 (267542, 11)
33

```

```

34 #-----
35 ## Tipos de dados das colunas
36 precos_carros.dtypes
37
38 year_of_reference    float64
39 month_of_reference   object
40 fiipe_code           object
41 authentication       object
42 brand                object
43 model                object
44 fuel                 object
45 gear                 object
46 engine_size          float64
47 year_model           float64
48 avg_price_brl        float64
49 dtype: object

```

b. Verificar se há valores faltantes

O código a seguir identifica a quantidade de valores nulos presentes em cada coluna do conjunto de dados. Posteriormente, verifica que as linhas com ausência de valor na variável referente ao preço médio possuem dados faltantes em todas as demais colunas. Com base nessa constatação, realiza a remoção dessas linhas incompletas, assegurando que o conjunto fique livre de valores nulos. Por fim, confirma a eliminação dos valores ausentes e apresenta a nova dimensão do conjunto resultante.

```

1 ## Verificando os valores nulos e faltantes das colunas
2 precos_carros.isnull().sum()
3
4 year_of_reference    65245
5 month_of_reference   65245
6 fiipe_code           65245
7 authentication       65245
8 brand                65245
9 model                65245
10 fuel                 65245
11 gear                 65245
12 engine_size          65245
13 year_model           65245
14 avg_price_brl        65245
15 dtype: int64
16

```

```

17 ## Existem valores faltantes em todas as colunas
18
19 #-----
20 ### Analisando quais valores estao nulos a partir do preco medio do carro
21 preco_medio_esta_nulo = precos_carros['avg_price_brl'].isna()
22 linhas_filtradas = precos_carros[preco_medio_esta_nulo]
23 linhas_filtradas.isna().sum()
24
25 year_of_reference      65245
26 month_of_reference     65245
27 fiipe_code             65245
28 authentication        65245
29 brand                  65245
30 model                  65245
31 fuel                   65245
32 gear                   65245
33 engine_size            65245
34 year_model             65245
35 avg_price_brl          65245
36 dtype: int64
37
38
39 ## Considerando que todas as colunas possuem valores nulos nas mesmas linhas, o
    melhor caminho para esse dataset esta em remover tais colunas.
40
41 #-----
42 Shape do dataset antes de apagar as linhas nulas (267542, 11)
43
44 #-----
45 ## Excluindos as linhas nulas
46 precos_carros.dropna(subset=['avg_price_brl'], inplace=True)
47
48 #-----
49 ## Verificar se ainda existe linhas nulas
50 precos_carros.isna().sum()
51
52
53 year_of_reference      0
54 month_of_reference     0

```



```

55 fipecode      0
56 authentication 0
57 brand         0
58 model         0
59 fuel          0
60 gear          0
61 engine_size   0
62 year_model    0
63 avg_price_brl 0
64 dtype: int64
65
66 #-----
67 Shape do dataset depois de apagar as linhas nulas (202297, 11)

```

c. Verificar se há valores duplicados

O código a seguir verifica a presença de valores duplicados no conjunto de dados e informa a quantidade encontrada. Em seguida, realiza a remoção desses registros duplicados, garantindo a unicidade das observações. Por fim, confirma a eliminação dos duplicados e apresenta a nova dimensão do conjunto resultante.

```

1  ## Conferindo se ha valores duplicados no dataset
2  print(f'Existem {precos_carros.duplicated().sum()} valores duplicados')
3
4  print('Removendo esses valores')
5  precos_carros.drop_duplicates(inplace=True)
6
7  print(f'Agora existem {precos_carros.duplicated().sum()} valores duplicados')
8
9  precos_carros.shape
10
11 #-----
12 Existem 2 valores duplicados
13 Removendo esses valores
14 Agora existem 0 valores duplicados
15 (202295, 11)

```

d. Separar os valores numéricos e categóricos.

O código a seguir classifica as variáveis do conjunto de dados em duas categorias: numéricas e categóricas. Essa separação é realizada com base no tipo de dado de cada coluna, facilitando as etapas subsequentes de exploração e análise dos dados.

```

1 # Criando categorias para separar colunas numericas e categoricas: facilita a AED
2 numericas_cols = [col for col in precos_carros.columns if precos_carros[col].
    dtype != 'object']
3 categoricas_cols = [col for col in precos_carros.columns if precos_carros[col].
    dtype == 'object']

```

e. Imprimir a contagem de valores por modelo e marca de carros.

```

1 ## Agrupando o dataset por modelo e marca e imprimindo a contagem
2 precos_carros.groupby(['brand', 'model']).count()

```

TABELA 2 – AMOSTRA DO DATASET DE PREÇOS DE CARROS

brand	model	year_of_reference	month_of_reference	type_code	authentication	fuel	gear	engine_size	year_model	avg_price_brl
Fiat	500 ABARTH MULTIAIR 1.4 TB 16V 3p	50	50	50	50	50	50	50	50	50
Fiat	500 Cabrio Dualogic Flex 1.4 8V	75	75	75	75	75	75	75	75	75
Fiat	500 Cabrio Flex 1.4 8V Mec.	50	50	50	50	50	50	50	50	50
Fiat	500 Cabrio/500 Coupe Gucci/Flex 1.4 Aut.	100	100	100	100	100	100	100	100	100
Fiat	500 Cult 1.4 Flex 8V EVO Dualogic	100	100	100	100	100	100	100	100	100
...										
VW - Volkswagen	up! move I MOTION 1.0 T. Flex 12V 3p	50	50	50	50	50	50	50	50	50
VW - Volkswagen	up! move I MOTION 1.0 T. Flex 12V 5p	125	125	125	125	125	125	125	125	125
VW - Volkswagen	up! take 1.0 T. Flex 12V 3p	100	100	100	100	100	100	100	100	100
VW - Volkswagen	up! take 1.0 Total Flex 12V 5p	150	150	150	150	150	150	150	150	150
VW - Volkswagen	up! track 1.0 Total Flex 12V 5p	25	25	25	25	25	25	25	25	25

2112 rows x 9 columns

FONTE: Dados processados pelo autor (2024).

f. Explicação sobre os resultados do dataset.

Diante da quantidade expressiva de valores ausentes e da ausência de relevância informacional, optou-se pela exclusão das linhas incompletas. Após essa etapa, verificou-se a presença de apenas duas observações duplicadas, as quais foram prontamente removidas. Com isso, o conjunto de dados passou a conter 202.295 linhas e 11 colunas. Em relação à tipologia das variáveis, identificou-se que o dataset é composto por 3 colunas numéricas e 8 categóricas, informação essencial para orientar as próximas etapas da análise.

2. Visualização dos dados

a. Gráfico da distribuição da quantidade de carros por marca.

O código a seguir gera uma visualização da distribuição da quantidade de veículos por marca presentes no conjunto de dados. Para isso, utiliza um gráfico de barras construído com a biblioteca Seaborn, ordenando as marcas de acordo com sua frequência. A representação gráfica permite identificar as montadoras com maior número de registros na base analisada, o que pode indicar padrões relevantes no mercado automotivo.

```

1 ## Distribuicao da quantidade de carros por marcas
2 plt.figure(figsize=(12,7))

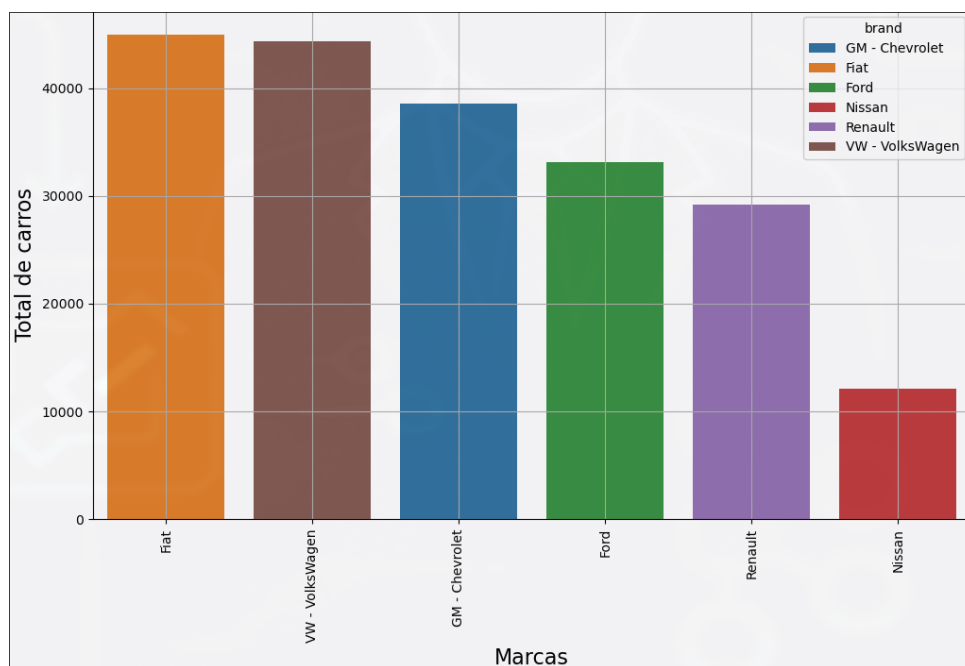
```

```

3 plt.title('Distribuicao da quantidade de carros por marcas', fontsize=20)
4 plt.ylabel('Total de carros', fontsize=16) # Rotulo do eixo Y
5 plt.xlabel('Marcas', fontsize=16) # Rotulo do eixo x
6 plt.xticks(rotation=90); # xticks para indicar a rotacao do texto no eixo X (90
    graus)
7 sns.countplot(
8 data=precos_carros,
9 x='brand',
10 hue='brand',
11 order=precos_carros['brand'].value_counts().index,
12 legend='full'
13 )
14 plt.grid(True)
15 plt.show()

```

FIGURA 5 – DISTRIBUIÇÃO DA QUANTIDADE DE CARROS POR MARCAS.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

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

O código a seguir realiza a contagem da quantidade de veículos de acordo com os diferentes tipos de engrenagem disponíveis no conjunto de dados. Em seguida, apresenta essa distribuição por meio de um gráfico de barras, facilitando a visualização da frequência de cada categoria.

```

1 ## Distribuicao da quantidade de carros por marcas

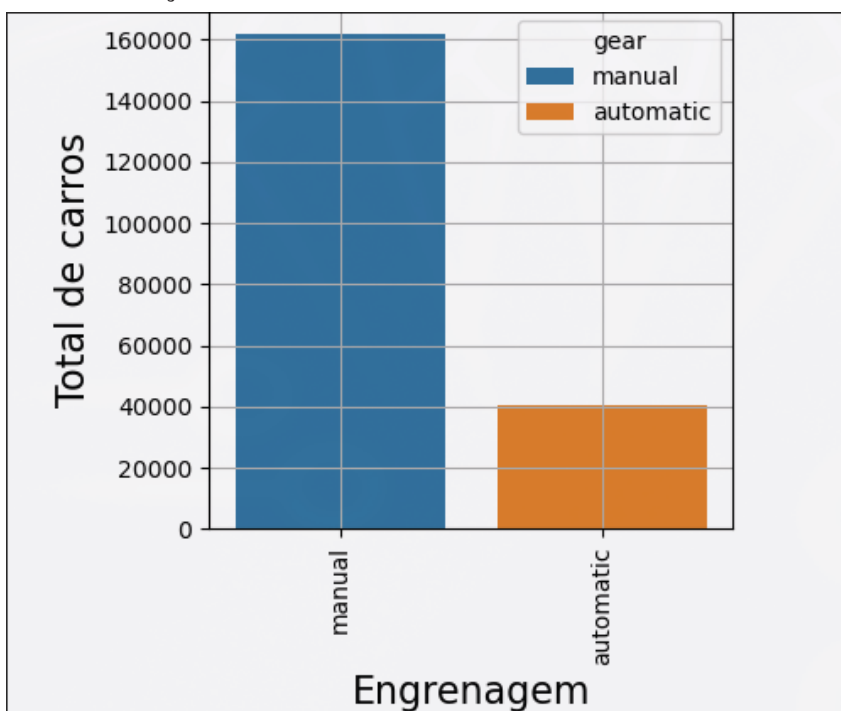
```

```

2 plt.figure(figsize=(4,4))
3 plt.title('Distribuicao da quantidade de carros por tipo de engrenagem', fontsize
    =20)
4 plt.ylabel('Total de carros', fontsize=16) # Rotulo do eixo Y
5 plt.xlabel('Engrenagem', fontsize=16) # Rotulo do eixo x
6 plt.xticks(rotation=90); # xticks para indicar a rotacao do texto no eixo X (90
    graus)
7 sns.countplot(data=precos_carros, x='gear', hue='gear', legend='full')
8 plt.grid(True)
9 plt.show()

```

FIGURA 6 – DISTRIBUIÇÃO DA QUANTIDADE DE CARROS POR TIPO DE ENGRENAGEM.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

c. **Gráfico da evolução da média de preço dos carros ao longo dos meses de 2022.**

O código a seguir seleciona os dados de veículos referentes ao ano de 2022 e calcula a média mensal dos preços. Em seguida, organiza essas informações em um novo conjunto de dados e as apresenta por meio de um gráfico de linha, permitindo visualizar a evolução dos preços médios dos veículos ao longo dos meses desse ano.

```

1 ## Dados que serao exibidos no grafico
2 precos_carros_2022 = precos_carros.query("year_of_reference == 2022")[['month_of_
    reference', 'avg_price_brl']]

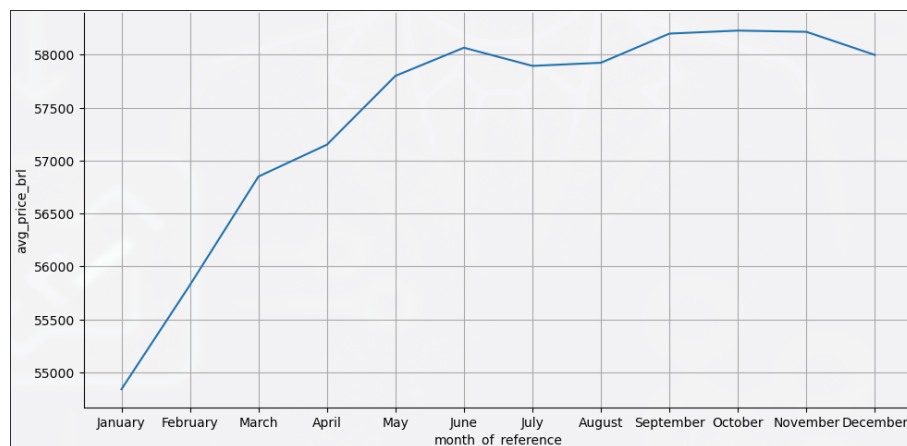
```

```

3 dados_grafico_preco_por_mes = precos_carros_2022.groupby('month_of_reference',
    sort=False)['avg_price_brl'].mean().round(2)
4 dados_grafico_preco_por_mes = dados_grafico_preco_por_mes .reset_index(name='avg_
    price_brl')
5
6 ## Grafico que sera exibido
7 sns.relplot(
8     data=dados_grafico_preco_por_mes,
9     x="month_of_reference",
10    y="avg_price_brl",
11    kind="line",
12    aspect=2,
13    dashes=True,
14 ).set(title="Evolucao da media de preco dos carros ao longo dos meses de 2022")
15 plt.grid(True)
16 plt.show()

```

FIGURA 7 – EVOLUÇÃO DA MÉDIA DE PREÇO DOS CARROS AO LONGO DOS MESES DE 2022.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

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

O código a seguir calcula o preço médio dos veículos agrupados por marca e tipo de câmbio, utilizando os dados disponíveis. Em seguida, gera um gráfico de barras que apresenta a distribuição desses preços médios por marca, diferenciando-os conforme o tipo de engrenagem. Essa visualização facilita a comparação dos valores médios entre diferentes categorias e pode auxiliar na identificação de tendências no mercado automotivo.

```

1 ## Agrupando o dataset por modelo e marca e imprimindo a contagem
2 precos_medio_carros_marca_cambio = precos_carros.groupby(['brand', 'gear'])['avg_
    price_brl'].mean().round(2)

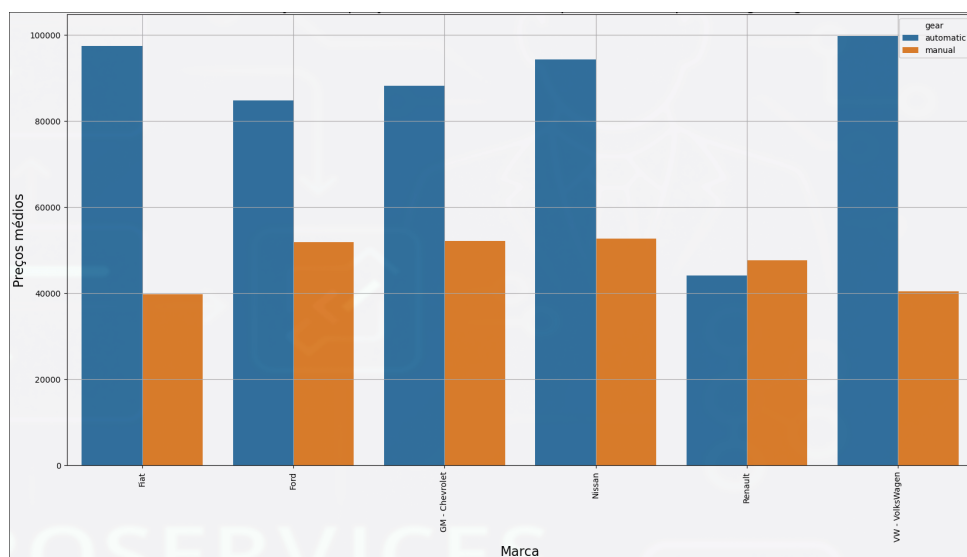
```

```

3
4 ## Resetando o index
5 precos_medio_carros_marca_cambio = precos_medio_carros_marca_cambio.reset_index(
    name='avg_price_brl')
6
7 ## Distribuicao da quantidade de carros por marcas
8 plt.figure(figsize=(20,10))
9 plt.title('Distribuicao dos precos medios dos carros por marca e tipo de
    engrenagem', fontsize=20)
10 plt.ylabel('Precos medios', fontsize=16) # Rotulo do eixo Y
11 plt.xlabel('Marca', fontsize=16) # Rotulo do eixo x
12 plt.xticks(rotation=90); # xticks para indicar a rotacao do texto no eixo X (90
    graus)
13 sns.barplot(
14 data=precos_medio_carros_marca_cambio,
15 x='brand',
16 y='avg_price_brl',
17 hue='gear',
18 legend='full'
19 )
20 plt.grid(True)
21 plt.show()

```

FIGURA 8 – DISTRIBUIÇÃO DOS PREÇOS MÉDIOS DOS CARROS POR MARCA E TIPO DE ENGRENAJEM.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

e. **Explicação sobre os resultados gerados no item d.**

O gráfico apresentado evidencia que as marcas Fiat e Volkswagen possuem os maiores preços médios no mercado brasileiro. Por outro lado, a marca Renault apresenta os preços médios mais baixos. Além disso, observa-se que, em geral, os veículos equipados com câmbio automático apresentam valores superiores em relação aos de câmbio manual, exceto para a marca Renault, na qual os modelos com câmbio manual se destacam por um preço médio mais elevado.

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

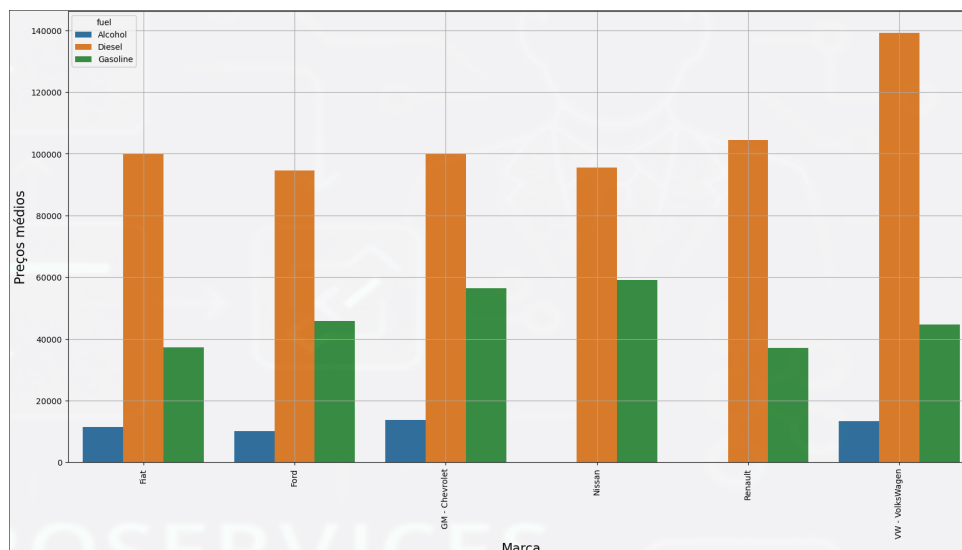
O código a seguir calcula o preço médio dos veículos agrupados por marca e tipo de combustível presentes no conjunto de dados. Em seguida, utiliza um gráfico de barras construído com a biblioteca Seaborn para visualizar essa distribuição, organizando as marcas no eixo x e destacando os diferentes tipos de combustível. Essa representação permite comparar os preços médios dos carros conforme a combinação entre marca e combustível, facilitando a análise das variações de preço relacionadas a essas características.

```

1  ## Agrupando o dataset por modelo e marca e imprimindo a contagem
2  precos_medio_carros_marca_combustivel = precos_carros.groupby(['brand', 'fuel'])[
    'avg_price_brl'].mean().round(2)
3
4  ## Resetando o index
5  precos_medio_carros_marca_combustivel = precos_medio_carros_marca_combustivel.
    reset_index(
6      name='avg_price_brl'
7  )
8
9  ## Distribuicao da quantidade de carros por marcas
10 plt.figure(figsize=(20,10))
11 plt.title('Distribuicao dos precos medios dos carros por marca e tipo de
    combustivel', fontsize=20)
12 plt.ylabel('Precos medios', fontsize=16) # Rotulo do eixo Y
13 plt.xlabel('Marca', fontsize=16) # Rotulo do eixo x
14 plt.xticks(rotation=90); # xticks para indicar a rotacao do texto no eixo X (90
    graus)
15 sns.barplot(
16     data=precos_medio_carros_marca_combustivel,
17     x='brand',
18     y='avg_price_brl',
19     hue='fuel',
20     legend='full'
21 )
22 plt.grid(True)
23 plt.show()

```

FIGURA 9 – DISTRIBUIÇÃO DOS PREÇOS MÉDIOS DOS CARROS POR MARCA E TIPO DE COMBUSTÍVEL.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

g. Explicação sobre os resultados gerados no item f.

O gráfico apresentado indica que, no Brasil, os veículos movidos a diesel possuem os maiores preços médios, possivelmente devido à predominância de caminhões e pickups nessa categoria. A marca Volkswagen apresenta a maior média de preço para veículos a diesel.

Além disso, observa-se que os veículos movidos a álcool possuem os menores preços médios. Já os veículos movidos a gasolina, produzidos por todas as marcas, apresentam preços médios geralmente superiores ao dobro daqueles movidos a álcool.

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

a. Seleção das variáveis numéricas para serem as variáveis independentes do modelo

O código a seguir realiza a codificação das variáveis categóricas presentes no conjunto de dados utilizando a técnica *Label Encoding*, facilitando a análise quantitativa subsequente. Em seguida, gera um mapa de correlação das variáveis numéricas, calculado pelo coeficiente de Spearman, que é apresentado por meio de um *heatmap* construído com a biblioteca Seaborn. Esta visualização permite identificar possíveis relações e dependências entre as variáveis numéricas e a variável alvo, contribuindo para a compreensão dos padrões presentes na base de dados.

```

1
2 precos_carros['cod_brand'] = LabelEncoder().fit_transform(precos_carros['brand']
3 precos_carros['cod_fuel'] = LabelEncoder().fit_transform(precos_carros['fuel'])

```



```

4 precos_carros['cod_gear'] = LabelEncoder().fit_transform(precos_carros['gear'])
5 precos_carros['cod_month_of_reference'] = LabelEncoder().fit_transform(precos_
  carros['month_of_reference'])
6
7 # ##### Correlação das variáveis #####
8 # Mapa de correlação das variáveis numéricas com variável Target
9 numericas_cols = [col for col in precos_carros.columns if precos_carros[col].
  dtype != 'object']
10 precos_carros_num = precos_carros.drop(categoricas_cols, axis = 1)
11 sns.heatmap(
12 precos_carros_num.corr("spearman"),
13 annot = True,
14 linewidth=.5,
15 fmt=".2f"
16 )
17 plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
18 plt.show()

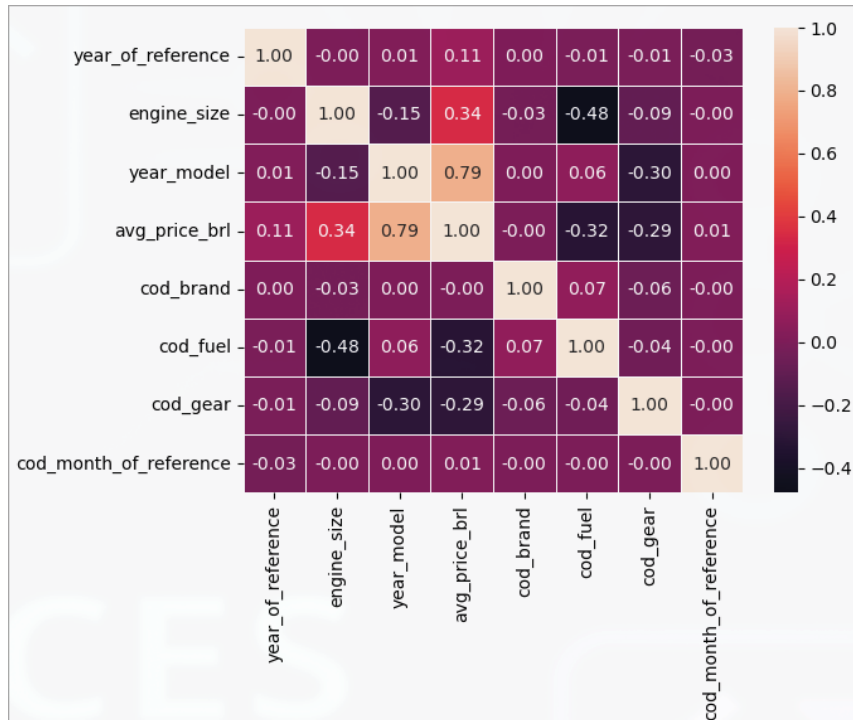
```

```

1 # Mapa de correlação das variáveis numéricas com variável Target
2 numericas_cols = [col for col in precos_carros.columns if precos_carros[col].
  dtype != 'object']
3 precos_carros_num = precos_carros.drop(categoricas_cols, axis = 1)
4 sns.heatmap(
5     precos_carros_num.corr("spearman"),
6     annot = True,
7     linewidth=.5,
8     fmt=".2f"
9 )
10 plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
11 #plt.grid(True)
12 plt.show()

```

FIGURA 10 – MAPA DE CORRELAÇÃO DAS VARIÁVEIS NUMÉRICAS.



FONTE: Elaborado pelo autor a partir de dados de Kaggle (2024).

b. Dividir o dataframe em partições de treino e teste.

O código a seguir prepara o conjunto de dados para a modelagem preditiva, separando as variáveis numéricas de interesse na matriz de atributos X , excluindo a variável alvo. A variável resposta Y contém os valores do preço médio dos veículos. Em seguida, realiza-se a divisão dos dados em conjuntos de treinamento e teste, destinando 75% para treinamento e 25% para teste, garantindo a aleatoriedade do processo por meio de uma semente fixa.

```

1 # Variável X contém apenas variáveis numéricas de interesse para a análise,
   excluindo a variável target
2 X = precos_carros_num.drop(['avg_price_brl'],axis = 1)
3
4 print(X.head())
5 # Variável Y contém apenas a variável target - avg_price_brl
6 Y = precos_carros_num['avg_price_brl']
7
8 # Divisão: 25% dos dados são de teste e 75% de treinamento
9 X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
   random_state = 42)

```

c. Treinar os modelos RandomForest e XGBoost.

O código a seguir implementa e ajusta modelos de regressão utilizando os algoritmos Random Forest e XGBoost. Inicialmente, são treinados modelos padrão, sem parâmetros personalizados. Em seguida, são configuradas versões otimizadas com parâmetros específicos para melhorar o desempenho. Todos os modelos são ajustados com base no conjunto de dados de treinamento previamente definido.

```

1 # Algoritmo xgboost sem parâmetros
2 model_rf = RandomForestRegressor()
3
4 # Ajuste do modelo, de acordo com as variáveis de treinamento
5 model_rf.fit(X_train, Y_train)
6
7 # Algoritmo Random Forest, sem parâmetros
8 model_rf_with_parameters = RandomForestRegressor(
9     min_samples_leaf=70,
10    random_state=43,
11    n_estimators = 60
12 )
13
14 # Ajuste do modelo, de acordo com as variáveis de treinamento
15 model_rf_with_parameters.fit(X_train, Y_train)
16
17 ## Algoritmo xgboost sem parametros
18 model_xgboost = XGBRegressor()
19
20 # Ajuste do modelo, de acordo com as variáveis de treinamento
21 model_xgboost.fit(X_train, Y_train)
22
23 ## Algoritmo xgboost com parametros
24 model_xgboost_with_parameters = XGBRegressor(
25     n_estimators=40
26 )
27
28 # Ajuste do modelo, de acordo com as variáveis de treinamento
29 model_xgboost_with_parameters.fit(X_train, Y_train)

```

d. Gravar os valores preditos em variáveis.

O código a seguir realiza a etapa de predição utilizando os modelos de regressão previamente treinados. São gerados os valores estimados de preço médio dos veículos a partir do conjunto de teste, considerando quatro variações: Random Forest e XGBoost, ambos com e sem parame-

trizações específicas. Os resultados obtidos servirão de base para a avaliação do desempenho preditivo dos modelos.

```

1 # Predição dos valores para random florest sem parâmetros nos dados de teste
2 valores_preditos_rf = model_rf.predict(X_test)
3 valores_preditos_rf
4
5 array([ 43147.42731706, 12192.20971598, 15380.63598232, ...,
6        113835.42040413, 16060.0635    , 21795.14863372])
7
8 #-----
9 # Predição dos valores para random florest com parâmetros nos dados de teste
10 valores_preditos_rf_with_parameters = model_rf_with_parameters.predict(X_test)
11 valores_preditos_rf_with_parameters
12
13 array([ 44861.33399553, 12635.99888517, 15348.59375211, ...,
14        109214.49732924, 15391.11869592, 21209.37990592])
15
16
17 #-----
18
19 # Predição dos valores para xgboost sem parâmetros nos dados de teste
20 valores_preditos_xgboost = model_xgboost.predict(X_test)
21 valores_preditos_xgboost
22
23 array([ 45747.688, 11782.867, 15458.265, ..., 115651.945, 14909.92 ,
24        21851.791], dtype=float32)
25
26
27 #-----
28 # Predição dos valores para xgboost com parâmetros nos dados de teste
29 valores_preditos_xgboost_with_parameters = model_xgboost_with_parameters.predict(
30     X_test)
31 valores_preditos_xgboost_with_parameters
32
33 array([ 46349.53 , 11270.773, 15133.414, ..., 116637.836, 14585.994,
34        22141.602], dtype=float32)

```

e. Análise de importância das variáveis.

O código a seguir realiza a análise da importância das variáveis preditoras no desempenho dos

modelos de regressão Random Forest e XGBoost, considerando versões com e sem parametrização. Para isso, são extraídos os valores de importância atribuídos a cada variável durante o treinamento, permitindo identificar quais atributos mais influenciam na previsão do preço médio dos veículos. Essa análise é fundamental para a interpretação dos modelos e para a seleção de variáveis mais relevantes em futuras iterações do processo preditivo.

```

1  ## Analise da importância da variáveis pelo algoritmo random florest (com
    parâmetros)
2  model_rf_with_parameters.feature_importances_
3  feature_importances_with_parameters = pd.DataFrame(model_rf.feature_importances_,
    index = X_train.columns, columns=['importance']).sort_values('importance',
    ascending = False)
4  feature_importances_with_parameters
5  """
6      importance
7  engine_size      0.474341
8  year_model       0.402385
9  cod_gear         0.037446
10 cod_fuel         0.032985
11 cod_brand        0.027031
12 year_of_reference 0.013656
13 cod_month_of_reference 0.012157
14 """
15
16 #-----
17 ## Analise da importância da variáveis pelo algoritmo random florest (sem
    parâmetros)
18 model_rf.feature_importances_
19 feature_importances = pd.DataFrame(model_rf.feature_importances_, index = X_train
    .columns, columns=['importance']).sort_values('importance', ascending = False
    )
20 feature_importances
21 """
22      importance
23 engine_size      0.474341
24 year_model       0.402385
25 cod_gear         0.037446
26 cod_fuel         0.032985
27 cod_brand        0.027031
28 year_of_reference 0.013656

```

```

29 cod_month_of_reference 0.012157
30 """
31
32 #-----
33 ## Analise da importância da variáveis pelo algoritmo xgboost (sem parâmetros)
34 model_xgboost.feature_importances_
35 feature_importances_xgboost = pd.DataFrame(model_xgboost.feature_importances_,
36                                             index = X_train.columns, columns=['importance']).sort_values('importance',
37                                             ascending = False)
38 feature_importances_xgboost
39 """
40
41             importance
42 engine_size      0.395016
43 year_model       0.287108
44 cod_fuel          0.182654
45 cod_gear          0.077372
46 cod_brand         0.030439
47 year_of_reference 0.022804
48 cod_month_of_reference 0.004608
49 """
50
51 #-----
52 ## Analise da importância da variáveis pelo algoritmo xgboost (com parâmetros)
53 model_xgboost_with_parameters.feature_importances_
54 feature_importances_xgboost_with_parameters = pd.DataFrame(model_xgboost_with_
55 parameters.feature_importances_, index = X_train.columns, columns=['
56 importance']).sort_values('importance', ascending = False)
57 feature_importances_xgboost_with_parameters
58
59 """
60             importance
61 engine_size      0.384562
62 year_model       0.320947
63 cod_fuel          0.148609
64 cod_gear          0.081993
65 cod_brand         0.031790
66 year_of_reference 0.025759
67 cod_month_of_reference 0.006339
68 """

```

64 """

f. **Explicação dos resultados encontrados na análise de importância das variáveis.**

Ambos os modelos atribuíram maior importância à cilindrada do motor (*engine size*) e ao ano de fabricação do veículo (*year model*), características tradicionalmente valorizadas no mercado automotivo e que influenciam diretamente no valor de revenda. Observa-se, entretanto, que o modelo XGBoost apresentou uma distribuição mais equilibrada da importância entre as variáveis, enquanto o Random Forest concentrou a relevância em um conjunto mais restrito de atributos, destacando ainda mais as duas variáveis mencionadas.

g. **Escolha do melhor modelo com base nas métricas de avaliação MSE, MAE e R2.**

O código a seguir realiza a avaliação da acurácia dos modelos preditivos treinados, com e sem ajuste de parâmetros, utilizando as métricas *Mean Squared Error* (MSE), *Mean Absolute Error* (MAE) e o coeficiente de determinação (R^2). Essas métricas permitem quantificar o desempenho dos modelos Random Forest e XGBoost na tarefa de previsão do preço médio dos veículos. Os valores obtidos possibilitam comparar a precisão entre os diferentes algoritmos e configurações adotadas, fornecendo subsídios para a escolha do modelo mais adequado à base de dados analisada.

```

1  ## Análise da acurácia do resultado obtido (Random Florest sem parâmetros)
2  """
3
4  MSE - calcula o erro quadrático médio das predições do nosso modelo. Quanto maior
      o MSE, pior é o modelo.
5  MAE - calcula a média da diferença absoluta entre o valor predito e o valor real.
6      Nesse caso, os erros são penalizados linearmente, ou seja, todos terão o
      mesmo peso na média.
7  R^2 - É uma métrica que varia entre 0 e 1 e é uma razão que indica o quão bom o
      nosso modelo.
8      Quanto maior seu valor, melhor é o modelo
9  """
10
11 mse_rf_no_parameter = mean_squared_error(Y_test, valores_preditos_rf).round(2)
12 mae_rf_no_parameter = mean_absolute_error(Y_test, valores_preditos_rf).round(2)
13 r2_rf_no_parameter = r2_score(Y_test, valores_preditos_rf).round(2)
14
15 metricas_acuracia_rf_no_parameter = {"MSE":mse_rf_no_parameter,
16                                     "MAE": mae_rf_no_parameter,
17                                     "R2": r2_rf_no_parameter}
18 metricas_acuracia_rf_no_parameter
19
20 {'MSE': 141495120.03, 'MAE': 5908.15, 'R2': 0.95}

```

```

21
22 #-----
23 ## Análise da acurácia do resultado obtido (Random Florest com parametros)
24 mse_rf_with_parameter = mean_squared_error(Y_test, valores_preditos_rf_with_
    parameters).round(2)
25 mae_rf_with_parameter = mean_absolute_error(Y_test, valores_preditos_rf_with_
    parameters).round(2)
26 r2_rf_with_parameter = r2_score(Y_test, valores_preditos_rf_with_parameters).
    round(2)
27
28 metricas_acuracia_rf_with_parameter = {"MSE":mse_rf_with_parameter,
29                                         "MAE": mae_rf_with_parameter,
30                                         "R2": r2_rf_with_parameter}
31 metricas_acuracia_rf_with_parameter
32
33 {'MSE': 266698714.75, 'MAE': 7000.1, 'R2': 0.9}
34
35
36 #-----
37 ## Análise da acurácia do resultado obtido (xgboost sem parametros)
38 mse_xgboost_no_parameter = mean_squared_error(Y_test, valores_preditos_xgboost).
    round(2)
39 mae_xgboost_no_parameter = mean_absolute_error(Y_test, valores_preditos_xgboost).
    round(2)
40 r2_xgboost_no_parameter = r2_score(Y_test, valores_preditos_xgboost).round(2)
41
42 metricas_acuracia_xgboost_no_parameter = {"MSE":mse_xgboost_no_parameter,
43                                         "MAE": mae_xgboost_no_parameter,
44                                         "R2": r2_xgboost_no_parameter}
45 metricas_acuracia_xgboost_no_parameter
46
47 {'MSE': 109249256.54, 'MAE': 5446.28, 'R2': 0.96}
48
49
50 #-----
51 ## Análise da acurácia do resultado obtido (xgboost com parametros)
52 mse_xgboost_with_parameters = mean_squared_error(Y_test, valores_preditos_xgboost
    _with_parameters).round(2)
53 mae_xgboost_with_parameters = mean_absolute_error(Y_test, valores_preditos_

```



```

    xgboost_with_parameters).round(2)
54 r2_xgboost_with_parameters = r2_score(Y_test, valores_preditos_xgboost_with_
    parameters).round(2)
55
56 metricas_acuracia_xgboost_with_parameters = {"MSE":mse_xgboost_with_parameters,
57                                               "MAE": mae_xgboost_with_parameters,
58                                               "R2": r2_xgboost_with_parameters}
59 metricas_acuracia_xgboost_with_parameters
60
61 {'MSE': 113209989.08, 'MAE': 5757.45, 'R2': 0.96}

```

h. Explicação sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada.

Com base nas métricas apresentadas, o modelo XGBoost sem ajustes de parâmetros foi o que obteve o melhor desempenho na previsão dos preços dos veículos. A avaliação foi feita com as métricas MSE (erro quadrático médio), MAE (erro absoluto médio) e R^2 (coeficiente de determinação). Este modelo alcançou o menor MSE (109 milhões²), o menor MAE (5.446,28) e o maior R^2 (0,96), indicando maior precisão e menor erro em relação aos demais.

APÊNDICE 3 – LINGUAGEM DE PROGRAMAÇÃO APLICADA - R

A – ENUNCIADO

1. Pesquisa com Dados de Satélite (Satellite)

O banco de dados consiste nos valores multispectrais 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 multispectrais.

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:

$$Volume = b0 + b1 * dap^2 * Ht$$

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

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B - RESOLUÇÃO

Bibliotecas utilizadas no trabalho

```
1 # Carregando a biblioteca caret
2 library("caret")
3 library("mlbench")
```

Setando a semente para geração de valores randômicos

```
1 set.seed(7)
```

1. Pesquisa com Dados de Satélite (Satellite)

1. Carregando a base de dados Satellite

```
1 data(Satellite)
2 dados_satelite <- as.data.frame(Satellite)[c("x.17", "x.18", "x.19", "x.20", "
  classes")]
3 head(dados_satelite)
4 #Saída
5 A data.frame:6 x 5
6 | x.17 x.18 x.19 x.20 classes
7 | <dbl> <dbl> <dbl> <dbl> <fct>
8 -----
```

```

9 1 | 92 112 118 85 grey soil
10 2 | 84 103 104 81 grey soil
11 3 | 84 99 104 78 grey soil
12 4 | 84 99 104 81 grey soil
13 5 | 76 99 104 81 grey soil
14 6 | 76 99 108 85 grey soil

```

2. Separando as partições contendo 80% para treino e 20% para teste

```

1 # Separando os índices para treino e teste
2 indices_satelite <- createDataPartition(dados_satelite$classes, p=0.80, list=
  FALSE)
3 satelite_treino <- dados_satelite[indices_satelite, ]
4 satelite_teste <- dados_satelite[-indices_satelite, ]
5 str(satelite_teste)
6
7 #Saída
8 'data.frame': 1284 obs. of 5 variables:
9 $ x.17 : num 84 76 76 88 88 84 84 53 56 75 ...
10 $ x.18 : num 99 99 87 99 99 103 99 54 51 95 ...
11 $ x.19 : num 104 104 91 104 104 108 108 75 67 100 ...
12 $ x.20 : num 78 81 67 85 85 85 85 70 63 79 ...
13 $ classes: Factor w/ 6 levels red soil,cotton crop,...: 3 3 4 3 3 3 3 5 5 4

```

3. Treinando os dados com os modelos RandomForest, SVM e RNA para predição

```

1 # Random Forest
2 model_rf <- train(classes ~., data=satelite_treino, method="rf")
3 predictions_rf <- predict(model_rf, satelite_teste)
4 unique(predictions_rf)
5
6 1. grey soil 2. damp grey soil 3. vegetation stubble 4. cotton crop 5. very damp
  grey soil 6. red soil
7
8 Levels:1. red soil 2. cotton crop 3. grey soil 4. damp grey soil 5. vegetation
  stubble 6. very damp grey soil
9
10 # SVM
11 model_svm <- train(classes ~., data=satelite_treino, method="svmRadial")
12 predictions_svm <- predict(model_svm, satelite_teste)

```

```

13 unique(predictions_svm)
14
15 1. grey soil 2. damp grey soil 3. vegetation stubble 4. very damp grey soil 5.
   cotton crop 6. red soil
16
17 Levels:1. red soil 2. cotton crop 3. grey soil 4. damp grey soil 5. vegetation
   stubble 6. very damp grey soil
18
19
20 # Redes Neurais
21 model_nnet <- train(classes ~., data=satelite_treino, method="nnet", trace=FALSE,
   linout=T)
22 predictions_nnet <- predict(model_nnet, satelite_teste)
23 unique(predictions_nnet)
24 1. grey soil 2. very damp grey soil 3. vegetation stubble 4. cotton crop 5. red
   soil 6. damp grey soil
25
26 Levels:1. red soil 2. cotton crop 3. grey soil 4. damp grey soil 5. vegetation
   stubble 6. very damp grey soil

```

4. Escolhendo o melhor modelo com base em suas matrizes de confusão

```

1 # Matriz de confusão do modelo nnet treinado
2 confusionMatrix(predictions_nnet, satelite_teste$classes)
3
4 Confusion Matrix and Statistics
5
6 Reference
7 Prediction red soil cotton crop grey soil damp grey soil
8
9 red soil 290 1 8 2
10 cotton crop 5 122 0 0
11 grey soil 3 0 262 79
12 damp grey soil 0 0 0 1
13 vegetation stubble 7 9 0 0
14 very damp grey soil 1 8 1 43
15
16 Reference
17 Prediction vegetation stubble very damp grey soil
18

```

```

19 red soil 10 0
20 cotton crop 10 0
21 grey soil 3 41
22 damp grey soil 2 2
23 vegetation stubble 100 2
24 very damp grey soil 16 256
25
26 Overall Statistics
27
28 Accuracy : 0.803
29 95% CI : (0.7801, 0.8244)
30
31 No Information Rate : 0.2383
32 P-Value [Acc > NIR] : < 2.2e-16
33
34 Kappa : 0.7528
35
36 McNemar's Test P-Value : NA
37
38 # Matriz de confusão do modelo svm treinado
39 confusionMatrix(predictions_svm, satelite_teste$classes)
40
41 Confusion Matrix and Statistics
42
43 Reference
44 Prediction red soil cotton crop grey soil damp grey soil
45
46 red soil 298 1 4 2
47 cotton crop 1 120 0 0
48 grey soil 4 0 260 29
49 damp grey soil 0 1 7 69
50 vegetation stubble 3 14 0 2
51 very damp grey soil 0 4 0 23
52
53 Reference
54 Prediction vegetation stubble very damp grey soil
55
56 red soil 7 0
57 cotton crop 4 0

```

```

58 grey soil 1 12
59 damp grey soil 2 32
60 vegetation stubble 117 3
61 very damp grey soil 10 254
62
63 Overall Statistics
64
65 Accuracy : 0.8707
66 95% CI : (0.8511, 0.8886)
67
68 No Information Rate : 0.2383
69 P-Value [Acc > NIR] : < 2.2e-16
70
71 Kappa : 0.8399
72
73 McNemar's Test P-Value : NA
74
75 ## Matriz de confusão do modelo random forest treinado
76 confusionMatrix(predictions_rf, satellite_teste$classes)
77
78 Confusion Matrix and Statistics
79
80 Reference
81 Prediction red soil cotton crop grey soil damp grey soil
82
83 red soil 296 1 5 2
84 cotton crop 0 123 0 0
85 grey soil 3 0 238 29
86 damp grey soil 1 0 21 62
87 vegetation stubble 6 10 0 0
88 very damp grey soil 0 6 7 32
89
90 Reference
91 Prediction vegetation stubble very damp grey soil
92
93 red soil 8 0
94 cotton crop 2 1
95 grey soil 1 10
96 damp grey soil 2 40

```



```

97 vegetation stubble 120 8
98 very damp grey soil 8 242
99
100 Overall Statistics
101
102 Accuracy : 0.8419
103 95% CI : (0.8208, 0.8614)
104
105 No Information Rate : 0.2383
106 P-Value [Acc > NIR] : < 2.2e-16
107
108 Kappa : 0.8047
109
110 McNemar's Test P-Value : NA

```

5. Indicação do melhor modelo

Com base na métrica de Acurácia, o modelo que apresentou o melhor desempenho foi o SVM, alcançando aproximadamente 87%, o que o torna a escolha mais adequada.

2. Estimativa de Volumes de Árvores

Funções utilizadas para calcular o desempenho dos modelos

```

1 # Função para calcular o R2
2 R2 <- function (Y_real, Y_pred){
3   return ( 1 - ( sum((Y_real - Y_pred)^2))/ (sum( (Y_real - mean(Y_real))^2 )))
4 }
5
6 # Função para calcular o erro padrão de estimativa Syx
7 Syx <- function (Y_real, Y_pred){
8   return ( sqrt(sum((Y_real - Y_pred)^2)/(length(Y_real) - 2)))
9 }
10
11 # Função para calcular o erro padrão de estimativa Syx percentual
12 SyxPerc <- function (Y_real, Y_pred){
13   return ( Syx(Y_real, Y_pred)/mean(Y_real) * 100 )
14 }

```

1. Carregando o arquivo Volumes.csv

```

1 dados_volumes <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")
2 str(dados_volumes)
3
4 'data.frame': 100 obs. of 5 variables:
5 $ NR : int 1 2 3 4 5 6 7 8 9 10 ...
6 $ DAP: num 34 41.5 29.6 34.3 34.5 29.9 28.4 29.5 36.3 36.3 ...
7 $ HT : num 27 27.9 26.4 27.1 26.2 ...
8 $ HP : num 1.8 2.75 1.15 1.95 1 1.9 2.3 2.4 1.8 1.5 ...
9 $ VOL: num 0.897 1.62 0.801 1.079 0.98 ...

```

2. Eliminando a coluna NR, que apresenta apenas números sequenciais.

```

1 dados_volumes$NR <- NULL
2 str(dados_volumes)
3
4 'data.frame': 100 obs. of 4 variables:
5 $ DAP: num 34 41.5 29.6 34.3 34.5 29.9 28.4 29.5 36.3 36.3 ...
6 $ HT : num 27 27.9 26.4 27.1 26.2 ...
7 $ HP : num 1.8 2.75 1.15 1.95 1 1.9 2.3 2.4 1.8 1.5 ...
8 $ VOL: num 0.897 1.62 0.801 1.079 0.98 ...

```

3. Separando a base de dados em base de treino e base de teste.

```

1 indices_vol_treino <- createDataPartition(dados_volumes$VOL, p=0.80, list=FALSE)
2 dados_volumes_treino <- dados_volumes[indices_vol_treino, ]
3 dados_volumes_teste <- dados_volumes[-indices_vol_treino, ]
4 str(dados_volumes_treino)
5
6 'data.frame': 80 obs. of 4 variables:
7 $ DAP: num 34 41.5 29.6 34.5 29.9 28.4 29.5 36.3 36.3 28.9 ...
8 $ HT : num 27 27.9 26.4 26.2 27.1 ...
9 $ HP : num 1.8 2.75 1.15 1 1.9 2.3 2.4 1.8 1.5 2.65 ...
10 $ VOL: num 0.897 1.62 0.801 0.98 0.907 ...
11
12 ## Dados para testes
13 str(dados_volumes_teste)
14
15 'data.frame': 20 obs. of 4 variables:
16 $ DAP: num 34.3 32.5 28 33.6 37.8 31.5 32.5 30.5 33.2 29.5 ...
17 $ HT : num 27.1 26.6 23.6 25.1 27.9 ...

```

```

18 $ HP : num 1.95 1.45 2.6 2 2.75 1.6 1.7 2.6 1.7 2.4 ...
19 $ VOL: num 1.079 1.055 0.708 0.967 1.575 ...

```

4. Treinando os modelos: Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR

```

1 # Modelo Random Forest
2 model_rf_volume <- caret::train(VOL~., data=dados_volumes_treino, method="rf")
3
4 note: only 2 unique complexity parameters in default grid. Truncating the grid
5 to 2 .
6
7 # Modelo SVM
8 model_svm_volume <- caret::train(VOL~., data=dados_volumes_treino, method="
  svmRadial")
9
10 # Modelo Redes Neurais
11 model_rn_volume <- caret::train(VOL~., data=dados_volumes_treino, method="nnet",
  linout=T)
12 model_rn_volume
13
14 # Neural Network
15
16 80 samples
17 3 predictor
18
19 No pre-processing
20 Resampling: Bootstrapped (25 reps)
21 Summary of sample sizes: 80, 80, 80, 80, 80, 80, ...
22 Resampling results across tuning parameters:
23
24 size decay RMSE Rsquared MAE
25 1 0e+00 0.3887074 0.5621765 0.3022262
26 1 1e-04 0.3474988 0.5863390 0.2715705
27 1 1e-01 0.1831501 0.8351462 0.1476095
28 3 0e+00 0.3019698 0.6498396 0.2306073
29 3 1e-04 0.2746637 0.7757514 0.2126920
30 3 1e-01 0.1772086 0.8440638 0.1400590
31 5 0e+00 0.2207606 0.7984239 0.1737718
32 5 1e-04 0.1750208 0.8243433 0.1386639

```

```

33 5 1e-01 0.1703584 0.8547071 0.1349266
34
35 RMSE was used to select the optimal model using the smallest value.
36 The final values used for the model were size = 5 and decay = 0.1.
37
38
39 # Modelo Alométrico de SPURR
40 model_spurr_volume <- nls( VOL ~ b0 + b1*DAP*DAP*HT, dados_volumes_treino, start=
    list(b0=0.5, b1=0.5) )

```

5. Efetuando as predições nos modelos treinados.

```

1 # Predição para o modelo Random Forest
2 predicao_rf_volume <- predict(model_rf_volume, dados_volumes_teste)
3 unique(predicao_rf_volume)
4
5 1. 1.280304885677032. 1.092516423976233. 0.8469698986826164. 1.215424575054035.
6 1.539735783964176. 1.105968643792977. 1.081533661178338. 0.981354046716743
7 1.096852570070610. 1.0474671869190611. 1.4425493003824312. 1.74297946393553
8 1.3554234872453314. 1.834641582816915. 1.618666739374316. 1.6902101496133
9 1.38921304666357 1.54507400459013 1.14575277559177 1.74831954645653
10
11
12 # Predição para o modelo SVM
13 predicao_svm_volume <- predict(model_svm_volume, dados_volumes_teste)
14 unique(predicao_svm_volume)
15
16 1. 1.2159091312252. 1.070786980687253. 0.9926504810105984. 1.23592039178091
17 1.535365105957356. 1.027653416284557. 1.098033573314428. 1.00199516922375
18 1.1416452655654610. 1.1382034963676211. 1.5848990336218912. 1.65767236365595
19 1.4088818670610714. 1.9042917349094715. 1.4327614003042416. 1.70744759254471
20 1.31898045040006 1.49113956412507 1.1433007353682 1.52732074672949
21
22 # Predição para o modelo Rede Neural
23 predicao_rn_volume <- predict(model_rn_volume, dados_volumes_teste)
24 unique(predicao_rn_volume)
25
26 1. 1.175793566173662. 1.035180046609573. 0.9489647917115984. 1.22919158959248
27 1.423339189198376. 1.068090646829717. 1.044323003440458. 1.03033146707945
28 1.0927339692604910. 0.8555642504837711. 1.5067189951530312. 1.6308001813253

```

```

29 1.3479204105497714. 1.7456824522419115. 1.5088170460478416. 1.72478140451454
30 1.27218723917318 1.44262811630278 0.96936803543582 1.75464351160939
31
32 # Predição para o modelo alométrico SPURR
33 predicao_spurr_volume <- predict(model_spurr_volume, dados_volumes_teste)
34 unique(predicao_spurr_volume)
35
36 1. 1.261080909249612. 1.115299972351613. 0.7445030798461634. 1.12251988681513
37 1.568367416072556. 0.9753674877016187. 1.125449965100638. 0.952260578765014
38 1.1730026114611910. 0.9565454372424461. 1.3559375283495612. 1.86511801420914
39 1.4514576422654614. 1.9009334347770615. 1.6165885349496416. 1.6465108961535
40 1.37296180248356 1.55452967750829 1.05754992016348 1.52162400160726

```

6. Calculando as métricas entre as predições e os dados observados.

```

1 #Métrica RMSE
2 cat(06.a) Métrica RMSE:\n)
3 cat(rmse rf: ,RMSE(predicao_rf_volume, dados_volumes_teste$VOL))
4 cat(nrmse rn: ,RMSE(predicao_rn_volume, dados_volumes_teste$VOL))
5 cat(nrmse svm: ,RMSE(predicao_svm_volume, dados_volumes_teste$VOL))
6 cat(nrmse spurr: ,RMSE(predicao_spurr_volume, dados_volumes_teste$VOL))
7
8 06.a) Métrica RMSE:
9 rmse rf: 0.1371347
10 rmse rn: 0.1235196
11 rmse svm: 0.1395021
12 rmse spurr: 0.1493509
13
14
15 #Métrica R2
16 cat(06.b) Métrica R2:)
17 cat(r2 rf: ,R2(predicao_rf_volume, dados_volumes_teste$VOL))
18 cat(r2 rn: ,R2(predicao_rn_volume, dados_volumes_teste$VOL))
19 cat(r2 svm: ,R2(predicao_svm_volume, dados_volumes_teste$VOL))
20 cat(r2 spurr: ,R2(predicao_spurr_volume, dados_volumes_teste$VOL))
21
22 06.b) Métrica R2:
23 r2 rf: 0.8499875
24 r2 rn: 0.8645165
25 r2 svm: 0.8186649

```

```

26 r2 spurr: 0.8503353
27
28 #Métrica Syx
29 cat(06.c) Métrica Syx:)
30 cat(Syx rf: ,Syx(predicao_rf_volume, dados_volumes_teste$VOL))
31 cat(Syx rn: ,Syx(predicao_rn_volume, dados_volumes_teste$VOL))
32 cat(Syx svm: ,Syx(predicao_svm_volume, dados_volumes_teste$VOL))
33 cat(Syx spurr: ,Syx(predicao_spurr_volume, dados_volumes_teste$VOL))
34
35 06.c) Métrica Syx:
36 Syx rf: 0.1445527
37 Syx rn: 0.1302011
38 Syx svm: 0.1470481
39 Syx spurr: 0.1574296
40
41 #Métrica Syx percentual
42 cat(06.c) Métrica Syx percentual:)
43 cat(Syx percentual rf: ,SyxPerc(predicao_rf_volume, dados_volumes_teste$VOL))
44 cat(\nSyx percentual rn: ,SyxPerc(predicao_rn_volume, dados_volumes_teste$VOL))
45 cat(\nSyx percentual svm: ,SyxPerc(predicao_svm_volume, dados_volumes_teste$VOL))
46 cat(\nSyx percentual spurr: ,SyxPerc(predicao_spurr_volume, dados_volumes_teste$
    VOL))
47
48 06.c) Métrica Syx percentual:
49 Syx percentual rf: 10.9554
50 Syx percentual rn: 9.946558
51 Syx percentual svm: 11.00145
52 Syx percentual spurr: 11.83288

```

7. Escolha do melhor modelo.

A seleção do modelo mais adequado baseou-se na análise conjunta de quatro métricas de desempenho: RMSE (Root Mean Square Error), R^2 (coeficiente de determinação), Syx (erro padrão da estimativa) e Syx percentual. O modelo de Redes Neurais (*nnet*) apresentou os melhores resultados em todas essas métricas quando comparado aos demais modelos avaliados (Random Forest, SVM e o modelo empírico de Spurr).

Especificamente, o modelo *nnet* obteve o menor RMSE (0,1235), indicando menor dispersão dos erros quadráticos médios. Além disso, atingiu o maior valor de R^2 (0,8645), evidenciando uma maior capacidade explicativa da variabilidade dos dados observados. Também apresentou

o menor Syx (0,1302) e o menor Syx percentual (9,95%), o que reforça sua superioridade em termos de precisão e acurácia.

Esses resultados demonstram que o modelo de Redes Neurais não apenas minimiza os erros de predição, como também mantém uma alta correlação com os valores reais, sendo, portanto, a escolha mais apropriada para a estimativa de volumes neste estudo.

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1. Gráficos e tabelas

- a) **(15 pontos)** Elaborar os gráficos box-plot e histograma das variáveis "age"(idade da esposa) e "husage"(idade do marido) e comparar os resultados.
- b) **(15 pontos)** Elaborar a tabela de frequências das variáveis "age"(idade da esposa) e "husage"(idade do marido) e comparar os resultados

2. Medidas de posição e dispersão

- a) **(15 pontos)** Calcular a média, mediana e moda das variáveis "age"(idade da esposa) e "husage"(idade do marido) e comparar os resultados.
- b) **(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

- a) **(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.

Observação:

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

B - RESOLUÇÃO

O código-fonte abaixo apresenta a preparação do ambiente para a análise estatística descritiva das variáveis `age` (idade da esposa) e `husage` (idade do marido), com base nos dados contidos no conjunto `salarios`. Foram utilizadas bibliotecas do ambiente R especializadas em manipulação de dados, geração de gráficos e cálculo de estatísticas descritivas. O objetivo principal é realizar a construção de gráficos (boxplot e histograma), tabelas de frequência, e o cálculo de medidas de tendência central e dispersão, conforme solicitado. A seguir, é apresentado o código responsável pela importação das bibliotecas necessárias e o carregamento da base de dados.


```

1  ## Bibliotecas utilizadas no trabalho
2  library(ggplot2)
3  library(tidyverse)
4  library(plotly)
5  library (lattice)
6  library(cowplot)
7  library(fdth)
8
9  # Carregar a base de dados "salarios" em formato R
10 load("../dados/salarios.RData")

```

1. Gráficos e tabelas

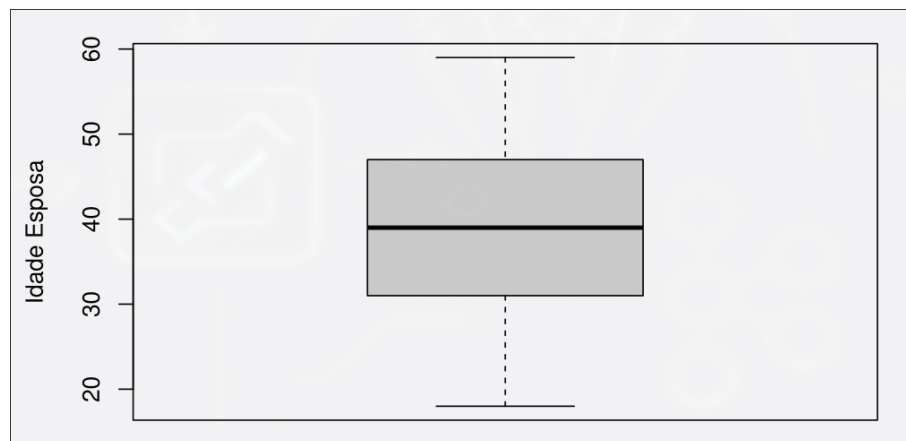
- a) Gráficos box-plot e histograma das variáveis "age"(idade da esposa) e "husage"(idade do marido).

```

1  boxplot(salarios$age,
2          main=Boxplot idade esposa,
3          ylab=Idade Esposa)
4

```

FIGURA 11 – BOXPLOT IDADE ESPOSA.



FONTE: Elaborado pelo autor a partir da base de dados salarios.RData (2024).

```

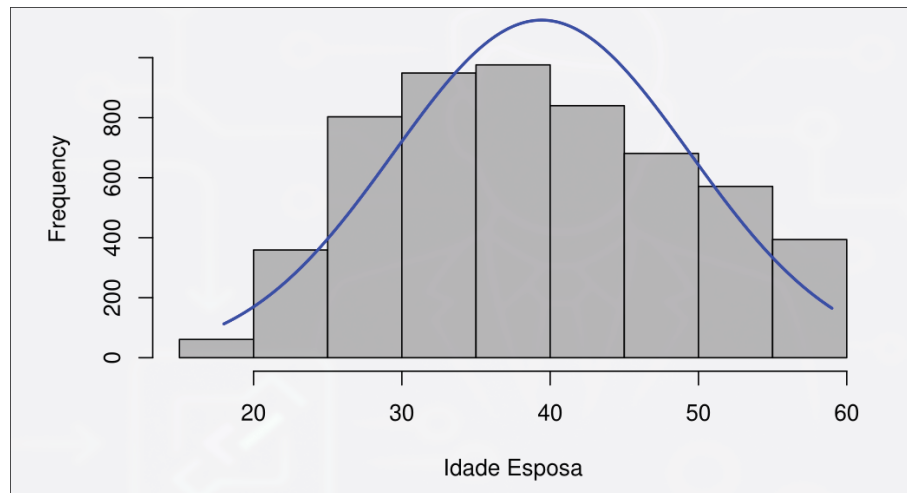
1  # Plotar o histograma com a curva normal para a variável "age"
2  plotNormalHistogram(salarios$age,
3                      prob = FALSE,
4                      xlab=Idade Esposa,
5                      main = Curva normal sobre o histograma,

```

```
6     length = 1000 )
```

```
7
```

FIGURA 12 – HISTOGRAMA IDADE ESPOSA.



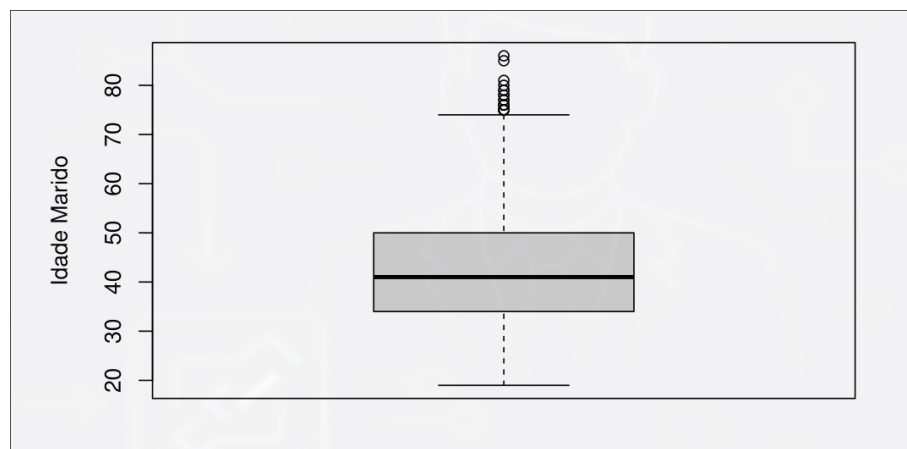
FONTE: Elaborado pelo autor a partir da base de dados `salarios.RData` (2024).

```
1 # Plotar o boxplot para a variável "husage"(idade do marido)
2 boxplot(salarios$husage,
3         main=Boxplot idade do c njuge,
4         ylab=Idade Marido)
```

```
5
```

```
6
```

FIGURA 13 – BOXPLOT IDADE C NJUGE.



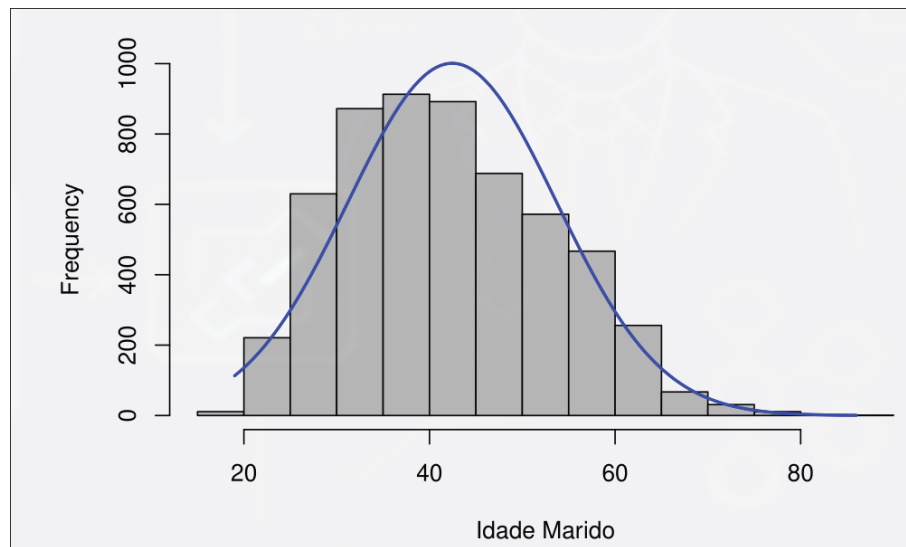
FONTE: Elaborado pelo autor a partir da base de dados `salarios.RData` (2024).

```

1 # Plotar histograma com a curva normal para a variável "husage"
2 plotNormalHistogram(salarios$husage, prob = FALSE,
3   xlab=Idade Marido,
4   main = Curva normal sobre o histograma,
5   length = 1000 )
6
7

```

FIGURA 14 – HISTOGRAMA IDADE CÔNJUGE.



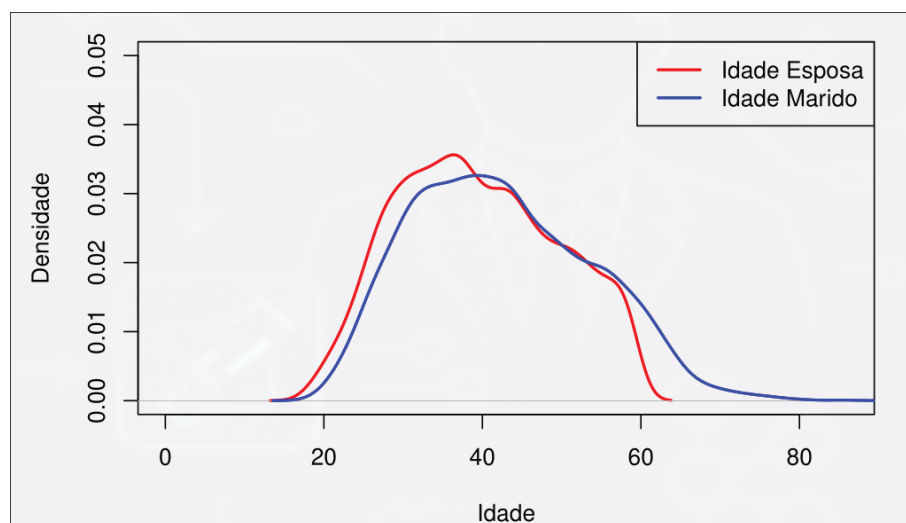
FONTE: Elaborado pelo autor a partir da base de dados salarios.RData (2024).

```

1 # Plotar gráfico de densidade
2 plot(density(salarios$age),
3   main=Comparação da Idade Esposa e Marido,
4   xlim=c(0, max(salarios$age, salarios$husage)),
5   ylim=c(0, 0.05),
6   xlab="Idade",
7   ylab="Densidade",
8   col="red",
9   lwd=2)
10 lines(density(salarios$husage), col="blue", lwd=2)
11 legend("topright", legend=c(Idade Esposa, Idade Marido),
12   col=c("red", "blue"),
13   lwd=2)
14

```

FIGURA 15 – COMPARAÇÃO DAS IDADES DA ESPOSA E DO CÔNJUGE.



FONTE: Elaborado pelo autor a partir da base de dados salarios.RData (2024).

b) **Tabelas de frequências das variáveis "age"(idade da esposa) e "husage"(idade do marido) e comparação dos resultados**

```
1 # Cálculo da distribuição de frequência
2 freq_idade_wife <- fdt(salarios$age)
3 freq_idade_husband <- fdt(salarios$husage)
4 print(freq_idade_wife)
5 print(freq_idade_husband)
```

TABELA 3 – DISTRIBUIÇÃO DE FREQUÊNCIA DAS IDADES DAS ESPOSAS

Class limits	f	rf	rf(%)	cf	cf(%)
17,82 – 20,804	61	0,01	1,08	61	1,08
20,804 – 23,787	161	0,03	2,86	222	3,94
23,787 – 26,771	312	0,06	5,54	534	9,48
26,771 – 29,754	505	0,09	8,96	1039	18,44
29,754 – 32,738	562	0,10	9,98	1601	28,42
32,738 – 35,721	571	0,10	10,13	2172	38,55
35,721 – 38,705	624	0,11	11,08	2796	49,63
38,705 – 41,689	510	0,09	9,05	3306	58,68
41,689 – 44,672	542	0,10	9,62	3848	68,30
44,672 – 47,656	432	0,08	7,67	4280	75,97
47,656 – 50,639	389	0,07	6,90	4669	82,87
50,639 – 53,623	358	0,06	6,35	5027	89,23
53,623 – 56,606	304	0,05	5,40	5331	94,62
56,606 – 59,590	303	0,05	5,38	5634	100,00

FONTE: Elaborado pelo autor a partir da base de dados salarios.

TABELA 4 – DISTRIBUIÇÃO DE FREQUÊNCIA DAS IDADES DOS MARIDOS

Class limits	f	rf	rf(%)	cf	cf(%)
18,81, – 23,671	102	0.02	1.81	102	1.81
23,671, – 28,531	466	0.08	8.27	568	10.08
28,531, – 33,392	809	0.14	14.36	1377	24.44
33,392, – 38,253	895	0.16	15.89	2272	40.33
38,253, – 43,114	917	0.16	16.28	3189	56.60
43,114, – 47,974	629	0.11	11.16	3818	67.77
47,974, – 52,835	649	0.12	11.52	4467	79.29
52,835, – 57,696	541	0.10	9.60	5008	88.89
57,696, – 62,556	394	0.07	6.99	5402	95.88
62,556, – 67,417	152	0.03	2.70	5554	98.58
67,417, – 72,278	51	0.01	0.91	5605	99.49
72,278, – 77,139	21	0.00	0.37	5626	99.86
77,139, – 81,999	6	0.00	0.11	5632	99.96
81,999, – 86,86	2	0.00	0.04	5634	100.00

FONTE: Elaborado pelo autor a partir da base de dados `salarios`.

Ao analisar as distribuições de frequência das idades das esposas e dos maridos na base de dados estudada, observa-se que aproximadamente 60% dos maridos têm entre 28 e 44 anos, o que corresponde ao intervalo [28.531, 43.114). Para as esposas, uma proporção semelhante (60%) está concentrada no intervalo de idade entre 26 e 44 anos, correspondente ao intervalo [26.771, 44.672). No entanto, esse intervalo para as esposas está dividido em um número maior de classes, o que indica uma distribuição etária mais dispersa. Isso sugere uma maior diversidade nas idades das esposas em comparação aos maridos, cuja distribuição apresenta faixas etárias mais concentradas.

2. Medidas de posição e dispersão

a) Média, mediana e moda das variáveis "age"(idade da esposa) e "husage"(idade do marido)

Nesta etapa, são determinadas as medidas de posição (média, mediana e moda) das variáveis `age` (idade da esposa) e `husage` (idade do marido). Tais medidas são utilizadas para descrever a tendência central dos dados e permitir a comparação entre os dois grupos.

A função para o cálculo da moda foi implementada manualmente, uma vez que essa medida não está disponível de forma nativa na linguagem R. Após o cálculo da média, mediana e moda para cada variável, os resultados foram organizados em uma tabela. Além disso, foi calculada a diferença percentual entre os valores correspondentes das duas variáveis, com o objetivo de quantificar possíveis variações entre os perfis etários dos cônjuges.

```

1 # Função para calcular a moda de um conjunto de valores
2 moda <- function(valores){
3   uniq_val <- unique(valores)
4   uniq_val[which.max(tabulate(match(valores, uniq_val)))]
5 }
6 # Idades dos maridos e das esposas
7 idades_wife <- salarios$age

```

```

8 idades_husband <- salarios$husage
9
10 #Cálculo das medidas de posicao para as esposas
11 medidas_posicao_wife <- c(mean(idades_wife), median(idades_wife), moda(idades_
    wife))
12 medidas_posicao_husband <- c(mean(idades_husband), median(idades_husband), moda(
    idades_husband))
13 medidas_posicao <- data.frame(c("media", "mediana", "moda"),medidas_posicao_wife,
    medidas_posicao_husband)
14 colnames(medidas_posicao ) <- c("medida", "wife", "husband")
15 medidas_posicao$perc_husb_wife <- (((medidas_posicao$husband/medidas_posicao$wife
    ) - 1 ) * 100)
16
17

```

TABELA 5 – MEDIDAS DE TENDÊNCIA CENTRAL DAS IDADES DOS CÔNJUGES

Medida	Esposas	Maridos	Diferença (%)
Média	39,43	42,45	7,67%
Mediana	39,00	41,00	5,13%
Moda	37,00	44,00	18,92%

FONTE: Elaborado pelo autor a partir da base de dados salarios.

Ao analisarmos as medidas de tendência central da variável "idade" para maridos e esposas, observa-se que, na base de dados estudada, a idade média dos maridos (42,45 anos) é aproximadamente 8% superior à das esposas (39,43 anos). A mediana das idades também segue essa tendência, sendo ligeiramente superior para os maridos (41 anos) em relação às esposas (39 anos), com uma diferença proporcional de cerca de 5%. No que diz respeito à moda, a diferença é ainda mais acentuada: a idade mais frequente entre os maridos é de 44 anos, enquanto entre as esposas é de 37 anos — um valor cerca de 19% menor. Esses resultados indicam que, de modo geral, os maridos tendem a ser mais velhos que as esposas, tanto em média quanto nas medidas de posição mais representativas da distribuição.

b) Variância, desvio padrão e coeficiente de variação das variáveis "age"(idade da esposa) e "husage"(idade do marido)

Nesta seção, são calculadas as principais medidas de dispersão das variáveis age (idade da esposa) e husage (idade do marido), com o objetivo de avaliar o grau de variabilidade dos dados em relação à média. As métricas utilizadas foram: variância (VAR), desvio padrão (SD) e coeficiente de variação (CVE).

O cálculo foi realizado utilizando a linguagem R. A variância e o desvio padrão foram obtidos por meio das funções estatísticas padrão da linguagem. O coeficiente de variação foi calculado como a razão entre o desvio padrão e a média da variável, multiplicada por 100, representando a

dispersão relativa em termos percentuais. Ao final, os resultados foram organizados em uma tabela, e a diferença percentual entre os valores dos maridos e das esposas também foi computada para efeito de comparação.

```

1 # Medida de dispersao das idades dos maridos e das esposas
2 coeficiente_variacao_wife <- (sd(idades_wife)/mean(idades_wife)) * 100
3 coeficiente_variacao_husband <- (sd(idades_husband)/mean(idades_husband)) * 100
4 medidas_dispersao_wife <- c(var(idades_wife), sd(idades_wife), coeficiente_
    variacao_wife)
5 medidas_dispersao_husband <- c(var(idades_husband), sd(idades_husband),
    coeficiente_variacao_husband)
6
7 # DataFrame com os dados
8 medidas_dispersao <- data.frame(c("VAR", "SD", "CVE"), medidas_dispersao_wife,
    medidas_dispersao_husband)
9 colnames(medidas_dispersao) <- c("medida", "wife", "husband")
10 medidas_dispersao$perc_husb_wife <- (((medidas_dispersao$husband/medidas_
    dispersao$wife) - 1) * 100)
11

```

TABELA 6 – MEDIDAS DE DISPERSÃO DAS IDADES DOS CÔNJUGES

Medida	Esposas	Maridos	Diferença (%)
Variância (VAR)	99,75	126,07	26,38%
Desvio padrão (SD)	9,99	11,23	12,42%
Coef. de Variação (CVE)	25,33%	26,45%	4,41%

FONTE: Elaborado pelo autor a partir da base de dados salarios.

Com relação às medidas de dispersão das idades, verifica-se que a variância entre os maridos (126,07) é cerca de 26% maior do que entre as esposas (99,75), indicando uma maior variabilidade nas idades masculinas. O desvio padrão acompanha esse padrão, sendo também mais elevado entre os maridos (11,23 anos) do que entre as esposas (9,99 anos), com uma diferença proporcional de aproximadamente 12%.

No entanto, ao analisarmos o coeficiente de variação (CVE) — que permite comparar a dispersão relativa entre as distribuições — observa-se que as idades das esposas apresentam, ligeiramente, uma menor homogeneidade (CVE de 25,33%) em comparação às idades dos maridos (CVE de 26,45%). Essa diferença, de aproximadamente 4%, sugere que, proporcionalmente à média, as idades dos maridos são um pouco mais dispersas.

3. Testes paramétricos ou não paramétricos

Nesta seção, é realizada a análise comparativa entre as idades das esposas (age) e dos maridos (husage), por meio de testes estatísticos paramétricos. Considerando o tamanho da amostra e a disponibilidade da variância populacional, opta-se pela aplicação do teste z para comparação de médias.

Os dados das variáveis de idade foram combinados em uma única estrutura, com a inclusão de uma variável categórica indicando o grupo (esposa ou marido) ao qual cada observação pertence. Em seguida, são calculadas as médias, os desvios padrão e outras estatísticas descritivas de cada grupo. Também foi incluída a representação gráfica da distribuição normal sobre o histograma, com o objetivo de avaliar visualmente a aderência dos dados à normalidade.

a) **Testar se as médias ou as medianas das variáveis "age"(idade da esposa) e "husage"(idade do marido) são iguais**

```

1 # Tendo em vista que temos como obter a variância populacional e temos uma grande
2 # amostra aplicamos o teste z
3 # empilhando as colunas age e husage (converter para formato long.)
4 # e criar as categoria marido e esposa na variável grupo
5 idade_emp <- data.frame(
6   grupo = rep(c("Esposa", "Marido"), each = nrow(salarios)),
7   idade = c(salarios$age, salarios$husage))
8 head(idade_emp, 3)
9
10 idade_emp <- data.frame(
11   grupo = rep(c("Esposa", "Marido"), each = nrow(salarios)),
12   idade = c(salarios$age, salarios$husage))
13
14 # Calculamos a média da idade geral
15 mean(idade_emp$idade)
16 ## [1] 40.94027
17 # média da idade geral 40.94
18
19 # Calculamos as estatísticas descritivas por grupo, Esposas e Maridos.
20 group_by(idade_emp, grupo) %>%
21   summarise(
22     count = n(),
23     mean = mean(idade, na.rm = TRUE),
24     sd = sd(idade, na.rm = TRUE)
25   )
26 ## # A tibble: 2 x 4
27 #       grupo    count    mean    sd
28 #   <chr>    <int>    <dbl>    <dbl>
29 ## 1   Esposa   5634     39.4     9.99
30 ## 2   Marido   5634     42.5    11.2
31
32 # Calculamos o desvio padrão de toda a amostra

```

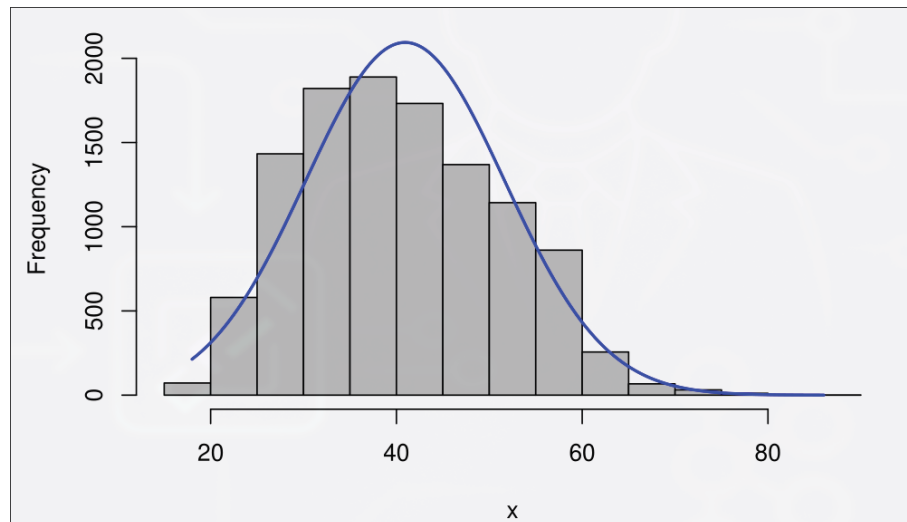


```

33 sd(idade_emp$idade)
34 ## [1] 10.73268
35 # Desvio padrão geral = 10.73
36
37 # Plotamos a curva normal sobre o histograma:
38 plotNormalHistogram(idade_emp$idade,
39   prob = FALSE,
40   main = Normal Distribution overlay on Histogram,
41   length = 1000 )
42

```

FIGURA 16 – HISTOGRAMA IDADES (ESPOSAS E MARIDOS) EMPILHADAS



FONTE: Elaborado pelo autor a partir da base de dados salarios.

```

1 # Executamos o teste de normalidade de Kolmogorov-Smirnov
2 lillie.test(idade_emp$idade)
3 ## Lilliefors (Kolmogorov-Smirnov) normality test
4 ## data: idade_emp$idade
5 ## D = 0.060083, p-value < 0.00000000000000022
6

```

A aplicação do teste de normalidade de Lilliefors (uma adaptação do teste de Kolmogorov-Smirnov) à variável *idade* empilhada resultou em um valor de $D = 0,0601$ com um p-valor praticamente nulo ($p < 2,2 \times 10^{-16}$).

Como o p-valor é significativamente inferior ao nível de significância usual de 0,05, rejeita-se a hipótese nula de que os dados seguem uma distribuição normal.

Dessa forma, não é adequado utilizar testes paramétricos que pressupõem normalidade, como o teste- z ou o teste- t . Nesse contexto, recomenda-se a adoção de testes não paramétricos, que são mais apropriados para distribuições que não atendem aos pressupostos da estatística paramétrica, como temos duas amostras, utilizaremos o teste de Mann-Whitney "U"test.

a.1) **Primeiro Teste: se a idade mediana dos maridos é igual a das esposas.** O teste de Mann-Whitney U foi aplicado para comparar as idades entre maridos e esposas, considerando que a suposição de normalidade foi previamente rejeitada. As hipóteses consideradas foram:

- H_0 : A mediana das idades dos maridos é estatisticamente igual à mediana das idades das esposas;
- H_a : A mediana das idades dos maridos é estatisticamente diferente da mediana das idades das esposas.

```

1 res <- wilcox.test(idade ~ grupo, data=idade_emp,
2   exact = FALSE,
3   conf.int=TRUE)
4 res
5 ## Wilcoxon rank sum test with continuity correction
6 ## data: idade by grupo
7 ## W = 13619912, p-value < 0.00000000000000022
8 ## alternative hypothesis: true location shift is not equal to 0
9 ## 95 percent confidence interval:
10 ## -3.000024 -2.000033
11 ## sample estimates:
12 ## difference in location -2.99996

```

O teste resultou em um valor de $W = 13.619.912$ com p-valor extremamente pequeno ($p < 2,2 \times 10^{-16}$), o que nos leva a rejeitar a hipótese nula ao nível de significância de 5%. O intervalo de confiança de 95% para a diferença entre as medianas está entre $-3,00$ e $-2,00$, indicando que, em média, as esposas são de 2 a 3 anos mais jovens que os maridos.

Dessa forma, conclui-se que há evidência estatística significativa para afirmar que as idades medianas dos maridos e das esposas diferem, com os maridos sendo, em média, mais velhos.

a.2) **Segundo Teste: se a idade mediana dos maridos é menor a das esposas.** No segundo teste, foi aplicada novamente a versão unilateral do teste de Mann-Whitney U para verificar se a idade mediana dos maridos é estatisticamente menor do que a das esposas. As hipóteses consideradas foram:

- H_0 : A idade mediana dos maridos é estatisticamente MENOR a idade mediana das esposas.
- H_a : A idade mediana dos maridos não é estatisticamente MENOR a idade mediana das esposas.

```

1 wilcox.test(idade ~ grupo, data = idade_emp,
2     exact = FALSE, alternative = "less",
3     conf.int = TRUE)
4 ## Wilcoxon rank sum test with continuity correction
5 ## data: idade by grupo
6 ## W = 13619912, p-value < 0.00000000000000022
7 ## alternative hypothesis: true location shift is less than 0
8 ## 95 percent confidence interval:
9 ## -Inf -2.000046
10 ## sample estimates:
11 ## difference in location -2.999966

```

O teste resultou em um valor de $W = 13.619.912$ com p-valor inferior a $2,2 \times 10^{-16}$, o que leva à rejeição da hipótese nula ao nível de significância de 5%. O intervalo de confiança de 95% para a diferença entre as medianas foi $(-\infty, -2,00)$, indicando uma diferença negativa estatisticamente significativa.

Portanto, os resultados não apenas rejeitam a hipótese de que os maridos sejam, em média, mais jovens, como reforçam a conclusão de que a idade mediana dos maridos é significativamente maior do que a das esposas.

a.3) **Terceiro Teste: se a idade mediana dos maridos é maior a das esposas.** Neste terceiro teste, foi utilizado o teste de Mann-Whitney U com hipótese alternativa unilateral, com o objetivo de verificar se a mediana das idades dos maridos é estatisticamente maior do que a mediana das idades das esposas. As hipóteses formuladas foram:

- H_0 : A idade mediana dos maridos é estatisticamente MAIOR a idade mediana das esposas.
- H_a : A idade mediana dos maridos não é estatisticamente MAIOR a idade mediana das esposas.

```

1 wilcox.test(idade ~ grupo, data = idade_emp,
2     exact = FALSE, alternative = "greater",
3     conf.int = TRUE)
4 ## Wilcoxon rank sum test with continuity correction
5 ## data: idade by grupo
6 ## W = 13619912, p-value = 1
7 ## alternative hypothesis: true location shift is greater than 0
8 ## 95 percent confidence interval:
9 ## -3.000034 Inf
10 ## sample estimates:
11 ## difference in location -2.999966
12
13

```

O teste resultou em um valor de $W = 13.619.912$, com p-valor igual a 1. Este valor representa ausência total de evidência estatística para rejeitar a hipótese nula, o que significa que não há qualquer suporte para afirmar que a idade mediana dos maridos seja maior que a das esposas.

O intervalo de confiança de 95% para a diferença entre as medianas foi $(-3,00, \infty)$, o que indica que essa diferença pode inclusive ser negativa. Assim, a hipótese de que os maridos sejam mais velhos do que as esposas, no sentido estritamente estatístico proposto por este teste unilateral, **não se sustenta**.

Conclui-se, portanto, que não há evidência estatística para afirmar que a mediana das idades dos maridos seja maior do que a das esposas.

TABELA 7 – RESUMO DOS TESTES DE HIPÓTESE PARA COMPARAÇÃO DAS IDADES

Teste	Hipótese nula (H_0)	Hipótese alternativa (H_a)	p-valor	Decisão e interpretação
1 (bilateral)	A idade mediana dos maridos é igual à das esposas	A idade mediana dos maridos é diferente da das esposas	$< 2,2 \times 10^{-16}$	Rejeita-se H_0 ; há diferença significativa entre as medianas
2 (unilateral – menor)	A idade mediana dos maridos é menor que a das esposas	A idade mediana dos maridos não é menor que a das esposas	$< 2,2 \times 10^{-16}$	Rejeita-se H_0 ; maridos não são mais jovens que as esposas
3 (unilateral – maior)	A idade mediana dos maridos é maior que a das esposas	A idade mediana dos maridos não é maior que a das esposas	1	Não se rejeita H_0 ; não há evidência de que maridos sejam estatisticamente mais velhos

FONTE: Elaborado pelo autor a partir da base de dados salarios.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

1. Regressões Ridge, Lasso e ElasticNet

- a) **(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 R2) 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)

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

B - RESOLUÇÃO

1. Regressões Ridge, Lasso e ElasticNet

Esta seção tem como objetivo aplicar e comparar três modelos de regressão penalizada — Ridge, Lasso e ElasticNet — com a variável dependente `lwage`, que representa o salário-hora da esposa em logaritmo neperiano. Todas as demais variáveis disponíveis no conjunto de dados são consideradas variáveis explicativas. Os modelos são avaliados com base no erro quadrático médio (RMSE) e no coeficiente de determinação (R^2), sendo escolhida a abordagem que apresentar melhor desempenho preditivo. Por fim, é realizada uma previsão com base em um conjunto específico de valores das variáveis explicativas, e os resultados são apresentados com intervalos de confiança. Como a variável dependente está em escala logarítmica, é necessário aplicar a exponenciação ao final para interpretar os valores previstos em sua escala original.

A seguir, apresenta-se o código-fonte implementado em linguagem R para construção dos modelos de regressão Ridge, Lasso e ElasticNet. O conjunto de dados é preparado por meio de limpeza, padronização e transformação das variáveis categóricas e contínuas. Em seguida, são realizados procedimentos de visualização de correlações, separação entre bases de treino e teste e preparação para a modelagem. Esta etapa é fundamental para garantir consistência estatística e reprodutibilidade dos resultados na análise dos salários-hora das esposas com base em múltiplas variáveis explicativas.

```
1 # Carregamento das bibliotecas necessárias para manipulação de dados e modelagem
2 library(readr)
3 library(dplyr)
4 library(caret)
5 library(repr)
6 library(glmnet)
7 library(ggplot2)
8 library(tidyverse)
9
10 # Armazena o conjunto de dados no objeto 'dados_ridge'
11 dados_ridge <- trabalhosalarios
12
13 # Visualiza a estrutura do conjunto de dados
14 glimpse(dados_ridge)
15
16 # Remove a variável 'earns' que não será utilizada
17 dados_ridge <- dados_ridge %>% select(-earns)
18
19 # Calcula a matriz de correlação entre todas as variáveis
20 correlation_matrix <- cor(dados_ridge)
21
22 # Extrai as correlações entre 'lwage' e as demais variáveis
```

```

23 cor_with_lwage <- data.frame(
24   variable = rownames(correlation_matrix),
25   correlation = correlation_matrix[, "lwage"]
26 )
27
28 # Remove a correlação de 'lwage' consigo mesma
29 cor_with_lwage <- cor_with_lwage[cor_with_lwage$variable != "lwage", ]
30
31 # Ordena as variáveis pela magnitude da correlação com 'lwage'
32 cor_with_lwage <- cor_with_lwage[order(abs(cor_with_lwage$correlation),
33   decreasing = TRUE),]
34
35 # Gera gráfico de barras com as correlações
36 ggplot(cor_with_lwage, aes(x = reorder(variable, correlation), y = correlation))
37   +
38   geom_bar(stat = "identity", fill = "skyblue") +
39   labs(title = "Correlação das variáveis com lwage",
40     x = "Variável",
41     y = "Correlação") +
42   theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
43   coord_flip()
44
45 # Define semente para reprodutibilidade dos resultados
46 set.seed(10)
47
48 # Converte variáveis categóricas para fator
49 cols_dummy <- c(2,5,6,8,10,12,13,15)
50 dados_ridge[,cols_dummy] <- lapply(dados_ridge[,cols_dummy], factor)
51
52 # Separa os dados em conjuntos de treino (80%) e teste (20%)
53 index = sample(1:nrow(dados_ridge), 0.8 * nrow(dados_ridge))
54 treinamento_ridge = dados_ridge[index,]
55 teste_ridge = dados_ridge[-index,]
56
57 # Visualiza as dimensões dos conjuntos de treino e teste
58 dim(treinamento_ridge)
59 dim(teste_ridge)
60
61 # Aplica exponencial à variável resposta para retornar à escala original

```

```

60 treinamento_ridge$lwage <- exp(treinamento_ridge$lwage)
61 teste_ridge$lwage <- exp(teste_ridge$lwage)
62
63 # Padroniza variáveis contínuas no conjunto de treino
64 trein_padr_ridge <- treinamento_ridge %>% mutate_at(c('husage',
65                                                         'husearns',
66                                                         'huseduc',
67                                                         'hushrs',
68                                                         'age',
69                                                         'educ',
70                                                         'exper',
71                                                         'lwage'),
72                                                         ~(scale(.) %>% as.vector))
73
74 # Padroniza variáveis contínuas no conjunto de teste
75 test_padr_ridge <- teste_ridge %>% mutate_at(c('husage',
76                                                         'husearns',
77                                                         'huseduc',
78                                                         'hushrs',
79                                                         'age',
80                                                         'educ',
81                                                         'exper',
82                                                         'lwage'),
83                                                         ~(scale(.) %>% as.vector))
84
85 # Prepara os dados para análise de correlação
86 dat_cor_ridge <- trein_padr_ridge
87 dat_cor_ridge$husunion <- as.numeric(dat_cor_ridge$husunion)
88 dat_cor_ridge$husblk <- as.numeric(dat_cor_ridge$husblk)
89 dat_cor_ridge$hushisp <- as.numeric(dat_cor_ridge$hushisp)
90 dat_cor_ridge$kid6 <- as.numeric(dat_cor_ridge$kid6)
91 dat_cor_ridge$black <- as.numeric(dat_cor_ridge$black)
92 dat_cor_ridge$hispanic <- as.numeric(dat_cor_ridge$hispanic)
93 dat_cor_ridge$union <- as.numeric(dat_cor_ridge$union)
94 dat_cor_ridge$kidlt6 <- as.numeric(dat_cor_ridge$kidlt6)
95
96 # Calcula a matriz de correlação para os dados padronizados
97 corr_ridge <- cor(dat_cor_ridge)
98

```



```

99 # Extrai a correlação das variáveis com 'lwage'
100 corr_with_lwage_ridge <- corr_ridge["lwage", ]
101
102 # Remove a autocorrelação da variável dependente
103 corr_with_lwage_ridge <- corr_with_lwage_ridge[!names(corr_with_lwage_ridge) == "
    lwage"]
104
105 # Gera gráfico de barras com correlação de cada variável com 'lwage'
106 barplot(corr_with_lwage_ridge,
107         main = "Correlação com lwage",
108         xlab = "Variáveis",
109         ylab = "Correlação")

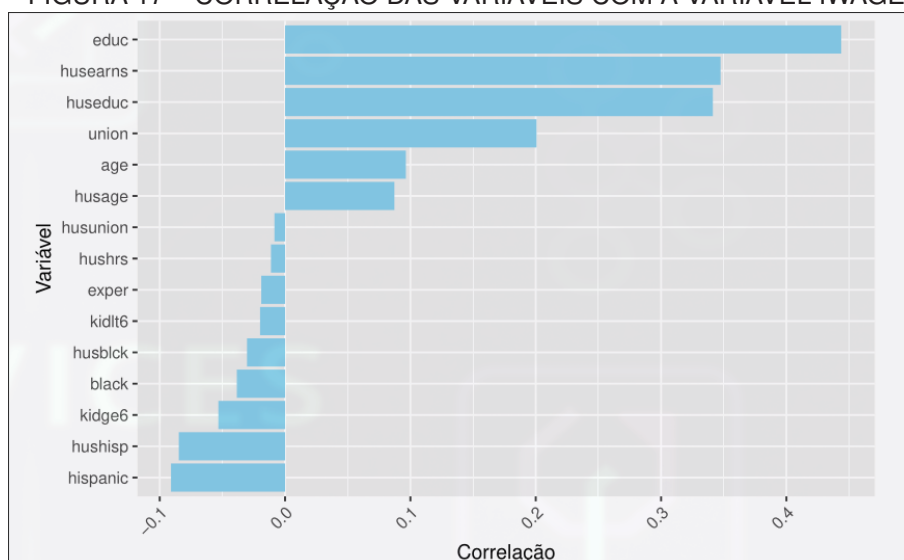
```

TABELA 8 – DESCRIÇÃO DAS VARIÁVEIS DO CONJUNTO DE DADOS DADOS_RIDGE.

Variável	Tipo / Exemplos
husage	Idade do marido (ex.: 56, 31, 33)
husunion	União estável do marido (0 = não, 1 = sim)
husearns	Renda semanal do marido em US\$ (ex.: 1500, 800)
huseduc	Anos de estudo do marido (ex.: 14, 17, 13)
husblck	Marido é preto (0 = não, 1 = sim)
hushisp	Marido é hispânico (0 = não, 1 = sim)
hushrs	Horas de trabalho semanais do marido (ex.: 40, 60)
kidge6	Possui filhos com mais de 6 anos (0 = não, 1 = sim)
earns	Renda semanal da esposa em US\$ (ex.: 100, 480)
age	Idade da esposa (ex.: 49, 29, 30)
black	Esposa é preta (0 = não, 1 = sim)
educ	Anos de estudo da esposa (ex.: 12, 14, 15)
hispanic	Esposa é hispânica (0 = não, 1 = sim)
union	União estável da esposa (0 = não, 1 = sim)
exper	Anos de experiência da esposa (ex.: 31, 9, 12)
kidlt6	Possui filhos menores de 6 anos (0 = não, 1 = sim)
lwage	Logaritmo natural do salário-hora da esposa

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

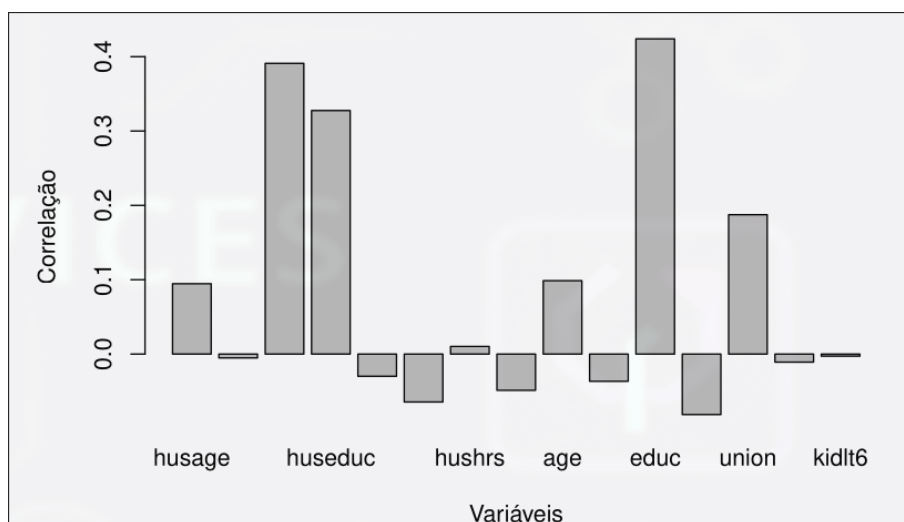
FIGURA 17 – CORRELAÇÃO DAS VARIÁVEIS COM A VARIÁVEL lwage



FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

A análise da figura 17 mostra que a variável `educ` possui a maior correlação positiva com `lwage`, o que sugere que o nível de escolaridade é o principal fator explicativo para a variação dos salários no conjunto de dados.

FIGURA 18 – GRÁFICO DE BARRAS DA CORRELAÇÃO DAS VARIÁVEIS COM A VARIÁVEL lwage



FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

Treinamento e avaliação do modelo Ridge

A regressão Ridge foi aplicada com o objetivo de estimar os parâmetros de um modelo linear multivariado com penalização do tipo ℓ_2 , visando reduzir multicolinearidade entre variáveis e

melhorar a capacidade preditiva. O procedimento compreende a reorganização dos dados, separação em conjuntos de treino e teste, definição do hiperparâmetro λ por validação cruzada, ajuste do modelo e avaliação por meio das métricas RMSE e R^2 , considerando os dados de treino e teste.

```

1 # Reordena as colunas para colocar 'lwage' como primeira
2 trein_padr_ridge <- trein_padr_ridge %>% select(lwage, everything())
3 test_padr_ridge <- test_padr_ridge %>% select(lwage, everything())
4
5 # Converte os dados de treino em matriz para o modelo
6 x_treino_ridge <- as.matrix(trein_padr_ridge[1:15])
7
8 # Define o vetor de resposta para os dados de treino
9 y_treino_ridge <- trein_padr_ridge$lwage
10
11 # Converte os dados de teste em matriz
12 x_teste_ridge <- as.matrix(test_padr_ridge[1:15])
13
14 # Define o vetor de resposta para os dados de teste
15 y_teste_ridge <- test_padr_ridge$lwage
16
17 # Define a sequência de valores de lambda para validação cruzada
18 lambdas <- 10^seq(2, -3, by = -.1)
19
20 # Realiza validação cruzada para obter o lambda ótimo (alpha = 0 para Ridge)
21 ridge_lamb <- cv.glmnet(x_treino_ridge, y_treino_ridge, alpha = 0, lambda =
    lambdas)
22
23 # Armazena o valor ótimo de lambda
24 best_lambda_ridge <- ridge_lamb$lambda.min
25
26 # Exibe o lambda ótimo
27 best_lambda_ridge
28
29 # Ajusta o modelo Ridge com o lambda ótimo
30 ridge_reg = glmnet( x_treino_ridge,
31                     y_treino_ridge,
32                     nlambdas = 25, alpha = 0,
33                     family = 'gaussian',
34                     lambda = best_lambda_ridge)

```

```
35
36 # Exibe os coeficientes estimados pelo modelo
37 ridge_reg[["beta"]]
38
39 # Função para calcular RMSE e R2 a partir dos valores previstos e observados
40 eval_results <- function(true, predicted, df) {
41   SSE <- sum((predicted - true)^2)
42   SST <- sum((true - mean(true))^2)
43   R_square <- 1 - SSE / SST
44   RMSE = sqrt(SSE/nrow(df))
45   data.frame(RMSE = RMSE, Rsquare = R_square)
46 }
47
48 # Gera previsões para os dados de treino
49 predictions_train <- predict(ridge_reg, s = best_lambda_ridge, newx = x_treino_
   ridge)
50
51 # Calcula métricas de desempenho (treino)
52 rmse_r2_ridge_trein <- eval_results(y_treino_ridge, predictions_train,
   treinamento_ridge)
53
54 # Exibe resultados do treino
55 rmse_r2_ridge_trein
56
57 # Gera previsões para os dados de teste
58 predictions_test <- predict(ridge_reg, s = best_lambda_ridge, newx = x_teste_
   ridge)
59
60 # Calcula métricas de desempenho (teste)
61 rmse_r2_ridge_test <- eval_results(y_teste_ridge, predictions_test, teste_ridge)
62
63 # Exibe resultados do teste
64 rmse_r2_ridge_test
```

TABELA 9 – COEFICIENTES ESTIMADOS PELO MODELO RIDGE PARA CADA VARIÁVEL EXPLICATIVA

Variável	Coefficiente
lwage	0,9986098
husage	0,0000836
husunion	-0,0000521
husearns	0,0003723
huseduc	0,0000644
husbck	0,0000937
hushisp	0,0002041
hushrs	-0,0000760
kidge6	-0,0001584
age	-0,0000063
black	-0,0000706
educ	0,0004313
hispanic	-0,0000491
union	0,0004995
exper	-0,0000003

FONTE: Elaborado pelo autor com base no dataset `trabalhosalarios(2024)`.

TABELA 10 – DESEMPENHO DO MODELO RIDGE NOS CONJUNTOS DE TREINO E TESTE

Conjunto	RMSE	R^2
Treino	0,001177862	0,9999986
Teste	0,001213452	0,9999985

FONTE: Elaborado pelo autor com base no dataset `trabalhosalarios(2024)`.

Ao comparar as métricas de desempenho obtidas nos conjuntos de treinamento e teste, observa-se que tanto o erro quadrático médio (RMSE) quanto o coeficiente de determinação (R^2) apresentam valores extremamente próximos. O RMSE foi de aproximadamente 0.00118 no treinamento e 0.00121 no teste, enquanto o R^2 atingiu 0.9999986 e 0.9999985, respectivamente. Esses resultados indicam que o modelo apresenta excelente capacidade de generalização, descartando indícios de overfitting ou underfitting. O valor de R^2 , próximo de 1, evidencia que o modelo é capaz de explicar quase a totalidade da variabilidade dos dados, o que demonstra um desempenho altamente satisfatório.

Predição e intervalo de confiança

A seguir, apresenta-se o procedimento utilizado para realizar a predição do salário-hora da esposa (em escala original) utilizando os valores fornecidos para as variáveis explicativas, previamente padronizadas com base no conjunto de treino. Além disso, são construídos um intervalo de confiança de 95% para a predição e duas visualizações gráficas: um gráfico de dispersão entre valores reais e preditos, e um QQ-plot dos resíduos. Essas análises são fundamentais para avaliar a precisão e a adequação do modelo Ridge à realidade observada.

```

1 # Padroniza a variável husage com base nos dados de treino
2 husage.scale = (40-mean(treinamento_ridge$husage))/sd(treinamento_ridge$husage)
3
4 # Padroniza a variável husearns com base nos dados de treino
5 husearns.scale = (600-mean(treinamento_ridge$husearns))/sd(treinamento_ridge$husearns)
6
7 # Padroniza a variável huseduc com base nos dados de treino
8 huseduc.scale = (13-mean(treinamento_ridge$huseduc))/sd(treinamento_ridge$huseduc)
9
10 # Padroniza a variável hushrs com base nos dados de treino
11 hushrs.scale = (40-mean(treinamento_ridge$hushrs))/sd(treinamento_ridge$hushrs)
12
13 # Padroniza a variável age com base nos dados de treino
14 age.scale = (38-mean(treinamento_ridge$age))/sd(treinamento_ridge$age)
15
16 # Padroniza a variável educ com base nos dados de treino
17 educ.scale = (13-mean(treinamento_ridge$educ))/sd(treinamento_ridge$educ)
18
19 # Padroniza a variável exper com base nos dados de treino
20 exper.scale = (18-mean(treinamento_ridge$exper))/sd(treinamento_ridge$exper)
21
22 # Cria uma matriz com os dados padronizados e valores fixos para variáveis
    categóricas
23 our_pred = as.matrix(data.frame(
24   husage=husage.scale, husearns=husearns.scale, huseduc=huseduc.scale,
25   hushrs=hushrs.scale, age=age.scale, educ=educ.scale, exper=exper.scale,
26   husunion=0, husblk=1, hushisp=0, kidge6=1, hispanic=1, black=0, union=0,
    kidlt6=1
27 ))
28
29 # Realiza a predição com o modelo Ridge ajustado
30 predict_our_ridge <- predict(ridge_reg, s = best_lambda_ridge, newx = our_pred)
31
32 # Converte o resultado padronizado para a escala original (antilog da predição)
33 wage_pred_ridge = (predict_our_ridge * sd(treinamento_ridge$lwage) + mean(
    treinamento_ridge$lwage))

```

```

34
35 # Número de observações do conjunto de treino
36 n <- nrow(treinamento_ridge)
37
38 # Média predita do salário (em escala original)
39 m <- wage_pred_ridge
40
41 # Desvio padrão da variável dependente no conjunto de treino
42 s <- sd(treinamento_ridge$lwage)
43
44 # Erro padrão da média
45 dam <- s / sqrt(n)
46
47 # Limite inferior do intervalo de confiança de 95%
48 CIlwr_ridge <- m + (qnorm(0.025)) * dam
49
50 # Limite superior do intervalo de confiança de 95%
51 CIupr_ridge <- m - (qnorm(0.025)) * dam
52
53 # Exibe o limite inferior
54 CIlwr_ridge
55
56 # Exibe o limite superior
57 CIupr_ridge
58
59 # Conversão das predições para vetor
60 predictions_test <- as.vector(predictions_test)
61
62 # Criação de dataframe com valores reais e valores preditos
63 df_pred <- data.frame(real = y_teste_ridge, predito = predictions_test)
64
65 # Geração de gráfico de dispersão entre os valores reais e preditos
66 ggplot(df_pred, aes(x = real, y = predito)) +
67   # Adição dos pontos ao gráfico
68   geom_point() +
69   # Linha de referência (reta 45 graus)
70   geom_abline(intercept = 0, slope = 1, color = "red") +
71   # Título e rótulos dos eixos
72   labs(title = "Valores Reais x Preditos",

```

```

73     x = "Valores Reais",
74     y = "Valores Preditos") +
75     # Tema visual minimalista
76     theme_minimal()
77
78 # Cálculo dos resíduos (diferença entre valores reais e preditos)
79 residuos <- y_teste_ridge - predictions_test
80
81 # Criação de dataframe com os resíduos
82 df_residuos <- data.frame(Residuos = residuos)
83
84 # Construção do gráfico QQ-plot para verificar a normalidade dos resíduos
85 ggplot(df_residuos, aes(sample = Residuos)) +
86     # Pontos do QQ-plot
87     geom_qq() +
88     # Linha de referência da normalidade
89     geom_qq_line(color = "red") +
90     # Título e rótulos dos eixos
91     labs(title = "QQ-plot dos Resíduos",
92          x = "Quantis Teóricos",
93          y = "Quantis Observados") +
94     # Tema visual minimalista
95     theme_minimal()

```

TABELA 11 – PREDIÇÃO DO SALÁRIO-HORA DA ESPOSA E INTERVALO DE CONFIANÇA AO NÍVEL DE 95%

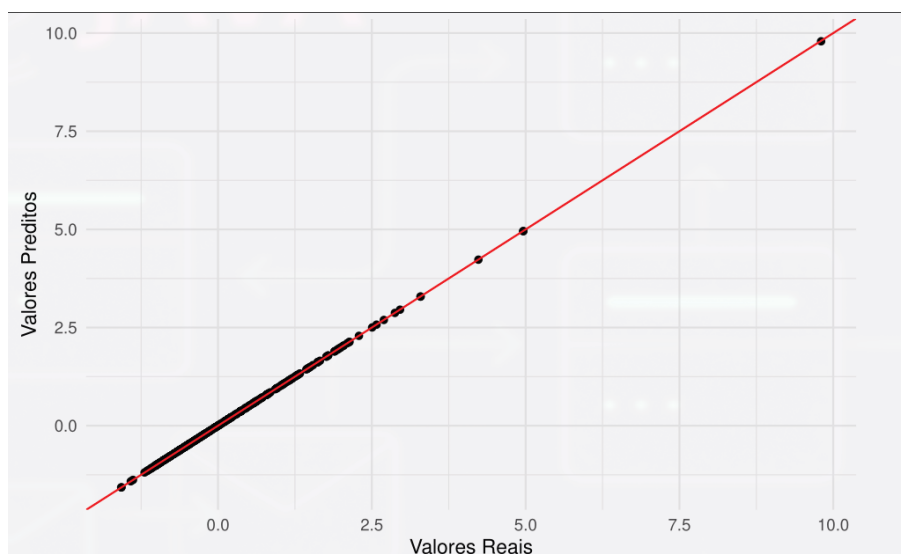
Métrica	Predição	Limite Inferior (IC 95%)	Limite Superior (IC 95%)
Salário-hora estimado	10,2668	10,0062	10,5274

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

Com base nas características fornecidas, o valor previsto para o salário por hora (lwage) pelo modelo Ridge foi de **10,26**. Considerando o intervalo de confiança estimado, esse valor pode variar entre aproximadamente **10,01** e **10,53**, o que indica uma boa precisão na predição.

A seguir, são apresentados dois gráficos gerados a partir do conjunto de teste, com o objetivo de avaliar visualmente o desempenho do modelo Ridge. Para viabilizar a construção dessas representações, o vetor `predictions_test` foi previamente convertido em um vetor unidimensional com valores contínuos.

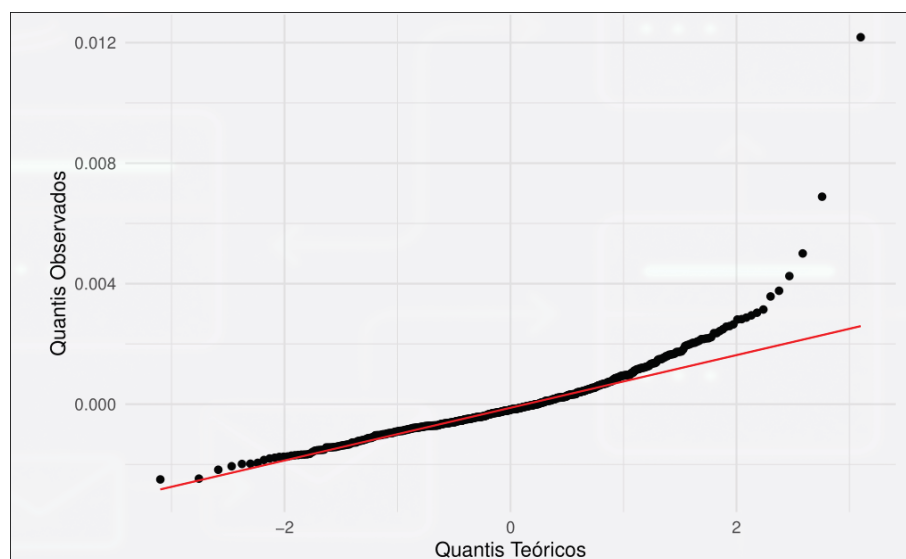
FIGURA 19 – GRÁFICO DE DISPERSÃO PARA COMPARAÇÃO DE VALORES REAIS DOS PREDITOS



FONTE: Elaborado pelo autor com base no dataset `trabalhosalarios(2024)`.

Observa-se na figura 19 um excelente desempenho do modelo, dado que os pontos estão fortemente alinhados à reta identidade $y = x$, onde o valor previsto coincide exatamente com o valor real. Como o eixo x representa os valores reais e o eixo y os valores preditos, esse alinhamento indica que o modelo foi capaz de reproduzir com alta fidelidade os dados observados, evidenciando uma acurácia praticamente perfeita nas previsões.

FIGURA 20 – GRÁFICO DE QUANTIS TEÓRICOS VS. QUANTIS OBSERVADOS DOS RESÍDUOS



FONTE: Elaborado pelo autor com base no dataset `trabalhosalarios(2024)`.

A figura 20 permite avaliar a aderência dos resíduos do modelo à distribuição normal. Observa-se


```

27
28 # Padroniza variáveis contínuas na base de teste
29 test_padr_lasso <- test_lasso %>% mutate_at(c('husage',
30                                             'husearns',
31                                             'huseduc',
32                                             'hushrs',
33                                             'age',
34                                             'educ',
35                                             'exper',
36                                             'lwage'),
37                                             ~(scale(.) %>% as.vector))
38
39 # Reorganiza as colunas para colocar 'lwage' na primeira posição
40 trein_padr_lasso <- trein_padr_lasso %>% select(lwage, everything())
41 test_padr_lasso <- test_padr_lasso %>% select(lwage, everything())
42
43 # Cria matriz de variáveis explicativas para o conjunto de treino
44 x_train_lasso <- as.matrix(trein_padr_lasso[, 2:16])
45
46 # Extrai vetor da variável dependente do treino
47 y_train_lasso <- trein_padr_lasso$lwage
48
49 # Cria matriz de variáveis explicativas para o conjunto de teste
50 x_test_lasso <- as.matrix(test_padr_lasso[, 2:16])
51
52 # Extrai vetor da variável dependente do teste
53 y_test_lasso <- test_padr_lasso$lwage
54
55 # Define sequência de valores de lambda para regularização
56 lambdas <- 10^seq(2, -3, by = -0.1)
57
58 # Ajusta modelo Lasso com validação cruzada (alpha = 1)
59 lasso_lamb <- cv.glmnet(x_train_lasso, y_train_lasso,
60                         alpha = 1,
61                         lambda = lambdas,
62                         standardize = TRUE,
63                         nfolds = 5)
64
65 # Armazena o valor de lambda que minimiza o erro

```

```

66 best_lambda_lasso <- lasso_lamb$lambda.min
67
68 # Exibe o melhor valor de lambda encontrado
69 best_lambda_lasso
70 ## [1] 0.01
71
72 # Ajusta o modelo Lasso final com lambda ótimo
73 lasso_model <- glmnet(x_train_lasso, y_train_lasso,
74                       alpha = 1,
75                       lambda = best_lambda_lasso,
76                       standardize = TRUE)
77
78 # Exibe os coeficientes estimados pelo modelo ajustado
79 lasso_model[["beta"]]
80
81 # Realiza previsões na base de treino
82 predictions_train <- predict(lasso_model,
83                               s = best_lambda_lasso,
84                               newx = x_train_lasso)
85
86 # Define função para calcular métricas de avaliação (RMSE e R2)
87 eval_results <- function(true, predicted, df) {
88   SSE <- sum((predicted - true)^2)      # Soma dos quadrados dos erros
89   SST <- sum((true - mean(true))^2)    # Soma total dos quadrados
90   R_square <- 1 - SSE / SST            # Coeficiente de determinação
91   RMSE <- sqrt(SSE / nrow(df))        # Raiz do erro quadrático médio
92
93   # Retorna dataframe com métricas
94   data.frame(
95     RMSE = RMSE,
96     R_square = R_square
97   )
98 }
99
100 # Avalia desempenho do modelo na base de treino
101 rmse_r2_lasso_trein <- eval_results(y_train_lasso, predictions_train, trein_lasso
102                                     )
103
104 # Realiza previsões na base de teste

```

```

104 predictions_test_lasso <- predict(lasso_model,
105     s = best_lambda_lasso,
106     newx = x_test_lasso)
107
108 # Avalia desempenho do modelo na base de teste
109 rmse_r2_lasso_test <- eval_results(y_test_lasso, predictions_test_lasso, test_
    lasso)
110
111 # Exibe as métricas de avaliação de treino
112 rmse_r2_lasso_trein
113
114 # Exibe as métricas de avaliação de teste
115 rmse_r2_lasso_test

```

TABELA 12 – COEFICIENTES ESTIMADOS PELO MODELO LASSO

Variável	Coeficiente
husage	0,0294
husunion	-0,0885
husearns	0,2777
huseduc	0,0073
husblack	0 (penalizado)
hushisp	0,0932
hushrs	-0,0727
kidge6	-0,1044
age	0,0120
black	-0,0074
educ	0,3232
hispanic	0 (penalizado)
union	0,3363
exper	0 (penalizado)
kidlt6	0 (penalizado)

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

As variáveis cujos coeficientes foram reduzidos a zero pelo Lasso foram consideradas irrelevantes para o modelo, sendo eliminadas automaticamente como parte do processo de regularização. Isso contribui para a simplificação do modelo e redução do risco de overfitting.

TABELA 13 – DESEMPENHO DO MODELO LASSO NAS BASES DE TREINO E TESTE

Conjunto	RMSE	R ²
Treinamento	0,8486	0,2795
Teste	0,8651	0,2501

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios(2024).

O modelo Lasso apresentou desempenho limitado na tarefa preditiva. O coeficiente de determinação (R^2) foi de aproximadamente 0,28 na base de treinamento e 0,25 na base de teste, revelando baixa capacidade explicativa do modelo sobre a variabilidade da variável dependente. Os valores do erro quadrático médio (RMSE) foram semelhantes entre os conjuntos — 0,85 no treino e 0,87 no teste — o que indica estabilidade, mas também sugere que o modelo não consegue generalizar com precisão suficiente. Esses resultados refletem a restrição do Lasso em capturar padrões mais complexos nos dados avaliados.

Predição e intervalo de confiança com modelo lasso

O código a seguir realiza a predição do salário (em escala original) para um indivíduo com características específicas, utilizando o modelo de regressão Lasso previamente ajustado. Inicialmente, os valores das variáveis contínuas são padronizados com base na média e desvio padrão da base de treinamento. Em seguida, é construída a matriz de predição com variáveis contínuas padronizadas e binárias definidas. O modelo retorna o valor predito de salário, que é então convertido da escala padronizada para a escala original. Por fim, calcula-se um intervalo de confiança aproximado de 95% para essa estimativa.

```

1 # Padroniza os valores das variáveis contínuas conforme média e desvio padrão da
  base de treinamento
2 husage.scale <- (40 - mean(trein_lasso$husage)) / sd(trein_lasso$husage)
3 husearns.scale <- (600 - mean(trein_lasso$husearns)) / sd(trein_lasso$husearns)
4 huseduc.scale <- (13 - mean(trein_lasso$huseduc)) / sd(trein_lasso$huseduc)
5 hushrs.scale <- (40 - mean(trein_lasso$hushrs)) / sd(trein_lasso$hushrs)
6 age.scale <- (38 - mean(trein_lasso$age)) / sd(trein_lasso$age)
7 educ.scale <- (13 - mean(trein_lasso$educ)) / sd(trein_lasso$educ)
8 exper.scale <- (18 - mean(trein_lasso$exper)) / sd(trein_lasso$exper)
9
10 # Atribui os valores padronizados às variáveis correspondentes
11 husage <- husage.scale
12 husearns <- husearns.scale
13 huseduc <- huseduc.scale
14 hushrs <- hushrs.scale
15 age <- age.scale
16 educ <- educ.scale
17 exper <- exper.scale
18
19 # Define os valores das variáveis binárias para o cenário de predição
20 husunion <- 0
21 husblck <- 1
22 hushisp <- 0

```

```

23 kidge6 <- 1
24 hispanic <- 1
25 black <- 0
26 union <- 0
27 kidlt6 <- 1
28
29 # Monta a matriz de predição com as variáveis explicativas padronizadas e
    binárias
30 our_pred_lasso <- as.matrix(data.frame(
31   husage = husage,
32   husearns = husearns,
33   huseduc = huseduc,
34   hushrs = hushrs,
35   age = age,
36   educ = educ,
37   exper = exper,
38   husunion = husunion,
39   husblck = husblck,
40   hushisp = hushisp,
41   kidge6 = kidge6,
42   hispanic = hispanic,
43   black = black,
44   union = union,
45   kidlt6 = kidlt6
46 ))
47
48 # Realiza a predição com o modelo Lasso usando o melhor lambda encontrado
49 predict_our_lasso <- predict(lasso_model,
50                               s = best_lambda_lasso,
51                               newx = our_pred_lasso)
52
53 # Converte a predição padronizada de volta para a escala original da variável
    salário (lwage)
54 wage_pred_lasso <- predict_our_lasso * sd(trein_lasso$lwage) + mean(trein_lasso$
    lwage)
55
56 # Exibe o valor do salário predito na escala original e com o melhor lambda
57 wage_pred_lasso
58 predict_our_lasso

```

```

59
60 # Define o tamanho da amostra da base de treinamento
61 n <- nrow(trein_lasso)
62
63 # Define o valor previsto do salário (wage) pelo modelo Lasso
64 m <- wage_pred_lasso
65
66 # Calcula o desvio padrão da variável salário na base de treinamento
67 s <- sd(trein_lasso$lwage)
68
69 # Calcula o erro padrão da média (desvio padrão amostral dividido pela raiz de n)
70 dam <- s / sqrt(n)
71
72 # Calcula o limite inferior do intervalo de confiança de 95%
73 CIlwr_lasso <- m + (qnorm(0.025)) * dam
74
75 # Calcula o limite superior do intervalo de confiança de 95%
76 CIupr_lasso <- m - (qnorm(0.025)) * dam
77
78 # Intervalo de confiança estimado para o salário previsto: aproximadamente entre
    11,60 e 12,11
79 CIlwr_lasso
80 CIupr_lasso

```

TABELA 14 – PREDIÇÃO E INTERVALO DE CONFIANÇA PARA O MODELO LASSO

Métrica	Valor	Descrição
Predição (original)	11,85542	Salário estimado na escala original
Predição (padronizada)	0,26939	Valor predito com o melhor lambda
Limite Inferior (IC 95%)	11,59967	Limite inferior do intervalo de confiança
Limite Superior (IC 95%)	12,11117	Limite superior do intervalo de confiança

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios (2024).

Com base nos valores preditos pelo modelo *Lasso*, observa-se que o salário estimado foi de aproximadamente 11,86, com um intervalo de confiança de 95% entre 11,60 e 12,11. Apesar da estabilidade do modelo, evidenciada pela amplitude relativamente estreita do intervalo, o valor predito encontra-se próximo da média, refletindo a limitada capacidade explicativa do *Lasso* observada anteriormente. Portanto, o modelo apresenta desempenho modesto, com baixa sensibilidade a variações individuais nas covariáveis.

Modelo Elastic Net

O código a seguir implementa o modelo Elastic Net, combinando os princípios da regressão Ridge e Lasso. O procedimento inclui a preparação dos dados, padronização das variáveis contínuas, criação de variáveis dummies para as categóricas e ajuste do modelo com validação cruzada repetida. Por fim, são realizadas predições e avaliadas métricas de desempenho nos conjuntos de treino e teste.

```

1
2 # Copia o dataset inicial para o objeto "dados_elastic"
3 dados_elastic <- trabalhosalarios
4
5 # Remove a coluna "earns" por não estar disponível para predição
6 dados_elastic <- dados_elastic %>% select(-earns)
7
8 # Aplica o antilog na variável "lwage" para retornar à escala original antes da
  padronização
9 dados_elastic$lwage <- exp(dados_elastic$lwage)
10
11 # Define a semente para reprodutibilidade
12 set.seed(10)
13
14 # Cria índice para dividir o dataset em 80% para treinamento
15 index_elastic <- sample(1:nrow(dados_elastic), 0.8 * nrow(dados_elastic))
16
17 # Define a base de treinamento com 80% dos dados
18 train_elastic <- dados_elastic[index_elastic,]
19
20 # Define a base de teste com os 20% restantes
21 test_elastic <- dados_elastic[-index_elastic,]
22
23 # Seleciona as variáveis contínuas para padronização
24 cols <- c('husage', 'husearns', 'huseduc', 'hushrs', 'age', 'educ', 'exper', '
  lwage')
25
26 # Aplica padronização (centralização e escala) na base de treinamento e teste
27 pre_proc_val <- preProcess(train_elastic[, cols], method = c("center", "scale"))
28 train_elastic[, cols] <- predict(pre_proc_val, train_elastic[, cols])
29 test_elastic[, cols] <- predict(pre_proc_val, test_elastic[, cols])
30
31 # Define as variáveis explicativas para o modelo
32 cols_reg <- c('husage', 'husunion', 'husearns', 'huseduc', 'husblck', 'hushisp',

```

```

    'hushrs',
33         'kidge6', 'age', 'black', 'educ', 'hispanic', 'union', 'exper', '
    kidlt6', 'lwage')
34
35 # Gera variáveis dummies para as categóricas e organiza os datasets em matrizes
36 dummies <- dummyVars(lwage ~ ., data = dados_elastic[, cols_reg])
37 train_dummies <- predict(dummies, newdata = train_elastic[, cols_reg])
38 test_dummies <- predict(dummies, newdata = test_elastic[, cols_reg])
39
40 # Cria matrizes com as variáveis explicativas para treinamento e teste
41 x_train_elastic <- as.matrix(train_dummies)
42 x_test_elastic <- as.matrix(test_dummies)
43
44 # Define os vetores da variável dependente para treinamento e teste
45 y_train_elastic <- train_elastic$lwage
46 y_test_elastic <- test_elastic$lwage
47
48 # Configura o controle de treino com validação cruzada repetida (10 folds, 5
    repetições) e busca aleatória
49 train_cont <- trainControl(method = "repeatedcv", number = 10, repeats = 5,
    search = "random", verboseIter = TRUE)
50
51 # Treina o modelo Elastic Net com ajuste automático dos parâmetros alpha (entre 0
    e 1) e lambda via cross-validation
52 elastic_reg <- train(lwage ~ ., data = train_elastic, method = "glmnet",
53                     tuneLength = 10, trControl = train_cont)
54
55 # Realiza predições no conjunto de treinamento e avalia a performance
56 predictions_train <- predict(elastic_reg, x_train_elastic)
57
58 # Função para calcular R2 e RMSE a partir dos valores verdadeiros e preditos
59 eval_results <- function(true, predicted, df) {
60   SSE <- sum((predicted - true)^2)
61   SST <- sum((true - mean(true))^2)
62   R_square <- 1 - SSE / SST
63   RMSE <- sqrt(SSE / nrow(df))
64
65   # Retorna as métricas de performance em um data.frame
66   data.frame(

```

```

67   RMSE = RMSE,
68   Rsquare = R_square
69 )
70 }
71
72 # Calcula métricas para o conjunto de treinamento
73 rmse_r2_elastic_trein <- eval_results(y_train_elastic, predictions_train, train_
    elastic)
74
75 # Realiza previsões no conjunto de teste
76 predictions_test_elastic <- predict(elastic_reg, x_test_elastic)
77
78 # Calcula métricas para o conjunto de teste
79 rmse_r2_elastic_test <- eval_results(y_test_elastic, predictions_test_elastic,
    test_elastic)
80
81 # Exibe as métricas de performance para treino e teste
82 rmse_r2_elastic_trein
83 rmse_r2_elastic_test

```

TABELA 15 – VALORES DE *ALPHA* E *LAMBDA* DURANTE A VALIDAÇÃO CRUZADA DO MODELO ELASTIC NET

Fold	Alpha	Lambda
Fold01.Rep1	0.29481	0.043777
Fold01.Rep1	0.11982	0.005046
Fold01.Rep1	0.34503	1.912272
⋮	⋮	⋮
Fold10.Rep5	0.99423	0.198967
Fold10.Rep5	0.36816	0.003910
Fold10.Rep5	0.36604	0.019913
Melhor combinação selecionada		
Final	0.366	0.0199

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios (2024).

TABELA 16 – DESEMPENHO DO MODELO LASSO NAS BASES DE TREINO E TESTE

Conjunto	RMSE	R ²
Treinamento	0,8474519	0,2814763
Teste	0,867134	0,2360335

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios (2024).

O modelo Elastic Net apresentou desempenho moderado, com coeficiente de determinação (R^2) de aproximadamente 0,28 na base de treinamento e 0,24 na base de teste, indicando capacidade explicativa razoável. Os valores de erro quadrático médio (RMSE) foram 0,85 no treino e 0,87 no teste, demonstrando consistência entre os conjuntos e boa generalização do modelo.

Predição com o modelo Elastic Net

O código a seguir realiza a predição do salário utilizando o modelo Elastic Net previamente treinado. Inicialmente, padroniza as variáveis explicativas com base nos parâmetros de treinamento, monta o conjunto de predição, gera a previsão para o salário, converte a predição para a escala original e calcula o intervalo de confiança de 95%.

```

1 # Padroniza os valores das variáveis explicativas para predição
2 husage <- (40 - pre_proc_val[["mean"]][["husage"]]) / pre_proc_val[["std"]][["
   husage"]]
3 husunion <- 0
4 husearns <- (600 - pre_proc_val[["mean"]][["husearns"]]) / pre_proc_val[["std"
   ]][["husearns"]]
5 huseduc <- (13 - pre_proc_val[["mean"]][["huseduc"]]) / pre_proc_val[["std"]][["
   huseduc"]]
6 husblack <- 1
7 hushisp <- 0
8 hushrs <- (40 - pre_proc_val[["mean"]][["hushrs"]]) / pre_proc_val[["std"]][["
   hushrs"]]
9 kidge6 <- 1
10 age <- (38 - pre_proc_val[["mean"]][["age"]]) / pre_proc_val[["std"]][["age"]]
11 black <- 0
12 educ <- (13 - pre_proc_val[["mean"]][["educ"]]) / pre_proc_val[["std"]][["educ"]]
13 hispanic <- 1
14 union <- 0
15 kidlt6 <- 1
16 exper <- (18 - pre_proc_val[["mean"]][["exper"]]) / pre_proc_val[["std"]][["exper
   "]]
17
18 # Monta um data.frame com as variáveis padronizadas para predição
19 our_pred_elastic <- data.frame(
20   husage = husage,
21   husearns = husearns,
22   huseduc = huseduc,
23   hushrs = hushrs,
24   age = age,

```

```

25   educ = educ,
26   exper = exper,
27   husunion = husunion,
28   husblck = husblck,
29   hushisp = hushisp,
30   kidge6 = kidge6,
31   hispanic = hispanic,
32   black = black,
33   union = union,
34   kidlt6 = kidlt6
35 )
36
37 # Realiza a predição com os valores especificados
38 predict_our_elastic <- predict(elastic_reg, our_pred_elastic)
39
40 # Converte o resultado padronizado para o valor original
41 wage_pred_elastic <- (predict_our_elastic * pre_proc_val[["std"]][["lwage"]]) +
   pre_proc_val[["mean"]][["lwage"]]
42
43 # Exibe a predição final do salário
44 predict_our_elastic
45 wage_pred_elastic
46
47 # Calcula o intervalo de confiança para a predição realizada
48 n <- nrow(train_elastic) # Número de observações no conjunto de treinamento
49 m <- wage_pred_elastic   # Valor predito (não padronizado)
50 s <- pre_proc_val[["std"]][["lwage"]] # Desvio padrão da variável 'lwage'
51 dam <- s / sqrt(n)       # Erro padrão da média
52
53 # Limites inferior e superior do intervalo de 95% de confiança
54 CIlwr_elastic <- m + qnorm(0.025) * dam
55 CIupr_elastic <- m - qnorm(0.025) * dam
56
57 # Exibe os limites do intervalo
58 CIlwr_elastic
59 CIupr_elastic

```

TABELA 17 – PREDIÇÃO E INTERVALO DE CONFIANÇA PARA O MODELO ELASTIC NET

Métrica	Valor
Predição padronizada do modelo	-0,1773
Predição convertida para a escala original (salário)	9,2366
Limite inferior do intervalo de confiança (95%)	8,9760
Limite superior do intervalo de confiança (95%)	9,4971

FONTE: Elaborado pelo autor com base no dataset `trabalhosalarios` (2024).

Os resultados indicam que a predição do modelo Elastic Net, após conversão para a escala original, estimou um salário de aproximadamente 9,24. O intervalo de confiança de 95% para essa estimativa varia entre 8,98 e 9,50, refletindo a precisão e a confiabilidade do modelo na predição do salário. A predição padronizada negativa sugere ajuste interno do modelo antes da reescala, o que é esperado devido à centralização dos dados.

Conclusão

A avaliação comparativa dos modelos Ridge, Lasso e Elastic Net é fundamental para compreender suas respectivas capacidades preditivas e ajustes frente ao conjunto de dados utilizado. Para tanto, foram compiladas as métricas de desempenho, especificamente o Erro Quadrático Médio (RMSE) e o coeficiente de determinação (R^2), calculadas sobre a base de treinamento. Essas medidas fornecem uma visão quantitativa da precisão e da capacidade explicativa dos modelos, possibilitando uma análise crítica e fundamentada acerca da adequação de cada abordagem para o problema em questão. O código-fonte a seguir demonstra a criação dos data frames que armazenam essas métricas, os quais serão utilizados para compor a tabela apresentada na sequência.

```

1 # Cria um data frame contendo os valores de RMSE para cada modelo
2 resultados_rmse <- data.frame(
3   Modelo = c("Ridge", "Lasso", "Elastic Net"),
4   RMSE = c(
5     rmse_r2_ridge_trein$RMSE, # RMSE do modelo Ridge no conjunto de treino
6     rmse_r2_lasso_trein$RMSE, # RMSE do modelo Lasso no conjunto de treino
7     rmse_r2_elastic_trein$RMSE # RMSE do modelo Elastic Net no conjunto de
      treino
8   )
9 )
10
11 # Cria um data frame contendo os valores de  $R^2$  para cada modelo
12 resultados_r2 <- data.frame(
13   Modelo = c("Ridge", "Lasso", "Elastic Net"),
14   R_squared = c(

```

```
15   rmse_r2_ridge_trein$Rsquare, # R2 do modelo Ridge no conjunto de treino
16   rmse_r2_lasso_trein$Rsquare, # R2 do modelo Lasso no conjunto de treino
17   rmse_r2_elastic_trein$Rsquare # R2 do modelo Elastic Net no conjunto de
    treino
18 )
19 )
```

TABELA 18 – COMPARAÇÃO DAS MÉTRICAS DE DESEMPENHO DOS MODELOS

Modelo	RMSE	R ²
Ridge	0,0012	0,9999986
Lasso	0,8486	0,2795
Elastic Net	0,8475	0,2815

FONTE: Elaborado pelo autor com base no dataset trabalhosalarios (2024).

Neste trabalho, foi avaliado o desempenho de três modelos de regressão penalizada: Ridge, Lasso e Elastic Net, com o objetivo de prever o salário-hora da esposa (*lwage*) a partir de variáveis socioeconômicas.

Com base nas métricas de avaliação — raiz do erro quadrático médio (RMSE) e coeficiente de determinação (R^2) — o modelo de regressão Ridge apresentou o melhor desempenho, com menor RMSE (0,0012) e R^2 próximo de 1 (0,9999986), indicando excelente ajuste aos dados de treinamento e baixa variabilidade dos resíduos. Os modelos Lasso e Elastic Net, embora úteis em cenários de alta dimensionalidade, apresentaram desempenho inferior neste caso específico.

Ao utilizar o modelo Ridge para predição, o salário-hora estimado foi de \$10,26. Com base no intervalo de confiança de 95%, esse valor pode variar entre \$10,01 e \$10,53, o que demonstra boa precisão na estimativa. Esses resultados reforçam a robustez do modelo Ridge na tarefa de predição neste conjunto de dados.

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 disponibilizado e crie as funções para calcular as diferentes características para o problema da identificação da língua do texto de entrada.

Nessa atividade é para "construir características".

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

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

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

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

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

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

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

B - RESOLUÇÃO

1. Construção de Características: Identificador automático de idioma

Identificador automático de idioma

Problema: Dado um texto de entrada, é possível identificar em qual idioma ele está escrito?

Exemplo de Entrada:

"texto qualquer"

Exemplo de Saída:

Português, Inglês, Espanhol, etc.

O Processo de Reconhecimento de Padrões

O objetivo deste trabalho é apresentar o processo de **construção de atributos** e destacar sua relevância no contexto do **Reconhecimento de Padrões (RP)**.

Inicialmente, utiliza-se um conjunto de *amostras previamente conhecidas e classificadas*, que serve como base para o treinamento do modelo. A partir dessas amostras, são extraídos atributos relevantes, que permitirão ao sistema aprender a identificar automaticamente o idioma de novos textos.

```
1 #
2 # amostras de texto em diferentes línguas
3 #
4 ingles = [
5     "Hello, how are you?",
6     "I love to read books.",
7     "The weather is nice today.",
8     "Where is the nearest restaurant?",
9     "What time is it?",
10    "I enjoy playing soccer.",
11    "Can you help me with this?",
12    "I'm going to the movies tonight.",
13    "This is a beautiful place.",
14    "I like listening to music.",
15    "Do you speak English?",
16    "What is your favorite color?",
17    "I'm learning to play the guitar.",
18    "Have a great day!",
19    "I need to buy some groceries.",
20    "Let's go for a walk.",
21    "How was your weekend?",
22    "I'm excited for the concert.",
23    "Could you pass me the salt, please?",
24    "I have a meeting at 2 PM.",
25    "I'm planning a vacation.",
26    "She sings beautifully.",
27    "The cat is sleeping.",
28    "I want to learn French.",
29    "I enjoy going to the beach.",
30    "Where can I find a taxi?",
31    "I'm sorry for the inconvenience.",
32    "I'm studying for my exams.",
33    "I like to cook dinner at home.",
34    "Do you have any recommendations for restaurants?",
35 ]
```

```
36
37 espanhol = [
38 "Hola, ¿cómo estás?",
39 "Me encanta leer libros.",
40 "El clima está agradable hoy.",
41 "¿Dónde está el restaurante más cercano?",
42 "¿Qué hora es?",
43 "Voy al parque todos los días.",
44 "¿Puedes ayudarme con esto?",
45 "Me gustaría ir de vacaciones.",
46 "Este es mi libro favorito.",
47 "Me gusta bailar salsa.",
48 "¿Hablas español?",
49 "¿Cuál es tu comida favorita?",
50 "Estoy aprendiendo a tocar el piano.",
51 "¡Que tengas un buen día!",
52 "Necesito comprar algunas frutas.",
53 "Vamos a dar un paseo.",
54 "¿Cómo estuvo tu fin de semana?",
55 "Estoy emocionado por el concierto.",
56 "¿Me pasas la sal, por favor?",
57 "Tengo una reunión a las 2 PM.",
58 "Estoy planeando unas vacaciones.",
59 "Ella canta hermosamente.",
60 "El perro está jugando.",
61 "Quiero aprender italiano.",
62 "Disfruto ir a la playa.",
63 "¿Dónde puedo encontrar un taxi?",
64 "Lamento las molestias.",
65 "Estoy estudiando para mis exámenes.",
66 "Me gusta cocinar la cena en casa.",
67 "¿Tienes alguna recomendación de restaurantes?"
68 ]
69
70 portugues = [
71 "Estou indo para o trabalho agora.",
72 "Adoro passar tempo com minha família.",
73 "Preciso comprar leite e pão.",
74 "Vamos ao cinema no sábado.",
```

```

75 "Gosto de praticar esportes ao ar livre.",
76 "O trânsito está terrível hoje.",
77 "A comida estava deliciosa!",
78 "Você já visitou o Rio de Janeiro?",
79 "Tenho uma reunião importante amanhã.",
80 "A festa começa às 20h.",
81 "Estou cansado depois de um longo dia de trabalho.",
82 "Vamos fazer um churrasco no final de semana.",
83 "O livro que estou lendo é muito interessante.",
84 "Estou aprendendo a cozinhar pratos novos.",
85 "Preciso fazer exercícios físicos regularmente.",
86 "Vou viajar para o exterior nas férias.",
87 "Você gosta de dançar?",
88 "Hoje é meu aniversário!",
89 "Gosto de ouvir música clássica.",
90 "Estou estudando para o vestibular.",
91 "Meu time de futebol favorito ganhou o jogo.",
92 "Quero aprender a tocar violão.",
93 "Vamos fazer uma viagem de carro.",
94 "O parque fica cheio aos finais de semana.",
95 "O filme que assisti ontem foi ótimo.",
96 "Preciso resolver esse problema o mais rápido possível.",
97 "Adoro explorar novos lugares.",
98 "Vou visitar meus avós no domingo.",
99 "Estou ansioso para as férias de verão.",
100 "Gosto de fazer caminhadas na natureza.",
101 "O restaurante tem uma vista incrível.",
102 "Vamos sair para jantar no sábado.",
103 ]

```

Representação de Padrões

A amostra de texto precisa ser transformada em um conjunto de padrões.

Um **padrão** é representado por um conjunto de características, geralmente estruturadas em forma de vetor. Um conjunto de padrões pode ser organizado em uma tabela, onde cada linha corresponde a um padrão distinto, e cada coluna representa uma característica. Frequentemente, a última coluna da tabela indica a *classe* à qual o padrão pertence.

```
1 # Construção do conjunto de padrões com classe conhecida.
2 # Para cada frase disponível em inglês, espanhol e português, uma entrada é
   criada
3 # contendo a frase (representando a amostra de texto) e sua respectiva classe (o
   idioma).
4 # Essas entradas são armazenadas na lista 'pre_padroes', que servirá como base
   para
5 # o treinamento ou avaliação do modelo de classificação.
6 # Ao final, o conjunto é embaralhado aleatoriamente com 'random.shuffle' para
   garantir
7 # que os dados fiquem distribuídos de forma não ordenada por classe, evitando
   viés
8 # durante o processamento posterior.
9
10 import random
11 import pandas as pd
12
13 pre_padroes = []
14 for frase in ingles:
15     pre_padroes.append( [frase, 'inglês'])
16
17 for frase in espanhol:
18     pre_padroes.append( [frase, 'espanhol'])
19
20 for frase in portugues:
21     pre_padroes.append( [frase, 'português'])
22
23 random.shuffle(pre_padroes)
24
25 # Criação do dataframe
26 dados = pd.DataFrame(pre_padroes, columns=['Frase', 'Idioma'])
```

TABELA 19 – AMOSTRA DO CONJUNTO DE PADRÕES GERADO A PARTIR DE FRASES ROTULADAS

ID	Frase	Idioma
0	Estoy emocionado por el concierto.	espanhol
1	¿Dónde puedo encontrar un taxi?	espanhol
2	Vamos ao cinema no sábado.	português
3	¿Cuál es tu comida favorita?	espanhol
4	I need to buy some groceries.	inglês
...		
87	What is your favorite color?	inglês
88	Estou estudando para o vestibular.	português
89	Estoy planeando unas vacaciones.	espanhol
90	Have a great day!	inglês
91	O parque fica cheio aos finais de semana.	português

FONTE: Elaborado pelo autor (2024)

Extração de Atributos

A extração de atributos consiste na conversão das frases rotuladas em vetores numéricos contendo informações relevantes para a tarefa de classificação. Nesse processo, cada frase é processada com o objetivo de identificar características linguísticas específicas, como o número de vogais, consoantes, letras acentuadas ou específicas de um idioma, bem como a ocorrência de palavras comuns em português, espanhol ou inglês.

Além disso, é calculado o tamanho médio das palavras presentes em cada sentença, com o intuito de capturar padrões estruturais do idioma. O resultado dessa transformação é um conjunto de vetores que representam os padrões derivados das frases originais.

O código-fonte a seguir implementa a extração dessas características e organiza o conjunto resultante em um *DataFrame*, que será utilizado nas etapas seguintes do pipeline de classificação.

```

1 # Criação dos padrões das frases
2 import re # Biblioteca para expressões regulares
3 import numpy as np # Biblioteca para operações numéricas
4
5 # Calcula o tamanho médio das palavras de uma frase
6 def tamanhoMedioFrases(texto):
7     palavras = re.findall(r'\b\w+\b', texto) # Extrai palavras
8     tamanhos = [len(s) for s in palavras] # Comprimento de cada palavra
9     return (sum(tamanhos) / len(tamanhos)) if len(tamanhos) > 0 else 0 # Média
    ou 0 se vazio
10
11 # Função que extrai características linguísticas específicas
12 def contaCaracteristicas(texto):
13     texto = texto.lower()

```

```

14
15 # Conjuntos de caracteres relevantes por idioma
16 vogais = "eiouáéíóúãõâêîôûäëïöü"
17 consoantes = "bcd fghjklmnpqrstvwxyz"
18 letras_espanhol = '¿¡!iñü'
19 letras_portugues = 'çããêõü'
20 letras_ingles = "wkyh"
21
22 # Contagem de caracteres específicos
23 num_vogais = sum(1 for char in texto if char in vogais)
24 num_consoantes = sum(1 for char in texto if char in consoantes)
25 num_letras_espanhol = sum(1 for char in texto if char in letras_espanhol)
26 num_letras_portugues = sum(1 for char in texto if char in letras_portugues)
27 num_letras_ingles = sum(1 for char in texto if char in letras_ingles)
28
29 # Palavras comuns em inglês (amostra representativa)
30 palavras_comuns_ingles = {
31     'the', 'be', 'to', 'of', 'and', 'a', 'in', 'that', 'have', 'i', 'gh', 'th',
    ', 'has', 'long', 'little', 'good',
32     'new', 'same', 'right', 'different', 'old', 'great', 'small', 'big', 'own',
    ', 'last', 'left', 'large', 'important',
33     'enough', 'second', 'first', 'well', 'next', 'live', 'high', 'following',
    ', 'used', 'even', 'is', 'was', 'are',
34     'had', 'were', 'can', 'said', 'do', 'will', 'would', 'could', 'make', ', '
    been', 'did', 'find', 'use', 'know', 'get',
35     'go', 'write', 'look', 'think', 'came', 'come', 'must', 'does', 'put', ', '
    went', 'tell', 'say', 'should', 'give',
36     'read', 'might', 'saw', 'asked', "don't", 'going', 'want', 'keep', 'took',
    ', 'began', 'got', 'need', 'let', 'being',
37     'see', 'made', 'may', 'take', 'found', 'set', 'looked', 'study', 'called',
    ', 'you', 'it', 'he', 'they', 'this', 'what',
38     'all', 'we', 'which', 'each', 'them', 'she', 'some', 'these', 'who', 'me',
    ', 'another', 'us', 'something', 'both',
39     'those', 'his', 'an', 'your', 'their', 'other', 'more', 'him', 'its', 'my',
    ', 'much', 'our', 'any', 'such', 'every',
40     'many', 'as', 'for', 'on', 'with', 'at', 'from', 'by', 'about', 'into', ', '
    like', 'after', 'through', 'before', 'under',
41     'along', 'until', 'without', 'between', 'or', 'but', 'if', 'than', ', '
    because', 'while', 'when', 'how', 'up', 'out',

```

```

42     'then', 'so', 'no', 'not', 'now', 'over', 'down', 'only', 'very', 'just',
    'where', 'most', 'too', 'also', 'around',
43     'here', 'why', 'away', 'again', 'off', 'still', 'never', 'below', 'always',
    'often', 'together', 'once', 'sometimes',
44     'above', 'almost', 'far', 'back', 'one', 'two', 'three', 'few', 'four'
45 }
46
47 # Palavras comuns em espanhol (com acentuação e estruturas frequentes)
48 palavras_comuns_espanhol = {
49     'i', '¿', 'll', 'ñ', 'aire', 'su', 'eres', 'es', 'una', 'tú', 'de', 'la',
    'que', 'el', 'en', 'y', 'a', 'los',
50     'hay', 'sin', 'del', 'se', 'tus', 'solo', 'me', 'mi', 'fue', 'ha', 'hacer',
    'hecho', 'hizo', 'haz', 'hoy',
51     'bien', 'ción', 'quie', 'tiem', 'pero', 'acción', 'acuerdo', 'además', '
    agua', 'ahora', 'año', 'aunque', 'bajo',
52     'cabeza', 'calidad', 'cambio', 'centro', 'ciudad', 'cuerpo', 'decir', '
    derecho', 'desarrollo', 'dice', 'dicho',
53     'donde', 'ejemplo', 'entonces', 'equipo', 'español', 'están', 'estaba', '
    familia', 'fin', 'frente', 'general',
54     'gente', 'gobierno', 'grupo', 'guerra', 'haber', 'había', 'hemos', 'hacia',
    'hasta', 'historia', 'hombre',
55     'incluso', 'internacional', 'lado', 'ley', 'llegar', 'lograr', 'luego', '
    madre', 'mal', 'manera', 'mayoría',
56     'medio', 'mejor', 'mercado', 'mientras', 'ministro', 'mismo', 'mujer', '
    muy', 'mucho', 'nacional', 'nada',
57     'nivel', 'nuevo', 'nunca', 'obra', 'organización', 'padre', 'palabras', '
    parece', 'partido', 'pasado', 'persona',
58     'poder', 'política', 'presidente', 'programa', 'propio', 'pueden', 'pues',
    'punto', 'quiere', 'realidad',
59     'relación', 'según', 'seguridad', 'semana', 'sentido', 'será', 'sistema',
    'situación', 'social', 'sociedad',
60     'también', 'tarde', 'tener', 'tiempo', 'tierra', 'tipo', 'todo', 'trabajo',
    'uno', 'ver', 'vida', 'vacaciones'
61 }
62
63 # Palavras comuns em português (com digramas, acentos e formas usuais)
64 palavras_comuns_portugues = {
65     'do', 'da', 'em', 'um', 'para', 'nh', 'lh', 'qua', 'di', 'du', 'je', '
    último', 'próprio', 'haver', 'fazer',

```

```

66     'dar', 'ficar', 'poder', 'ver', 'coisa', 'casa', 'tempo', 'ano', 'dia', '
vez', 'homem', 'senhor', 'senhora',
67     'moço', 'moça', 'não', 'mais', 'muito', 'já', 'quando', 'mesmo', 'depois'
, 'ainda', 'dois', 'primeiro',
68     'cem', 'mil', 'com', 'até', 'ou', 'também', 'assim', 'como', 'porque', '
eu', 'você', 'ele', 'esse', 'isso',
69     'sua', 'ai', 'ah', 'au', 'ui', 'hum'
70 }
71
72 # Tokenização simples para comparação com as listas de palavras
73 palavras = re.findall(r'\b\w+\b', texto)
74 num_palavras_ingles = sum(1 for palavra in palavras if palavra in palavras_
comuns_ingles)
75 num_palavras_espanhol = sum(1 for palavra in palavras if palavra in palavras_
comuns_espanhol)
76 num_palavras_portugues = sum(1 for palavra in palavras if palavra in palavras
_comuns_portugues)
77
78 return (
79     num_vogais, num_consoantes,
80     num_letras_espanhol, num_letras_portugues, num_letras_ingles,
81     num_palavras_ingles, num_palavras_espanhol, num_palavras_portugues
82 )
83
84 # Extrai todas as características de uma frase rotulada
85 def extraiCaracteristicas(frase):
86     texto = frase[0]
87     pattern_regex = re.compile('[^\w+]', re.UNICODE)
88     texto = re.sub(pattern_regex, ' ', texto) # Remove símbolos especiais
89
90     num_vogais, num_consoantes, num_letras_espanhol, num_letras_portugues, num_
letras_ingles, \
91     num_palavras_ingles, num_palavras_espanhol, num_palavras_portugues =
contaCaracteristicas(texto)
92
93     caracteristica1 = tamanhoMedioFrases(texto) # Média do tamanho das palavras
94
95     # Vetor de características com classe final (idioma)
96     padrao = [

```



```

97     caracteristcal, num_vogais, num_consoantes,
98     num_letras_espanhol, num_letras_portugues, num_letras_ingles,
99     num_palavras_ingles, num_palavras_espanhol, num_palavras_portugues,
100     frase[1] # Classe (idioma)
101 ]
102
103     return padrao
104
105 # Aplica extração em todas as frases do conjunto
106 def geraPadroes(frases):
107     padroes = []
108     for frase in frases:
109         padrao = extraiCaracteristicas(frase)
110         padroes.append(padrao)
111     return padroes
112
113 # Gera os vetores de padrões com base nas frases rotuladas
114 padroes = geraPadroes(pre_padroes)
115
116 # Visualiza os vetores extraídos
117 print(padroes)
118
119 # Converte para DataFrame para facilitar análise e modelagem
120 dados = pd.DataFrame(padroes)
121
122 # Ajusta uma das características apenas para a classe "português"
123 dados[0] = np.where(dados[9] == 'português', ((dados[0]) + 1) / 2, dados[0])
124
125 # Exibe os primeiros 60 registros
126 dados.head(60)

```

TABELA 20 – PADRÕES EXTRAÍDOS DAS FRASES: CADA LINHA REPRESENTA UM VETOR DE CARACTERÍSTICAS ASSOCIADO AO RESPECTIVO IDIOMA

ID	Méd. Pal.	Vogais	Cons.	L. Es	L. Pt	L. En	P. En	P. Es	P. Pt	Classe
0	5.800000	13	15	2	0	1	0	1	0	espanhol
1	5.000000	9	14	1	0	0	0	0	0	espanhol
2	2.600000	7	10	1	1	0	1	0	0	português
3	4.400000	8	11	2	1	0	0	1	0	espanhol
4	3.833333	11	12	2	0	1	4	0	0	inglês
5	3.333333	5	14	2	0	2	3	0	0	inglês
6	3.200000	4	8	0	0	0	1	1	1	espanhol
7	4.200000	9	12	5	0	1	3	0	0	inglês
8	4.333333	5	7	0	1	1	0	0	0	espanhol
9	3.666667	8	14	2	0	1	3	0	0	inglês
...										
50	6.000000	11	16	2	1	1	0	0	1	espanhol
51	3.083333	10	17	2	0	0	0	0	0	português
52	4.750000	6	12	2	0	2	1	0	0	inglês
53	4.200000	7	11	3	0	1	3	1	0	inglês
54	2.357143	14	11	4	2	0	0	1	2	português
55	3.416667	11	20	1	0	1	1	1	0	português
56	4.250000	6	10	0	0	6	3	0	0	inglês
57	4.000000	5	10	2	0	1	2	0	0	inglês
58	3.285714	10	12	3	0	3	4	0	0	inglês
59	3.333333	6	13	2	0	5	5	1	0	inglês

FONTE: Elaborado pelo autor (2024)

Treinamento do modelo com SVM

Para a etapa de classificação, foi utilizado o algoritmo *Support Vector Machines* (SVM), amplamente empregado em problemas de reconhecimento de padrões devido à sua capacidade de determinar hiperplanos ótimos que separam as classes de forma eficaz.

Antes do treinamento, os dados foram divididos em dois subconjuntos: um para o treinamento do modelo e outro para a avaliação do seu desempenho. Essa divisão é fundamental para assegurar que a validação seja realizada de maneira imparcial, utilizando-se dados inéditos durante a fase de aprendizagem.

A divisão dos dados foi realizada de forma estratificada, mantendo a proporção original entre as classes. A seguir, apresenta-se o código responsável por essa separação e treinamento:

```

1 from sklearn.model_selection import train_test_split
2 import numpy as np
3 from sklearn.preprocessing import StandardScaler
4
5 # Converte o DataFrame 'dados' em um array NumPy para facilitar a manipulação
  numérica
6 vet = np.array(dados)
7

```

```

8 # Seleciona a última coluna do array, que contém as classes (idiomas)
9 classes = vet[:, -1]
10
11 # print(classes) # Linha comentada para possível debug: exibe as classes
12
13 # Seleciona todas as colunas, exceto a última, que representam os padrões/
    características
14 padroes_sem_classe = vet[:, 0:-1]
15
16 # Imprime os padrões extraídos para verificação
17 print(padroes_sem_classe)
18
19 # Divide os dados em conjuntos de treinamento e teste
20 # 25% dos dados são reservados para teste
21 # A divisão mantém a proporção original das classes com 'stratify=classes'
22 # 'random_state=70' fixa a semente para garantir resultados reproduzíveis
23 X_train, X_test, y_train, y_test = train_test_split(
24     padroes_sem_classe, classes, test_size=0.25, stratify=classes, random_state
        =70
25 )
26
27 #####
28 ## Treinamento com SVM
29 from sklearn import svm
30 from sklearn.metrics import confusion_matrix
31 from sklearn.metrics import classification_report
32
33 treinador = svm.SVC() #algoritmo escolhido
34 modelo = treinador.fit(X_train, y_train)
35
36 #
37 # score com os dados de treinamento
38 acuracia = modelo.score(X_train, y_train)
39 print("Acurácia nos dados de treinamento: {:.2f}%".format(acuracia * 100))
40
41 #
42 # melhor avaliar com a matriz de confusão
43 y_pred = modelo.predict(X_train)
44 cm = confusion_matrix(y_train, y_pred)

```

```

45 print(cm)
46 print(classification_report(y_train, y_pred))
47
48 # score com os dados de teste
49 acuracia = modelo.score(X_test , y_test)
50 print("Acurácia nos dados de teste: {:.2f}%".format(acuracia * 100))
51
52 #
53 # com dados de teste que não foram usados no treinamento
54 print('métricas mais confiáveis')
55 y_pred2 = modelo.predict(X_test)
56 cm = confusion_matrix(y_test, y_pred2)
57 print(cm)
58 print(classification_report(y_test, y_pred2))

```

TABELA 21 – MATRIZ DE CONFUSÃO DO MODELO SVM NOS DADOS DE TREINAMENTO

Classe prevista	Espanhol	Inglês	Português
Espanhol	19	0	3
Inglês	2	21	0
Português	4	0	20
Acurácia: 86,96%			

FONTE: Elaborado pelo autor (2024)

TABELA 22 – MÉTRICAS DE DESEMPENHO DO MODELO SVM NOS DADOS DE TREINAMENTO

Classe	Precisão	Recall	F1-Score	Support
Espanhol	0,76	0,86	0,81	22
Inglês	1,00	0,91	0,95	23
Português	0,87	0,83	0,85	24
Acurácia: 86,96%				
Macro avg	0,88	0,87	0,87	69
Weighted avg	0,88	0,87	0,87	69

FONTE: Elaborado pelo autor (2024)

TABELA 23 – MATRIZ DE CONFUSÃO DO MODELO SVM NOS DADOS DE TESTE

Classe prevista	Espanhol	Inglês	Português
Espanhol	6	0	2
Inglês	0	6	1
Português	0	0	8
Acurácia: 86,96%			

FONTE: Elaborado pelo autor (2024)

TABELA 24 – MÉTRICAS DE DESEMPENHO DO MODELO SVM NOS DADOS DE TESTE

Classe	Precisão	Recall	F1-Score	Support
Espanhol	1,00	0,75	0,86	8
Inglês	1,00	0,86	0,92	7
Português	0,73	1,00	0,84	8
Acurácia: 0,87				
Macro avg	0,91	0,87	0,87	23
Weighted avg	0,91	0,87	0,87	23

FONTE: Elaborado pelo autor (2024)

Execução do modelo com entrada de texto

A etapa de execução do modelo consiste em classificar uma nova entrada de texto fornecida pelo usuário em tempo real. Inicialmente, o texto é capturado por meio da função `input`, permitindo que o usuário digite uma frase a ser classificada.

Em seguida, o texto é encapsulado em uma estrutura compatível com as funções de extração de características, adicionando um rótulo fictício, que não será utilizado na predição. A função `extraíCaracterísticas` é aplicada para converter o texto em um vetor de atributos numéricos que representam as propriedades relevantes para a classificação.

Após a extração, o vetor de características é ajustado para remover o rótulo fictício, ficando apto para ser fornecido como entrada ao modelo treinado. Finalmente, o modelo realiza a predição do idioma com base nas características extraídas, e o resultado é exibido ao usuário.

A seguir, apresenta-se o código responsável por essa etapa:

```

1
2 #Entrada do texto para execução
3 texto = input('Digite o texto a ser classificado pelo Modelo: ')
4 texto_vetor = [texto, '0']
5 print(texto[2])
6 # Extraímos as características do texto de exemplo
7 caracteristicas_texto = extraíCaracterísticas(texto_vetor)
8 print(caracteristicas_texto)
9
10 # Removemos a última característica que é o rótulo '0'
11 caracteristicas_texto = caracteristicas_texto[:-1]
12 print(caracteristicas_texto)
13 # Usamos o modelo treinado para prever o idioma do texto
14 idioma_predito = modelo.predict([caracteristicas_texto])
15
16 print("O texto foi classificado como:", idioma_predito[0])

```

TABELA 25 – RESULTADO DA EXECUÇÃO DA CLASSIFICAÇÃO DE TEXTO

Item	Resultado
Entrada	celebrating the birthday of a person
Caractere impresso (posição 2)	1
Características extraídas (com rótulo fictício)	[5.17, 8, 20, 2, 0, 3, 3, 1, 0, '0']
Características para predição (sem rótulo)	[5.17, 8, 20, 2, 0, 3, 3, 1, 0]
Idioma predito	inglês

FONTE: Elaborado pelo autor (2024)

Conclusão

Os resultados indicam uma predominância na classificação de frases como pertencentes ao *português*, o que pode ser atribuído ao peso significativo da característica relacionada ao comprimento da frase no modelo. Frases curtas em *espanhol* são corretamente identificadas; entretanto, ao aumentar o tamanho do texto com a inclusão de termos em espanhol, o modelo tende a classificá-las erroneamente como *português*. Isso sugere que o atributo referente ao tamanho do texto exerce influência desproporcional na decisão do classificador, comprometendo a precisão na identificação correta de frases mais extensas em espanhol.

2. Melhore uma base de dados ruim

A presente atividade tem como objetivo melhorar o desempenho de um modelo de reconhecimento de padrões (RP) aplicado a um conjunto de dados previamente existente. O conjunto de dados em questão é composto por instâncias de e-mails, os quais devem ser classificados quanto à presença de características associadas a mensagens do tipo *spam*. A tarefa propõe a análise e aplicação de técnicas que otimizem o processo de classificação, visando maior acurácia e desempenho computacional. Dessa forma, busca-se aprimorar a capacidade preditiva do modelo por meio de ajustes metodológicos, como seleção de atributos, escolha de algoritmos ou modificação de parâmetros.

```

1 # O comando abaixo realiza a instalação do pacote ucimlrepo, que permite acessar
   conjuntos de dados disponibilizados pelo UCI Machine Learning Repository
   diretamente em ambientes Python, facilitando a importação e manipulação dos
   dados utilizados na atividade.
2 pip install ucimlrepo

```

Carregamento e estruturação inicial do conjunto de dados

O código-fonte a seguir é responsável por carregar e preparar o conjunto de dados *Spambase*, disponibilizado pela UCI Machine Learning Repository, para tarefas de classificação de e-mails em mensagens legítimas ou *spam*. Inicialmente, são importadas as bibliotecas necessárias e configurado o ambiente de execução. Em seguida, o conjunto de dados é carregado por duas abordagens distintas: diretamente via URL e utilizando o pacote *ucimlrepo*, que fornece uma

interface estruturada para o acesso ao repositório da UCI. Essa redundância permite verificar a consistência entre diferentes métodos de obtenção dos dados e facilita o processo de exploração e modelagem em etapas posteriores.

```

1  # Importação da biblioteca para ignorar alertas de execução
2  import warnings
3  warnings.filterwarnings("ignore")
4
5  # Bibliotecas para manipulação e visualização de dados
6  import pandas as pd
7  import numpy as np
8  import seaborn as sns
9  import matplotlib.pyplot as plt
10 from numpy import nan
11
12 # Bibliotecas para pré-processamento e modelagem
13 from sklearn.impute import KNNImputer
14 from sklearn.ensemble import RandomForestClassifier
15 from sklearn.pipeline import Pipeline
16 from sklearn.impute import SimpleImputer
17 from sklearn.model_selection import RepeatedStratifiedKFold
18 from sklearn.model_selection import cross_val_score
19 from sklearn.model_selection import train_test_split
20 from sklearn.svm import SVC
21 from sklearn.metrics import accuracy_score, confusion_matrix
22 from sklearn.preprocessing import LabelEncoder
23 from sklearn.preprocessing import StandardScaler
24 from sklearn.preprocessing import MinMaxScaler
25 from sklearn.feature_selection import SelectKBest, f_classif
26 from imblearn.over_sampling import SMOTE
27 from sklearn.metrics import classification_report
28
29 # URL do conjunto de dados Spambase (UCI)
30 url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/spambase/
      spambase.data'
31
32 # Nomes das colunas conforme documentação oficial do dataset
33 column_names = [
34     "word_freq_make", "word_freq_address", "word_freq_all", "word_freq_3d", "word
      _freq_our",

```

```

35     "word_freq_over", "word_freq_remove", "word_freq_internet", "word_freq_order"
    , "word_freq_mail",
36     "word_freq_receive", "word_freq_will", "word_freq_people", "word_freq_report"
    , "word_freq_addresses",
37     "word_freq_free", "word_freq_business", "word_freq_email", "word_freq_you", "
word_freq_credit",
38     "word_freq_your", "word_freq_font", "word_freq_000", "word_freq_money", "word
_freq_hp",
39     "word_freq_hpl", "word_freq_george", "word_freq_650", "word_freq_lab", "word_
freq_labs",
40     "word_freq_telnet", "word_freq_857", "word_freq_data", "word_freq_415", "word
_freq_85",
41     "word_freq_technology", "word_freq_1999", "word_freq_parts", "word_freq_pm",
"word_freq_direct",
42     "word_freq_cs", "word_freq_meeting", "word_freq_original", "word_freq_project
", "word_freq_re",
43     "word_freq_edu", "word_freq_table", "word_freq_conference", "char_freq;", "
char_freq(",
44     "char_freq[", "char_freq!", "char_freq$", "char_freq#", "capital_run_
length_average",
45     "capital_run_length_longest", "capital_run_length_total", "class"
46 ]
47
48 # Carrega o dataset diretamente da URL, utilizando os nomes das colunas definidos
    acima
49 data = pd.read_csv(url, header=None, names=column_names)
50
51 # Importação do utilitário ucimlrepo para acessar a base de dados estruturada da
    UCI
52 from ucimlrepo import fetch_ucirepo
53
54 # Obtém o dataset Spambase via ID (94) do repositório UCI usando o ucimlrepo
55 spambase = fetch_ucirepo(id=94)
56
57 # Separação das variáveis preditoras (features) e da variável alvo (class)
58 X = spambase.data.features # Características extraídas dos e-mails
59 y = spambase.data.targets # Classe (spam ou não spam)

```

Com o objetivo de preparar os dados para a etapa de modelagem, este trecho de código realiza

a consolidação das variáveis preditoras e da variável-alvo em um único *DataFrame*, facilitando a manipulação e a análise conjunta das informações. Em seguida, são eliminadas algumas colunas com baixa correlação, de forma a melhorar a legibilidade e o foco visual da matriz de correlação. A matriz resultante é calculada utilizando o coeficiente de Spearman, adequado para variáveis com distribuições não necessariamente lineares. Por fim, a correlação entre as variáveis é visualizada por meio de um *heatmap*, possibilitando a identificação de possíveis redundâncias e relações significativas entre os atributos e a classe.

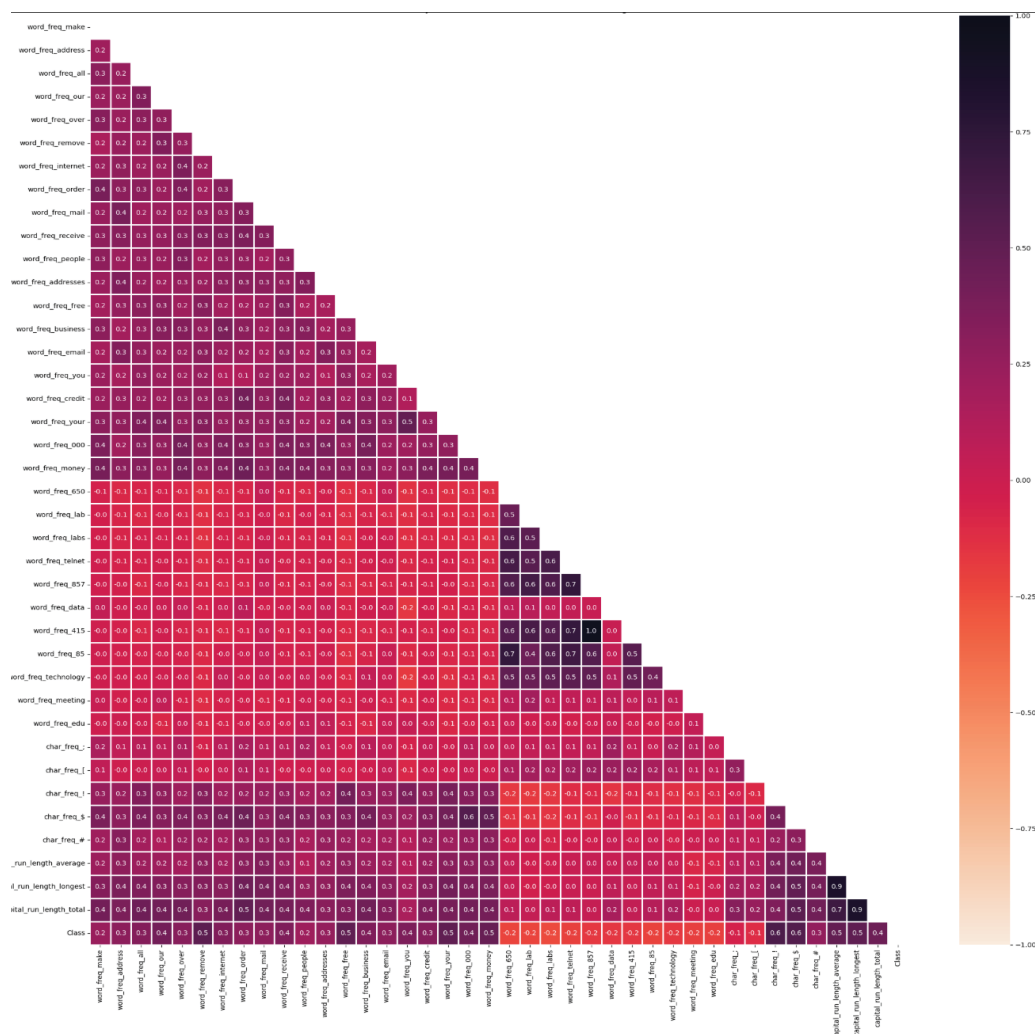
```

1 # Cria um DataFrame com as características (variáveis preditoras)
2 caracteristicas = pd.DataFrame(X)
3
4 # Cria um DataFrame com as classes (variável-alvo)
5 classes = pd.DataFrame(y)
6
7 # Concatena as características e a classe em um único DataFrame
8 # Isso facilita a análise conjunta das variáveis e a visualização dos dados
9 dadosGeral: pd.DataFrame = pd.concat([caracteristicas, classes], axis=1)
10
11 # Remove colunas com baixa correlação para simplificar a visualização
12 # Essa exclusão é feita apenas para melhorar a legibilidade do gráfico,
13 # sem impacto na análise mais aprofundada ou no modelo final
14 dadosGeralAltaCorrelacao = dadosGeral.drop([
15     'word_freq_3d', 'word_freq_will', 'word_freq_report', 'word_freq_font',
16     'word_freq_pm', 'word_freq_direct', 'word_freq_cs', 'word_freq_original',
17     'word_freq_hp', 'word_freq_hpl', 'word_freq_george', 'word_freq_1999',
18     'char_freq(', 'word_freq_parts', 'word_freq_project', 'word_freq_re',
19     'word_freq_table', 'word_freq_conference'
20 ], axis=1)
21
22 # Verifica os valores únicos presentes na variável de classe
23 # Isso permite confirmar se há mais de uma classe e se os dados estão corretos
24 unique_classes = dadosGeralAltaCorrelacao['Class'].unique()
25 print("Classes:", unique_classes)
26
27 # Calcula a matriz de correlação entre todas as variáveis do DataFrame
28 # Utiliza o método de Spearman, adequado para capturar relações monotônicas
29 matriz_correlacao = dadosGeralAltaCorrelacao.corr(method='spearman')
30
31 # Define o tamanho da figura do gráfico (largura x altura em polegadas)
32 plt.figure(figsize=(26, 26))

```

```
33
34 # Cria uma máscara para ocultar a metade superior da matriz (duplicata)
35 mask = np.triu(np.ones_like(matriz_correlacao, dtype=float))
36
37 # Gera o heatmap com os coeficientes de correlação
38 # O mapa de cores utilizado é 'rocket_r' (inverso do rocket) para melhor
    contraste
39 sns.heatmap(
40     matriz_correlacao,
41     mask=mask,
42     vmin=-1,
43     vmax=1,
44     annot=True,
45     cmap='rocket_r',
46     fmt="0.1f",
47     linewidth=.9
48 )
49
50 # Adiciona o título ao gráfico
51 plt.title('Matriz de Correlação entre Características e Classes - Dados Originais
    ')
52
53 # Exibe o gráfico na tela
54 plt.show()
```

FIGURA 21 – MATRIZ DE CORRELAÇÃO ENTRE CARACTERÍSTICAS E CLASSES - DADOS ORIGINAIS



FONTE: Elaboração do autor (2024).

Avaliação do impacto de estratégias de tratamento de valores ausentes

Esta etapa tem como objetivo avaliar o impacto de diferentes estratégias de tratamento de valores ausentes no desempenho de modelos de classificação. Para isso, utilizam-se os métodos oferecidos pelo *SimpleImputer*, sendo eles: média, mediana, valor mais frequente e valor constante. Cada estratégia é aplicada dentro de um pipeline junto a um classificador *Random Forest*, e a acurácia é avaliada utilizando validação cruzada estratificada repetida (com 10 divisões e 3 repetições). Ao final, os resultados são apresentados graficamente, permitindo a comparação entre as abordagens e a escolha da mais adequada para o conjunto de dados em questão.

```
1 # Comentários sobre técnicas de codificação de dados (não utilizadas diretamente
   neste código, mas relevantes para contexto futuro)
2 # OrdinalEncoder: codifica categorias em inteiros, preservando a ordem.
3 # Binarizer: transforma valores contínuos em binários (0 ou 1).
```

```

4 # KBinsDiscretizer: discretiza valores contínuos em faixas (bins).
5 # LabelBinarizer: codifica rótulos em formato binário (um contra todos).
6 # LabelEncoder: codifica rótulos em inteiros (geralmente usado para y).
7 # OneHotEncoder: transforma categorias em colunas binárias (codificação esparsa).
8 # TargetEncoder: codifica categorias com base na média do alvo (target), útil em
  problemas supervisionados.
9
10 # Exibe as 10 primeiras linhas do DataFrame com variáveis de maior correlação
11 dadosGeralAltaCorrelacao.head(10)
12
13 # AVALIAÇÃO DE ESTRATÉGIAS DE TRATAMENTO DE VALORES AUSENTES
14
15 # Lista para armazenar os resultados de cada estratégia de tratamento
16 results = list()
17
18 # Separa os dados em variáveis preditoras (X) e alvo (y)
19 X = dadosGeral.iloc[:, :-1].values # Características (todas as colunas, exceto a
  última)
20 y = dadosGeral.iloc[:, -1].values # Classe (última coluna)
21
22 # Define as estratégias de tratamento a serem avaliadas
23 strategies = ['mean', 'median', 'most_frequent', 'constant']
24
25 # Loop para aplicar e avaliar cada estratégia
26 for strat in strategies:
27     # Cria um pipeline com duas etapas: tratamento dos valores ausentes e
  classificação
28     pipeline = Pipeline(steps=[
29         ('i', SimpleImputer(strategy=strat)),          # Imputador com a
  estratégia atual
30         ('m', RandomForestClassifier())                # Classificador Random
  Forest
31     ])
32
33     # Configura a validação cruzada estratificada repetida
34     crossval = RepeatedStratifiedKFold(
35         n_splits=10,      # Número de subdivisões (folds)
36         n_repeats=3,      # Número de repetições da validação cruzada
37         random_state=42   # Define a semente para reprodutibilidade

```

```

38     )
39
40     # Avalia o pipeline utilizando validação cruzada
41     scores = cross_val_score(
42         pipeline, X, y,
43         scoring='accuracy',    # Métrica de avaliação: acurácia
44         cv=crossval,
45         n_jobs=-1              # Usa todos os núcleos disponíveis para paralelismo
46     )
47
48     # Armazena os resultados de acurácia para posterior comparação
49     results.append(scores)
50
51     # Exibe a média e o desvio padrão da acurácia para a estratégia atual
52     print('> %s %.3f (%.3f)' % (strat, np.mean(scores), np.std(scores)))
53
54 # Gera um boxplot para comparar visualmente os resultados das estratégias de
    tratamento de valores ausentes
55 plt.boxplot(results, labels=strategies, showmeans=True)
56 plt.title("Comparação entre estratégias de tratamento de valores ausentes")
57 plt.xlabel("Estratégia de tratamento")
58 plt.ylabel("Acurácia")
59 plt.grid(True)
60 plt.show()

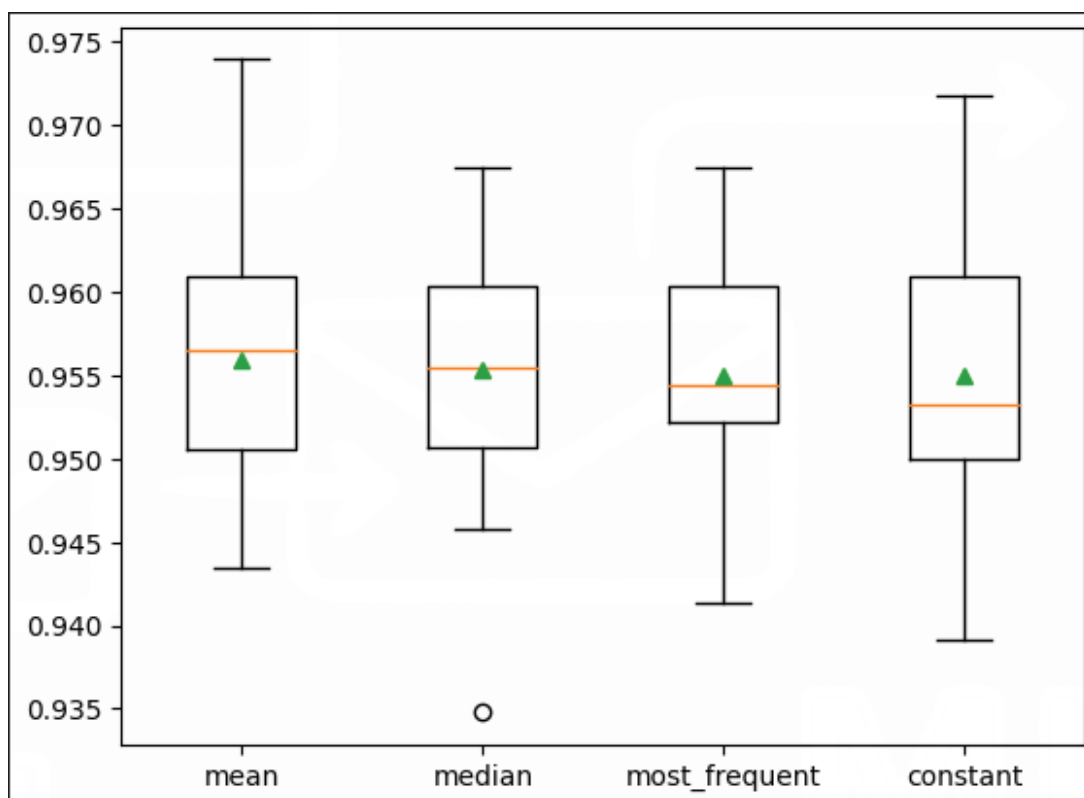
```

TABELA 26 – RESULTADOS MÉDIOS DE ACURÁCIA PARA DIFERENTES ESTRATÉGIAS DE TRATAMENTO DE VALORES AUSENTES

Estratégia de Tratamento	Acurácia Média	Desvio Padrão
Média (mean)	0,956	0,007
Mediana (median)	0,955	0,007
Mais frequente (most_frequent)	0,955	0,006
Constante (constant)	0,955	0,008

FONTE: Elaboração do autor (2024).

FIGURA 22 – COMPARAÇÃO ENTRE ESTRATÉGIAS DE TRATAMENTO DE VALORES AUSENTES



FONTE: Elaboração do autor (2024).

Aplicação de técnicas de escalonamento dos dados

Nesta etapa, aplicam-se duas técnicas clássicas de escalonamento ao conjunto de dados: a normalização e a padronização. A normalização, realizada por meio do *MinMaxScaler*, transforma os valores das variáveis para o intervalo $[0, 1]$, mantendo a proporcionalidade dos dados. Já a padronização, utilizando o *StandardScaler*, centraliza os dados em torno da média zero, com desvio padrão igual a um. Ambas as transformações são amplamente utilizadas em algoritmos de aprendizado de máquina que são sensíveis à escala dos dados, sendo uma etapa essencial no pré-processamento para garantir melhor desempenho e estabilidade dos modelos.

```

1 # ESTUDO DE NORMALIZAÇÃO E PADRONIZAÇÃO DOS DADOS GERAIS
2
3 # Inicializa o escalonador Min-Max para normalização dos dados (escala entre 0 e
  1)
4 min_max_scaler = MinMaxScaler()
5
6 # Inicializa o escalonador padrão para padronização dos dados (média 0 e desvio
  padrão 1)
7 std_scaler      = StandardScaler()

```

```

8
9 # Separa as características (todas as colunas, exceto a última) e as classes (
    última coluna)
10 caracteristicas = dadosGeral.iloc[:, :-1] # Características
11 classes = dadosGeral.iloc[:, -1]          # Classes
12
13 # Aplica normalização aos dados completos
14 data_normalized = min_max_scaler.fit_transform(dadosGeral)
15
16 # Converte o resultado da normalização em DataFrame
17 df_data_normalized = pd.DataFrame(data_normalized)
18
19 # Aplica padronização aos dados completos
20 data_standardized = std_scaler.fit_transform(dadosGeral)
21
22 # Converte o resultado da padronização em DataFrame
23 df_data_standardized = pd.DataFrame(data_standardized)
24
25 # Neste ponto, os dados estão prontos para serem utilizados em modelos de
    aprendizado de máquina,
26 # podendo-se escolher entre os dados normalizados ou padronizados conforme a
    natureza do algoritmo.

```

Avaliação do desempenho inicial do classificador SVM

Nesta etapa, realiza-se a primeira avaliação do desempenho do modelo de classificação *Support Vector Machine* (SVM) utilizando o conjunto de dados original, ou seja, sem aplicação de técnicas de balanceamento. O objetivo é estabelecer uma linha de base (baseline) de desempenho, observando o comportamento do modelo frente à possível desproporção entre as classes. A base de dados é dividida em subconjuntos de treino e teste utilizando amostragem estratificada, de modo a preservar a proporção das classes. A métrica de desempenho utilizada é a acurácia, acompanhada da matriz de confusão e do relatório de classificação, os quais auxiliam na análise do impacto do desbalanceamento na capacidade preditiva do modelo.

```

1 ## Acurácia na Base de Teste antes do Balanceamento
2
3 # Divide o conjunto de dados em treino e teste (75% treino, 25% teste),
4 # utilizando amostragem estratificada para preservar a proporção das classes
5 X_train, X_test, y_train, y_test = train_test_split(
6     X, y, stratify=classes, test_size=0.25, random_state=42
7 )

```

```

8
9 # Inicializa o classificador SVM com kernel radial (RBF)
10 svm_classifier = SVC(kernel='rbf')
11
12 # Treina o modelo com os dados de treino
13 svm_classifier.fit(X_train, y_train)
14
15 # Realiza previsões com os dados de teste
16 y_pred = svm_classifier.predict(X_test)
17
18 # Calcula a acurácia do modelo no conjunto de teste
19 initial_accuracy = accuracy_score(y_test, y_pred)
20 print("Acurácia na Base de Teste antes do Balanceamento:", initial_accuracy)
21
22 # Gera a matriz de confusão para avaliar os acertos e erros por classe
23 initial_conf_matrix = confusion_matrix(y_test, y_pred)
24 print("Matriz de Confusão - Dados de Teste:")
25 print(initial_conf_matrix)
26
27 # Exibe o relatório de classificação (precisão, recall, f1-score)
28 print(classification_report(y_test, y_pred))

```

TABELA 27 – MATRIZ DE CONFUSÃO E MÉTRICAS DE DESEMPENHO DO CLASSIFICADOR SVM ANTES DO BALANCEAMENTO

Classe real	Classe prevista 0	Classe prevista 1	Precisão	Revocação	F1-Score	Suporte
Não Spam (0)	587	110	0,72	0,84	0,77	697
Spam (1)	233	221	0,67	0,49	0,56	454
Acurácia geral	0,70 (com 1151 amostras)					
Média macro	–		0,69	0,66	0,67	1151
Média ponderada	–		0,70	0,70	0,69	1151

FONTE: Elaboração do autor com base nos resultados obtidos (2024).

Avaliação do classificador SVM após o balanceamento com SMOTE

Nesta etapa, aplica-se a técnica *SMOTE* (Synthetic Minority Over-sampling Technique) com o intuito de balancear o conjunto de dados de treino, aumentando sinteticamente a quantidade de exemplos da classe minoritária. O balanceamento é uma etapa essencial em problemas de classificação com distribuição desproporcional entre as classes, como é o caso da base de dados utilizada. Após o balanceamento, o classificador *Support Vector Machine* (SVM), com o kernel do tipo *rbf*, é treinado novamente e avaliado com o mesmo conjunto de teste anterior. As métricas de desempenho são então comparadas com os resultados obtidos antes do balanceamento para verificar se houve ganho na capacidade de generalização do modelo frente à classe minoritária.


```

1 # BALANCEAR os dados de treino com SMOTE
2 # O SMOTE cria amostras sintéticas para a classe minoritária, reduzindo o
  desbalanceamento
3 smote = SMOTE(random_state = 42)
4 X_train_balanced, y_train_balanced = smote.fit_resample(X_train, y_train)
5
6 # Comentário explicativo da SVC (não executável, apenas informativo)
7 # SVC(
8 #     C=1,
9 #     kernel='rbf',
10 #     degree=3,
11 #     gamma='scale',
12 #     coef0=0,
13 #     shrinking=True,
14 #     probability=False,
15 #     tol=0.001,
16 #     cache_size=200,
17 #     class_weight=None,
18 #     verbose=False,
19 #     max_iter=-1,
20 #     decision_function_shape='ovr',
21 #     break_ties=False,
22 #     random_state=None
23 # )
24
25 # Treinar o classificador SVM com os dados balanceados
26 svm_classifier_balanced = SVC(kernel='rbf')
27 svm_classifier_balanced.fit(X_train_balanced, y_train_balanced)
28
29 # Fazer previsões no conjunto de teste
30 y_pred_balanced = svm_classifier_balanced.predict(X_test)
31
32 # Calcular e exibir a acurácia após o balanceamento
33 balanced_accuracy = accuracy_score(y_test, y_pred_balanced)
34 print("Acurácia na Base de Teste após o Balanceamento:", balanced_accuracy)
35
36 # Exibir a matriz de confusão
37 balanced_conf_matrix = confusion_matrix(y_test, y_pred_balanced)

```

```

38 print("Matriz de Confusão - Dados de Teste após o Balanceamento:")
39 print(balanced_conf_matrix)
40
41 # Exibir o relatório de classificação detalhado
42 print(classification_report(y_test, y_pred_balanced))

```

TABELA 28 – MATRIZ DE CONFUSÃO E MÉTRICAS DE DESEMPENHO DO CLASSIFICADOR SVM APÓS O BALANCEAMENTO COM SMOTE

Classe real	Classe prevista 0	Classe prevista 1	Precisão	Revocação	F1-Score	Suporte
Não Spam (0)	483	214	0,75	0,69	0,72	697
Spam (1)	157	297	0,58	0,65	0,62	454
Acurácia geral	0,68 (com 1151 amostras)					
Média macro	–		0,67	0,67	0,67	1151
Média ponderada	–		0,69	0,68	0,68	1151

FONTE: Elaboração do autor com base nos resultados obtidos (2024).

Melhoria do desempenho do classificador SVM com normalização e seleção de características

Nesta etapa, o desempenho do classificador *Support Vector Machine* (SVM) é aprimorado por meio da aplicação de técnicas adicionais de pré-processamento. Primeiramente, os dados balanceados são normalizados utilizando o método *Min-Max Scaling*, que transforma os valores das características para um intervalo fixo, geralmente entre 0 e 1, preservando a proporcionalidade dos dados. Em seguida, é aplicada a técnica de seleção de características *SelectKBest*, com base na função estatística *ANOVA F-value* (`f_classif`), visando reduzir a dimensionalidade e manter apenas as variáveis mais relevantes para o modelo.

Com esses aprimoramentos, o classificador SVM é treinado novamente e os resultados obtidos — como acurácia, matriz de confusão e métricas de desempenho — são avaliados e comparados às etapas anteriores, permitindo analisar os ganhos efetivos dessas técnicas no desempenho do modelo.

```

1 # Normalização Min-Max dos dados de treino balanceados
2 # Transforma os dados para o intervalo [0, 1]
3 scaler = MinMaxScaler()
4 X_train_scaled = scaler.fit_transform(X_train_balanced) # Ajusta e transforma os
   dados de treino
5 X_test_scaled = scaler.transform(X_test) # Aplica a mesma transformação aos
   dados de teste
6

```

```

7 # Seleção das 20 características mais relevantes com base no teste ANOVA F-value
8 selector = SelectKBest(f_classif, k=20)
9 X_train_selected = selector.fit_transform(X_train_scaled, y_train_balanced) #
    Ajusta e transforma os dados de treino
10 X_test_selected = selector.transform(X_test_scaled) # Aplica a transformação nos
    dados de teste
11
12 # Treinar o classificador SVM com os dados balanceados, normalizados e com
    seleção de características
13 svm_classifier_improved = SVC(kernel='rbf')
14 svm_classifier_improved.fit(X_train_selected, y_train_balanced)
15
16 # Realizar previsões usando os dados de teste processados
17 y_pred_improved = svm_classifier_improved.predict(X_test_selected)
18
19 # Calcular a acurácia final após as melhorias no pipeline
20 improved_accuracy = accuracy_score(y_test, y_pred_improved)
21 print("Acurácia na Base de Teste após Melhorias:", improved_accuracy)
22
23 # Exibir a matriz de confusão
24 improved_conf_matrix = confusion_matrix(y_test, y_pred_improved)
25 print("Matriz de Confusão - Dados de Teste após Melhorias:")
26 print(improved_conf_matrix)
27
28 # Exibir o relatório detalhado com precisão, revocação e F1-score
29 print(classification_report(y_test, y_pred_improved))

```

TABELA 29 – MATRIZ DE CONFUSÃO E MÉTRICAS DE DESEMPENHO DO CLASSIFICADOR SVM APÓS NORMALIZAÇÃO E SELEÇÃO DE CARACTERÍSTICAS

Classe real	Classe prevista 0	Classe prevista 1	Precisão	Revocação	F1-Score	Suporte
Não Spam (0)	658	39	0,92	0,94	0,93	697
Spam (1)	56	398	0,91	0,88	0,89	454
Acurácia geral	0,92 (com 1151 amostras)					
Média macro	–		0,92	0,91	0,91	1151
Média ponderada	–		0,92	0,92	0,92	1151

FONTE: Elaboração do autor com base nos resultados obtidos (2024).

Conclusão

Neste trabalho, buscou-se melhorar o desempenho da classificação de emails em spam por meio da aplicação de técnicas de pré-processamento, balanceamento e seleção de características no

conjunto de dados Spambase. Os resultados mostraram que o balanceamento dos dados com SMOTE contribuiu para maior equilíbrio entre as classes, ainda que com leve redução na acurácia geral. A utilização conjunta de normalização e seleção das melhores variáveis proporcionou um ganho significativo no desempenho do classificador, elevando a acurácia para mais de 91%. Esses resultados evidenciam que o aprimoramento do pré-processamento e a seleção criteriosa de atributos são fundamentais para melhorar o desempenho de métodos de redução de dimensionalidade aplicados a conjuntos de dados reais, confirmando o objetivo do trabalho.

APÊNDICE 7 – 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 PDF:

- O conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada);
- Explicação do contexto e o público-alvo da visualização de dados e do storytelling que será desenvolvido;
- A visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha) explicando a escolha do tipo de visualização e da ferramenta usada; (50 pontos)
- A descrição da narrativa/storytelling dessa visualização de dados. (50 pontos)

B - RESOLUÇÃO

Evolução da Indústria Automotiva no Brasil

Base de Dados

Os dados trabalhados representam uma série temporal relacionada ao setor automotivo no Brasil, com registros desde janeiro de 1957 até dezembro de 2019. As colunas analisadas foram:

- Data: período da observação (mês/ano).
- Emplacamento Total: número total de veículos emplacados.
- Emplacamento Nacionais: número de veículos nacionais emplacados.
- Emplacamento Importados: número de veículos importados emplacados.
- Produção: número de veículos produzidos.
- Exportação: número de veículos exportados.

Base de dados https://anfavea.com.br/docs/SeriesTemporais_Autoveiculos.xlsm ¹.

¹ Acesso realizado no dia 05/03/2025 as 10h.

FIGURA 23 – TABELA DOS DADOS DOS VEÍCULOS

Data	Emplacamento Total	Emplacamento Nacionais	Emplacamento Importados	Produção	Exportação
jan-1957	2009	2009	0	2317	0
fev-1957	2091	2091	0	2144	0
mar-1957	2802	2802	0	2660	0
abr-1957	2530	2530	0	2449	0
mai-1957	2889	2889	0	2788	0
jun-1957	2913	2913	0	2740	0
jul-1957	3053	3053	0	2846	0
ago-1957	2922	2922	0	2740	0
set-1957	2373	2373	0	2635	0
out-1957	2170	2170	0	2646	0
nov-1957	1793	1793	0	2264	0
dez-1957	3432	3432	0	2313	0
jan-1958	4712	4712	0	4633	0
fev-1958	4594	4594	0	4333	0
mar-1958	5612	5612	0	5308	0
abr-1958	4956	4956	0	4920	0
mai-1958	5535	5535	0	5563	0
jun-1958	5867	5867	0	5453	0
jul-1958	5632	5632	0	5566	0
ago-1958	5476	5476	0	5561	0
set-1958	4689	4689	0	5202	0
out-1958	4297	4297	0	5264	0
nov-1958	3501	3501	0	4584	0
dez-1958	6055	6055	0	4596	0

FONTE: anfavea.com.

Contexto

A indústria automotiva brasileira desempenha um papel crucial na economia nacional, destacando-se como um dos setores mais relevantes em termos de produção, exportação e geração de empregos. Ao longo das décadas, essa indústria passou por diversas transformações, impulsionadas por fatores econômicos, políticos e tecnológicos. Desde a chegada das primeiras montadoras na década de 1950, quando empresas como a Volkswagen e a General Motors se estabeleceram no país, até a ascensão da mobilidade elétrica e digitalização no século XXI, a produção de veículos no Brasil reflete tanto momentos de crescimento quanto períodos de crise (Parasmo, 2024) (Revista Pegada - UNESP, 2021) (TG Poli, 2023). Este estudo busca compreender essa evolução por meio de uma análise das séries temporais de produção, emplacamento e exportação de veículos no Brasil. Lançando mão de ferramentas de visualização gráfica, o objetivo é identificar padrões, tendências e fatores que influenciaram a indústria ao longo dos anos, além de projetar possíveis cenários futuros para o setor.

Público-Alvo

Formuladores de Políticas Públicas

- Governos e entidades reguladoras interessadas no impacto das políticas econômicas sobre a indústria automotiva e na criação de incentivos para o setor.
- Organizações que desenvolvem políticas de mobilidade urbana e sustentabilidade, considerando a transição para veículos elétricos e outras inovações.

Visualização do dados

Gráfico de Linhas

Neste trabalho, os gráficos de linha foram escolhidos para a análise da série temporal automotiva por serem a opção ideal para visualizar a evolução de variáveis ao longo do tempo. Essa escolha se justifica pelos seguintes motivos:

- Representação Clara da Tendência Temporal

Como a análise dos dados está distribuída ao longo de várias décadas, os gráficos de linha permitem visualizar claramente as tendências de crescimento, queda ou estabilidade das variáveis (produção, emplacamento e exportação).

- Facilidade de Comparação Entre as Séries

Utilizar múltiplas linhas no mesmo gráfico permite comparar diferentes variáveis ao longo do tempo. No caso deste estudo, é possível ver simultaneamente como a produção, o emplacamento e a exportação se comportaram em cada período.

- Destaque para Padrões e Anomalias

Com uma linha conectando os pontos ao longo dos anos, é possível identificar facilmente padrões como ciclos de crescimento e crise, bem como eventuais anomalias nos dados, como quedas bruscas durante recessões econômicas ou picos de crescimento em períodos de expansão.

- Adequação para Séries Temporais

Gráficos de linha são amplamente utilizados para representar séries temporais, pois enfatizam a continuidade dos dados ao longo do tempo. Já que análise tem um forte componente histórico, este tipo de gráfico é a forma mais intuitiva de visualizar as informações.

Ferramenta Escolhida

Os gráficos foram gerados com a biblioteca Matplotlib em Python, uma das ferramentas mais utilizadas para visualização de dados, especialmente séries temporais. Sua popularidade se deve à capacidade de criar gráficos personalizáveis e de alta qualidade. A seguir, destacam-se alguns motivos para a escolha dessa ferramenta:

- Flexibilidade – Permite personalizar cores, rótulos, legendas e escalas para melhor compreensão dos dados.
- Adequação para Séries Temporais – Funciona bem para gráficos de linha, que são ideais para acompanhar tendências ao longo do tempo.
- Compatibilidade – Pode ser usado junto com outras bibliotecas como Pandas e Seaborn para análises mais avançadas.

- Conhecimento do Autor – A escolha do Matplotlib também se deve à familiaridade e experiência do autor com a ferramenta, garantindo um uso eficiente e a geração de visualizações bem estruturadas.

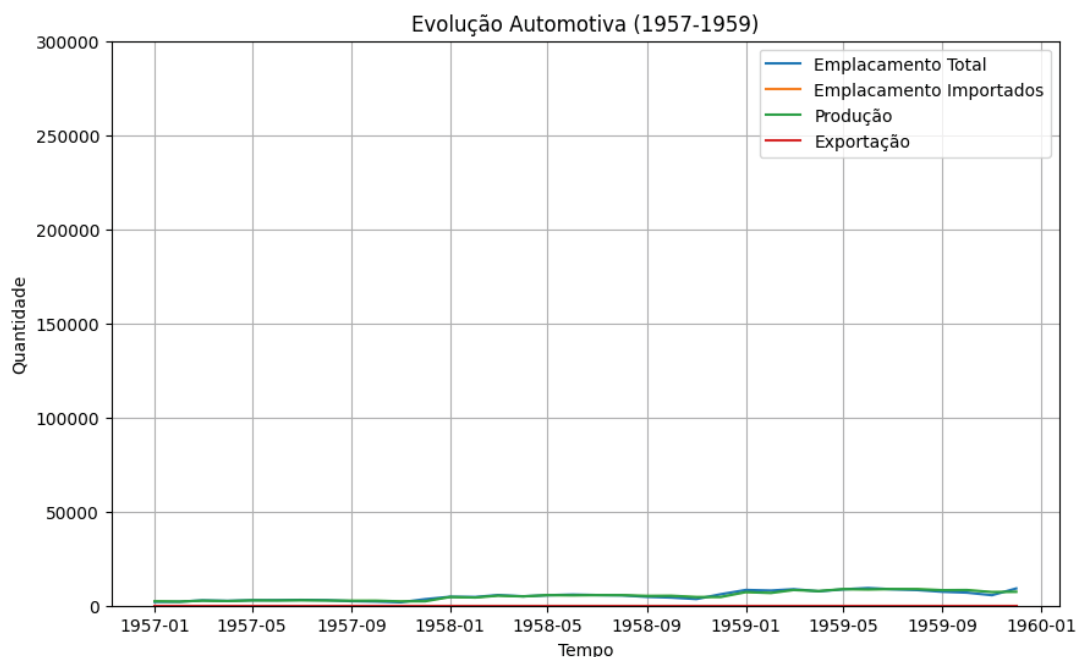
Análise por Décadas

Evolução da Indústria Automotiva no Brasil

1957-1959: O Início da Indústria

- A produção e o emplacamento eram relativamente baixos, pois a indústria estava em seus primeiros anos.
- O governo Juscelino Kubitschek incentivou a criação de montadoras nacionais, resultando em um crescimento inicial.
- Exportações eram praticamente inexistentes, pois o foco estava no mercado interno.

FIGURA 24 – INÍCIO DA INDÚSTRIA AUTOMOTIVA NO BRASIL

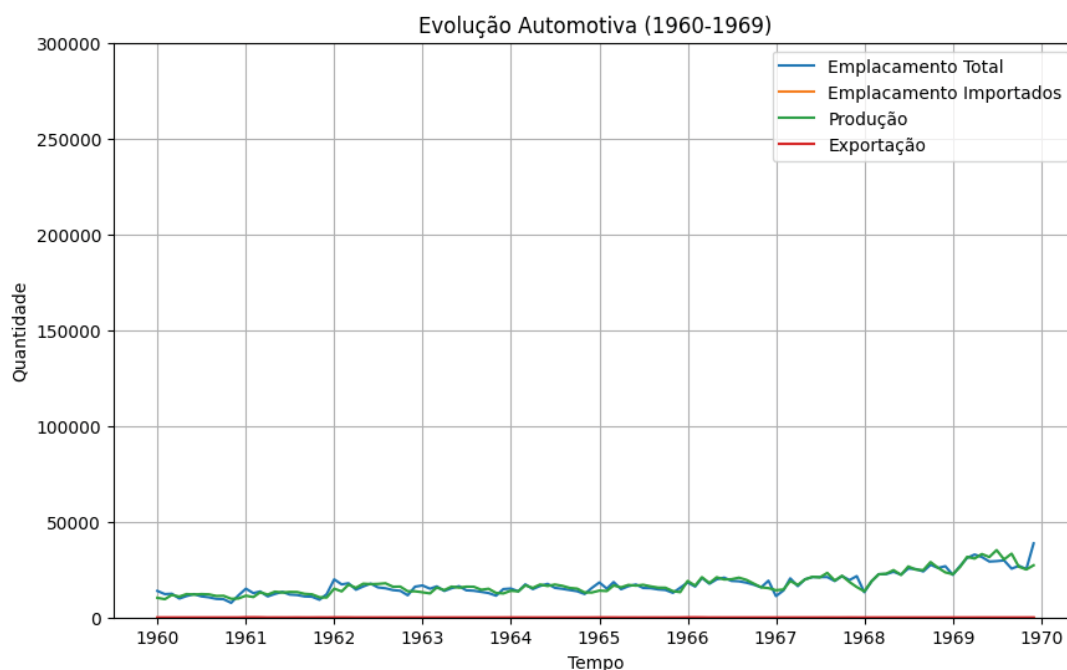


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025).

1960-1969: Consolidação e Crescimento Moderado

- Produção e emplacamento cresceram à medida que mais montadoras se estabeleceram no Brasil.
- A vinda da Volkswagen e da Ford para o país impulsionou o setor.
- O regime militar (1964) trouxe incentivos à industrialização, mas a economia ainda era instável.

FIGURA 25 – CONSOLIDAÇÃO DA INDÚSTRIA AUTOMOTIVA NO BRASIL

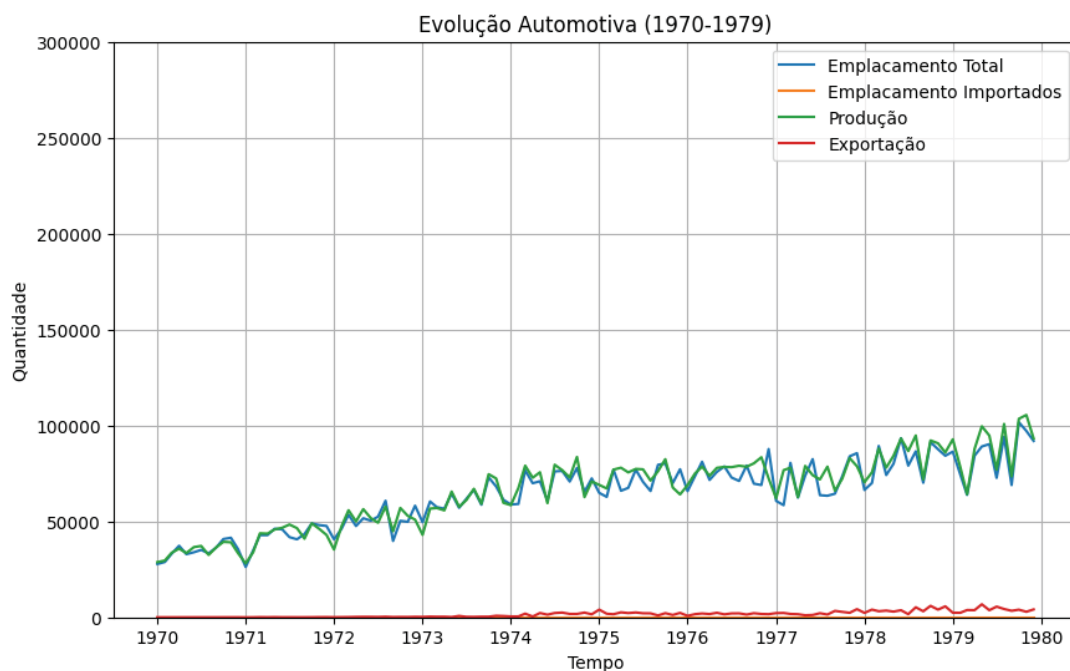


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025).

1970-1979: Expansão e Crise do Petróleo

- Crescimento significativo na produção e no número de emplacamentos.
- Criação de modelos icônicos como o Fusca e a Brasília, que dominaram o mercado interno.
- A crise do petróleo (1973) aumentou o custo dos combustíveis, reduzindo o crescimento da demanda.
- Exportações começam a aumentar com acordos comerciais internacionais.

FIGURA 26 – EXPANSÃO DA INDÚSTRIA AUTOMOTIVA NO BRASIL E CRISE DO PETRÓLEO

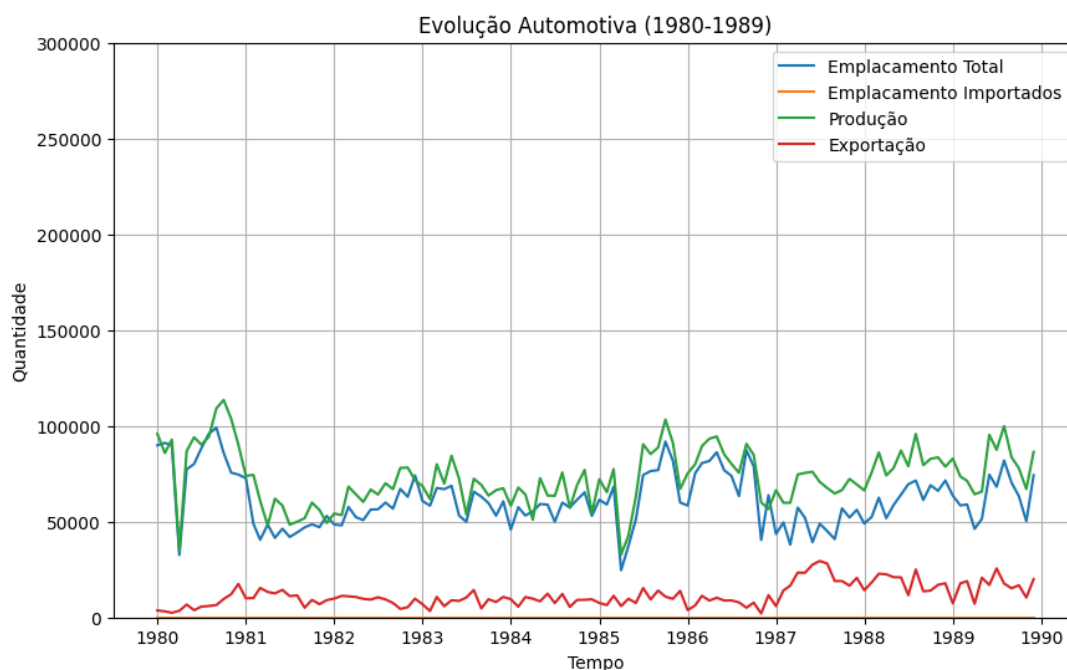


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025) .

1980-1989: Crise Econômica e Estagnação

- A hiperinflação e a recessão afetaram a venda de veículos no Brasil.
- A produção se manteve estável, mas o mercado interno foi impactado pela perda de poder de compra da população.
- Exportações ajudaram a manter a indústria funcionando, mas ainda eram baixas.

FIGURA 27 – CRISE ECONÔMICA E ESTAGNAÇÃO DA INDÚSTRIA AUTOMOTIVA NO BRASIL

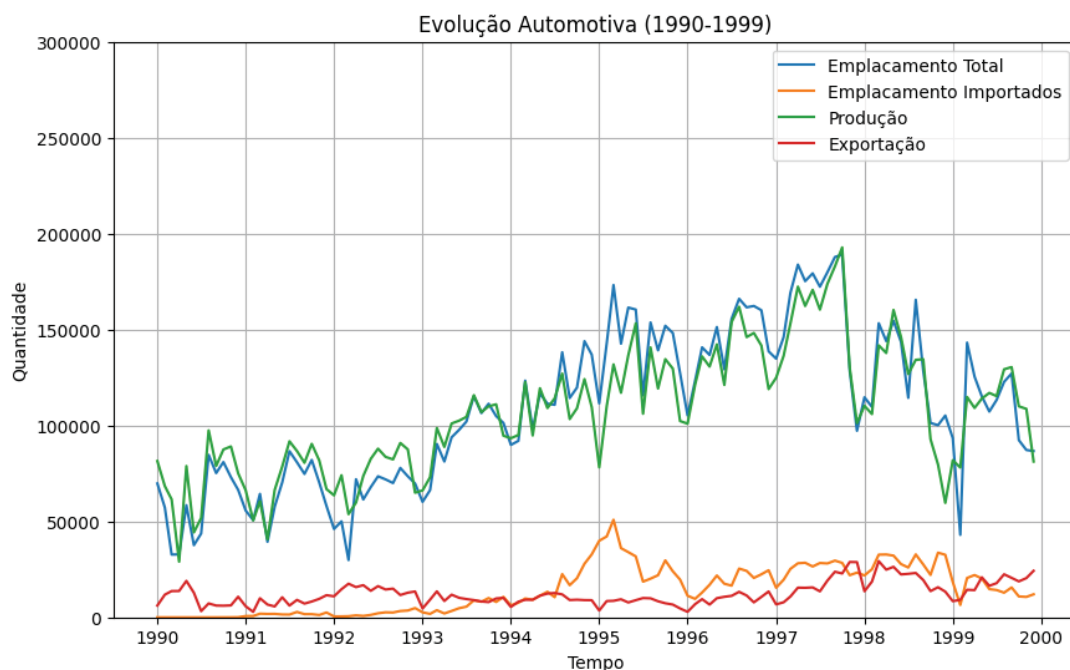


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025).

1990-1999: Abertura Econômica e Concorrência

- O governo Collor removeu barreiras para importação, resultando no aumento do número de veículos estrangeiros no Brasil.
- A indústria nacional precisou se modernizar para competir com os importados.
- Produção cresceu devido à entrada de novas montadoras e fábricas no Brasil.
- A exportação aumentou, impulsionada pelo Mercosul e pela globalização da indústria.

FIGURA 28 – ABERTURA ECONÔMICA E CONCORRÊNCIA NA INDÚSTRIA AUTOMOTIVA NO BRASIL

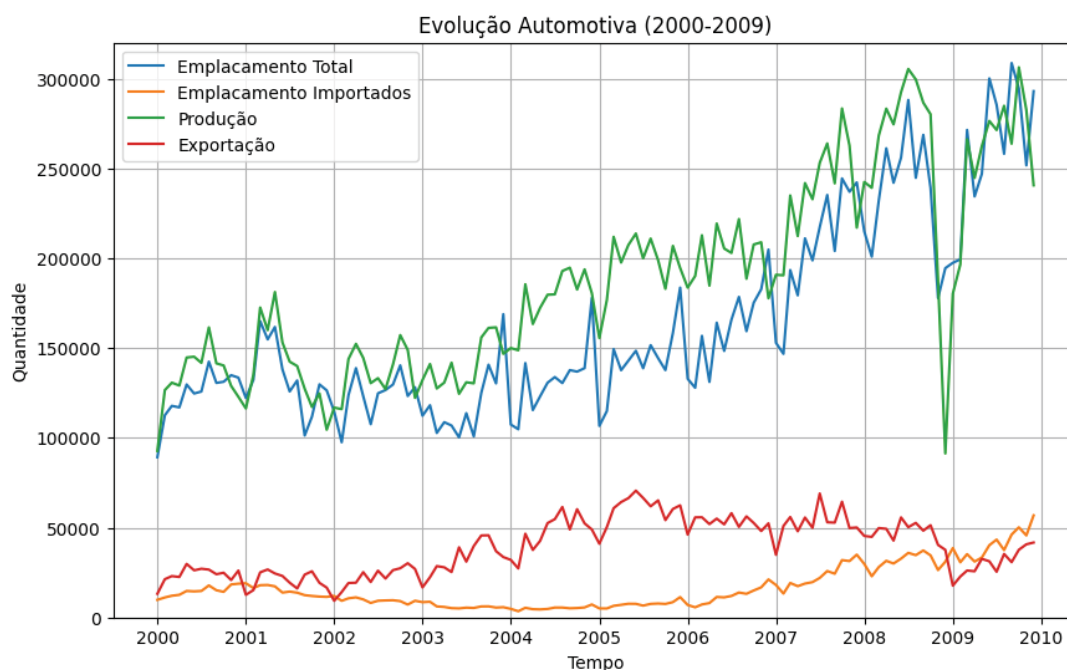


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025) .

2000-2009: Recordes de Produção

- O Brasil vive um boom econômico, aumentando a demanda por veículos.
- A indústria bate recordes de produção e exportação, impulsionada pelo crescimento da classe média.
- Acordos comerciais fortaleceram as exportações para Argentina, México e outros países da América Latina.

FIGURA 29 – RECORDES DE PRODUÇÃO DA INDÚSTRIA AUTOMOTIVA NO BRASIL

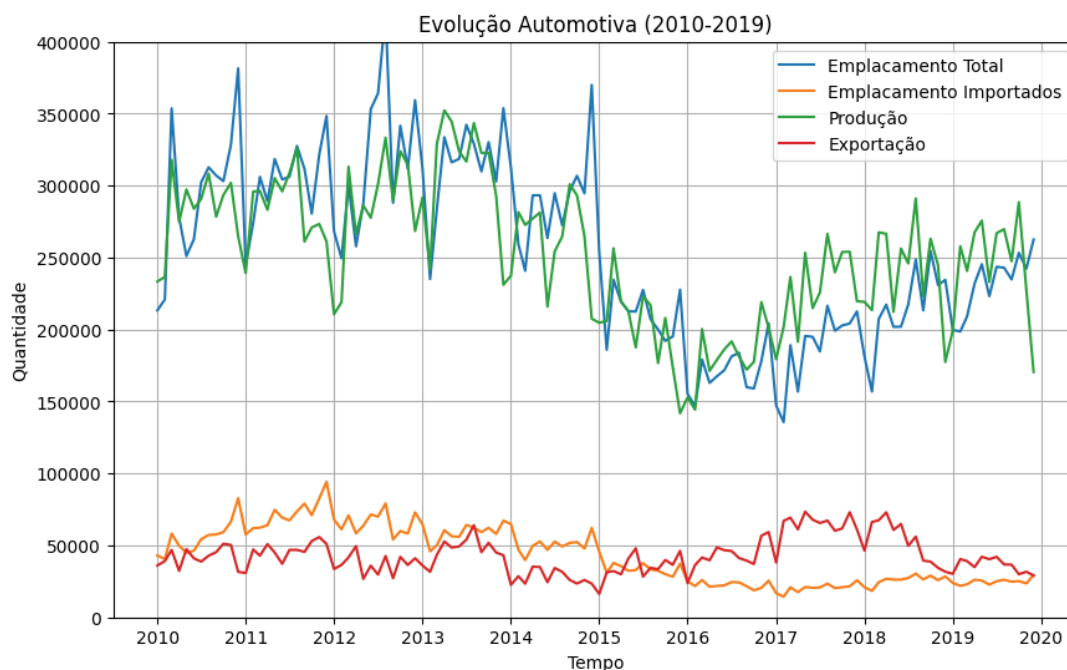


FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025).

2010-2019: Crise Econômica e Novas Tendências

- A crise econômica de 2014-2016 impactou o setor, reduzindo as vendas e a produção.
- A exportação se manteve forte, mas não foi suficiente para compensar a queda na demanda interna.
- A indústria começou a se digitalizar, com avanços tecnológicos e novas tendências de mobilidade (carros elétricos e compartilhados).

FIGURA 30 – CRISE ECONÔMICA E NOVAS TENDÊNCIAS NA INDÚSTRIA AUTOMOTIVA NO BRASIL



FONTE: Elaboração do autor com base na tabela de dados carregada no início (2025).

Conclusão

A análise demonstra que o setor automotivo é altamente sensível às condições econômicas e políticas. Momentos de estabilidade favorecem o crescimento da produção e do emplacamento, enquanto períodos de crise resultam em retração e adaptação do mercado. A exportação, que antes era apenas uma alternativa secundária, tornou-se essencial para garantir a sustentabilidade da indústria, principalmente durante períodos de recessão no mercado interno.

Para o futuro, a indústria automotiva brasileira enfrenta novos desafios, como a transição para veículos elétricos e híbridos, a digitalização das fábricas e as mudanças no comportamento do consumidor. O setor precisará investir em inovação e sustentabilidade para continuar competitivo no cenário global.

APÊNDICE 8 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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.

TABELA 30 – VEÍCULOS

Técnica	Parâmetros	Acurácia	Matriz de Confusão
KNN	$k = XX$		
RNA – Hold-out	size = XX, decay = XX		
RNA – CV	size = XX, decay = XX		
SVM – Hold-out	$C = XX, \sigma = XX$		
SVM – CV	$C = XX, \sigma = XX$		
RF – Hold-out	mtry = XX		
RF – CV	mtry = XX		

FONTE: Tarefa proposta

TABELA 31 – DIABETES

Técnica	Parâmetros	Acurácia	Matriz de Confusão
KNN	$k = XX$		
RNA – Hold-out	size = XX, decay = XX		
RNA – CV	size = XX, decay = XX		
SVM – Hold-out	$C = XX, \sigma = XX$		
SVM – CV	$C = XX, \sigma = XX$		
RF – Hold-out	mtry = XX		
RF – CV	mtry = XX		

FONTE: Tarefa proposta

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.
 - A lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

TABELA 32 – ADMISSÃO

Técnica	Parâmetros	R ²	Syx	Pearson	RMSE	MAE
KNN	$k = XX$					
RNA – Hold-out	size = XX, decay = XX					
RNA – CV	size = XX, decay = XX					
SVM – Hold-out	$C = XX, \sigma = XX$					
SVM – CV	$C = XX, \sigma = XX$					
RF – Hold-out	mtry = XX					
RF – CV	mtry = XX					

FONTE: Tarefa proposta

TABELA 33 – BIOMASSA

Técnica	Parâmetros	R ²	Syx	Pearson	RMSE	MAE
KNN	$k = XX$					
RNA – Hold-out	size = XX, decay = XX					
RNA – CV	size = XX, decay = XX					
SVM – Hold-out	$C = XX, \sigma = XX$					
SVM – CV	$C = XX, \sigma = XX$					
RF – Hold-out	mtry = XX					
RF – CV	mtry = XX					

FONTE: Tarefa proposta

Agrupamento

Veículo, lista de clusters gerados:

- 10 primeiras linhas do arquivo com o cluster correspondente.
- Usa 10 clusters no experimento.
- Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

Regras de associação

Musculação:

- Regras geradas com uma configuração de Suporte e Confiança.
- Colocar a lista de comandos emitidos no RStudio para conseguir os resultados obtidos.

B - RESOLUÇÃO

A fim de assegurar a reprodutibilidade dos resultados, todos os experimentos foram executados utilizando a seed 202460.

Classificação

Veículos

O experimento a seguir tem como objetivo comparar o desempenho de diferentes técnicas de classificação aplicadas a um conjunto de dados sobre veículos. Os resultados estão organizados em ordem decrescente de acurácia, destacando a técnica com melhor desempenho. Em seguida, são apresentadas as predições para três novos casos utilizando o modelo com maior acurácia, bem como os comandos executados em R para obtenção desses resultados.

TABELA 34 – RESULTADOS DOS CLASSIFICADORES PARA VEÍCULOS

Técnica	Parâmetro	Acurácia	Matriz de Confusão				
SVM – CV	C=100, Sigma=0.015	84,43%	Predição	bus	opel	saab	van
			bus	41	0	1	1
			opel	0	30	10	0
			saab	0	10	32	0
			van	2	2	0	38
RNA – CV	size=31, decay=0.7	83,83%	Predição	bus	opel	saab	van
			bus	41	0	0	0
			opel	0	29	12	0
			saab	1	10	31	0
			van	1	3	0	39
SVM – Hold-out	C=1, Sigma=0.07900225	76,65%	Predição	bus	opel	saab	van
			bus	40	0	0	0
			opel	0	23	16	0
			saab	0	14	26	0
			van	3	5	1	39
RF – Hold-out	mtry=7	74,25%	Predição	bus	opel	saab	van
			bus	41	0	1	0
			opel	0	22	15	0
			saab	0	17	22	0
			van	2	3	5	39
RF – CV	mtry=10	74,25%	Predição	bus	opel	saab	van
			bus	41	1	2	0
			opel	0	24	16	0
			saab	0	14	20	0
			van	2	3	5	39
KNN	k=XX	61,18%	Predição	bus	opel	saab	van
			bus	29	6	6	3
			opel	0	11	15	2
			saab	2	26	21	0
			van	2	3	1	43
RNA – Hold-out	size=5, decay=0.1	58,08%	Predição	bus	opel	saab	van
			bus	27	11	5	2
			opel	9	24	28	0
			saab	3	3	9	0
			van	4	4	1	37

FONTE: Resultados gerados pelo autor a partir da execução dos experimentos (2024).

Os classificadores SVM e RNA, ambos com validação cruzada, apresentaram os melhores desempenhos, com acurácias de 84,43% e 83,83%, respectivamente. Em contrapartida, modelos com hold-out, especialmente a RNA, tiveram desempenho inferior, destacando a vantagem da validação cruzada na avaliação dos algoritmos.

Comandos utilizados na execução do experimento de classificação - base de veículos.

Nesta parte, são apresentados os comandos executados em R que possibilitaram a obtenção dos resultados deste trabalho. O código cobre desde a leitura e pré-processamento dos dados até o treinamento e avaliação dos modelos de classificação, com destaque para a técnica de Máquinas de Vetores de Suporte (SVM). São utilizados procedimentos de validação cruzada, ajuste de hiperparâmetros e análise dos resultados por meio da matriz de confusão e métricas de desempenho.

```

1  ### Pacotes utilizados no experimento
2  library("caret") # Pacote principal para treinamento e avaliação de modelos
3  library(mice)    # Para tratamento de dados ausentes (não utilizado neste trecho)
4
5  ### SEED para garantir reprodutibilidade dos resultados
6  set.seed(202460)
7
8  ### Leitura dos dados
9  setwd("./Veículos")
10 dados <- read.csv("6 - Veiculos - Dados.csv")
11
12 ### Remoção da variável 'a', que representa o índice (não relevante para o modelo)
13 dados$a <- NULL
14
15 ### Divisão dos dados em treino (80%) e teste (20%)
16 indices <- createDataPartition(dados$po, p=0.80, list=FALSE)
17 treino <- dados[indices, ]
18 teste <- dados[-indices, ]
19
20 ### Validação cruzada com SVM
21
22 # Define o método de treino como validação cruzada com 10 folds
23 ctrl <- trainControl(method = "cv", number = 10)
24
25 # Treina um modelo SVM com kernel radial (RBF), usando validação cruzada
26 svm <- train(
27   po ~ .,
28   data = treino,
29   method = "svmRadial",
30   trControl = ctrl
31 )
32
33 ### Avaliação do modelo com o conjunto de teste

```

```

34 predict.svm <- predict(svm, teste)
35 confusionMatrix(predict.svm, as.factor(teste$po)) # Matriz de confusão
36
37 ### Ajuste de hiperparâmetros (C e sigma)
38
39 # Define uma grade de valores para busca de hiperparâmetros
40 tuneGrid <- expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(0.01, 0.015, 0.2))
41
42 # Re-treina o modelo com busca por melhores hiperparâmetros usando validação cruzada
43 svm <- train(
44   po ~ .,
45   data = treino,
46   method = "svmRadial",
47   trControl = ctrl,
48   tuneGrid = tuneGrid
49 )
50
51 ### Avaliação final do modelo com o conjunto de teste
52 predict.svm <- predict(svm, teste)
53 confusionMatrix(predict.svm, as.factor(teste$po))
54
55 ### Predição de novos casos
56 # Leitura do arquivo contendo novos exemplos
57 dados_novos_casos <- read.csv("6 - Veiculos - Novos Casos.csv")
58 dados_novos_casos$a <- NULL # Remove a coluna de índice, se presente
59
60 # Realiza predições usando o modelo final
61 predict.svm <- predict(svm, dados_novos_casos)
62
63 # Junta as predições aos dados originais
64 resultado <- cbind(dados_novos_casos, predict.svm)
65
66 # Remove a variável 'po' original, se presente
67 resultado$po <- NULL

```

TABELA 35 – PREDIÇÃO DE NOVOS RESULTADOS PARA O CONJUNTO DE DADOS DE VEÍCULOS

Entrada																			
a	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
2	92	40	85	140	57	9	149	45	19	143	169	329	157	72	9	14	185	199	?
3	103	49	107	208	66	10	207	32	23	158	222	632	221	73	14	9	186	196	?
6	105	58	107	171	50	6	255	26	28	169	274	959	265	85	5	9	187	183	?
Resultado																			
a	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
2	92	40	85	140	57	9	149	45	19	143	169	329	157	72	9	14	185	199	van
3	103	49	107	208	66	10	207	32	23	158	222	632	221	73	14	9	186	196	saab
6	105	58	107	171	50	6	255	26	28	169	274	959	265	85	5	9	187	183	bus

FONTE: Elaborado pelo autor com base nos dados experimentais (2024).

Diabetes

Este experimento tem como objetivo aplicar diferentes técnicas de classificação ao conjunto de dados relacionado a diagnósticos de diabetes, avaliando o desempenho de cada modelo com base na métrica de acurácia. Assim como no experimento anterior, os resultados são organizados em uma tabela ordenada pela acurácia de forma decrescente, permitindo a identificação da técnica mais eficaz.

Após a apresentação da tabela, são incluídos os seguintes elementos complementares: (i) a predição de novos casos utilizando o modelo com melhor desempenho; e (ii) a lista de comandos utilizados em R para execução do experimento.

TABELA 36 – RESULTADOS DOS CLASSIFICADORES PARA DIABETES

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RNA – Hold-out	size=3 decay=0.1	81,05%	Predição neg pos neg 89 18 pos 11 35
SVM – Hold-out	C=0.25 Sigma=0.1330529	80,39%	Predição neg pos neg 95 25 pos 5 28
SVM – CV	C=1 Sigma=0.015	80,39%	Predição neg pos neg 94 24 pos 6 29
RF – Hold-out	mtry=2	78,43%	Predição neg pos neg 87 20 pos 13 33
KNN	k=9	75,97%	Predição neg pos neg 90 20 pos 17 27
RNA – CV	size=21 decay=0.4	73,86%	Predição neg pos neg 78 18 pos 22 35

FONTE: Elaboração do autor a partir dos resultados obtidos (2024).

Observa-se que a técnica *RNA – Hold-out*, com os parâmetros *size=3* e *decay=0.1*, apresentou o melhor desempenho entre os modelos testados, atingindo uma acurácia de 81,05%. Apesar de pequenas variações nos resultados, os demais modelos demonstraram desempenhos competitivos, com destaque para as técnicas baseadas em *SVM*. A matriz de confusão permite visualizar a capacidade de cada modelo em distinguir corretamente as classes positivas e negativas.

Comandos utilizados na execução do experimento de classificação - base de diagnósticos de diabetes.

A seguir, são apresentados os comandos utilizados na linguagem R para a execução do experimento de classificação aplicado à base de dados sobre diagnósticos de diabetes. O modelo foi treinado com validação Hold-out, e sua acurácia foi avaliada por meio da matriz de confusão gerada a partir do conjunto de teste. Além disso, foram realizadas predições para novos casos com base no modelo treinado.

```

1  ### Carregamento das bibliotecas utilizadas
2  library(caret)
3  library(mlbench)
4
5  ### Leitura da base de dados
6  setwd("./10 - Diabetes")
7  dados <- read.csv("10 - Diabetes - Dados.csv")
8
9  ### Remoção da variável de índice (coluna num)
10 dados$num <- NULL
11
12 ### Visualização inicial dos dados
13 View(dados)
14
15 ### Separação dos dados em conjunto de treino (80%) e teste (20%)
16 set.seed(202460) # Semente para reprodutibilidade
17 indices <- createDataPartition(dados$diabetes, p = 0.80, list = FALSE)
18 treino <- dados[indices, ]
19 teste <- dados[-indices, ]
20
21 ### Treinamento do modelo de Rede Neural Artificial (RNA) com validação Hold-out
22 set.seed(202460)
23 rna <- train(diabetes ~ ., data = treino, method = "nnet", trace = FALSE)
24 rna # Exibe os resultados do treinamento
25
26 ### Realização das predições com os dados de teste
27 predict.rna <- predict(rna, teste)
28
29 ### Avaliação do modelo com matriz de confusão
30 confusionMatrix(predict.rna, as.factor(teste$diabetes))
31
32 ### PREDIÇÃO DE NOVOS CASOS
33 dados_novos_casos <- read.csv("10 - Diabetes - Novos Casos.csv")
34 dados_novos_casos$num <- NULL # Remove a coluna de índice

```

```

35 View(dados_novos_casos)
36
37 #Novos casos
38 predict.rna <- predict(rna, dados_novos_casos) # Realiza a predição
39 dados_novos_casos$diabetes <- NULL # Remove a coluna de classe (se houver)
40 resultado <- cbind(dados_novos_casos, predict.rna) # Junta os dados com as predições
41
42 ### Visualização do resultado final
43 View(resultado)

```

TABELA 37 – PREDIÇÃO DE NOVOS CASOS PARA O EXPERIMENTO DE CLASSIFICAÇÃO - BASE DE DADOS DE DIABETES

Entrada									
num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	2	102	79	10	81	19,4	490	23	?
2	5	135	71	0	0	23,8	278	61	?
3	2	152	83	43	482	40,6	686	24	?
Resultado									
num	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes
1	2	102	79	10	81	19,4	490	23	pos
2	5	135	71	0	0	23,8	278	61	pos
3	2	152	83	43	482	40,6	686	24	pos

FONTE: Elaborado pelo autor com base nos dados experimentais (2024).

Regressão

Admissão

O presente experimento tem por objetivo avaliar o desempenho de diferentes técnicas de regressão aplicadas a um conjunto de dados relacionado a admissões. As variáveis preditoras incluem características relevantes ao perfil dos indivíduos, e a variável resposta representa um indicador contínuo associado ao processo de admissão.

A comparação entre os modelos será realizada com base no coeficiente de determinação (R^2), ordenando-se os resultados de forma decrescente. Serão apresentados: (i) a tabela de desempenho dos modelos, (ii) a predição de novos casos utilizando a técnica com maior R^2 , (iii) o gráfico de resíduos da técnica selecionada e (iv) a lista dos comandos utilizados na linguagem R para obtenção dos resultados.

TABELA 38 – RESULTADOS DO EXPERIMENTO DE REGRESSÃO - BASE DE ADMISSÕES

Técnica	Parâmetro	R^2	Syx	Pearson	RMSE	MAE
RF – Hold-out	mtry=2	0,7778	0,0718	0,8895	0,0688	0,0495
RF – CV	mtry=2	0,7764	0,0720	0,8889	0,0690	0,0496
RNA – CV	size=10, decay=0,1	0,7656	0,0738	0,8874	0,0707	0,0506
SVM – CV	C=0,5, Sigma=0,2054744	0,7378	0,0780	0,8715	0,0748	0,0502
SVM – Hold-out	C=0,5, Sigma=0,2054744	0,7378	0,0780	0,8715	0,0748	0,0502
RNA – Hold-out	size=5, decay=0,1	0,7115	0,0818	0,8529	0,0784	0,0582
KNN	k=9	0,6817	0,0860	0,8277	0,0824	0,0623

FONTE: Resultados obtidos pelo autor por meio de experimentos com diferentes técnicas de regressão aplicadas à base de admissões (2024).

Observa-se que a técnica *Random Forest* (RF) com validação do tipo *Hold-out* apresentou o melhor desempenho entre os modelos avaliados, com $R^2 = 0,7778$ e alto coeficiente de correlação de Pearson ($r = 0,8895$). Esses resultados indicam que o modelo é capaz de explicar uma parcela significativa da variabilidade da variável dependente, além de apresentar baixos valores de erro (RMSE e MAE). Portanto, a técnica RF com *Hold-out* é a mais adequada para o problema proposto com esta base de dados.

Comandos utilizados na execução do experimento de regressão - base de admissões

A seguir, apresenta-se a sequência de comandos utilizados na linguagem R para realizar o experimento de regressão com a base de dados sobre admissões. O modelo que apresentou o melhor desempenho foi o Random Forest, conforme os critérios de avaliação definidos. As métricas de desempenho, as predições e o gráfico de resíduos foram gerados com base nesse modelo.

```

1 # Carrega os pacotes necessários
2 library(e1071)
3 library(kernlab)
4 library(caret)
5 library(Metrics)
6
7 # Define o diretório de trabalho
8 setwd("./09 - Admissão")
9
10 # Lê os dados principais
11 dados <- read.csv("9 - Admissao - Dados.csv", header = TRUE)
12
13 # Remove a variável identificadora
14 dados$num <- NULL
15
16 # Define a semente aleatória

```

```

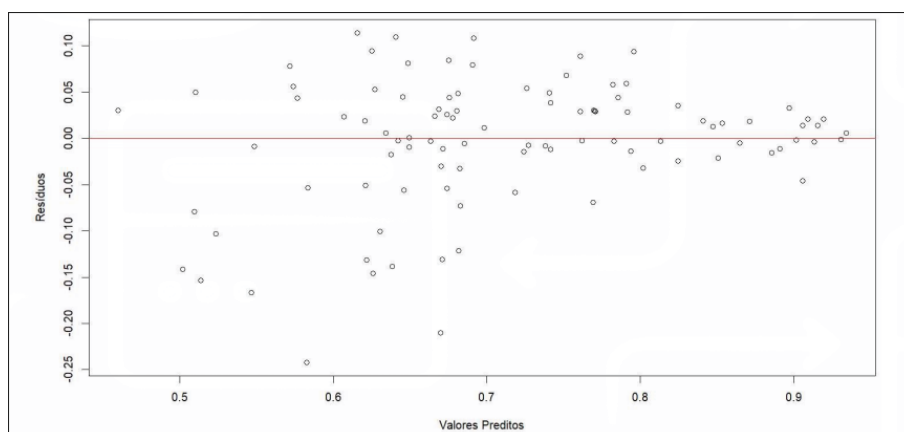
17 set.seed(202460)
18
19 # Cria os índices para divisão treino/teste (80% treino)
20 indices <- createDataPartition(dados$ChanceOfAdmit, p = 0.80, list = FALSE)
21
22 # Separa os dados em treino e teste
23 treino <- dados[indices, ]
24 teste <- dados[-indices, ]
25
26 # Treina o modelo Random Forest
27 set.seed(202460)
28 rf <- train(ChanceOfAdmit ~ ., data = treino, method = "rf")
29
30 # Realiza predições com os dados de teste
31 predicoes.rf <- predict(rf, teste)
32
33 # Calcula o RMSE
34 rmse(teste$ChanceOfAdmit, predicoes.rf)
35
36 # Função para cálculo do R2
37 r2 <- function(predito, observado) {
38   return(1 - (sum((predito - observado)^2) / sum((observado - mean(observado))^2)))
39 }
40
41 # Aplica o cálculo do R2
42 r2(predicoes.rf, teste$ChanceOfAdmit)
43
44 # Prepara variáveis para cálculo do Syx
45 observado <- teste$ChanceOfAdmit
46 predito <- predicoes.rf
47 n <- length(observado)      # Número de observações
48 p <- ncol(teste) - 1        # Número de variáveis preditoras (exclui a target)
49
50 # Calcula o Syx
51 Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
52 Syx
53
54 # Calcula o coeficiente de correlação de Pearson
55 correlacao <- cor(observado, predito)

```



```
56 correlacao
57
58 # Calcula o erro absoluto médio (MAE)
59 mae <- mean(abs(observado - predito))
60 mae
61
62 # Calcula os resíduos
63 residuos <- observado - predito
64
65 # Gera o gráfico de resíduos
66 plot(predito, residuos,
67       xlab = "Valores Preditos",
68       ylab = "Resíduos",
69       main = "Gráfico de Resíduos")
70 abline(h = 0, col = "red")
71
72 # Lê os dados com novos casos
73 dados_novos_casos <- read.csv("9 - Admissao - Novos Casos.csv")
74
75 # Visualiza os novos dados
76 View(dados_novos_casos)
77
78 # Realiza predições com novos dados
79 predict.rf <- predict(rf, dados_novos_casos)
80
81 # Combina os dados com as predições
82 resultado <- cbind(dados_novos_casos, predict.rf)
83
84 # Visualiza o resultado final
85 View(resultado)
```

FIGURA 31 – GRÁFICO DE RESÍDUOS - BASE ADMISSÃO



FONTE: Elaborado pelo autor (2024).

TABELA 39 – PREDIÇÃO DE NOVOS CASOS PARA O EXPERIMENTO DE REGRESSÃO - BASE DE ADMISSÃO

Entrada							
GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
318	110	3	4	3	8.8	0	?
313	107	2	2.5	2	8.5	1	?
339	119	5	4.5	4	9.7	0	?
Resultado							
GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
318	110	3	4	3	8.8	0	0.6886672
313	107	2	2.5	2	8.5	1	0.6011606
339	119	5	4.5	4	9.7	0	0.9013246

FONTE: Elaborado pelo autor (2024).

Conforme apresentado na Tabela 38, o modelo de *Random Forest* com particionamento *Hold-out* e parâmetro `mtry = 2` apresentou o melhor desempenho entre as técnicas avaliadas, com um coeficiente de determinação R^2 de aproximadamente 0,778. Utilizando esse modelo, foi realizada a predição da chance de admissão acadêmica para três novos candidatos, cujos resultados são apresentados na Tabela 39. As estimativas variaram entre aproximadamente 60% e 90%, demonstrando coerência com os perfis fornecidos. Esses resultados reforçam a capacidade preditiva do modelo selecionado e evidenciam sua aplicabilidade prática no contexto analisado.

Biomassa

Este experimento tem como objetivo avaliar o desempenho de diferentes técnicas de regressão aplicadas a um conjunto de dados relacionados à estimativa de biomassa. As variáveis preditoras consideradas foram o diâmetro à altura do peito (dap), a altura (h) e a massa específica da madeira (Me), sendo a biomassa a variável dependente.

A avaliação dos modelos foi realizada com base nos indicadores R^2 , Syx (erro padrão da estimativa), coeficiente de correlação de Pearson, RMSE (Root Mean Square Error) e MAE (Mean Absolute Error). As técnicas testadas foram ordenadas de forma decrescente com base no valor de R^2 , de modo que os modelos com melhor ajuste apareçam nas primeiras posições da tabela.

Ao final, apresenta-se a predição de novos casos utilizando o modelo com maior R^2 , bem como o gráfico de resíduos e os comandos executados em linguagem R.

TABELA 40 – RESULTADOS DO EXPERIMENTO DE REGRESSÃO - BASE DE BIOMASSA

Técnica	Parâmetro	R^2	Syx	Pearson	RMSE	MAE
RF – CV	mtry = 2	0,9798	94,14	0,9932	90,94	44,87
RF – Hold-out	mtry = 2	0,9784	97,41	0,9930	94,11	47,03
SVM – Hold-out	C = 1, Sigma = 0.8166	0,9444	153,48	0,9836	150,90	93,71
SVM – CV	C = 1, Sigma = 0.8166	0,9444	153,48	0,9836	150,90	93,71
KNN	k = 1	0,9358	164,98	0,9674	162,20	72,19
RNA – CV	size = 4, decay = 0,1	0,9032	202,55	0,9806	199,15	83,50
RNA – Hold-out	size = 3, decay = 0,1	0,8448	256,43	0,9722	252,12	110,09

FONTE: Resultados obtidos com experimentação utilizando a linguagem R (2024).

Com base nos resultados apresentados, observa-se que a técnica *Random Forest* com validação cruzada obteve o melhor desempenho preditivo, com $R^2 = 0,9798$, menor erro padrão da estimativa (Syx) e o menor RMSE. Esses indicadores evidenciam sua maior capacidade de generalização em relação às demais técnicas avaliadas, sendo, portanto, a mais adequada para estimar a biomassa com base nas variáveis *dap*, *h* e *Me*.

Comandos utilizados na execução do experimento de regressão - base de biomassa

Esta seção apresenta os comandos utilizados na linguagem R para a execução do experimento de regressão com a base de dados de biomassa. O objetivo é avaliar a capacidade preditiva dos modelos, utilizando validação cruzada e métricas como R^2 , erro padrão da estimativa (Syx), RMSE, MAE, entre outras. O modelo de maior desempenho foi então aplicado para gerar predições sobre novos casos.

```

1 # Carrega os pacotes necessários
2 library(e1071)
3 library(kernlab)
4 library(caret)
5 library(Metrics)
6
7 # Define o diretório de trabalho e lê o arquivo com os dados
8 setwd("./05 - Biomassa")
9 dados <- read.csv("5 - Biomassa - Dados.csv", header = TRUE)
10
11 # Separa os dados em treino (80%) e teste (20%)

```

```

12 set.seed(202460)
13 indices <- createDataPartition(dados$biomassa, p = 0.80, list = FALSE)
14 treino <- dados[indices, ]
15 teste <- dados[-indices, ]
16
17 # Define a validação cruzada com 10 folds
18 ctrl <- trainControl(method = "cv", number = 10)
19
20 # Treina o modelo Random Forest com validação cruzada
21 set.seed(202460)
22 rf <- train(biomassa ~ ., data = treino, method = "rf", trControl = ctrl)
23
24 # Realiza predições na base de teste
25 predicoes.rf <- predict(rf, teste)
26
27 # Calcula o RMSE
28 rmse(teste$biomassa, predicoes.rf)
29
30 # Função para cálculo do coeficiente de determinação (R2)
31 r2 <- function(predito, observado) {
32   return(1 - (sum((predito - observado)^2) / sum((observado - mean(observado))^2)))
33 }
34
35 # Calcula o R2
36 r2(predicoes.rf, teste$biomassa)
37
38 # Calcula o erro padrão da estimativa (Syx)
39 observado <- teste$biomassa
40 predito <- predicoes.rf
41 n <- length(observado)      # Número de observações
42 p <- ncol(teste) - 1        # Número de variáveis preditoras
43 Syx <- sqrt(sum((observado - predito)^2) / (n - p - 1))
44
45 # Calcula o coeficiente de correlação de Pearson
46 correlacao <- cor(observado, predito)
47
48 # Calcula o erro absoluto médio (MAE)
49 mae <- mean(abs(observado - predito))
50

```

```

51 # Calcula os resíduos
52 residuos <- observado - predito
53
54 # Plota o gráfico de resíduos
55 plot(predito, residuos,
56       xlab = "Valores Preditos",
57       ylab = "Resíduos",
58       main = "Gráfico de Resíduos")
59 abline(h = 0, col = "red")
60
61 # Lê os novos casos e remove a variável alvo
62 dados_novos_casos <- read.csv("5 - Biomassa - Novos Casos.csv")
63 View(dados_novos_casos)
64 dados_novos_casos$biomassa <- NULL
65
66 # Realiza predições para os novos casos
67 predict.rf <- predict(rf, dados_novos_casos)
68 resultado <- cbind(dados_novos_casos, predict.rf)
69 View(resultado)

```

TABELA 41 – PREDIÇÃO DE NOVOS CASOS PARA O EXPERIMENTO DE REGRESSÃO - BASE DE BIOMASSA

Entrada			
DAP (cm)	Altura (m)	Me (g/cm ³)	Biomassa (kg)
9,7	14,8	0,38	?
16,7	9,4	0,63	?
23	12,4	0,40	?
Resultado			
DAP (cm)	Altura (m)	Me (g/cm ³)	Biomassa (kg)
9,7	14,8	0,38	38,63899
16,7	9,4	0,63	113,71063
23	12,4	0,40	200,897285

FONTE: Elaborado pelo autor (2024).

A Tabela 41 apresenta três novos casos para os quais foram aplicadas as predições do modelo selecionado. Conforme os resultados do experimento (ver Tabela 40), o algoritmo Random Forest com validação cruzada (RF – CV, com $mtry = 2$) foi o que obteve o melhor desempenho, alcançando um coeficiente de determinação (R^2) de 0,98, menor erro padrão residual (S_{yx}) e menor erro quadrático médio (RMSE) dentre todas as técnicas avaliadas.

Dessa forma, as predições exibidas na Tabela 41 foram geradas com base neste modelo, garantindo, assim, maior confiabilidade na estimativa da variável de interesse, a biomassa. Esses resultados

evidenciam a robustez da técnica selecionada para aplicações práticas que demandam precisão na modelagem de dados florestais.

Agrupamento

Neste experimento, foi realizada a aplicação de técnicas de agrupamento sobre a base de dados de veículos, com o objetivo de identificar estruturas latentes nos dados por meio da formação de grupos com características semelhantes. Para isso, foram utilizados dois algoritmos distintos: *k-means* e *k-modes*, ambos configurados para gerar dez clusters.

Embora o algoritmo *k-modes* tenha emitido mensagens de alerta durante sua execução, os resultados obtidos foram equivalentes aos gerados pelo *k-means*, sendo, portanto, mantidos no experimento. A seguir, são apresentados os dez primeiros registros da base de dados com a indicação do cluster ao qual cada um foi atribuído. Ao final do exercício, é apresentada a listagem completa dos comandos utilizados na linguagem R para obtenção dos resultados.

TABELA 42 – DEZ PRIMEIRAS LINHAS DO ARQUIVO COM O CLUSTER CORRESPONDENTE

ID	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HolIRa	Class	Cluster
1	70	6	16	187	197	van	2
2	72	9	14	189	199	van	1
3	73	14	9	188	196	saab	4
4	63	6	10	199	207	van	5
5	127	9	11	180	183	bus	5
6	85	5	9	181	183	bus	5
7	66	13	1	200	204	bus	5
8	67	3	3	193	202	van	2
9	64	2	14	200	208	van	2
10	64	4	14	195	204	saab	3

FONTE: Elaborado pelo autor com base nos resultados obtidos via linguagem R (2024).

Comandos utilizados na execução do experimento de agrupamento - base de veículos

Esta etapa apresenta os comandos utilizados em linguagem R para a realização do agrupamento dos dados da base de veículos, utilizando-se o algoritmo *K-means*. Foram configurados 10 clusters para o experimento. Inicialmente, os dados foram preparados com a remoção de atributos não numéricos e a análise de correlação entre variáveis. Em seguida, foi aplicada a técnica de agrupamento e realizadas visualizações para análise dos resultados, como gráficos de dispersão com os centroides de cada grupo. Todos os comandos foram executados de forma a garantir reprodutibilidade e clareza na metodologia empregada.

```

1 # Define o diretório de trabalho e lê o arquivo CSV
2 setwd("./4 - Veiculos")
3 dados <- read.csv("4 - Veiculos - Dados.csv", header = TRUE)
4

```

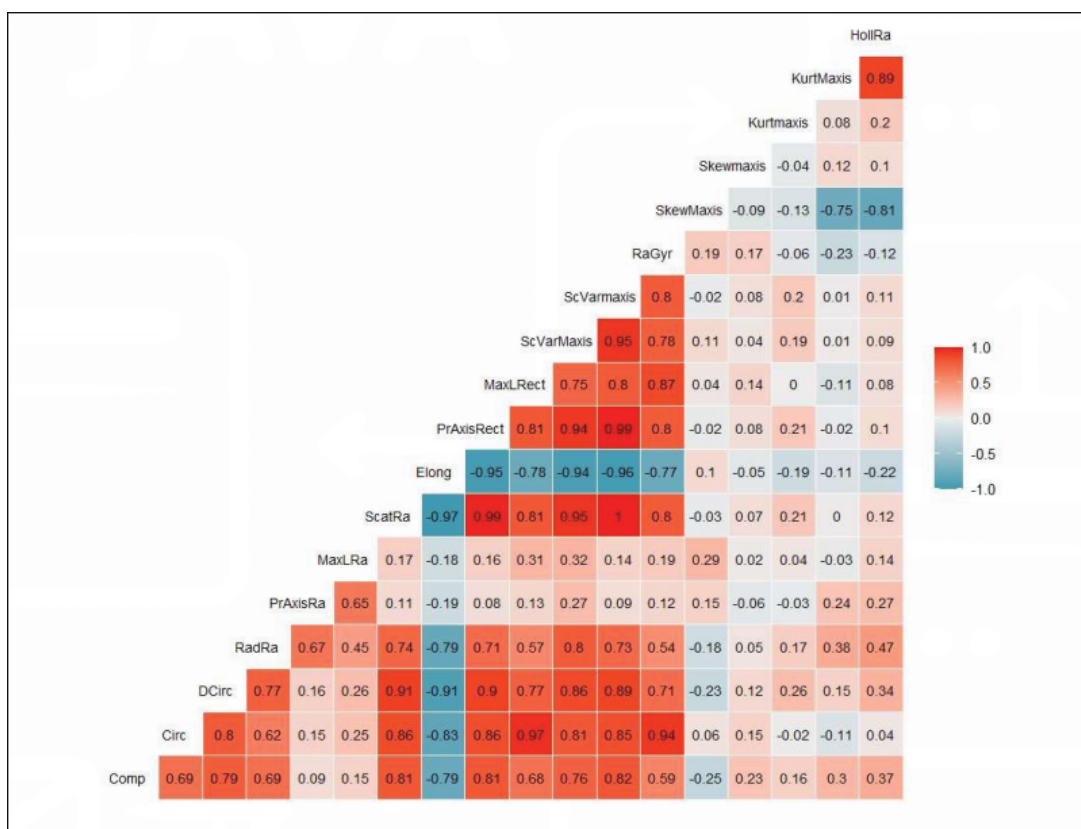
```

5 # Remove a coluna de índice
6 dados$a <- NULL
7
8 # Visualiza a estrutura dos dados
9 str(dados)
10
11 # Carrega os pacotes necessários
12 library(corrplot)
13 library(GGally)
14 library(klaR)
15 library(ggplot2)
16 library(dplyr)
17
18 # Seleciona apenas variáveis numéricas
19 dados_numericos <- dados[, sapply(dados, is.numeric)]
20
21 # Calcula a matriz de correlação
22 matriz_correlacao <- cor(dados_numericos)
23
24 # Gera gráfico de correlação com rótulos
25 ggcorr(dados_numericos, label = TRUE, label_round = 2, label_size = 3, hjust = 0.75,
26        size = 3)
27
28 # Cria gráfico de dispersão com coloração por classe
29 ggplot(dados, aes(ScVarMaxis, HollRa, color = Class)) + geom_point()
30
31 # Calcula os centroides por classe
32 centroides <- dados %>%
33   group_by(Class) %>%
34   summarize(ScVarMaxis = mean(ScVarMaxis), HollRa = mean(HollRa))
35
36 # Cria gráfico de dispersão com os centroides destacados
37 ggplot(dados, aes(x = ScVarMaxis, y = HollRa, color = Class)) +
38   geom_point(alpha = 0.5) +
39   geom_point(data = centroides, aes(x = ScVarMaxis, y = HollRa, color = Class), shape
40             = 8, size = 5) +
41   labs(title = "Gráfico de Dispersão com Centroides Destacados") +
42   theme_minimal()

```

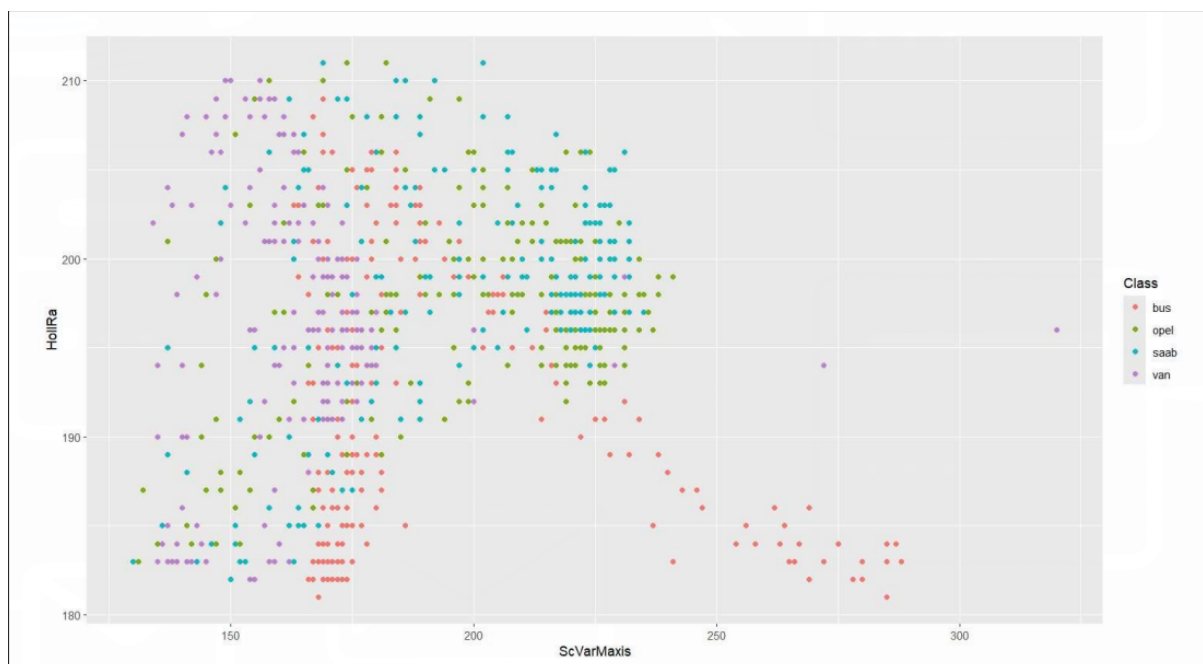
```
42 # Executa o algoritmo K-means com 10 clusters
43 set.seed(202460)
44 veiculosCluster <- kmeans(dados_numericos, 10)
45
46 # Exibe resultado do agrupamento
47 veiculosCluster
48
49 # Define a semente para reprodutibilidade dos resultados
50 set.seed(202460)
51
52 # Executa o algoritmo K-means com 10 clusters utilizando apenas variáveis numéricas
53 veiculosCluster <- kmeans(dados_numericos, 10)
54
55 # Exibe os resultados do agrupamento com K-means
56 veiculosCluster
57
58 # Gera uma tabela de contingência entre os clusters obtidos e a variável alvo (Class)
59 table(veiculosCluster$cluster, dados$Class)
60
61 # Adiciona os rótulos dos clusters obtidos aos dados originais
62 resultado <- cbind(dados, Cluster = veiculosCluster$cluster)
63
64 # Exibe o conjunto de dados com a coluna de clusters adicionada
65 resultado
66
67 # Executa o algoritmo K-modes com 5 clusters (sem pesos e no máximo 10 iterações)
68 cluster.results <- kmodes(dados, 5, iter.max = 10, weighted = FALSE)
69
70 # Exibe os resultados do agrupamento com K-modes
71 cluster.results
72
73 # Adiciona os rótulos dos clusters K-modes aos dados originais
74 resultado_kmodes <- cbind(dados, Cluster = cluster.results$cluster)
75
76 # Salva os resultados do agrupamento com K-modes em um arquivo CSV
77 write.csv(resultado_kmodes, "resultados_kmodes.csv", row.names = FALSE)
78
79 # Exibe o conjunto de dados com a coluna de clusters adicionada (K-modes)
80 resultado_kmodes
```


FIGURA 32 – GRÁFICO DE CORRELAÇÃO - AGRUPAMENTO - BASE VEÍCULOS



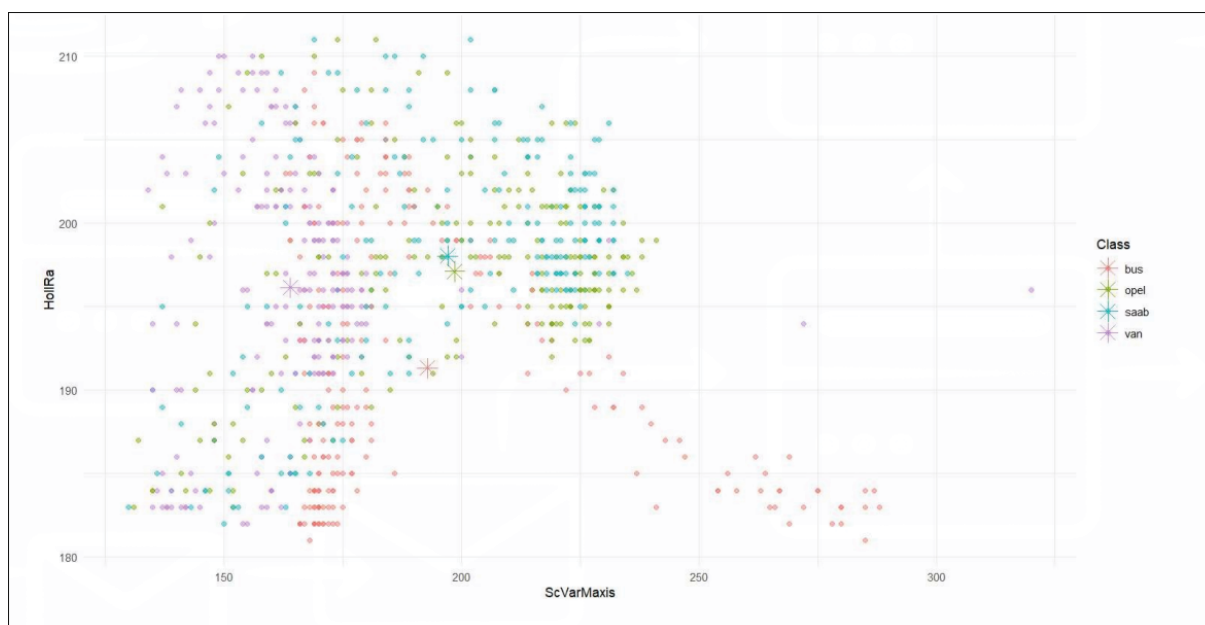
FONTE: Elaborado pelo autor (2024).

FIGURA 33 – GRÁFICO DE DISPERSÃO - AGRUPAMENTO - BASE VEÍCULOS



FONTE: Elaborado pelo autor (2024).

FIGURA 34 – GRÁFICO DE DISPERSÃO COM CENTROIDES DESTACADOS - AGRUPAMENTO - BASE VEÍCULOS



FONTE: Elaborado pelo autor (2024).

TABELA 43 – AMOSTRA DOS RESULTADOS OBTIDOS COM O ALGORITMO K-MODES SOBRE OS DADOS DE VEÍCULOS.

ID	Comp	Circ	DCirc	RadHa	PrAxisHa	MaxLHa	ScatHa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarMaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	Class	Cluster
1	95	48	83	178	72	10	162	42	20	159	176	379	184	70	6	16	187	197	van	1
2	91	41	84	141	57	9	149	45	19	143	170	330	158	72	9	14	189	199	van	5
3	104	50	106	209	86	10	207	32	23	158	223	635	220	73	14	9	188	196	saab	2
4	95	41	82	159	83	9	144	46	19	143	160	309	127	63	6	10	199	207	van	1
5	85	44	70	205	103	52	149	45	19	144	241	325	188	127	9	11	180	183	bus	5
6	107	57	106	172	50	6	255	26	28	169	280	957	264	85	5	9	181	183	bus	5
7	97	43	73	173	65	6	153	42	19	143	176	361	172	66	13	1	200	204	bus	5
8	90	43	66	157	85	9	137	48	18	146	162	281	164	67	3	3	193	202	van	3
9	86	34	62	140	61	7	122	54	17	127	141	223	112	64	2	14	200	208	van	5
10	93	44	98	197	62	11	183	36	22	146	202	505	152	64	4	14	195	204	saab	3
11	86	36	70	143	61	9	133	50	18	130	153	266	127	66	2	10	194	202	van	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
49	89	47	81	147	64	11	156	44	20	163	170	352	188	76	6	13	184	193	van	1
50	91	45	79	176	59	9	163	40	20	148	184	404	179	62	0	10	199	208	saab	1

FONTE: Elaborado pelo autor com base nos resultados obtidos via linguagem R (2024).

A aplicação dos algoritmos de agrupamento *K-means* e *K-modes* ao conjunto de dados de veículos possibilitou a formação de grupos coerentes com os padrões observados na variável categórica *Class*. A análise dos agrupamentos revelou que as estruturas dos clusters se alinham com categorias previamente conhecidas, evidenciando a eficácia dos métodos utilizados. Embora o algoritmo *K-modes* tenha emitido alertas durante sua execução, os resultados obtidos foram consistentes com os gerados pelo *K-means*, validando a robustez das abordagens adotadas e confirmando a viabilidade do uso dessas técnicas em problemas de segmentação de dados multidimensionais.

Regras de associação

Este experimento tem como objetivo explorar a aplicação de regras de associação sobre um conjunto de dados relacionado à musculação. A análise foi conduzida com base na utilização da linguagem R, empregando-se parâmetros mínimos de suporte e confiança para a geração das regras.

A metodologia adotada permite identificar padrões frequentes entre os itens registrados na base, revelando associações relevantes que podem ser úteis para compreender o comportamento dos praticantes de musculação. As regras foram extraídas utilizando algoritmos apropriados para este tipo de tarefa, respeitando os critérios previamente definidos.

O experimento inclui a apresentação das principais regras obtidas, bem como a listagem dos comandos utilizados na linguagem R para reprodução dos resultados.

TABELA 44 – REGRAS GERADAS COM A CONFIGURAÇÃO DE SUPORTE, CONFIANÇA E COBERTURA

ID	lhs	rhs	Suporte	Confiança	Cobertura
1		LegPress	0,8077	0,8077	1,0000
2	Adutor	Agachamento	0,1154	1,0000	0,1154
3	Adutor	LegPress	0,1154	1,0000	0,1154
4	Agachamento	LegPress	0,3077	1,0000	0,3077
5	Afundo	Gemeos	0,3462	1,0000	0,3462
6	AgachamentoSmith	Esteira	0,3077	0,8000	0,3846
7	AgachamentoSmith	Extensor	0,3462	0,9000	0,3846
8	AgachamentoSmith	Bicicleta	0,3077	0,8000	0,3846
9	Esteira	Extensor	0,4231	0,9167	0,4615
10	Extensor	Esteira	0,4231	0,8462	0,5000
11	Esteira	Bicicleta	0,3846	0,8333	0,4615
12	Extensor	Bicicleta	0,4615	0,9231	0,5000
13	Bicicleta	Extensor	0,4615	0,8571	0,5385
14	Adutor, Agachamento	LegPress	0,1154	1,0000	0,1154
15	Adutor, LegPress	Agachamento	0,1154	1,0000	0,1154
16	Agachamento, Gemeos	LegPress	0,1923	1,0000	0,1923
17	Afundo, Bicicleta	Gemeos	0,1538	1,0000	0,1538
18	Afundo, LegPress	Gemeos	0,2692	1,0000	0,2692
19	AgachamentoSmith, Esteira	Extensor	0,2692	0,8750	0,3077
20	AgachamentoSmith, LegPress	Esteira	0,2308	0,8571	0,2692
21	AgachamentoSmith, Extensor	Bicicleta	0,3077	0,8889	0,3462
22	AgachamentoSmith, Bicicleta	Extensor	0,3077	1,0000	0,3077
23	AgachamentoSmith, Gemeos	Extensor	0,1923	0,8333	0,2308

TABELA 44 – REGRAS GERADAS COM A CONFIGURAÇÃO DE SUPORTE, CONFIANÇA E COBERTURA

ID	lhs	rhs	Suporte	Confiança	Cobertura
24	AgachamentoSmith, LegPress	Extensor	0,2308	0,8571	0,2692
25	Esteira, Extensor	Bicicleta	0,3846	0,9091	0,4231
26	Bicicleta, Esteira	Extensor	0,3846	1,0000	0,3846
27	Bicicleta, Extensor	Esteira	0,3846	0,8333	0,4615
28	Esteira, Gemeos	Extensor	0,1923	0,8333	0,2308
29	Esteira, LegPress	Extensor	0,2692	0,8750	0,3077
30	Extensor, LegPress	Esteira	0,2692	0,8750	0,3077
31	Extensor, Gemeos	Bicicleta	0,2308	0,8571	0,2692
32	Extensor, LegPress	Bicicleta	0,2692	0,8750	0,3077
33	AgachamentoSmith, Esteira, Extensor	Bicicleta	0,2308	0,8571	0,2692
34	AgachamentoSmith, Bicicleta, Esteira	Extensor	0,2308	1,0000	0,2308
35	AgachamentoSmith, Esteira, LegPress	Extensor	0,1923	0,8333	0,2308
36	AgachamentoSmith, Extensor, LegPress	Esteira	0,1923	0,8333	0,2308
37	AgachamentoSmith, Bicicleta, LegPress	Esteira	0,1538	0,8000	0,1923
38	AgachamentoSmith, Extensor, Gemeos	Bicicleta	0,1538	0,8000	0,1923
39	AgachamentoSmith, Bicicleta, Gemeos	Extensor	0,1538	1,0000	0,1538
40	AgachamentoSmith, Extensor, LegPress	Bicicleta	0,1923	0,8333	0,2308
41	AgachamentoSmith, Bicicleta, LegPress	Extensor	0,1923	1,0000	0,1923
42	Esteira, Extensor, Gemeos	Bicicleta	0,1538	0,8000	0,1923
43	Bicicleta, Esteira, Gemeos	Extensor	0,1538	1,0000	0,1538
44	Esteira, Extensor, LegPress	Bicicleta	0,2308	0,8571	0,2692
45	Bicicleta, Esteira, LegPress	Extensor	0,2308	1,0000	0,2308
46	Bicicleta, Extensor, LegPress	Esteira	0,2308	0,8571	0,2692

TABELA 44 – REGRAS GERADAS COM A CONFIGURAÇÃO DE SUPORTE, CONFIANÇA E COBERTURA

ID	lhs	rhs	Suporte	Confiança	Cobertura
47	AgachamentoSmith, Esteira, Extensor, LegPress	Bicicleta	0,1538	0,8000	0,1923
48	AgachamentoSmith, Bicicleta, Esteira, LegPress	Extensor	0,1538	1,0000	0,1538
49	AgachamentoSmith, Bicicleta, Extensor, LegPress	Esteira	0,1538	0,8000	0,1923

FONTE: Elaborado pelo autor (2024).

Comandos utilizados para extração das regras de associação - base musculação

Esta etapa apresenta os comandos utilizados em linguagem R para a extração e visualização das regras de associação a partir da base de dados de musculação. O processo foi realizado com o uso do pacote *arules*, aplicando o algoritmo *Apriori* com parâmetros de suporte mínimo de 10% e confiança mínima de 80%, visando ampliar o número de regras extraídas para análise. As regras obtidas refletem padrões frequentes entre os exercícios realizados.

Adicionalmente, foram empregadas técnicas de visualização por meio de grafos, utilizando os pacotes *arulesViz*, *igraph* e *ggplot2*, com o objetivo de representar as relações entre os itens de forma intuitiva. Todos os comandos foram executados de forma a garantir a reprodutibilidade dos resultados e a clareza na metodologia empregada.

```

1 # Carregamento do pacote arules, necessário para mineração de regras de associação
2 library(arules)
3
4 # Carregamento de conjuntos de dados padrão do R (não utilizado diretamente)
5 library(datasets)
6
7 # Carregamento dos pacotes necessários para visualização das regras
8 library(arulesViz)
9 library(igraph)
10
11 # Carregamento do pacote ggplot2 para personalização de gráficos
12 library(ggplot2)
13
14 # Definição do diretório de trabalho onde se encontra o arquivo CSV
15 setwd("./2 - Musculacao")

```

```

16
17 # Leitura do arquivo de dados em formato de transações, separadas por ponto e vírgula
18 dados <- read.transactions(file = "2 - Musculacao - Dados.csv", format = "basket", sep
    = ";")
19
20 # Definição de semente para garantir reprodutibilidade dos resultados
21 set.seed(202460)
22
23 # Geração das regras de associação utilizando o algoritmo Apriori
24 # Parâmetros definidos: suporte mínimo de 0.1 e confiança mínima de 0.8
25 rules <- apriori(dados, parameter = list(supp = 0.1, conf = 0.8, target = "rules"))
26
27
28 # Visualização das regras em forma de grafo simples
29 plot(rules, method = "graph")
30
31 # Visualização alternativa com coloração vermelha nos elementos do grafo
32 plot(rules, method = "graph", color = "red")
33
34 # Geração de grafo com personalização avançada: setas, cores e tamanhos proporcionais
    ao suporte
35 plot(rules,
36     method = "graph",
37     control = list(
38         edges = ggraph::geom_edge_link(
39             end_cap = ggraph::circle(4, "mm"),
40             start_cap = ggraph::circle(4, "mm"),
41             color = "black",
42             arrow = arrow(length = unit(2, "mm"), angle = 20, type = "closed"),
43             alpha = .2
44         ),
45         nodes = ggraph::geom_node_point(aes(size = support)),
46         nodetext = ggraph::geom_node_label(aes(label = label), alpha = .8, repel = TRUE),
47         limit = 10
48     )
49 ) +
50 scale_color_gradient(low = "yellow", high = "red") +
51 scale_size(range = c(2, 10))
52

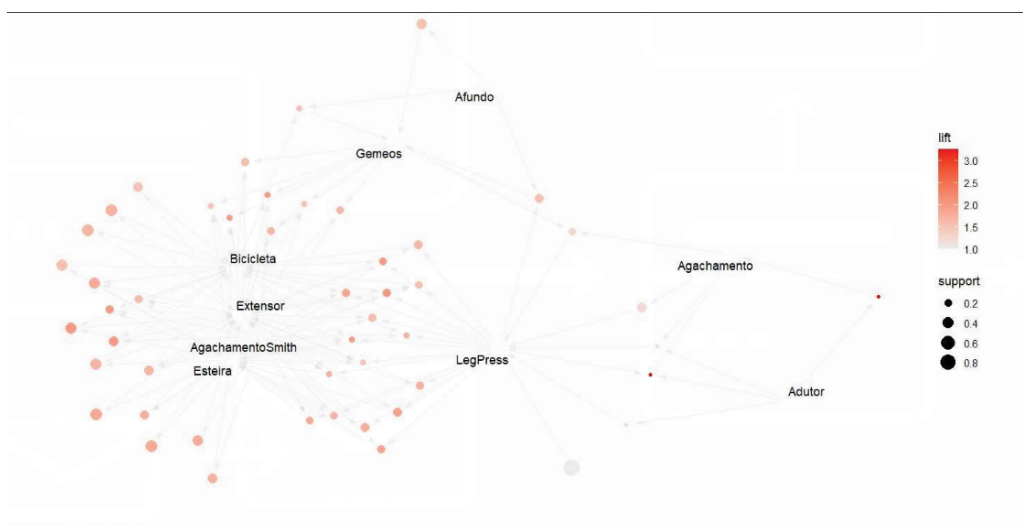
```

```

53 # Visualização das 10 principais regras como arestas (grafo com foco nas conexões)
54 plot(rules, method = "graph", asEdges = TRUE, limit = 10)
55
56 # Visualização semelhante, com layout linear (não circular)
57 plot(rules, method = "graph", asEdges = TRUE, circular = FALSE, limit = 10)
58
59 # Visualização utilizando o engine igraph para renderização
60 plot(rules, method = "graph", engine = "igraph", limit = 10)
61
62 # Visualização com coloração em tons de cinza e alta opacidade
63 plot(rules,
64       method = "graph", engine = "igraph",
65       nodeCol = grey.colors(10), edgeCol = grey(.7), alpha = 1,
66       limit = 10
67 )

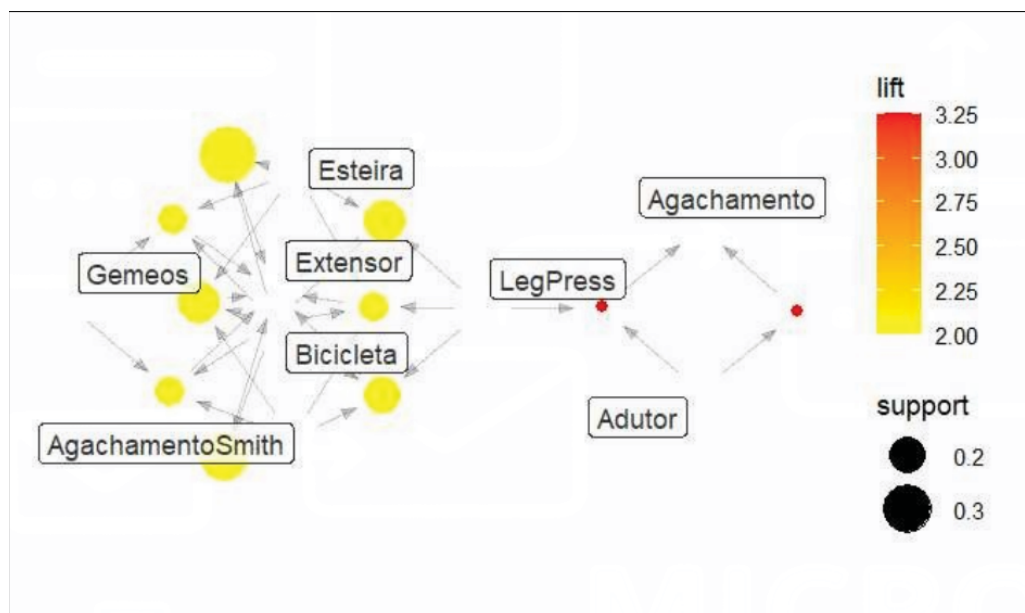
```

FIGURA 35 – REGRAS DE ASSOCIAÇÃO COMO GRAFO DE ITENS CONECTADOS - BASE DE MUSCULAÇÃO



FONTE: Elaborado pelo autor (2024).

FIGURA 36 – GRAFO DAS REGRAS COM ÊNFASE EM SUPORTE E INTENSIDADE - BASE DE MUSCULAÇÃO



FONTE: Elaborado pelo autor (2024).

FIGURA 37 – GRAFO COM VISUALIZAÇÃO DAS REGRAS COMO ARESTAS EXPLÍCITAS - BASE DE MUSCULAÇÃO



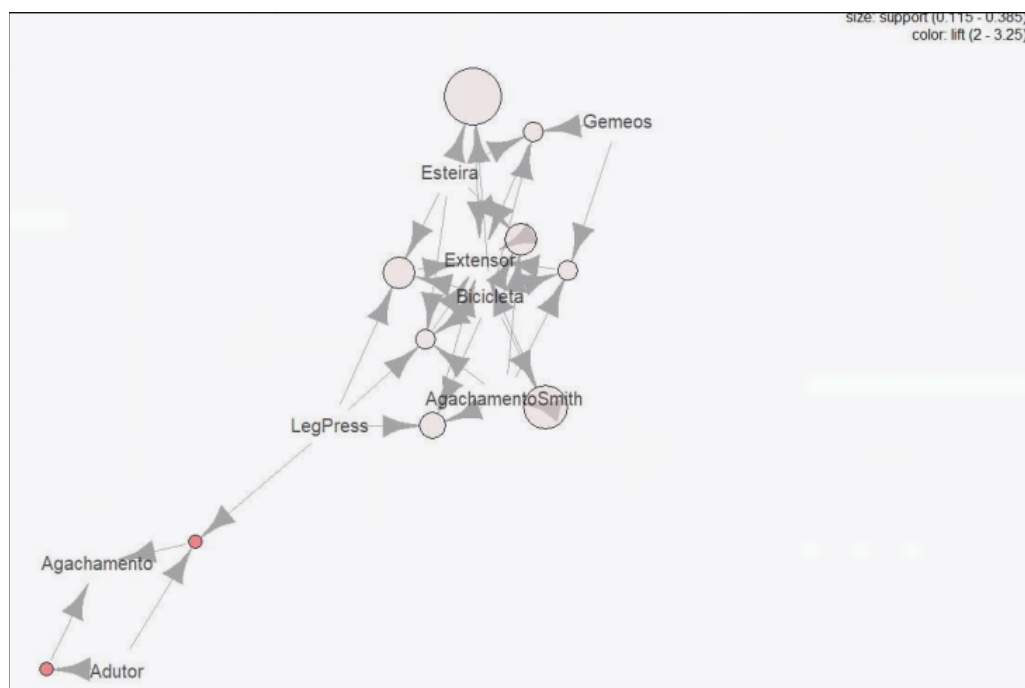
FONTE: Elaborado pelo autor (2024).

FIGURA 38 – GRAFO DAS REGRAS COM ARESTAS EXPLÍCITAS EM DISPOSIÇÃO HIERÁRQUICA - BASE DE MUSCULAÇÃO



FONTE: Elaborado pelo autor (2024).

FIGURA 39 – GRAFO DAS 10 PRINCIPAIS REGRAS DE ASSOCIAÇÃO - BASE DE MUSCULAÇÃO



FONTE: Elaborado pelo autor (2024).

A aplicação do algoritmo Apriori para a extração de regras de associação na base de dados de musculação permitiu identificar padrões frequentes e relações significativas entre os exercícios analisados. Os resultados indicam que o exercício *LegPress* atua como um item central, frequentemente associado a outros movimentos como *Agachamento* e *Adutor*, refletindo padrões consistentes de coocorrência. A

visualização por meio dos grafos contribuiu para a compreensão das interações entre os itens, evidenciando a estrutura das regras extraídas. Dessa forma, a metodologia adotada demonstrou-se eficaz para a identificação de associações relevantes, confirmando a aplicabilidade das técnicas de mineração de dados na análise de conjuntos de dados esportivos.

APÊNDICE 9 – 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)

Esta atividade teve como objetivo desenvolver e treinar uma rede neural convolucional (CNN) para a tarefa de classificação de imagens utilizando a base de dados CIFAR-10. A implementação foi realizada na linguagem Python com o uso das bibliotecas TensorFlow, Keras, NumPy, Matplotlib, Seaborn e Scikit-learn.

Base de dados CIFAR-10

A base CIFAR-10 contém 60.000 imagens coloridas de resolução 32×32 pixels, distribuídas em 10 classes distintas: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* e *truck*, sendo 50.000 destinadas ao treinamento e 10.000 ao teste do modelo Krizhevsky (2009). As imagens foram normalizadas para o intervalo $[0, 1]$ e os rótulos convertidos para codificação *one-hot*.

Arquitetura do modelo

A arquitetura proposta para a CNN consiste em três camadas convolucionais com 32, 64 e 128 filtros, respectivamente, todas com função de ativação ReLU e kernel de tamanho 3×3 , seguidas por camadas de *MaxPooling2D* com pool size 2×2 . Após o empilhamento das camadas convolucionais, os mapas de ativação são achatados com uma camada *Flatten*, conectados a uma camada densa

com 128 neurônios e ativação ReLU. Para reduzir o risco de sobreajuste, foi aplicada uma camada *Dropout* com taxa de 0,5 antes da camada de saída, que possui 10 neurônios com ativação *softmax*, correspondendo às 10 classes do problema.

Treinamento do modelo

TABELA 45 – HISTÓRICO DE TREINAMENTO: ACURÁCIA E PERDA POR ÉPOCA

Época	Acurácia	Perda	Val. Acurácia	Val. Perda
1	0,2727	1,9248	0,5188	1,3336
2	0,5062	1,3692	0,5811	1,1979
3	0,5758	1,2063	0,6237	1,0665
4	0,6239	1,0833	0,6494	0,9843
5	0,6518	0,9999	0,6564	0,9726
6	0,6749	0,9325	0,6788	0,9175
7	0,6973	0,8799	0,6979	0,8692
8	0,7085	0,8389	0,6985	0,8783
9	0,7262	0,7779	0,7026	0,8670
10	0,7451	0,7401	0,7138	0,8321

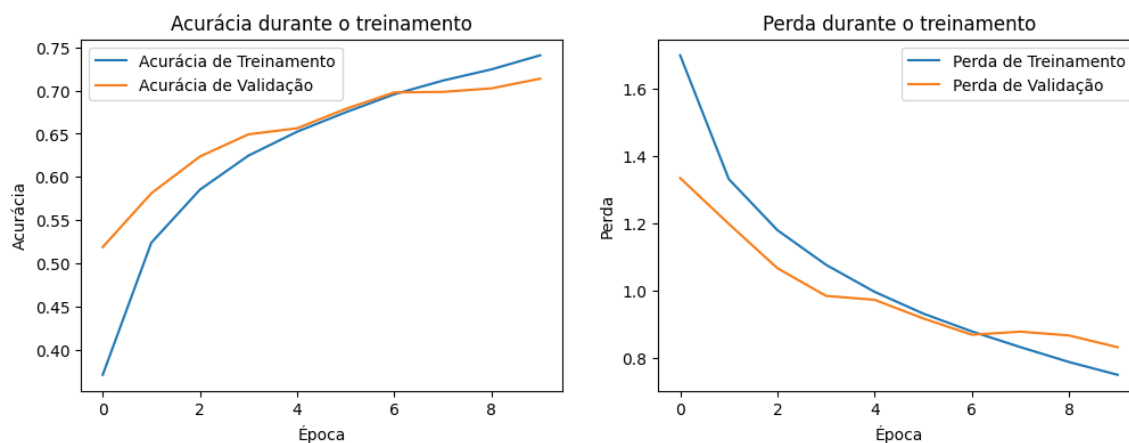
FONTE: Elaborado pelo autor (2024).

O modelo foi treinado por 10 épocas, com *batch size* de 64 e utilizando a função de perda *categorical crossentropy*. O otimizador utilizado foi o *Adam*. A Tabela 45 apresenta a evolução da acurácia e da perda ao longo das épocas.

Avaliação do modelo

Após o treinamento, o modelo atingiu uma acurácia de **71,38%** sobre o conjunto de teste, indicando desempenho satisfatório dado o nível de complexidade da rede. A Figura 40 apresenta a evolução da acurácia e da perda durante o treinamento e a validação.

FIGURA 40 – CURVAS DE ACURÁCIA E PERDA DURANTE O TREINAMENTO



FONTE: Elaborado pelo autor (2024).

A Figura 41 mostra a matriz de confusão obtida com o conjunto de teste, evidenciando o desempenho por classe. As maiores confusões ocorreram entre categorias similares, como *cat* e *dog*.

FIGURA 41 – MATRIZ DE CONFUSÃO DA CNN NO CONJUNTO DE TESTE

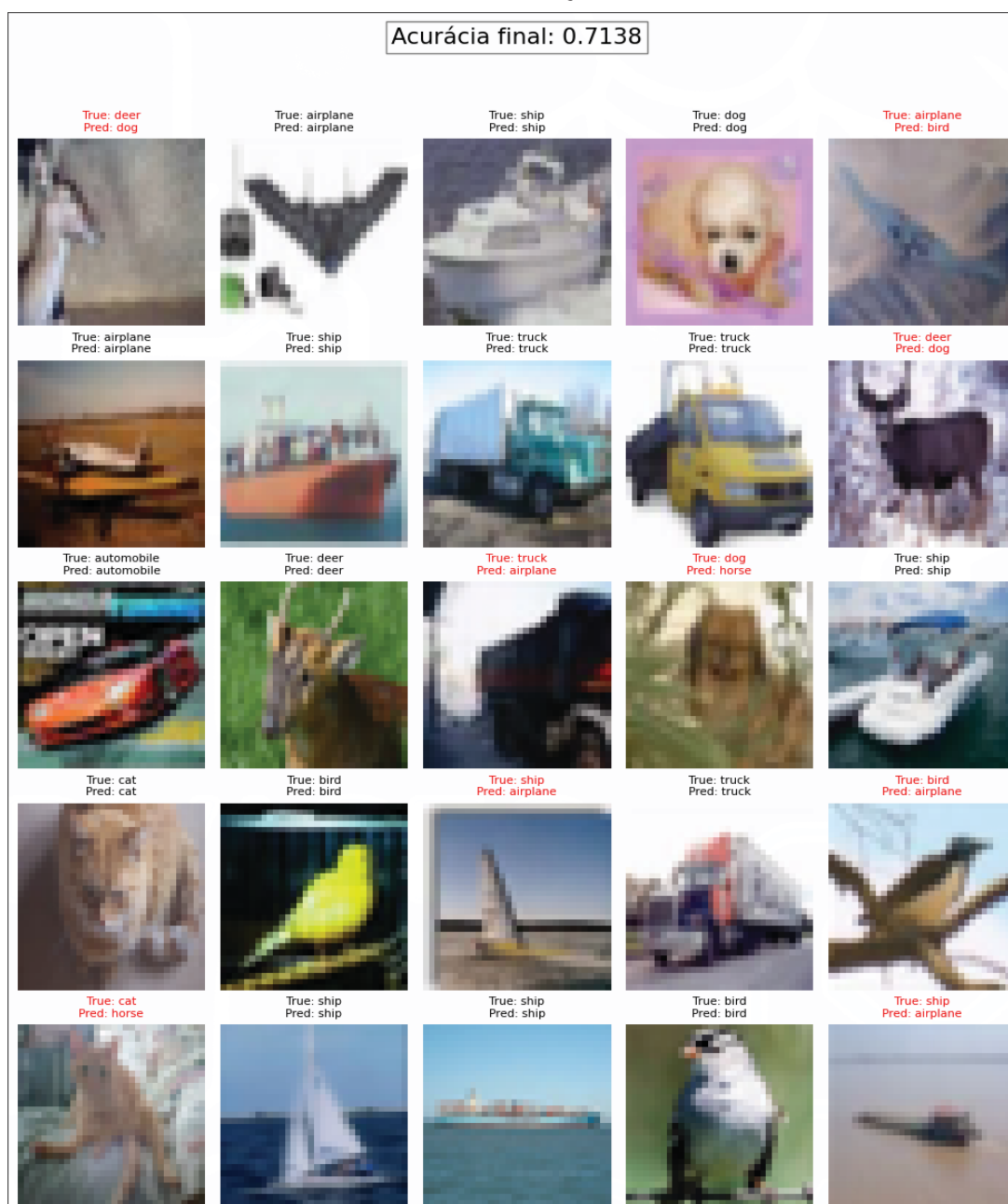
airplane	818	17	45	13	14	7	6	17	39	24
automobile	31	874	8	7	2	4	9	7	16	42
bird	80	6	617	38	68	78	62	36	7	8
cat	22	15	92	419	58	200	101	54	15	24
deer	22	4	82	40	635	40	70	99	5	3
dog	10	8	64	114	43	660	29	66	1	5
frog	8	5	66	41	18	21	825	9	2	5
horse	17	3	35	18	36	58	10	812	1	10
ship	129	48	20	6	6	7	8	15	741	20
truck	57	114	10	11	6	8	10	25	22	737
	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck

FONTE: Elaborado pelo autor (2024).

Análise qualitativa das previsões

Para complementar a avaliação, foram selecionadas aleatoriamente 25 imagens do conjunto de teste, indicando se foram corretamente classificadas. Imagens mal classificadas são destacadas com texto vermelho, conforme a Figura 42.

FIGURA 42 – AMOSTRAS DE CLASSIFICAÇÕES CORRETAS E INCORRETAS



FONTE: Elaborado pelo autor (2024).

Código-fonte

```

1 # Importação das bibliotecas necessárias
2 import tensorflow as tf

```

```

3 from tensorflow.keras.datasets import cifar10
4 from tensorflow.keras.models import Sequential
5 from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout,
    Input
6 from tensorflow.keras.utils import to_categorical
7 import matplotlib.pyplot as plt
8 import numpy as np
9 import seaborn as sns
10 from sklearn.metrics import confusion_matrix, accuracy_score
11
12 # Carregamento da base de dados CIFAR-10
13 (x_train, y_train), (x_test, y_test) = cifar10.load_data()
14
15 # Normalização das imagens para o intervalo [0, 1]
16 x_train, x_test = x_train / 255.0, x_test / 255.0
17
18 # Conversão das classes para codificação one-hot
19 y_train = to_categorical(y_train, 10)
20 y_test = to_categorical(y_test, 10)
21
22 # Definição da arquitetura da rede neural convolucional (CNN)
23 model = Sequential([
24     Input(shape=(32, 32, 3)),          # Camada de entrada (imagens 32x32 RGB)
25     Conv2D(32, (3, 3), activation='relu'), # Primeira camada convolucional
26     MaxPooling2D((2, 2)),              # Primeira camada de pooling
27
28     Conv2D(64, (3, 3), activation='relu'), # Segunda camada convolucional
29     MaxPooling2D((2, 2)),              # Segunda camada de pooling
30
31     Conv2D(128, (3, 3), activation='relu'), # Terceira camada convolucional
32     MaxPooling2D((2, 2)),              # Terceira camada de pooling
33
34     Flatten(),                        # Transformação dos mapas em vetor
35     Dense(128, activation='relu'),     # Camada totalmente conectada
36     Dropout(0.5),                     # Regularização com dropout
37     Dense(10, activation='softmax')    # Camada de saída (10 classes)
38 ])
39
40 # Compilação do modelo com otimizador, função de perda e métrica

```

```

41 model.compile(optimizer='adam',
42               loss='categorical_crossentropy',
43               metrics=['accuracy'])
44
45 # Treinamento do modelo com conjunto de validação
46 history = model.fit(x_train, y_train, epochs=10,
47                   validation_data=(x_test, y_test),
48                   batch_size=64)
49
50 # Avaliação do desempenho final no conjunto de teste
51 test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
52 print('\nTest accuracy:', test_acc)
53
54 # Geração dos gráficos de acurácia e perda por época
55 plt.figure(figsize=(12, 4))
56
57 plt.subplot(1, 2, 1)
58 plt.plot(history.history['accuracy'], label='Acurácia de Treinamento')
59 plt.plot(history.history['val_accuracy'], label='Acurácia de Validação')
60 plt.xlabel('Época')
61 plt.ylabel('Acurácia')
62 plt.legend()
63 plt.title('Acurácia durante o treinamento')
64
65 plt.subplot(1, 2, 2)
66 plt.plot(history.history['loss'], label='Perda de Treinamento')
67 plt.plot(history.history['val_loss'], label='Perda de Validação')
68 plt.xlabel('Época')
69 plt.ylabel('Perda')
70 plt.legend()
71 plt.title('Perda durante o treinamento')
72
73 plt.tight_layout()
74 plt.savefig('imagens/curvas_treinamento.png')
75 plt.show()
76
77 # Geração de previsões a partir do conjunto de teste
78 y_pred = model.predict(x_test).argmax(axis=1) # Converte para classe prevista
79 y_true = np.argmax(y_test, axis=1)           # Converte rótulo verdadeiro

```



```

80
81 # Cálculo da matriz de confusão
82 cm = confusion_matrix(y_true, y_pred)
83
84 # Exibição da matriz de confusão com seaborn
85 plt.figure(figsize=(7, 7))
86 sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
87             xticklabels=['airplane', 'automobile', 'bird', 'cat', 'deer',
88                         'dog', 'frog', 'horse', 'ship', 'truck'],
89             yticklabels=['airplane', 'automobile', 'bird', 'cat', 'deer',
90                         'dog', 'frog', 'horse', 'ship', 'truck'],
91             cbar=False, square=True, linewidths=0.5, linecolor='black')
92 plt.xlabel('Predicted Label')
93 plt.ylabel('True Label')
94 plt.title('Matriz de Confusão')
95 plt.tight_layout()
96 plt.savefig('imagens/matriz_confusao.png')
97 plt.show()
98
99 # Definição dos rótulos das classes
100 labels = ['airplane', 'automobile', 'bird', 'cat', 'deer',
101          'dog', 'frog', 'horse', 'ship', 'truck']
102
103 # Seleção de uma amostra incorretamente classificada
104 misclassified = np.where(y_pred != y_true)[0]
105 i = np.random.choice(misclassified)
106
107 # Exibição da imagem com erro de classificação
108 plt.imshow(x_test[i])
109 plt.title("True label: %s Predicted: %s" % (labels[y_true[i]], labels[y_pred[i]]))
110 plt.axis('off')
111 plt.tight_layout()
112 plt.show()
113
114 # Cálculo da acurácia final usando sklearn
115 accuracy = accuracy_score(np.argmax(y_test, axis=1), y_pred)
116
117 # Número de imagens a serem exibidas na amostra final
118 num_images = 25

```

```

119 indices = np.random.choice(len(x_test), num_images, replace=False)
120
121 # Exibição das 25 imagens com previsão e rótulo verdadeiro
122 fig = plt.figure(figsize=(10, 12))
123
124 fig.suptitle(f"Acurácia final: {accuracy:.4f}", fontsize=16,
125             bbox=dict(facecolor='white', alpha=0.5, edgecolor='black'))
126 fig.text(0.5, 1.05, 'As imagens com texto em vermelho não foram classificadas
127             corretamente',
128          ha='center', va='center', fontsize=12,
129          bbox=dict(facecolor='white', alpha=0.5, edgecolor='black'))
130
131 # Plotagem das imagens com legendas coloridas
132 for i, idx in enumerate(indices):
133     ax = fig.add_subplot(5, 5, i + 1)
134     ax.imshow(x_test[idx])
135     true_label = labels[np.argmax(y_test[idx])]
136     pred_label = labels[y_pred[idx]]
137     color = 'black' if true_label == pred_label else 'red'
138     ax.set_title(f"True: {true_label}\nPred: {pred_label}", fontsize=8, color=color)
139     ax.axis('off')
140
141 plt.tight_layout(rect=[0, 0, 1, 0.95])
142 plt.savefig('imagens/resultados_amostrais.png')
143 plt.show()

```

Conclusão

A CNN implementada obteve uma acurácia final de 71,38% na base CIFAR-10, demonstrando capacidade razoável de generalização. A análise dos resultados, incluindo a matriz de confusão e imagens incorretamente classificadas, confirma a efetividade do modelo e aponta possíveis direções para aperfeiçoamentos futuros.

2. Detector de SPAM (RNN)

Nesta atividade, foi implementado um modelo de detecção de mensagens SPAM utilizando Redes Neurais Recorrentes (RNN), mais especificamente uma arquitetura baseada em Long Short-Term Memory (LSTM). O objetivo é classificar automaticamente mensagens de texto como "spam" ou "ham" (não spam), com base em um conjunto de dados rotulado. Este tipo de tarefa é bastante comum em aplicações de filtragem de e-mails e mensagens automáticas.

Base de dados

A base de dados utilizada é a conhecida *SMS Spam Collection*, disponível no link <<http://www.razer.net.br/datasets/spam.csv>>. Este conjunto de dados contém 5.572 mensagens de texto rotuladas manualmente como *spam* ou *ham*. Para facilitar o processamento, o arquivo CSV foi carregado com a codificação ISO-8859-1, sendo posteriormente tratado para remoção de colunas irrelevantes. As mensagens foram então convertidas para valores numéricos por meio da técnica de tokenização, limitando-se ao vocabulário de 20.000 palavras mais frequentes. A base foi dividida em 67% para treino e 33% para teste.

Arquitetura do modelo

O modelo foi construído utilizando a API funcional do Keras e possui as seguintes camadas principais:

- **Input Layer:** entrada com formato de sequência (frase tokenizada com padding).
- **Embedding Layer:** mapeia os tokens para vetores densos de dimensão 20.
- **LSTM Layer:** camada recorrente com 5 unidades, capaz de capturar padrões sequenciais.
- **Dense Layer:** camada final com ativação sigmoide, responsável pela classificação binária.

A Tabela 46 apresenta o resumo da arquitetura implementada.

TABELA 46 – RESUMO DA ARQUITETURA DO MODELO RNN PARA DETECÇÃO DE SPAM

Camada	Saída Esperada	Parâmetros Treináveis
Input	(None, 121)	0
Embedding	(None, 121, 20)	143.960
LSTM	(None, 5)	520
Dense	(None, 1)	6
Total	-	144.486

FONTE: Elaborado pelo autor (2024).

Treinamento e validação

O treinamento do modelo foi realizado com o otimizador Adam e função de perda `binary_crossentropy`, adequada para problemas de classificação binária. O modelo foi treinado por 5 épocas, e os dados de validação permitiram acompanhar o desempenho em termos de acurácia e perda.

A Tabela 47 resume os principais indicadores por época.

TABELA 47 – DESEMPENHO DO MODELO POR ÉPOCA

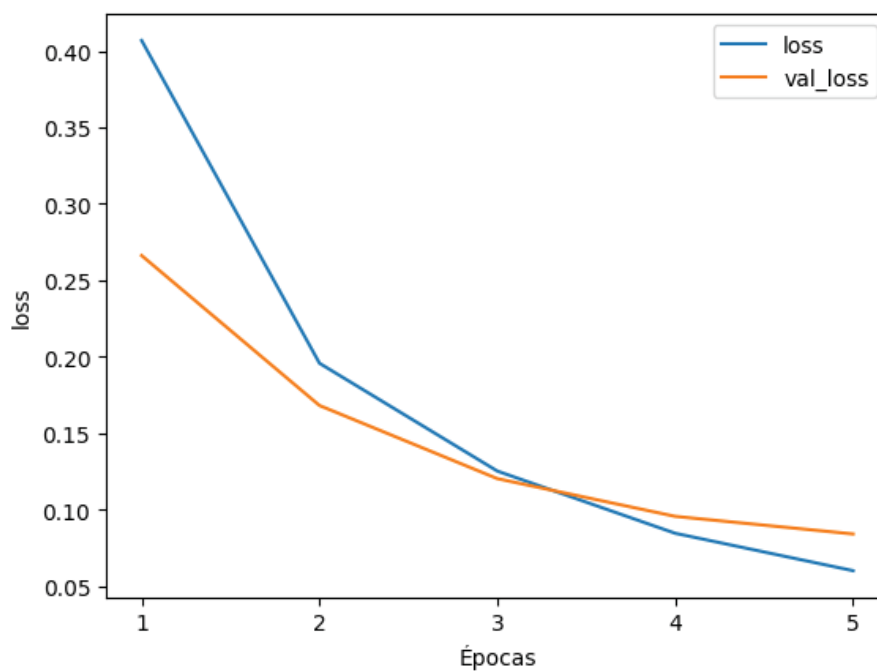
Época	Acurácia (Treino)	Perda (Treino)	Acurácia (Val)	Perda (Val)
1	85,34%	0,5135	92,39%	0,2661
2	95,82%	0,2213	96,52%	0,1680
3	98,36%	0,1390	98,21%	0,1202
4	99,19%	0,0918	98,53%	0,0955
5	99,67%	0,0637	98,37%	0,0840

FONTE: Elaborado pelo autor (2024).

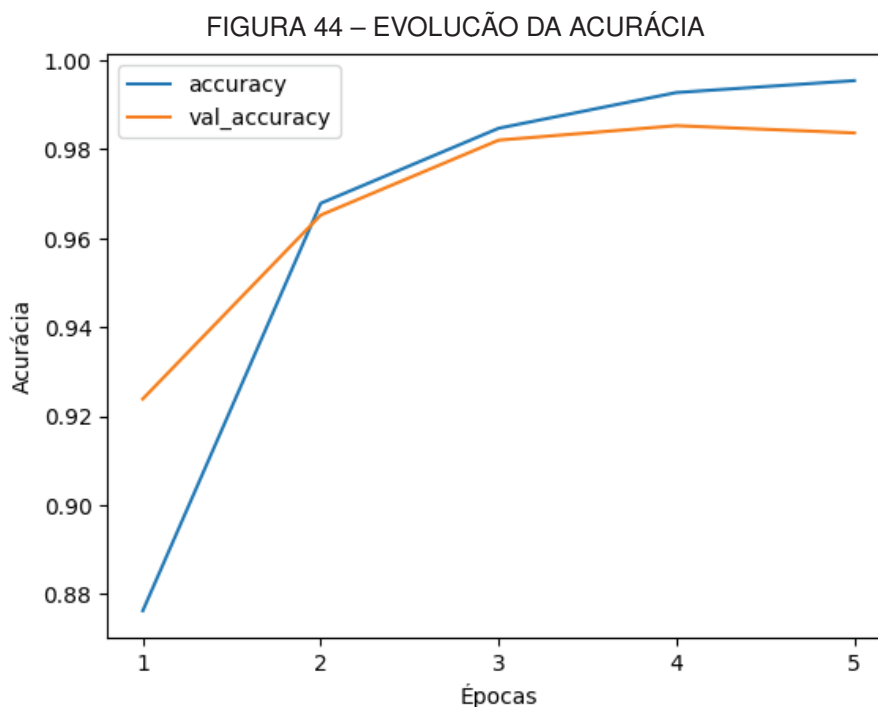
Visualização dos resultados

As Figuras 43 e 44 apresentam a evolução da função de perda e da acurácia ao longo das épocas de treinamento.

FIGURA 43 – EVOLUÇÃO DA FUNÇÃO DE PERDA



FONTE: Elaborado pelo autor (2024).



FONTE: Elaborado pelo autor (2024).

Avaliação final do modelo

Para validar a aplicabilidade do modelo, foi utilizada uma nova mensagem com características típicas de spam:

Is your car dirty? Discover our new product. Free for all. Click the link.

Após o pré-processamento (tokenização e padding), a saída do modelo foi 0.5793. Como o limiar de classificação foi definido como 0,5, o modelo corretamente classificou a mensagem como SPAM.

Código-fonte

```

1 # Importação das Bibliotecas
2 import tensorflow as tf
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import pandas as pd
6 from sklearn.model_selection import train_test_split
7 from tensorflow.keras.layers import Input, Embedding, LSTM, Dense
8 from tensorflow.keras.models import Model
9 from tensorflow.keras.preprocessing.sequence import pad_sequences
10 from tensorflow.keras.preprocessing.text import Tokenizer
11

```

```
12 # Baixar a base de dados
13 !wget http://www.razer.net.br/datasets/spam.csv
14
15 # Carregar a base de dados CSV com codificação ISO-8859-1
16 df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
17
18 # Exibir as primeiras linhas do dataset
19 df.head()
20
21 # Remover colunas irrelevantes para a tarefa
22 df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
23
24 # Renomear colunas para melhor identificação
25 df.columns = ["labels", "data"]
26
27 # Mapear labels para valores binários: ham -> 0, spam -> 1
28 df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
29
30 # Extrair os valores das labels binárias
31 y = df["b_labels"].values
32
33 # Dividir o conjunto em treino (67%) e teste (33%)
34 x_train, x_test, y_train, y_test = train_test_split(df["data"], y, test_size=0.33)
35
36 # Definir número máximo de palavras a serem consideradas
37 num_words = 20000
38
39 # Inicializar o tokenizer com limite de palavras
40 tokenizer = Tokenizer(num_words=num_words)
41
42 # Ajustar o tokenizer ao conjunto de treino
43 tokenizer.fit_on_texts(x_train)
44
45 # Converter textos em sequências de tokens (índices)
46 sequences_train = tokenizer.texts_to_sequences(x_train)
47 sequences_test = tokenizer.texts_to_sequences(x_test)
48
49 # Obter o índice das palavras no vocabulário
50 word2index = tokenizer.word_index
```

```

51
52 # Número total de tokens únicos
53 V = len(word2index)
54 print(f"{V} tokens")
55
56 # Ajustar o tamanho das sequências com padding (tamanho da maior sequência)
57 data_train = pad_sequences(sequences_train)
58 T = data_train.shape[1] # tamanho da sequência
59 data_test = pad_sequences(sequences_test, maxlen=T)
60
61 print("data_train.shape: ", data_train.shape)
62 print("data_test.shape: ", data_test.shape)
63
64 # Definir parâmetros do modelo
65 D = 20 # dimensão do embedding
66 M = 5 # número de unidades na camada LSTM
67
68 # Construção do modelo
69 i = Input(shape=(T,)) # entrada com tamanho fixo T
70 x = Embedding(V+1, D)(i) # camada embedding
71 x = LSTM(M)(x) # camada LSTM
72 x = Dense(1, activation="sigmoid")(x) # camada densa com saída binária
73
74 model = Model(i, x)
75
76 # Exibir resumo da arquitetura do modelo
77 model.summary()
78
79 # Compilar o modelo com função de perda binária e otimizador Adam
80 model.compile(loss="binary_crossentropy", optimizer="adam", metrics=["accuracy"])
81
82 # Número de épocas para treinamento
83 epochs = 5
84
85 # Treinamento do modelo com validação
86 r = model.fit(data_train, y_train, epochs=epochs, validation_data=(data_test, y_test))
87
88 # Visualização da função de perda durante o treinamento
89 plt.plot(r.history["loss"], label="loss")

```

```

90 plt.plot(r.history["val_loss"], label="val_loss")
91 plt.xlabel("Épocas")
92 plt.ylabel("Loss")
93 plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
94 plt.legend()
95 plt.show()
96
97 # Visualização da acurácia durante o treinamento
98 plt.plot(r.history["accuracy"], label="accuracy")
99 plt.plot(r.history["val_accuracy"], label="val_accuracy")
100 plt.xlabel("Épocas")
101 plt.ylabel("Acurácia")
102 plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs+1))
103 plt.legend()
104 plt.show()
105
106 # Predição para texto novo
107 texto = "Is your car dirty? Discover our new product. Free for all. Click the link."
108 seq_texto = tokenizer.texts_to_sequences([texto]) # tokenização
109 data_texto = pad_sequences(seq_texto, maxlen=T) # padding
110
111 pred = model.predict(data_texto) # predição
112 print(pred)
113 print("SPAM" if pred >= 0.5 else "OK")

```

Conclusão

A atividade demonstrou a eficácia da aplicação de RNNs com LSTM na tarefa de detecção de SPAM. O modelo obteve acurácia superior a 98%, mesmo com uma arquitetura simples e treinamento curto. A utilização de embeddings e o ajuste adequado do tamanho das sequências permitiram um bom desempenho geral, evidenciando a viabilidade do uso de redes recorrentes em problemas de classificação de texto. Esta implementação reforça o entendimento prático dos conceitos estudados em sala sobre PLN e Deep Learning.

3. Gerador de Dígitos Fake (GAN)

Esta atividade propôs a implementação de uma Rede Adversária Generativa (*GAN — Generative Adversarial Network*) para a geração de imagens sintéticas semelhantes aos dígitos manuscritos presentes na base de dados MNIST. O modelo desenvolvido é composto por duas redes neurais concor-

rentes: um gerador, responsável por criar imagens falsas, e um discriminador, encarregado de distinguir entre imagens reais e falsas.

O objetivo principal desta atividade foi avaliar a capacidade da GAN em aprender a distribuição de probabilidade dos dados de entrada, de modo a gerar amostras visivelmente realistas.

Base de dados

A base de dados utilizada foi a MNIST, amplamente empregada para tarefas de classificação e geração de imagens de dígitos manuscritos. Cada amostra possui dimensão 28×28 pixels em escala de cinza. As imagens foram normalizadas para o intervalo $[-1, 1]$, de modo a facilitar a convergência do treinamento da rede adversária.

Arquitetura do modelo

A arquitetura do modelo foi dividida em dois módulos principais:

- **Gerador:** Recebe vetores de ruído com dimensão 100 e os transforma em imagens por meio de camadas Dense, BatchNormalization, LeakyReLU e Conv2DTranspose. O objetivo do gerador é produzir imagens indistinguíveis das reais pelo discriminador.
- **Discriminador:** Implementado como uma rede convolucional composta por camadas Conv2D, LeakyReLU e Dropout. Tem como função classificar as imagens como reais ou geradas.

As funções de ativação utilizadas foram a LeakyReLU, no interior das redes, e a função tanh, na saída do gerador, a fim de compatibilizar a faixa de saída com os dados normalizados.

Treinamento do modelo

O processo de treinamento foi realizado por meio de um laço adversarial entre o gerador e o discriminador, durante 100 épocas. A função de perda utilizada foi a BinaryCrossentropy, com rótulos reais atribuídos a imagens reais e rótulos nulos atribuídos às imagens falsas. A cada época, foram geradas e salvas amostras das imagens produzidas para fins de validação visual da evolução do modelo.

Para estabilizar o treinamento, foram empregados otimizadores do tipo Adam, com taxa de aprendizado igual a 10^{-4} . A execução do treinamento considerou também a criação de checkpoints para salvamento intermediário do modelo.

Validação do modelo

A avaliação do desempenho foi realizada de forma qualitativa, por meio da inspeção visual das imagens geradas ao longo das épocas. A Figura 45 apresenta um exemplo das imagens geradas na última época do treinamento.

FIGURA 45 – IMAGENS GERADAS PELO MODELO GAN APÓS 100 ÉPOCAS DE TREINAMENTO



FONTE: Elaborado pelo autor (2024).

Além disso, foi gerado um arquivo no formato GIF contendo a progressão das imagens ao longo das épocas, evidenciando a melhoria na qualidade visual à medida que o treinamento avançou.

Código-fonte

```

1 # Importações de bibliotecas necessárias
2 import tensorflow as tf
3 import glob
4 import imageio
5 import matplotlib.pyplot as plt
6 import numpy as np
7 import os
8 import PIL
9 from tensorflow.keras import layers
10 import time
11 from IPython import display
12
13 # Carregamento da base de dados MNIST
14 (train_images, train_labels), (_, _) = tf.keras.datasets.mnist.load_data()
15
16 # Normalização das imagens para o intervalo [-1, 1]
17 train_images = train_images.reshape(train_images.shape[0], 28, 28, 1).astype('float32')
18     )
19 train_images = (train_images - 127.5) / 127.5

```

```

20 # Parâmetros de buffer e tamanho do lote
21 BUFFER_SIZE = 60000
22 BATCH_SIZE = 256
23
24 # Criação do dataset com embaralhamento e divisão em lotes
25 train_dataset = tf.data.Dataset.from_tensor_slices(train_images).shuffle(BUFFER_SIZE).
    batch(BATCH_SIZE)
26
27 # Definição do modelo do gerador
28 def make_generator_model():
29     model = tf.keras.Sequential()
30     model.add(layers.Dense(7 * 7 * 256, use_bias=False, input_shape=(100,)))
31     model.add(layers.BatchNormalization())
32     model.add(layers.LeakyReLU())
33     model.add(layers.Reshape((7, 7, 256))) # Saída esperada: (None, 7, 7, 256)
34     model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1), padding='same', use_
        bias=False))
35     model.add(layers.BatchNormalization())
36     model.add(layers.LeakyReLU())
37     model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2), padding='same', use_
        bias=False))
38     model.add(layers.BatchNormalization())
39     model.add(layers.LeakyReLU())
40     model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_
        bias=False, activation='tanh'))
41     return model
42
43 # Instanciação e teste inicial do gerador (não treinado)
44 generator = make_generator_model()
45 noise = tf.random.normal([1, 100])
46 generated_image = generator(noise, training=False)
47 plt.imshow(generated_image[0, :, :, 0], cmap='gray')
48
49 # Definição do modelo do discriminador
50 def make_discriminator_model():
51     model = tf.keras.Sequential()
52     model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same', input_shape
        =[28, 28, 1]))
53     model.add(layers.LeakyReLU())

```

```

54     model.add(layers.Dropout(0.3))
55     model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
56     model.add(layers.LeakyReLU())
57     model.add(layers.Dropout(0.3))
58     model.add(layers.Flatten())
59     model.add(layers.Dense(1))
60     return model
61
62 # Instanciação e teste inicial do discriminador (não treinado)
63 discriminator = make_discriminator_model()
64 decision = discriminator(generated_image)
65 print(decision)
66
67 # Funções de perda para o gerador e discriminador
68 cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
69
70 def discriminator_loss(real_output, fake_output):
71     real_loss = cross_entropy(tf.ones_like(real_output), real_output)
72     fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
73     total_loss = real_loss + fake_loss
74     return total_loss
75
76 def generator_loss(fake_output):
77     return cross_entropy(tf.ones_like(fake_output), fake_output)
78
79 # Definição dos otimizadores para ambos os modelos
80 generator_optimizer = tf.keras.optimizers.Adam(1e-4)
81 discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
82
83 # Configuração de checkpoints para salvar o progresso do treinamento
84 checkpoint_dir = './training_checkpoints'
85 checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
86 checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
87                                   discriminator_optimizer=discriminator_optimizer,
88                                   generator=generator,
89                                   discriminator=discriminator)
89
90
91 # Definições do treinamento
92 EPOCHS = 100

```

```

93 noise_dim = 100
94 num_examples_to_generate = 16
95
96 # Semente fixa para visualização do progresso do gerador
97 seed = tf.random.normal([num_examples_to_generate, noise_dim])
98
99 # Etapa de treinamento (compilada com @tf.function)
100 @tf.function
101 def train_step(images):
102     noise = tf.random.normal([BATCH_SIZE, noise_dim])
103     with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
104         generated_images = generator(noise, training=True)
105         real_output = discriminator(images, training=True)
106         fake_output = discriminator(generated_images, training=True)
107         gen_loss = generator_loss(fake_output)
108         disc_loss = discriminator_loss(real_output, fake_output)
109         gradients_of_generator = gen_tape.gradient(gen_loss, generator.trainable_variables)
110         gradients_of_discriminator = disc_tape.gradient(disc_loss, discriminator.trainable_variables)
111         generator_optimizer.apply_gradients(zip(gradients_of_generator, generator.trainable_variables))
112         discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator, discriminator.trainable_variables))
113
114 # Função principal de treinamento
115 def train(dataset, epochs):
116     for epoch in range(epochs):
117         start = time.time()
118         for image_batch in dataset:
119             train_step(image_batch)
120         display.clear_output(wait=True)
121         generate_and_save_images(generator, epoch + 1, seed)
122         if (epoch + 1) % 15 == 0:
123             checkpoint.save(file_prefix=checkpoint_prefix)
124             print('Time for epoch {} is {} sec'.format(epoch + 1, time.time() - start))
125         display.clear_output(wait=True)
126         generate_and_save_images(generator, epochs, seed)
127

```

```

128 # Geração e salvamento de imagens após cada época
129 def generate_and_save_images(model, epoch, test_input):
130     predictions = model(test_input, training=False)
131     fig = plt.figure(figsize=(4, 4))
132     for i in range(predictions.shape[0]):
133         plt.subplot(4, 4, i + 1)
134         plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
135         plt.axis('off')
136     plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
137     plt.show()
138
139 # Início do treinamento
140 train(train_dataset, EPOCHS)
141
142 # Restauração do último checkpoint
143 checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))
144
145 # Função para exibir imagem de uma época específica
146 def display_image(epoch_no):
147     return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))
148
149 display_image(EPOCHS)
150
151 # Criação do GIF animado com os resultados gerados
152 anim_file = 'dcgan.gif'
153 with imageio.get_writer(anim_file, mode='I') as writer:
154     filenames = glob.glob('image*.png')
155     filenames = sorted(filenames)
156     for filename in filenames:
157         image = imageio.imread(filename)
158         writer.append_data(image)
159     image = imageio.imread(filename)
160     writer.append_data(image)
161
162 import tensorflow_docs.vis.embed as embed
163 embed.embed_file(anim_file)

```

Conclusão

A atividade demonstrou a aplicação de redes adversárias generativas (GANs) na geração de dígitos manuscritos sintéticos, utilizando a base de dados MNIST. O modelo gerador foi capaz de produzir imagens visualmente coerentes, enquanto o discriminador aprendeu a distinguir entre imagens reais e falsas ao longo do treinamento. A tarefa permitiu consolidar os conceitos apresentados em aula e reforçou a importância das técnicas de normalização, arquitetura e validação em problemas de geração de dados.

4. Tradutor de Textos (Transformer)

Neste experimento foi desenvolvido um modelo de tradução automática de sentenças do português para o inglês, utilizando a arquitetura *Transformer*, amplamente adotada em tarefas de processamento de linguagem natural. Essa arquitetura permite capturar dependências de longo alcance entre palavras por meio de mecanismos de atenção multi-cabeça, oferecendo desempenho significativamente superior aos modelos sequenciais tradicionais.

Base de dados

A base de dados utilizada foi a *TED Talks Translation*, disponibilizada pelo *TensorFlow Datasets* com o identificador `ted_hrlr_translate/pt_to_en`. Esta base contém pares de sentenças paralelas em português e inglês extraídas de transcrições de palestras TED. O conjunto foi dividido automaticamente em dados de treino e validação.

A seguir, um exemplo de par de sentenças extraído da base:

- **Português:** “mas eles não tinham a curiosidade de me testar .”
- **Inglês:** “but they didn’t test for curiosity .”

A base já vem pré-processada com separação entre conjuntos de treino e validação, sendo que a tokenização foi realizada com o modelo `ted_hrlr_translate_pt_en_converter`, contendo os vocabulários dos dois idiomas.

Arquitetura do modelo

O modelo desenvolvido segue a arquitetura *Transformer*, composta por blocos de codificadores (*encoders*) e decodificadores (*decoders*) empilhados. Os principais componentes implementados incluem:

- **Codificação Posicional:** incorpora informações de ordem sequencial sem o uso de recorrência.
- **Atenção Escalonada:** permite que cada posição na entrada preste atenção a todas as outras posições, ponderando a relevância.
- **Atenção Multi-cabeça:** projeta as representações em diferentes subespaços e aplica atenção paralelamente.

- **Rede Feedforward:** composta por duas camadas densas aplicadas ponto a ponto.

O modelo foi configurado com os seguintes hiperparâmetros:

- Camadas (*num_layers*): 4
- Dimensão do modelo (*d_model*): 128
- Unidades da camada feedforward (*dff*): 512
- Número de cabeças de atenção: 8
- Taxa de dropout: 0,1

A tokenização das sentenças foi feita utilizando os tokenizadores da própria base de dados, com vocabulário específico para cada idioma.

Treinamento do modelo

O treinamento foi realizado com 20 épocas sobre o respectivo conjunto de dados. A função de perda utilizada foi a entropia cruzada categórica com *masking*, para ignorar os tokens de padding. O otimizador escolhido foi o *Adam*, com agendamento de taxa de aprendizado personalizado, conforme sugerido na literatura para Transformers.

Durante o treinamento, foram salvos pontos de verificação a cada 5 épocas. Em cada época, a acurácia e a perda foram registradas, permitindo acompanhar a evolução do modelo. A Tabela 48 apresenta um resumo dos resultados finais em épocas selecionadas.

Observa-se uma redução progressiva no valor da função de perda e aumento da acurácia ao longo do processo, indicando aprimoramento do desempenho do modelo. Nas etapas finais, a função de perda estabilizou próxima a 1,45, enquanto a acurácia alcançou cerca de 68%, evidenciando capacidade do modelo em aprender e generalizar os dados.

TABELA 48 – RESUMO DO DESEMPENHO DO MODELO POR ÉPOCA

Época	Loss Final	Accuracy Final
1	6.7044	0.1187
2	4.9122	0.2434
3	4.4245	0.2892
18	1.5288	0.6676
19	1.4872	0.6747
20	1.4514	0.6799

FONTE: Elaborado pelo autor (2024).

Validação do modelo

A validação foi realizada utilizando o conjunto separado da base de dados. Após o treinamento, foi implementado um módulo `Translator`, responsável por gerar as traduções a partir de uma sentença de entrada em português.

Como exemplo, dada a sentença:

Eu li sobre triceratops na enciclopédia.

O modelo retornou a seguinte tradução:

i read about triopholes in egypt .

Observa-se que, apesar da estrutura sintática estar adequada, o modelo cometeu um erro semântico ao traduzir “triceratops” como “triopholes” e “enciclopédia” como “egypt”. Esse comportamento pode ser explicado por limitações no vocabulário e pela quantidade reduzida de dados envolvendo termos específicos.

Código-fonte

```
1 #imports e Configurações Iniciais
2 import os
3 import time
4 import numpy as np
5 import matplotlib.pyplot as plt
6 import tensorflow as tf
7 import tensorflow_datasets as tfds
8 import tensorflow_text as text
9 import logging
10
11 # Evita logs excessivos do TensorFlow
12 logging.getLogger('tensorflow').setLevel(logging.ERROR)
13
14 # Carrega os dados PTEN do TensorFlow Datasets
15 examples, metadata = tfds.load('ted_hrlr_translate/pt_to_en', with_info=True, as_supervised=True)
16 train_examples, val_examples = examples['train'], examples['validation']
17
18 #baixar Tokenizer pré-treinado do TensorFlow
19 model_name = "ted_hrlr_translate_pt_en_converter"
20
21 # Baixa e extrai o modelo de tokenização (um para PT e outro para EN)
```

```

22 tf.keras.utils.get_file(
23     f"{model_name}.zip",
24     f"https://storage.googleapis.com/download.tensorflow.org/models/{model_name}.zip",
25     cache_dir='.', cache_subdir='', extract=True
26 )
27
28 # Carrega o tokenizer salvo
29 tokenizers = tf.saved_model.load(model_name)
30
31
32 # Função para tokenizar pares (pt, en)
33 def tokenize_pairs(pt, en):
34     pt = tokenizers.pt.tokenize(pt).to_tensor() # Tokeniza e faz padding
35     en = tokenizers.en.tokenize(en).to_tensor()
36     return pt, en
37
38 # Parâmetros do pipeline de dados
39 BUFFER_SIZE = 20000
40 BATCH_SIZE = 64
41
42 # Pipeline completo: shuffle, batch, tokenize, prefetch
43 def make_batches(ds):
44     return (ds.cache()
45             .shuffle(BUFFER_SIZE)
46             .batch(BATCH_SIZE)
47             .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
48             .prefetch(tf.data.AUTOTUNE))
49
50 # Aplicar pipeline
51 train_batches = make_batches(train_examples)
52 val_batches = make_batches(val_examples)
53
54
55 # Calcula os ângulos para a codificação posicional
56 def get_angles(pos, i, d_model):
57     angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
58     return pos * angle_rates
59
60 # Gera a matriz de codificação posicional

```

```

61 def positional_encoding(position, d_model):
62     angle_rads = get_angles(
63         np.arange(position)[: , np.newaxis],
64         np.arange(d_model)[np.newaxis, :],
65         d_model
66     )
67
68     # Aplica seno nos índices pares e cosseno nos ímpares
69     angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
70     angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
71
72     pos_encoding = angle_rads[np.newaxis, ...] # Adiciona dimensão batch
73     return tf.cast(pos_encoding, dtype=tf.float32)
74
75
76 # Máscara de padding (usada para ignorar tokens com valor 0)
77 def create_padding_mask(seq):
78     return tf.cast(tf.math.equal(seq, 0), tf.float32)[: , tf.newaxis, tf.newaxis, :]
79
80 # Máscara look-ahead para impedir que o modelo "veja o futuro" durante o treinamento
81 def create_look_ahead_mask(size):
82     return 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
83
84 # Implementação da atenção escalonada: atenção(Q, K, V)
85 def scaled_dot_product_attention(q, k, v, mask):
86     matmul_qk = tf.matmul(q, k, transpose_b=True) # Produto escalar entre Q e K^T
87     dk = tf.cast(tf.shape(k)[-1], tf.float32) # Escalonamento pela raiz de d_k
88     scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
89
90     if mask is not None:
91         # Máscara define - onde necessário (valores muito negativos)
92         scaled_attention_logits += (mask * -1e9)
93
94     # Softmax na última dimensão (seq_len_k)
95     attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
96
97     # Produto da atenção com V
98     output = tf.matmul(attention_weights, v)
99     return output, attention_weights

```

```

100
101 # Atenção Multi-cabeças
102 class MultiHeadAttention(tf.keras.layers.Layer):
103     def __init__(self, d_model, num_heads):
104         super().__init__()
105         self.num_heads = num_heads
106         self.d_model = d_model
107
108         assert d_model % num_heads == 0 # d_model deve ser divisível pelo número de
cabeças
109         self.depth = d_model // num_heads
110
111         # Camadas lineares para Q, K, V
112         self.wq = tf.keras.layers.Dense(d_model)
113         self.wk = tf.keras.layers.Dense(d_model)
114         self.wv = tf.keras.layers.Dense(d_model)
115         self.dense = tf.keras.layers.Dense(d_model)
116
117     def split_heads(self, x, batch_size):
118         # Divide o vetor de embedding em múltiplas cabeças
119         x = tf.reshape(x, (batch_size, -1, self.num_heads, self.depth))
120         return tf.transpose(x, perm=[0, 2, 1, 3]) # (batch, heads, seq_len, depth)
121
122     def call(self, v, k, q, mask):
123         batch_size = tf.shape(q)[0]
124
125         # Passa pelas camadas lineares
126         q = self.split_heads(self.wq(q), batch_size)
127         k = self.split_heads(self.wk(k), batch_size)
128         v = self.split_heads(self.wv(v), batch_size)
129
130         # Aplica atenção
131         scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v,
mask)
132
133         # Concatena as cabeças novamente
134         scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
135         concat_attention = tf.reshape(scaled_attention, (batch_size, -1, self.d_model)
)

```

```

136
137     # Camada final densa
138     output = self.dense(concat_attention)
139     return output, attention_weights
140
141 # Rede neural feedforward aplicada a cada posição da sequência
142 def point_wise_feed_forward_network(d_model, dff):
143     return tf.keras.Sequential([
144         tf.keras.layers.Dense(dff, activation='relu'), # Expande dimensionalidade
145         tf.keras.layers.Dense(d_model)                # Retorna para d_model
146     ])
147
148
149 class EncoderLayer(tf.keras.layers.Layer):
150     def __init__(self, d_model, num_heads, dff, rate=0.1):
151         # Inicializa uma camada do encoder Transformer
152         super().__init__()
153         self.mha = MultiHeadAttention(d_model, num_heads) # Atenção multi-cabeças
154         self.ffn = point_wise_feed_forward_network(d_model, dff) # Rede feed-forward
155
156         self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6) #
157         Normalização 1
158
159         self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6) #
160         Normalização 2
161
162         self.dropout1 = tf.keras.layers.Dropout(rate) # Dropout após atenção
163         self.dropout2 = tf.keras.layers.Dropout(rate) # Dropout após FFN
164
165
166     def call(self, x, training, mask):
167         # Aplica atenção multi-cabeças
168         attn_output, _ = self.mha(x, x, x, mask)
169
170         # Residual + dropout + norm (atenção)
171         out1 = self.layernorm1(x + self.dropout1(attn_output, training=training))
172
173         # Residual + dropout + norm (feed-forward)
174         out2 = self.layernorm2(out1 + self.dropout2(self.ffn(out1), training=training))
175     )

```

```

172     return out2
173
174
175 class DecoderLayer(tf.keras.layers.Layer):
176     def __init__(self, d_model, num_heads, dff, rate=0.1):
177         # Inicializa uma camada do decoder Transformer
178         super().__init__()
179         self.mha1 = MultiHeadAttention(d_model, num_heads) # Autoatenção com look-
180         ahead
181         self.mha2 = MultiHeadAttention(d_model, num_heads) # Atenção ao encoder
182         self.ffn = point_wise_feed_forward_network(d_model, dff) # Feed-forward
183
184         self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6) # Norm 1
185         self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6) # Norm 2
186         self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6) # Norm 3
187
188         self.dropout1 = tf.keras.layers.Dropout(rate) # Dropout 1
189         self.dropout2 = tf.keras.layers.Dropout(rate) # Dropout 2
190         self.dropout3 = tf.keras.layers.Dropout(rate) # Dropout 3
191
192     def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
193         # Aplica autoatenção com máscara look-ahead
194         attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
195         out1 = self.layernorm1(x + self.dropout1(attn1, training=training))
196
197         # Aplica atenção ao encoder (usa enc_output como K e V)
198         attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1, padding_
199         mask)
200         out2 = self.layernorm2(out1 + self.dropout2(attn2, training=training))
201
202         # Aplica rede feed-forward
203         out3 = self.layernorm3(out2 + self.dropout3(self.ffn(out2), training=training)
204         )
205
206         return out3, attn_weights_block1, attn_weights_block2
207
208 class Encoder(tf.keras.layers.Layer):

```

```

208 def __init__(self, num_layers, d_model, num_heads, dff,
209               input_vocab_size, maximum_position_encoding, rate=0.1):
210     # Inicializa o encoder completo com múltiplas camadas
211     super().__init__()
212     self.d_model = d_model
213     self.num_layers = num_layers
214
215     self.embedding = tf.keras.layers.Embedding(input_vocab_size, d_model) #
Embedding dos tokens
216     self.pos_encoding = positional_encoding(maximum_position_encoding, d_model) #
Codificação posicional
217
218     # Lista de camadas do encoder
219     self.enc_layers = [
220         EncoderLayer(d_model, num_heads, dff, rate)
221         for _ in range(num_layers)
222     ]
223
224     self.dropout = tf.keras.layers.Dropout(rate) # Dropout após embedding
225
226 def call(self, x, training, mask):
227     # Converte tokens em vetores e aplica codificação posicional
228     x = self.embedding(x)
229     x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32)) # Escalonamento
230     x += self.pos_encoding[:, :tf.shape(x)[1], :] # Soma posição
231
232     x = self.dropout(x, training=training) # Dropout
233
234     # Passa por cada camada do encoder
235     for i in range(self.num_layers):
236         x = self.enc_layers[i](x, training, mask)
237
238     return x # Retorna saída final do encoder
239
240
241 class Decoder(tf.keras.layers.Layer):
242     def __init__(self, num_layers, d_model, num_heads, dff,
243                 target_vocab_size, maximum_position_encoding, rate=0.1):
244         # Inicializa o decoder completo com múltiplas camadas

```

```

245     super().__init__()
246     self.d_model = d_model
247     self.num_layers = num_layers
248
249     self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model) #
Embedding do target
250     self.pos_encoding = positional_encoding(maximum_position_encoding, d_model) #
Codificação posicional
251
252     # Lista de camadas do decoder
253     self.dec_layers = [
254         DecoderLayer(d_model, num_heads, dff, rate)
255         for _ in range(num_layers)
256     ]
257
258     self.dropout = tf.keras.layers.Dropout(rate) # Dropout após embedding
259
260
261 # Classe que define o modelo Transformer completo (encoder + decoder)
262 class Transformer(tf.keras.Model):
263     # Inicializa o Transformer com encoder, decoder e camada final de saída
264     def __init__(self, num_layers, d_model, num_heads, dff,
265                 input_vocab_size, target_vocab_size, pe_input, pe_target, rate=0.1):
266         super().__init__()
267         # Inicializa o encoder
268         self.encoder = Encoder(num_layers, d_model, num_heads, dff,
269                               input_vocab_size, pe_input, rate)
270         # Inicializa o decoder
271         self.decoder = Decoder(num_layers, d_model, num_heads, dff,
272                               target_vocab_size, pe_target, rate)
273         # Camada densa final que gera logits para o vocabulário de saída
274         self.final_layer = tf.keras.layers.Dense(target_vocab_size)
275
276     # Define o forward pass do Transformer
277     def call(self, inputs, training):
278         # Desempacota as entradas: inp (entrada) e tar (target)
279         inp, tar = inputs
280
281         # Cria as máscaras necessárias para atenção

```



```

282     enc_padding_mask, look_ahead_mask, dec_padding_mask = self.create_masks(inp,
tar)
283
284     # Passa a entrada pelo encoder
285     enc_output = self.encoder(inp, training, enc_padding_mask)
286
287     # Passa a entrada do decoder + saída do encoder pelo decoder
288     dec_output, attention_weights = self.decoder(
289         tar, enc_output, training, look_ahead_mask, dec_padding_mask)
290
291     # Passa a saída do decoder pela camada final densa
292     final_output = self.final_layer(dec_output)
293
294     # Retorna os logits e os pesos de atenção
295     return final_output, attention_weights
296
297     # Cria as máscaras de padding e look-ahead para encoder e decoder
298     def create_masks(self, inp, tar):
299         # Máscara de padding para o encoder
300         enc_padding_mask = create_padding_mask(inp)
301
302         # Máscara de padding para o decoder (2ª atenção)
303         dec_padding_mask = create_padding_mask(inp)
304
305         # Máscara look-ahead para evitar ver tokens futuros (1ª atenção do decoder)
306         look_ahead_mask = create_look_ahead_mask(tf.shape(tar)[1])
307
308         # Máscara de padding para a entrada do decoder
309         dec_target_padding_mask = create_padding_mask(tar)
310
311         # Combina look-ahead com padding mask
312         look_ahead_mask = tf.maximum(dec_target_padding_mask, look_ahead_mask)
313
314         # Retorna todas as máscaras
315         return enc_padding_mask, look_ahead_mask, dec_padding_mask
316
317
318     # Define hiperparâmetros do modelo Transformer
319     num_layers = 4

```

```

320 d_model = 128
321 dff = 512
322 num_heads = 8
323 dropout_rate = 0.1
324
325 # Classe para agendamento customizado da taxa de aprendizado (learning rate)
326 class CustomSchedule(tf.keras.optimizers.schedules.LearningRateSchedule):
327     # Inicializa com d_model e warmup_steps
328     def __init__(self, d_model, warmup_steps=4000):
329         super(CustomSchedule, self).__init__()
330         self.d_model = d_model
331         self.d_model = tf.cast(self.d_model, tf.float32) # Converte para float32
332         self.warmup_steps = warmup_steps
333
334     # Calcula o learning rate baseado no passo atual (step)
335     def __call__(self, step):
336         step = tf.cast(step, tf.float32) # Converte step para float32 para evitar
337         erro
338         arg1 = tf.math.rsqrt(step) # 1 / sqrt(step)
339         arg2 = step * (self.warmup_steps ** -1.5) # step * warmup_steps^-1.5
340         # Retorna o mínimo entre os dois argumentos multiplicado por 1 / sqrt(d_model)
341         return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)
342
343 # Instancia o scheduler de learning rate
344 learning_rate = CustomSchedule(d_model)
345
346 # Define o otimizador Adam com parâmetros específicos para Transformer
347 optimizer = tf.keras.optimizers.Adam(
348     learning_rate, beta_1=0.9, beta_2=0.98, epsilon=1e-9)
349
350 # Define a função de perda usando SparseCategoricalCrossentropy com logits
351 loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
352     from_logits=True, reduction='none')
353
354 # Função de perda que ignora tokens de padding (valor 0)
355 def loss_function(real, pred):
356     mask = tf.math.logical_not(tf.math.equal(real, 0)) # Máscara para ignorar padding
357     loss_ = loss_object(real, pred) # Calcula a perda sem máscara

```

```

357     mask = tf.cast(mask, dtype=loss_.dtype) # Converte máscara para mesmo tipo da
        perda
358     loss_ *= mask # Aplica máscara para zerar perdas em padding
359     return tf.reduce_sum(loss_) / tf.reduce_sum(mask) # Média ponderada da perda
360
361 # Função de acurácia que considera apenas tokens não padding
362 def accuracy_function(real, pred):
363     accuracies = tf.equal(real, tf.argmax(pred, axis=2)) # Verifica acertos token a
        token
364     mask = tf.math.logical_not(tf.math.equal(real, 0)) # Máscara para ignorar padding
365     accuracies = tf.math.logical_and(mask, accuracies) # Aplica máscara nos acertos
366     accuracies = tf.cast(accuracies, dtype=tf.float32) # Converte para float32
367     mask = tf.cast(mask, dtype=tf.float32) # Converte máscara para float32
368     return tf.reduce_sum(accuracies) / tf.reduce_sum(mask) # Média ponderada da
        acurácia
369
370 # Métrica para armazenar a média da perda durante o treinamento
371 train_loss = tf.keras.metrics.Mean(name='train_loss')
372
373 # Métrica para armazenar a média da acurácia durante o treinamento
374 train_accuracy = tf.keras.metrics.Mean(name='train_accuracy')
375
376 # Instancia o modelo Transformer com os hiperparâmetros definidos
377 transformer = Transformer(
378     num_layers=num_layers,
379     d_model=d_model,
380     num_heads=num_heads,
381     dff=dff,
382     input_vocab_size=tokenizers.pt.get_vocab_size().numpy(), # Tamanho vocabulário
        português
383     target_vocab_size=tokenizers.en.get_vocab_size().numpy(), # Tamanho vocabulário
        inglês
384     pe_input=1000, # Máximo comprimento para codificação posicional do input
385     pe_target=1000, # Máximo comprimento para codificação posicional do target
386     rate=dropout_rate)
387
388 # Define caminho para salvar checkpoints do treinamento
389 checkpoint_path = "./checkpoints/train"
390

```

```

391 # Cria checkpoint do modelo e do otimizador
392 ckpt = tf.train.Checkpoint(transformer=transformer, optimizer=optimizer)
393
394 # Gerenciador para salvar múltiplos checkpoints e controlar quantidade máxima
395 ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path, max_to_keep=5)
396
397 # Verifica se existe checkpoint salvo e restaura o último, se houver
398 if ckpt_manager.latest_checkpoint:
399     ckpt.restore(ckpt_manager.latest_checkpoint)
400     print('Latest checkpoint restored!!')
401
402 # Define número total de épocas para treinamento
403 EPOCHS = 20
404
405 # Assinatura do train_step para tf.function, especificando tipos e formas dos tensores
    de entrada
406 train_step_signature = [
407     tf.TensorSpec(shape=(None, None), dtype=tf.int64), # Entrada: batch_size x seq_
    len
408     tf.TensorSpec(shape=(None, None), dtype=tf.int64), # Target: batch_size x seq_len
409 ]
410
411 # Função de treinamento de um passo (batch), compilada com tf.function para otimização
412 @tf.function(input_signature=train_step_signature)
413 def train_step(inp, tar):
414     # Divide o target em input e real para o decoder (shifted right)
415     tar_inp = tar[:, :-1] # Target de entrada para decoder (sem último token)
416     tar_real = tar[:, 1:] # Target esperado para cálculo da perda (sem primeiro token
    )
417
418     # Grava as operações para cálculo de gradiente
419     with tf.GradientTape() as tape:
420         # Executa o transformer no modo treinamento e obtém previsões
421         predictions, _ = transformer([inp, tar_inp], training=True)
422         # Calcula a perda entre as previsões e o target real
423         loss = loss_function(tar_real, predictions)
424
425     # Calcula gradientes da perda em relação às variáveis treináveis do transformer
426     gradients = tape.gradient(loss, transformer.trainable_variables)

```

```

427     # Aplica gradientes para atualizar os pesos do modelo
428     optimizer.apply_gradients(zip(gradients, transformer.trainable_variables))
429
430     # Atualiza as métricas globais de perda e acurácia para o batch
431     train_loss(loss)
432     train_accuracy(accuracy_function(tar_real, predictions))
433
434 # Loop principal de treinamento para o número definido de épocas
435 for epoch in range(EPOCHS):
436     start = time.time() # Marca o tempo de início da época
437     train_loss.reset_state() # Reseta métricas de perda para nova época
438     train_accuracy.reset_state() # Reseta métricas de acurácia para nova época
439
440     # Loop pelos batches de dados de treinamento
441     # inp = frase em português, tar = frase em inglês
442     for (batch, (inp, tar)) in enumerate(train_batches):
443         train_step(inp, tar) # Executa passo de treinamento
444
445         # A cada 50 batches, imprime progresso parcial
446         if batch % 50 == 0:
447             print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
448             Accuracy {train_accuracy.result():.4f}')
449
450         # A cada 5 épocas, salva checkpoint do modelo
451         if (epoch + 1) % 5 == 0:
452             ckpt_save_path = ckpt_manager.save()
453             print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')
454
455         # Imprime resumo da época (perda e acurácia)
456         print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy {train_accuracy.
457         result():.4f}')
458         print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')
459
460 # Classe para traduzir sentenças usando o transformer treinado
461 class Translator(tf.Module):
462     # Inicializa com tokenizadores e modelo transformer
463     def __init__(self, tokenizers, transformer):
464         self.tokenizers = tokenizers
465         self.transformer = transformer

```

```

464
465 # Método para realizar a tradução, recebe uma sentença e tamanho máximo de saída
466 def __call__(self, sentence, max_length=20):
467     # Assegura que a sentença é um tensor tf.Tensor
468     assert isinstance(sentence, tf.Tensor)
469
470     # Se tensor escalar, adiciona dimensão batch
471     if len(sentence.shape) == 0:
472         sentence = sentence[tf.newaxis]
473
474     # Tokeniza a sentença em português e converte para tensor
475     sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
476     encoder_input = sentence # Entrada do encoder
477
478     # Obtém tokens de start e end para o idioma inglês (target)
479     start_end = self.tokenizers.en.tokenize([' '])[0]
480     start = start_end[0][tf.newaxis]
481     end = start_end[1][tf.newaxis]
482
483     # Inicializa array dinâmico para armazenar tokens gerados na saída
484     output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
485     output_array = output_array.write(0, start) # Primeiro token é token start
486
487     # Loop para gerar tokens até max_length
488     for i in tf.range(max_length):
489         output = tf.transpose(output_array.stack()) # Pilha dos tokens gerados
até agora
490         # Obtém predições do transformer no modo inferência
491         predictions, _ = self.transformer([encoder_input, output], training=False)
492         predictions = predictions[:, -1:, :] # Última predição (token mais
recente)
493         predicted_id = tf.argmax(predictions, axis=-1) # Token com maior
probabilidade
494
495         # Escreve token predito no array de saída
496         output_array = output_array.write(i+1, predicted_id[0])
497
498         # Para o loop se token end for gerado
499         if predicted_id == end:

```

```

500         break
501
502     # Converte array de tokens para tensor
503     output = tf.transpose(output_array.stack()) # Shape (1, tokens)
504
505     # Detokeniza para texto e obtém tokens legíveis
506     text = tokenizers.en.detokenize(output)[0]
507     tokens = tokenizers.en.lookup(output)[0]
508
509     # Também obtém pesos de atenção para análise (não usado aqui)
510     _, attention_weights = self.transformer([encoder_input, output[:, :-1]],
511                                           training=False)
512
513     return text, tokens, attention_weights
514
515 # Instancia o tradutor com tokenizadores e modelo Transformer
516 translator = Translator(tokenizers, transformer)
517
518 # Exemplo de frase para traduzir
519 sentence = "Eu li sobre triceratops na enciclopédia."
520
521 # Realiza a tradução usando o tradutor instanciado
522 translated_text, translated_tokens, attention_weights = translator(tf.constant(
523     sentence))
524
525 # Imprime o resultado da tradução
526 print(f'{"Prediction":15s}: {translated_text}')

```

Conclusão

Este experimento demonstrou a implementação completa da arquitetura *Transformer* para tradução automática de sentenças. Apesar de limitações nos resultados pontuais, o modelo foi capaz de capturar a estrutura das sentenças e realizar traduções com coerência parcial. Melhorias podem ser alcançadas com mais épocas de treinamento, ajuste fino dos hiperparâmetros e ampliação da base de dados.

APÊNDICE 10 – BIG DATA

A – ENUNCIADO

Projeto de Big Data

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

Escopo de Projeto: Avaliação de Preços no Contexto Supermercadista no Brasil

1. Objetivo do Projeto

Desenvolver um sistema de Big Data para integração e análise de preços que permita a avaliação precisa e em tempo real dos preços praticados nos setores de varejo e atacado, no contexto supermercadista no território nacional. O sistema integrará dados de múltiplas fontes, incluindo bases de dados de pontos comerciais físicos, informações de representantes de produtos e dados de compras online, proporcionando uma visão abrangente e atualizada do mercado.

2. Escopo

O projeto abrangerá as seguintes etapas:

a) Levantamento de Requisitos

- Identificar todas as fontes de dados disponíveis, como bases de dados de pontos comerciais (varejo e atacado), informações fornecidas pelos representantes de produtos, planilhas de contato e vendas e dados de compras online.
- Definir os indicadores-chave de desempenho (KPIs) para avaliação de preços, como variação de preços por região, sazonalidade, concorrência, e margem de lucro.

b) Integração de Dados

- Conectar e integrar as diferentes bases de dados de pontos comerciais, utilizando ETL (Extração, Transformação e Carga) para padronização e limpeza dos dados.
- Integrar os dados de representantes de produtos, com foco em negociações de preços, promoções e acordos comerciais.
- Conectar as bases de dados de compras online, mapeando o comportamento do consumidor e identificando tendências de mercado.

c) Desenvolvimento do Sistema de Análise

- Implementar um sistema de Big Data utilizando tecnologias como Hadoop, Spark, ou outras, para processamento e análise em larga escala.
- Desenvolver algoritmos de machine learning para prever tendências de preços e identificar oportunidades de ajuste.
- Criar dashboards interativos para visualização dos dados e geração de relatórios customizados, utilizando ferramentas como Power BI, Tableau ou soluções baseadas em Python.

d) **Validação e Testes**

- Realizar testes de desempenho e acurácia com conjuntos de dados históricos e em tempo real.
- Validar os modelos de previsão de preços e ajustar os algoritmos conforme necessário.
- Implementar um ambiente de testes com amostras de dados para validação contínua.

e) **Treinamento e Capacitação**

- Capacitar os colaboradores dos supermercados e representantes de produtos na utilização do sistema.
- Desenvolver manuais e tutoriais para a correta interpretação dos dados e utilização das ferramentas de análise.

f) **Implementação e Manutenção**

- Implementar o sistema nos pontos comerciais selecionados, garantindo a integração contínua com as bases de dados.
- Estabelecer um cronograma de manutenção e atualização do sistema, garantindo que os dados permaneçam atualizados e precisos.
- Monitorar a performance do sistema e realizar ajustes conforme necessário para otimizar o desempenho.

3. **Benefícios Esperados**

- **Melhoria na competitividade:** Ajustes de preços mais rápidos e precisos, alinhados com as tendências de mercado.
- **Redução de custos:** Identificação de oportunidades de compra e negociação com fornecedores.
- **Satisfação do consumidor:** Preços mais justos e competitivos, resultando em maior fidelização.
- **Decisões baseadas em dados:** Capacidade de tomar decisões estratégicas baseadas em análises profundas e preditivas.

4. **Recursos Necessários**

- **Tecnologia:** Servidores de processamento de dados, ferramentas de Big Data (Hadoop, Spark), ferramentas de visualização (Power BI, Tableau).
- **Equipe:** Cientistas de dados, engenheiros de dados, especialistas em BI, desenvolvedores de software, e analistas de mercado.
- **Orçamento:** Investimento inicial para aquisição de tecnologia, contratação de especialistas, e desenvolvimento do sistema.

5. Cronograma

- **Fase 1:** Levantamento de Requisitos e Integração de Dados (3 meses)
- **Fase 2:** Desenvolvimento do Sistema e Algoritmos de Análise (6 meses)
- **Fase 3:** Validação, Testes, e Treinamento (2 meses)
- **Fase 4:** Implementação e Monitoramento (3 meses)

6. Riscos e Mitigações

- **Riscos de Integração de Dados:** Problemas de compatibilidade entre diferentes bases de dados podem ser mitigados com um processo robusto de ETL.
- **Riscos de Precisão nas Previsões:** Implementação de validações contínuas para garantir que os modelos de machine learning estejam sempre alinhados com as condições de mercado.
- **Riscos de Adoção:** Investir em treinamento e suporte contínuo para garantir a adoção bem-sucedida do sistema pelos usuários finais.

Este escopo servirá como base para o planejamento detalhado e a execução do projeto, visando a criação de um sistema robusto e eficiente de avaliação de preços no setor supermercadista brasileiro.

Tipos de Bancos de Dados: SGDB e NoSQL

1. Sistemas de Gerenciamento de Banco de Dados (SGDB)

Os SGDBs são usados para gerenciar bancos de dados relacionais (RDBMS), que armazenam dados em tabelas organizadas em linhas e colunas. Esses sistemas utilizam SQL (Structured Query Language) para a manipulação de dados.

Principais Tipos de SGDBs:

- **MySQL:** Um dos SGDBs mais populares, é conhecido por sua performance e robustez em ambientes web. Ele suporta ACID (Atomicidade, Consistência, Isolamento e Durabilidade) e é ideal para transações que exigem consistência e integridade.

- **PostgreSQL:** Um SGDB avançado, com suporte a transações complexas e consultas avançadas. É altamente extensível e oferece suporte para JSON, permitindo armazenamento de dados semiestruturados.
- **Oracle:** Utilizado em grandes corporações, oferece alta performance, escalabilidade e funcionalidades avançadas de segurança e replicação.
- **SQL Server:** Desenvolvido pela Microsoft, é conhecido por sua integração com outras ferramentas da Microsoft, como o Power BI, e por sua robustez em ambientes empresariais.

Conexões Entre SGDBs:

- **ETL (Extract, Transform, Load):** Ferramentas como Talend, Apache Nifi e Informatica podem ser usadas para conectar, extrair, transformar e carregar dados de diferentes SGDBs, garantindo a integração entre eles.
- **ODBC/JDBC:** Protocolos que permitem a comunicação entre diferentes SGDBs e aplicações, possibilitando consultas e manipulação de dados de forma centralizada.

2. Bancos de Dados NoSQL

Bancos de dados NoSQL são projetados para lidar com grandes volumes de dados distribuídos e para oferecer flexibilidade no armazenamento de dados semiestruturados ou não estruturados. Eles não dependem de esquemas fixos e oferecem escalabilidade horizontal.

Principais Tipos de NoSQL:

- **MongoDB:** Um banco de dados orientado a documentos, que armazena dados em formato JSON, ideal para aplicações que precisam de flexibilidade no armazenamento de dados semiestruturados.
- **Cassandra:** Um banco de dados de larga escala, projetado para alta disponibilidade e escalabilidade, utilizado para grandes volumes de dados distribuídos.
- **Redis:** Um banco de dados em memória, que oferece alta performance para operações de leitura e escrita, frequentemente usado como cache.
- **HBase:** Um banco de dados distribuído e escalável, que funciona sobre o HDFS (Hadoop Distributed File System), adequado para grandes volumes de dados com necessidades de leitura e escrita rápidas.

Conexões Entre NoSQL:

- **Data Pipelines:** Ferramentas como Apache Kafka ou Apache NiFi permitem o fluxo contínuo de dados entre diferentes bancos de dados NoSQL, garantindo que os dados sejam transmitidos e transformados conforme necessário.
- **APIs e SDKs:** NoSQL geralmente fornece APIs e SDKs para integração com outras aplicações ou sistemas de bancos de dados, facilitando a comunicação entre diferentes sistemas.

Hadoop e Ferramentas de Integração de Big Data

O Hadoop é um framework de código aberto que permite o processamento distribuído de grandes volumes de dados. Ele é composto por vários módulos que podem ser usados para compatibilizar e traduzir informações provenientes de diferentes fontes de dados, sejam elas SGDB ou NoSQL.

Principais Componentes do Hadoop:

- **HDFS (Hadoop Distributed File System):** Sistema de arquivos distribuído que armazena grandes volumes de dados de forma distribuída e resiliente.
- **MapReduce:** Modelo de programação para processamento paralelo de grandes conjuntos de dados. Ele divide o processamento em duas etapas: Map (mapeamento) e Reduce (redução), permitindo a análise de dados em larga escala.
- **YARN (Yet Another Resource Negotiator):** Gerenciador de recursos que aloca e monitora recursos do cluster Hadoop para diferentes aplicações.
- **Hive:** Um sistema de data warehousing sobre Hadoop que permite consultas em grandes volumes de dados usando uma linguagem semelhante ao SQL chamada HiveQL.
- **Pig:** Uma plataforma de alto nível para criar programas que rodam sobre Hadoop. Usa uma linguagem chamada Pig Latin, que abstrai as complexidades do MapReduce.
- **HBase:** Banco de dados NoSQL distribuído que roda sobre o HDFS, permitindo leitura e escrita rápidas de dados estruturados e semiestruturados.
- **Sqoop:** Ferramenta usada para importar e exportar dados entre bancos de dados relacionais e o Hadoop.
- **Flume:** Serviço para coletar, agregar e mover grandes quantidades de dados de log para o HDFS.
- **Oozie:** Gerenciador de fluxo de trabalho que coordena os trabalhos de MapReduce, Pig e Hive.

Integração de SGDBs e NoSQL com Hadoop no Projeto de Big Data

Para o contexto do projeto de avaliação de preços no setor supermercadista:

1. Coleta e Integração de Dados:

- **Apache Sqoop** pode ser usado para importar dados dos SGDBs (MySQL, PostgreSQL, etc.) para o HDFS.
- **Flume** pode ser utilizado para coletar dados de logs de compras online e armazená-los no sistema de arquivos HDFS.
- **Kafka** pode ser empregado para transmitir dados em tempo real de diferentes fontes, como registros de vendas e dados de representantes de produtos.

2. Processamento e Análise:

- **MapReduce** pode ser usado para processar grandes volumes de dados e calcular métricas de avaliação de preços.
- **Hive** ou **Pig** podem ser utilizados para consultas e análises complexas sobre os dados coletados e armazenados no HDFS.

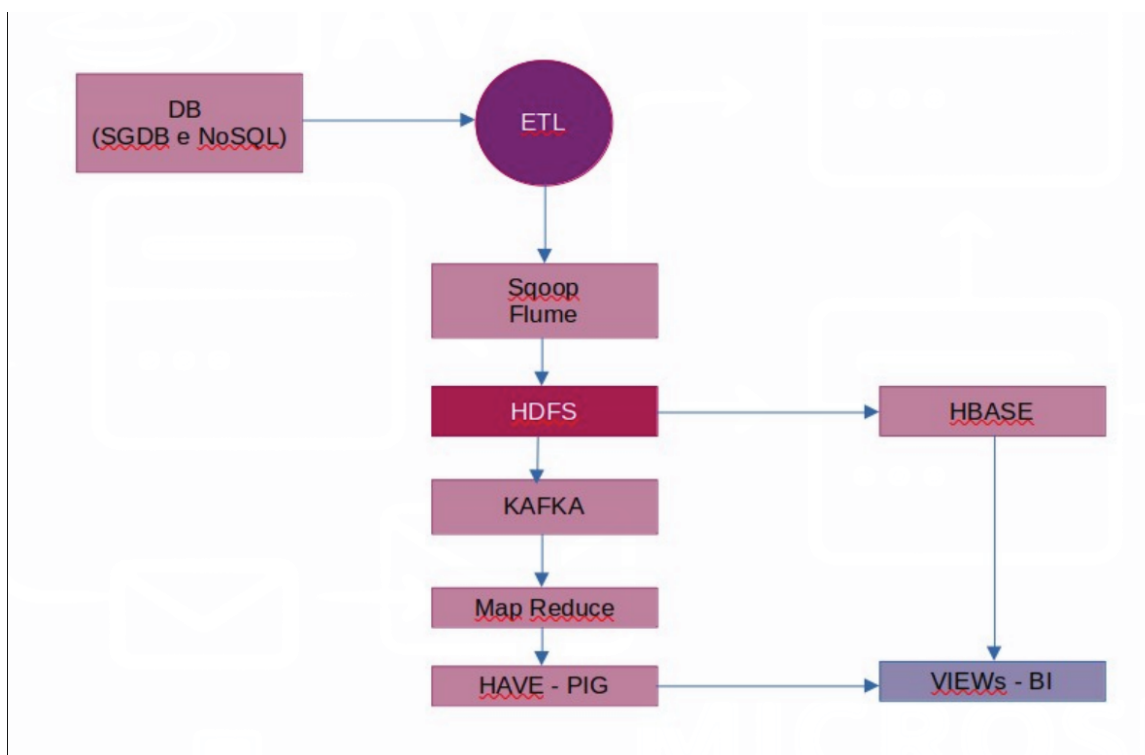
3. Armazenamento e Consulta de Dados:

- **HBase** pode ser utilizado para armazenamento de dados que necessitam de alta velocidade de leitura e escrita, como registros de preços em tempo real.
- **Hive** permite consultas SQL-like sobre dados armazenados no HDFS, facilitando a extração de insights.

4. Visualização e Relatórios:

- **Tableau** ou **Power BI** podem ser conectados ao Hadoop via ODBC/JDBC para a criação de dashboards interativos e geração de relatórios em tempo real.

FIGURA 46 – MAPA DE INTEGRAÇÃO DO PROJETO DE BIG DATA



FONTE: Elaborado pelo autor (2024).

Conclusão

A integração dos diversos tipos de SGDBs e NoSQL com a infraestrutura Hadoop permitirá um fluxo contínuo de dados e análises avançadas, garantindo que o sistema de avaliação de preços no contexto supermercadista seja eficiente, escalável e robusto. Essa combinação de tecnologias assegura que os dados de múltiplas fontes sejam compatibilizados, traduzidos e analisados de forma a oferecer insights valiosos para a tomada de decisões estratégicas.

APÊNDICE 11 – 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:

1. Carregue a base de dados de Treino.
2. Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes).
3. Extraia características utilizando LBP e a CNN VGG16 (gerando um `csv` para cada extrator).
4. Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos.
5. Carregue a base de Teste e execute a tarefa 3 nesta base.
6. Aplique os modelos treinados nos dados de teste.
7. Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão.
8. Indique qual modelo dá o melhor 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. Treine só as novas camadas. Compare os treinos de 10 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:

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

B - RESOLUÇÃO

1. Extração de Características

Importação de Bibliotecas

O trecho de código a seguir realiza a importação das bibliotecas necessárias para o desenvolvimento da tarefa. Essas bibliotecas oferecem suporte para manipulação de imagens, tratamento de dados, construção de modelos de aprendizado de máquina e visualização de resultados. As importações estão organizadas conforme sua finalidade, abrangendo desde operações básicas com arquivos até o uso de modelos pré-treinados de redes neurais convolucionais.

```
1 # Fornece funções para manipulação de diretórios e arquivos
2 import os
3
4 # Biblioteca para operações numéricas e vetoriais
5 import numpy as np
6
7 # Biblioteca OpenCV para processamento de imagens
8 import cv2
9
10 # Permite leitura e escrita de arquivos CSV
11 import csv
12
13 # Função para extração de textura via LBP
14 from skimage.feature import local_binary_pattern
15
16 # Importa o modelo VGG16 pré-treinado
17 from tensorflow.keras.applications import VGG16
```



```
18
19 # Permite definir e manipular modelos no Keras
20 from tensorflow.keras.models import Model
21
22 # Funções para carregar e converter imagens
23 from tensorflow.keras.preprocessing.image import load_img, img_to_array
24
25 # Pré-processamento específico do VGG16
26 from tensorflow.keras.applications.vgg16 import preprocess_input
27
28 # Biblioteca para manipulação de dados em formato tabular (DataFrames)
29 import pandas as pd
30
31 # Biblioteca para geração de gráficos
32 import matplotlib.pyplot as plt
33
34 # Para leitura e exibição de imagens com Matplotlib
35 import matplotlib.image as mpimg
36
37 # Biblioteca para visualização de dados estatísticos
38 import seaborn as sns
39
40 # Métricas de avaliação de desempenho de modelos
41 from sklearn.metrics import confusion_matrix, classification_report, accuracy_score,
    recall_score, f1_score
42
43 # Classificador baseado em rede neural (perceptron multicamada)
44 from sklearn.neural_network import MLPClassifier
45
46 # Classificador de vetores de suporte (SVM)
47 from sklearn.svm import SVC
48
49 # Cria pipelines de processamento e classificação
50 from sklearn.pipeline import make_pipeline
51
52 # Classificador baseado em floresta aleatória
53 from sklearn.ensemble import RandomForestClassifier
54
55 # Função para dividir dados em treino e teste
```

```

56 from sklearn.model_selection import train_test_split
57
58 # Normaliza os dados com média 0 e desvio padrão 1
59 from sklearn.preprocessing import StandardScaler
60
61 # Biblioteca para salvar e carregar modelos treinados
62 import joblib
63
64 # Biblioteca para serializar e desserializar objetos Python
65 import pickle
66
67 #montar o drive para acessar os arquivos
68 from google.colab import drive
69 drive.mount('/content/drive')

```

1. Carregamento da base de dados de Treino.

Esta seção do código tem como objetivo inicializar o processo de pré-processamento das imagens e extração de características. Para isso, são definidos os caminhos de acesso às bases de dados de treino e teste, que serão utilizados nas etapas subsequentes de análise e treinamento dos modelos. A organização adequada dos diretórios é essencial para garantir a correta leitura e manipulação dos dados ao longo do pipeline computacional.

```

1 # Carregar as bases de dados
2 train_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Train'
3 test_data_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'

```

2. Extração das características utilizando LBP e a CNN VGG16..

Esta etapa do código realiza a extração de características das imagens por meio de duas abordagens distintas: o método de textura *Local Binary Pattern* (LBP) e a rede neural convolucional pré-treinada VGG16. As características extraídas são organizadas e armazenadas em arquivos .csv, que servirão como entrada para os modelos de classificação nas etapas posteriores.

```

1 ### Extração de Características (LBP)
2 radius = 1
3 n_points = 8 * radius
4
5 def extract_lbp_features(image):
6     gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
7     lbp = local_binary_pattern(gray_image, n_points, radius, method='uniform')

```

```

8     (hist, _) = np.histogram(lbp.ravel(), bins=np.arange(0, n_points + 3), range
    =(0, n_points + 2))
9     hist = hist.astype("float")
10    hist /= (hist.sum() + 1e-6)
11    return hist
12
13    def process_images_for_lbp(data_dir, output_csv):
14        with open(output_csv, mode='w', newline='') as file:
15            writer = csv.writer(file)
16            writer.writerow(['Image', 'Classe', 'Cliente', 'Features'])
17            for class_dir in os.listdir(data_dir):
18                class_path = os.path.join(data_dir, class_dir)
19                if os.path.isdir(class_path):
20                    for image_name in os.listdir(class_path):
21                        image_path = os.path.join(class_path, image_name)
22                        image = cv2.imread(image_path)
23                        if image is not None:
24                            features = extract_lbp_features(image)
25                            cliente = image_name[:2]
26                            writer.writerow([image_name, class_dir, cliente, features
    .tolist()])
27            print(f"Extração de LBP para {os.path.basename(data_dir)} concluída. Arquivo
    '{output_csv}' criado.")
28
29    process_images_for_lbp(train_data_dir, 'lbp_features_train.csv')
30    process_images_for_lbp(test_data_dir, 'lbp_features_test.csv')
31
32    # ---
33    ### Extração de Características (VGG16)
34    print("\nIniciando a extração de características VGG16 das bases de treino e
    teste...")
35
36    base_model = VGG16(weights='imagenet', include_top=True)
37    model = Model(inputs=base_model.input, outputs=base_model.get_layer('fc2').output
    )
38
39    img_width, img_height = 224, 224
40
41    def extract_vgg16_features(image_path):

```

```

42     image = load_img(image_path, target_size=(img_width, img_height))
43     image = img_to_array(image)
44     image = np.expand_dims(image, axis=0)
45     image = preprocess_input(image)
46     features = model.predict(image, verbose=0)
47     return features.flatten()
48
49 def process_images_for_vgg16(data_dir, output_csv):
50     with open(output_csv, mode='w', newline='') as file:
51         writer = csv.writer(file)
52         writer.writerow(['Image', 'Classe', 'Cliente', 'Features'])
53         total_files = sum(len(files) for _, _, files in os.walk(data_dir))
54         current_file = 0
55         for class_dir in os.listdir(data_dir):
56             class_path = os.path.join(data_dir, class_dir)
57             if os.path.isdir(class_path):
58                 for image_name in os.listdir(class_path):
59                     current_file += 1
60                     image_path = os.path.join(class_path, image_name)
61                     cliente = image_name[:2]
62                     features = extract_vgg16_features(image_path)
63                     writer.writerow([image_name, class_dir, cliente, features.
64                                     tolist()])
65                     print(f'Processando arquivo {current_file}/{total_files} em {
66                           os.path.basename(data_dir)}', end='\r')
67                     print(f"\nExtração de VGG16 para {os.path.basename(data_dir)} concluída.
68                           Arquivo '{output_csv}' criado.")
69
70 process_images_for_vgg16(train_data_dir, 'vgg16_features_train.csv')
71 process_images_for_vgg16(test_data_dir, 'vgg16_features_test.csv')

```

3. Particionamento dos Dados.

Neste trecho do código, realiza-se o particionamento dos dados extraídos previamente por meio dos métodos LBP e VGG16. Os dados são carregados a partir de arquivos .csv, convertidos para o formato de matriz numérica e, em seguida, divididos em conjuntos de treino (80%) e validação (20%). A divisão é estratificada para garantir a proporcionalidade entre as classes nas partições, o que contribui para a robustez da avaliação dos modelos.

```

1  ## Particionamento dos Dados de Treino
2  # Funções auxiliares

```

```

3 def convert_to_array(feature_str):
4     return np.array(eval(feature_str))
5
6 # ---
7 ### Partição para LBP
8 print("\nRealizando a partição dos dados LBP...")
9 df_lbp = pd.read_csv('lbp_features_train.csv')
10 df_lbp['Features'] = df_lbp['Features'].apply(convert_to_array)
11 X_lbp = np.vstack(df_lbp['Features'].values)
12 y_lbp = df_lbp['Classe'].values
13 X_train_lbp, X_val_lbp, y_train_lbp, y_val_lbp = train_test_split(X_lbp, y_lbp,
14     test_size=0.2, random_state=42, stratify=y_lbp)
15 print("Partição de LBP concluída.")
16
17 # ---
18 ### Partição para VGG16
19 print("\nRealizando a partição dos dados VGG16...")
20 df_vgg16 = pd.read_csv('vgg16_features_train.csv')
21 df_vgg16['Features'] = df_vgg16['Features'].apply(convert_to_array)
22 X_vgg16 = np.vstack(df_vgg16['Features'].values)
23 y_vgg16 = df_vgg16['Classe'].values
24 X_train_vgg16, X_val_vgg16, y_train_vgg16, y_val_vgg16 = train_test_split(X_vgg16,
25     y_vgg16, test_size=0.2, random_state=42, stratify=y_vgg16)
26 print("Partição de VGG16 concluída.")

```

TABELA 49 – DISTRIBUIÇÃO DAS CLASSES NO CONJUNTO DE TREINO E TESTE

Classe	Treino	Teste	Total
0	116	29	145
1	117	30	147
2	120	30	150
3	120	29	149
Total	473	118	591

FONTE: Elaborado pelo autor (2024).

4. Treinamento dos modelos.

Nesta etapa, realizam-se o treinamento e a validação de três algoritmos de classificação: Random Forest, Máquina de Vetores de Suporte (SVM) e Rede Neural Artificial (RNA), aplicados aos conjuntos de dados extraídos por meio das técnicas LBP e VGG16. O desempenho dos modelos é avaliado utilizando a acurácia, e os modelos treinados são armazenados em arquivos para uso

posterior. Também é definida uma função auxiliar para o cálculo da especificidade a partir das matrizes de confusão geradas.

```

1  ## Treinamento e Validação de Modelos
2  # Funções auxiliares para métricas
3  def calculate_specificity(cm):
4      specifics = []
5      for i in range(cm.shape[0]):
6          TN = np.sum(cm) - (np.sum(cm[i, :]) + np.sum(cm[:, i]) - cm[i, i])
7          FP = np.sum(cm[:, i]) - cm[i, i]
8          specificity = TN / (TN + FP) if (TN + FP) != 0 else 0
9          specifics.append(specificity)
10     return specifics
11
12 # Dicionários para armazenar as acurácias
13 acuracies_lbp = {}
14 acuracies_vgg16 = {}
15
16 # ---
17 ### Treinamento e avaliação com LBP (Validação)
18 print("\nIniciando treinamento e avaliação com dados LBP...")
19
20 # Random Forest
21 print("Treinando Random Forest com LBP...")
22 rf_model_lbp = RandomForestClassifier(n_estimators=100, random_state=42)
23 rf_model_lbp.fit(X_train_lbp, y_train_lbp)
24 y_pred_lbp_rf = rf_model_lbp.predict(X_val_lbp)
25 acuracies_lbp["RF"] = accuracy_score(y_val_lbp, y_pred_lbp_rf) * 100
26 joblib.dump(rf_model_lbp, 'rf_model_lbp.pkl')
27 print(f"Acurácia RF com LBP: {acuracies_lbp['RF']:.2f}%")
28
29 # SVM
30 print("Treinando SVM com LBP...")
31 svm_model_lbp = make_pipeline(StandardScaler(), SVC(kernel='linear', random_state
    =42))
32 svm_model_lbp.fit(X_train_lbp, y_train_lbp)
33 y_pred_lbp_svm = svm_model_lbp.predict(X_val_lbp)
34 acuracies_lbp["SVM"] = accuracy_score(y_val_lbp, y_pred_lbp_svm) * 100
35 joblib.dump(svm_model_lbp, 'svm_model_lbp.pkl')
36 print(f"Acurácia SVM com LBP: {acuracies_lbp['SVM']:.2f}%")

```

```

37
38 # RNA
39 print("Treinando RNA com LBP...")
40 nn_model_lbp = MLPClassifier(hidden_layer_sizes=(100,), alpha=0.001, max_iter
    =2000, random_state=42)
41 nn_model_lbp.fit(X_train_lbp, y_train_lbp)
42 y_pred_lbp_rna = nn_model_lbp.predict(X_val_lbp)
43 acuracies_lbp["RNA"] = accuracy_score(y_val_lbp, y_pred_lbp_rna) * 100
44 joblib.dump(nn_model_lbp, 'rna_model_lbp.pkl')
45 print(f"Acurácia RNA com LBP: {acuracies_lbp['RNA']:.2f}%")
46
47 # ---
48 ### Treinamento e avaliação com VGG16 (Validação)
49 print("\nIniciando treinamento e avaliação com dados VGG16...")
50
51 # Random Forest
52 print("Treinando Random Forest com VGG16...")
53 rf_model_vgg16 = RandomForestClassifier(n_estimators=100, random_state=42)
54 rf_model_vgg16.fit(X_train_vgg16, y_train_vgg16)
55 y_pred_vgg16_rf = rf_model_vgg16.predict(X_val_vgg16)
56 acuracies_vgg16["RF"] = accuracy_score(y_val_vgg16, y_pred_vgg16_rf) * 100
57 joblib.dump(rf_model_vgg16, 'rf_model_vgg16.pkl')
58 print(f"Acurácia RF com VGG16: {acuracies_vgg16['RF']:.2f}%")
59
60 # SVM
61 print("Treinando SVM com VGG16...")
62 svm_model_vgg16 = make_pipeline(StandardScaler(), SVC(kernel='linear', random_
    state=42))
63 svm_model_vgg16.fit(X_train_vgg16, y_train_vgg16)
64 y_pred_vgg16_svm = svm_model_vgg16.predict(X_val_vgg16)
65 acuracies_vgg16["SVM"] = accuracy_score(y_val_vgg16, y_pred_vgg16_svm) * 100
66 joblib.dump(svm_model_vgg16, 'svm_model_vgg16.pkl')
67 print(f"Acurácia SVM com VGG16: {acuracies_vgg16['SVM']:.2f}%")
68
69 # RNA
70 print("Treinando RNA com VGG16...")
71 nn_model_vgg16 = MLPClassifier(hidden_layer_sizes=(100,), max_iter=1000, random_
    state=42)
72 nn_model_vgg16.fit(X_train_vgg16, y_train_vgg16)

```

```

73 y_pred_vgg16_rna = nn_model_vgg16.predict(X_val_vgg16)
74 acuracies_vgg16["RNA"] = accuracy_score(y_val_vgg16, y_pred_vgg16_rna) * 100
75 joblib.dump(nn_model_vgg16, 'rna_model_vgg16.pkl')
76 print(f"Acurácia RNA com VGG16: {acuracies_vgg16['RNA']:.2f}%")

```

5. Aplicação e análise na base de teste.

Esta etapa tem como finalidade aplicar os modelos treinados às amostras da base de teste, a fim de realizar a avaliação final de desempenho. São utilizados os conjuntos de características extraídas por LBP e VGG16, e para cada abordagem são aplicados os modelos Random Forest, SVM e Rede Neural Artificial (RNA).

As previsões geradas são comparadas com os rótulos reais por meio de métricas como acurácia e relatório de classificação. Além disso, são construídas as matrizes de confusão para cada modelo, com visualização gráfica via mapas de calor, facilitando a interpretação dos acertos e erros por classe. Por fim, os resultados são salvos em arquivos .csv contendo as previsões associadas a cada imagem, permitindo análises posteriores.

```

1  ## Aplicação e Análise na Base de Teste
2  # Funções auxiliares para plotar matriz de confusão
3  def plot_confusion_matrix(cm, title, cmap, class_names):
4      plt.figure(figsize=(10, 7))
5      sns.heatmap(cm, annot=True, fmt='d', cmap=cmap, cbar=False, xticklabels=class
6                  _names, yticklabels=class_names)
7      plt.xlabel('Preditas')
8      plt.ylabel('Reais')
9      plt.title(title)
10     plt.show()
11
12     class_names = ['Classe 0', 'Classe 1', 'Classe 2', 'Classe 3']
13     confusion_matrices = []
14
15     # ---
16     ### Avaliação dos modelos LBP na base de teste
17     print("\nAvaliando modelos com base de teste (LBP)...")
18     test_df_lbp = pd.read_csv('lbp_features_test.csv')
19     X_test_final_lbp = np.array(test_df_lbp['Features']).apply(eval).tolist()
20     y_true_lbp = test_df_lbp['Classe']
21
22     # Random Forest
23     print("Avaliando Random Forest com LBP...")

```



```

23 rf_model_lbp = joblib.load('rf_model_lbp.pkl')
24 y_pred_lbp_rf = rf_model_lbp.predict(X_test_final_lbp)
25 cm_lbp_rf = confusion_matrix(y_true_lbp, y_pred_lbp_rf)
26 confusion_matrices.append(cm_lbp_rf)
27 print(f"Acurácia RF-LBP: {accuracy_score(y_true_lbp, y_pred_lbp_rf) * 100:.2f}%\n")
28 print(classification_report(y_true_lbp, y_pred_lbp_rf, target_names=class_names))
29
30 # SVM
31 print("Avaliando SVM com LBP...")
32 svm_model_lbp = joblib.load('svm_model_lbp.pkl')
33 y_pred_lbp_svm = svm_model_lbp.predict(X_test_final_lbp)
34 cm_lbp_svm = confusion_matrix(y_true_lbp, y_pred_lbp_svm)
35 confusion_matrices.append(cm_lbp_svm)
36 print(f"Acurácia SVM-LBP: {accuracy_score(y_true_lbp, y_pred_lbp_svm) * 100:.2f}%\n")
37 print(classification_report(y_true_lbp, y_pred_lbp_svm, target_names=class_names))
38
39 # RNA
40 print("Avaliando RNA com LBP...")
41 nn_model_lbp = joblib.load('rna_model_lbp.pkl')
42 y_pred_lbp_rna = nn_model_lbp.predict(X_test_final_lbp)
43 cm_lbp_rna = confusion_matrix(y_true_lbp, y_pred_lbp_rna)
44 confusion_matrices.append(cm_lbp_rna)
45 print(f"Acurácia RNA-LBP: {accuracy_score(y_true_lbp, y_pred_lbp_rna) * 100:.2f}%\n")
46 print(classification_report(y_true_lbp, y_pred_lbp_rna, target_names=class_names))
47
48 ### Avaliação dos modelos VGG16 na base de teste
49 print("\nAvaliando modelos com base de teste (VGG16)...")
50 test_df_vgg16 = pd.read_csv('vgg16_features_test.csv')
51 X_test_final_vgg16 = np.array(test_df_vgg16['Features']).apply(eval).tolist()
52 y_true_vgg16 = test_df_vgg16['Classe']
53
54 # Random Forest
55 print("Avaliando Random Forest com VGG16...")
56 rf_model_vgg16 = joblib.load('rf_model_vgg16.pkl')

```

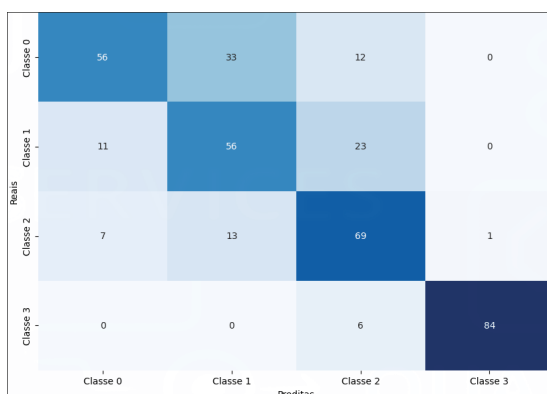
```

57 y_pred_vgg16_rf = rf_model_vgg16.predict(X_test_final_vgg16)
58 cm_vgg16_rf = confusion_matrix(y_true_vgg16, y_pred_vgg16_rf)
59 confusion_matrices.append(cm_vgg16_rf)
60 test_df_vgg16['RF_Prediction'] = y_pred_vgg16_rf
61 print(f"Acurácia RF-VGG16: {accuracy_score(y_true_vgg16, y_pred_vgg16_rf) *
      100:.2f}%\n")
62 print(classification_report(y_true_vgg16, y_pred_vgg16_rf, target_names=class_
      names))
63
64 # SVM
65 print("Avaliando SVM com VGG16...")
66 svm_model_vgg16 = joblib.load('svm_model_vgg16.pkl')
67 y_pred_vgg16_svm = svm_model_vgg16.predict(X_test_final_vgg16)
68 cm_vgg16_svm = confusion_matrix(y_true_vgg16, y_pred_vgg16_svm)
69 confusion_matrices.append(cm_vgg16_svm)
70 test_df_vgg16['SVM_Prediction'] = y_pred_vgg16_svm
71 print(f"Acurácia SVM-VGG16: {accuracy_score(y_true_vgg16, y_pred_vgg16_svm) *
      100:.2f}%\n")
72 print(classification_report(y_true_vgg16, y_pred_vgg16_svm, target_names=class_
      names))
73
74 # RNA
75 print("Avaliando RNA com VGG16...")
76 nn_model_vgg16 = joblib.load('rna_model_vgg16.pkl')
77 y_pred_vgg16_rna = nn_model_vgg16.predict(X_test_final_vgg16)
78 cm_vgg16_rna = confusion_matrix(y_true_vgg16, y_pred_vgg16_rna)
79 confusion_matrices.append(cm_vgg16_rna)
80 test_df_vgg16['RNA_Prediction'] = y_pred_vgg16_rna
81 print(f"Acurácia RNA-VGG16: {accuracy_score(y_true_vgg16, y_pred_vgg16_rna) *
      100:.2f}%\n")
82 print(classification_report(y_true_vgg16, y_pred_vgg16_rna, target_names=class_
      names))
83
84 test_df_vgg16.to_csv('vgg16_features_test_with_predictions.csv', index=False)
85 test_df_lbp.to_csv('lbp_features_test_with_predictions.csv', index=False)

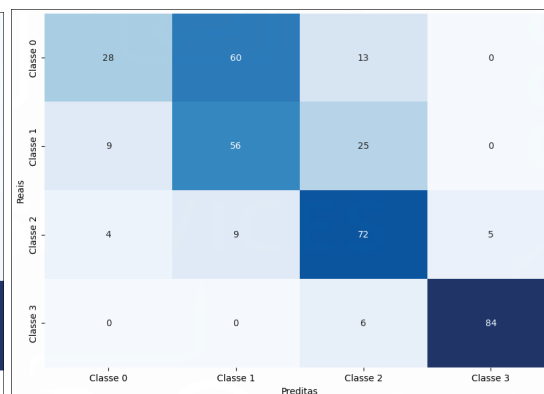
```

FIGURA 47 – MAPAS DE CALOR PARA MATRIZES DE CONFUSÃO LBP

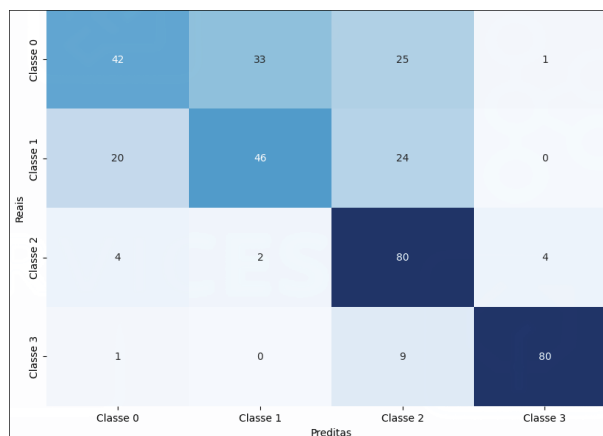
(a) LBP com SVM



(b) LBP com RNA



(c) LBP com Random Forest



FONTE: Elaborado pelo autor (2024).

TABELA 50 – MATRIZ DE CONFUSÃO LBP COM RANDOM FOREST

Classe	Precision	Recall	F1-Score	Support
0	0.75	0.72	0.74	29
1	0.72	0.77	0.74	30
2	0.81	0.73	0.77	30
3	0.90	0.97	0.93	29
Accuracy	0.80 (118 amostras)			
Macro Avg	0.80	0.80	0.80	118
Weighted Avg	0.80	0.80	0.80	118

FONTE: Elaborado pelo autor (2024).

TABELA 51 – MATRIZ DE CONFUSÃO LBP COM SVM

Classe	Precision	Recall	F1-Score	Support
0	0.74	0.69	0.71	29
1	0.67	0.60	0.63	30
2	0.66	0.70	0.68	30
3	0.91	1.00	0.95	29
Accuracy	0.75 (118 amostras)			
Macro Avg	0.74	0.75	0.74	118
Weighted Avg	0.74	0.75	0.74	118

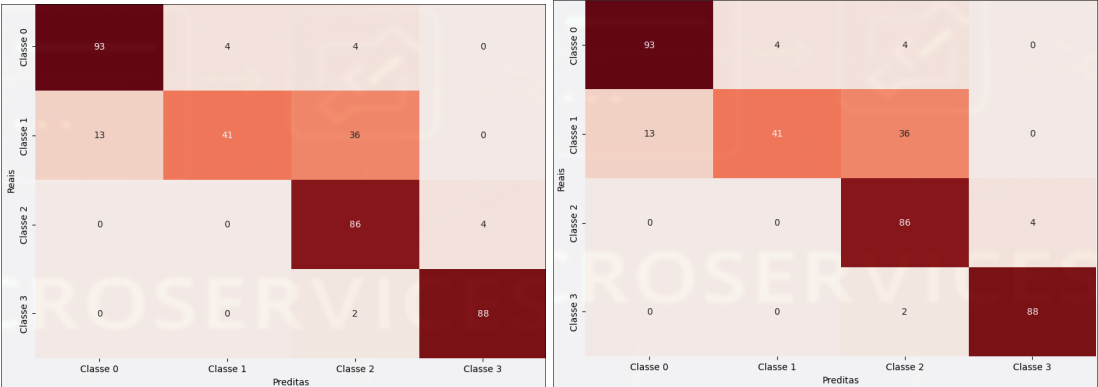
FONTE: Elaborado pelo autor (2024).

TABELA 52 – MATRIZ DE CONFUSÃO LBP COM RNA

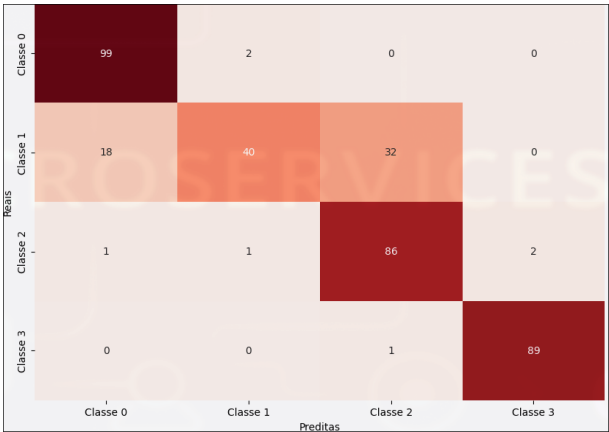
Classe	Precision	Recall	F1-score	Support
0	0.64	0.48	0.55	29
1	0.60	0.60	0.60	30
2	0.61	0.67	0.63	30
3	0.85	0.97	0.90	29
Accuracy	0.68 (118 amostras)			
Macro Avg	0.67	0.68	0.67	118
Weighted Avg	0.67	0.68	0.67	118

FONTE: Elaborado pelo autor (2024).

FIGURA 48 – MAPAS DE CALOR PARA MATRIZES DE CONFUSÃO VGG16
(a) VGG16 com SVM (b) VGG16 com RNA



(c) VGG16 com Random Forest



FONTE: Elaborado pelo autor (2024).

TABELA 53 – MATRIZ DE CONFUSÃO VGG16 COM RANDOM FOREST

Classe	Precision	Recall	F1-score	Support
0	0.90	0.97	0.93	29
1	0.93	0.90	0.92	30
2	0.90	0.93	0.92	30
3	1.00	0.93	0.96	29
Accuracy	0.93 (118 amostras)			
Macro Avg	0.93	0.93	0.93	118
Weighted Avg	0.93	0.93	0.93	118

FONTE: Elaborado pelo autor (2024).

TABELA 54 – MATRIZ DE CONFUSÃO VGG16 COM SVM

Classe	Precision	Recall	F1-score	Support
0	0.96	0.93	0.95	29
1	0.96	0.90	0.93	30
2	0.83	1.00	0.91	30
3	1.00	0.90	0.95	29
Accuracy	0.93 (118 amostras)			
Macro Avg	0.94	0.93	0.93	118
Weighted Avg	0.94	0.93	0.93	118

FONTE: Elaborado pelo autor (2024).

TABELA 55 – MATRIZ DE CONFUSÃO VGG16 COM RNA

Classe	Precision	Recall	F1-score	Support
0	0.97	0.97	0.97	29
1	0.97	0.93	0.95	30
2	0.94	1.00	0.97	30
3	1.00	0.97	0.98	29
Accuracy	0.97 (118 amostras)			
Macro Avg	0.97	0.97	0.97	118
Weighted Avg	0.97	0.97	0.97	118

FONTE: Elaborado pelo autor (2024).

6. Cálculo das métricas de Sensibilidade, Especificidade e F1-Score.

TABELA 56 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (LBP COM RANDOM FOREST)

Classe	Especificidade	Sensibilidade	F1-score
0	0.92	0.72	0.74
1	0.90	0.77	0.74
2	0.94	0.73	0.77
3	0.97	0.97	0.93

FONTE: Elaborado pelo autor (2024).

TABELA 57 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (LBP COM SVM)

Classe	Especificidade	Sensibilidade	F1-score
0	0.92	0.69	0.71
1	0.90	0.60	0.63
2	0.88	0.70	0.68
3	0.97	1.00	0.95

FONTE: Elaborado pelo autor (2024).

TABELA 58 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (LBP COM RNA)

Classe	Especificidade	Sensibilidade	F1-score
0	0.91	0.48	0.55
1	0.86	0.60	0.60
2	0.85	0.67	0.63
3	0.94	0.97	0.90

FONTE: Elaborado pelo autor (2024).

TABELA 59 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (VGG16 COM RANDOM FOREST)

Classe	Especificidade	Sensibilidade	F1-score
0	0.97	0.97	0.93
1	0.98	0.90	0.92
2	0.97	0.93	0.92
3	1.00	0.93	0.96

FONTE: Elaborado pelo autor (2024).

TABELA 60 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (VGG16 COM SVM)

Classe	Especificidade	Sensibilidade	F1-score
0	0.99	0.93	0.95
1	0.99	0.90	0.93
2	0.93	1.00	0.91
3	1.00	0.90	0.95

FONTE: Elaborado pelo autor (2024).

TABELA 61 – ESPECIFICIDADE, SENSIBILIDADE E F1-SCORE (VGG16 COM RNA)

Classe	Especificidade	Sensibilidade	F1-score
0	0.99	0.97	0.97
1	0.99	0.93	0.95
2	0.98	1.00	0.97
3	1.00	0.97	0.98

FONTE: Elaborado pelo autor (2024).

7. Galeria de imagens com previsões

Nesta etapa, a geração da galeria de imagens com previsões tem como objetivo apresentar e avaliar o desempenho do modelo VGG16 com Rede Neural Artificial (RNA), que foi o modelo com as melhores métricas de desempenho. O código carregado realiza a visualização das previsões do modelo para um conjunto de imagens, comparando-as com as classes reais. Para cada classe, um número fixo de imagens é exibido, com as previsões corretas destacadas por bordas azuis e as incorretas por bordas vermelhas. Essa abordagem permite uma análise visual

direta da acurácia do modelo, facilitando a identificação de padrões de erro e a avaliação do desempenho da RNA.

```

1 # Definição do diretório base onde as imagens estão armazenadas
2 base_dir = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'
3
4 # Carregamento do arquivo CSV contendo as previsões do modelo e as classes reais
5 df_vgg16 = pd.read_csv('vgg16_features_test_with_predictions.csv')
6
7 # Definição do número de imagens a serem exibidas por classe
8 num_images_per_class = 5
9
10 # Função para encontrar o caminho completo da imagem a partir do nome do arquivo
11 def find_image_path(base_dir, img_filename):
12     # Percorre o diretório base para localizar o arquivo de imagem
13     for root, dirs, files in os.walk(base_dir):
14         if img_filename in files:
15             return os.path.join(root, img_filename)
16     # Caso o arquivo não seja encontrado, lança um erro
17     raise FileNotFoundError(f"Arquivo {img_filename} não encontrado.")
18
19 # Extração das classes únicas presentes no dataframe
20 classes = df_vgg16['Classe'].unique()
21
22 # Seleção das primeiras imagens por classe, conforme definido em num_images_per_
    class
23 selected_images = pd.concat([df_vgg16[df_vgg16['Classe'] == c].head(num_images_
    per_class) for c in classes])
24
25 # Inicialização da figura para exibição das imagens
26 plt.figure(figsize=(12, len(classes) * 2.5))
27
28 # Inicialização da posição do gráfico para inserção das imagens
29 plot_position = 1
30
31 # Loop para cada classe
32 for c in classes:
33     # Criação de um subtítulo para a classe atual
34     plt.subplot(len(classes), num_images_per_class + 1, plot_position)
35     plt.text(0.5, 0.5, f"Classe: {c}", fontsize=14, ha='center', va='center',

```



```

weight='bold')
36 # Desativa os eixos para melhorar a visualização
37 plt.axis('off')
38 plot_position += 1
39
40 # Filtra as imagens pertencentes à classe atual
41 class_images = selected_images[selected_images['Classe'] == c]
42
43 # Loop para cada imagem dentro da classe
44 for i in range(len(class_images)):
45     # Criação de um subtítulo para cada imagem
46     plt.subplot(len(classes), num_images_per_class + 1, plot_position)
47     img_filename = class_images.iloc[i]['Image']
48     try:
49         # Busca o caminho da imagem no diretório
50         img_path = find_image_path(base_dir, img_filename)
51         # Carrega a imagem
52         img = mpimg.imread(img_path)
53
54         # Determina a cor da borda com base na acurácia da previsão
55         if class_images.iloc[i]['RNA_Prediction'] == class_images.iloc[i]['
Classe']:
56             edge_color = 'blue' # Cor azul para previsões corretas
57         else:
58             edge_color = 'red' # Cor vermelha para previsões incorretas
59
60         # Exibe a imagem com a borda colorida
61         plt.imshow(img, aspect='auto')
62         plt.gca().add_patch(plt.Rectangle((0, 0), img.shape[1], img.shape[0],
63                                         linewidth=4, edgecolor=edge_color,
64                                         facecolor='none'))
65         # Adiciona o título com as informações da previsão e classe real
66         plt.title(f"Prev: {class_images.iloc[i]['RNA_Prediction']}\nReal: {
class_images.iloc[i]['Classe']}")
67     except FileNotFoundError:
68         # Caso a imagem não seja encontrada, exibe uma mensagem
69         plt.text(0.5, 0.5, 'Imagem não encontrada', ha='center', va='center')
70     # Desativa os eixos para cada imagem
71     plt.axis('off')

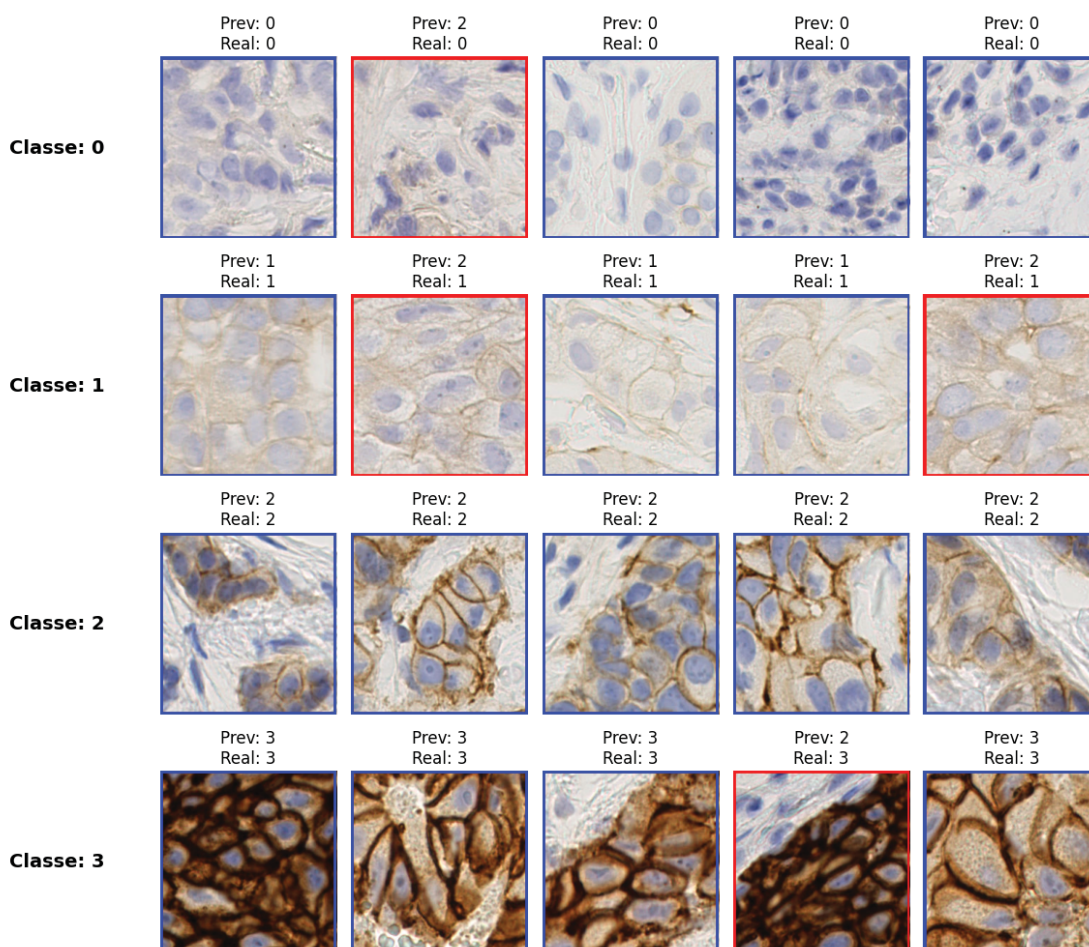
```

```

71     plot_position += 1
72
73 # Ajusta o layout da figura para evitar sobreposição
74 plt.tight_layout()
75
76 # Exibe a figura gerada com as imagens
77 plt.show()

```

FIGURA 49 – IMAGENS GERADAS COM PREVISÕES



FONTE: Elaborado pelo autor (2024).

8. Indique o modelo com melhor resultado e a métrica utilizada.

Com base nos resultados obtidos, o modelo que apresentou o melhor desempenho na tarefa de classificação de imagens foi a combinação da extração de características utilizando a rede pré-treinada **VGG16** com uma **Rede Neural Artificial (RNA)**. Este modelo alcançou os maiores valores em todas as métricas avaliadas, com destaque para o **F1-score médio de 0,97**, evidenciando um excelente equilíbrio entre precisão e sensibilidade nas predições realizadas.

A escolha do F1-score como métrica principal justificou-se por sua capacidade de refletir simultaneamente a qualidade da detecção das amostras positivas (sensibilidade) e a minimização de falsos positivos (precisão), sendo particularmente relevante em cenários de classificação multi-classe com distribuição de classes equilibrada.

Dessa forma, conclui-se que a abordagem baseada em **VGG16 com RNA** é a mais adequada para a resolução da tarefa proposta neste trabalho, apresentando resultados consistentes e robustos frente aos demais modelos testados.

2. Redes Neurais

Bibliotecas Utilizadas

O trecho de código apresentado abaixo realiza o carregamento das bibliotecas essenciais para o desenvolvimento e avaliação de modelos de redes neurais convolucionais, com ênfase no uso do framework TensorFlow/Keras. As bibliotecas incluem módulos para definição de modelos, pré-processamento de imagens, ajuste de hiperparâmetros, visualização de resultados e cálculo de métricas de desempenho.

```
1 # Carrega arquiteturas de redes neurais convolucionais pré-treinadas VGG16 e ResNet50
2 from tensorflow.keras.applications import VGG16, ResNet50
3
4 # Habilita a definição de modelos funcionais e o carregamento de modelos salvos
5 from tensorflow.keras.models import Model, load_model
6
7 # Disponibiliza camadas densas, achatamento de tensores e regularização via dropout
8 from tensorflow.keras.layers import Dense, Flatten, Dropout
9
10 # Define o otimizador Adam para ajuste dos pesos durante o treinamento
11 from tensorflow.keras.optimizers import Adam
12
13 # Fornece callbacks para controle do treinamento, como checkpoints e parada antecipada
14 from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
    ReduceLROnPlateau
15
16 # Permite a criação de geradores de dados com técnicas de aumento de imagens (data
    augmentation)
17 from tensorflow.keras.preprocessing.image import ImageDataGenerator
18
19 # Permite a visualização gráfica da matriz de confusão
20 from sklearn.metrics import ConfusionMatrixDisplay
21
```

```

22 # Disponibiliza métricas para avaliação de desempenho do modelo de classificação
23 from sklearn.metrics import classification_report, recall_score, precision_score, f1_
    score
24
25 # Permite o cálculo da matriz de confusão e da acurácia do modelo
26 from sklearn.metrics import confusion_matrix, accuracy_score
27
28 # Disponibiliza estrutura para contagem de elementos em coleções
29 from collections import Counter
30
31 # Define acesso às funcionalidades principais do framework TensorFlow
32 import tensorflow as tf
33
34 # Habilita a criação de visualizações estatísticas com gráficos avançados
35 import seaborn as sns
36
37 # Permite a geração de gráficos e visualizações em 2D
38 import matplotlib.pyplot as plt
39
40 # Disponibiliza suporte para operações numéricas com arrays e matrizes
41 import numpy as np
42
43 # Permite a manipulação de diretórios e caminhos no sistema de arquivos
44 import os
45
46 # Habilita operações de cópia e movimentação de arquivos e diretórios
47 import shutil
48
49 # Define funcionalidades para geração de números aleatórios
50 import random
51
52 # Carrega o drive para acesso aos arquivos
53 from google.colab import drive
54 drive.mount('/content/drive')

```

1. Separação e pré-processamento dos dados

O trecho a seguir realiza a preparação dos dados para o treinamento das redes neurais convolucionais. As etapas incluem: definição de diretórios, organização dos dados em conjuntos de

treino e validação com base na identificação dos pacientes, redimensionamento das imagens para 224x224 pixels e criação dos geradores de dados para treino, validação e teste.

```

1 # Define os parâmetros de entrada das imagens e o tamanho do batch
2 IMG_WIDTH, IMG_HEIGHT = 224, 224
3 BATCH_SIZE = 64
4
5 # Define os caminhos dos diretórios de dados
6 TRAIN_DIR = '/content/drive/MyDrive/VisaoComputacional/trabalho/Train'
7 TEST_DIR = '/content/drive/MyDrive/VisaoComputacional/trabalho/Test'
8 NEW_TRAIN_DIR = '/content/drive/MyDrive/VisaoComputacional/trabalho/novo_train'
9 NEW_VAL_DIR = '/content/drive/MyDrive/VisaoComputacional/trabalho/novo_val'
10
11 # Verifica a existência dos diretórios de treino e teste originais
12 if not os.path.exists(TRAIN_DIR):
13     print(f"Erro: Diretório não encontrado - {TRAIN_DIR}")
14 if not os.path.exists(TEST_DIR):
15     print(f"Erro: Diretório não encontrado - {TEST_DIR}")
16
17 # Remove e recria os diretórios de treino e validação
18 if os.path.exists(NEW_TRAIN_DIR):
19     shutil.rmtree(NEW_TRAIN_DIR)
20 if os.path.exists(NEW_VAL_DIR):
21     shutil.rmtree(NEW_VAL_DIR)
22 os.makedirs(NEW_TRAIN_DIR)
23 os.makedirs(NEW_VAL_DIR)
24
25 # Define função auxiliar para copiar arquivos de imagem para diretórios
    organizados por classe
26 def copy_images(file_list, source_dir, target_dir):
27     """Copia arquivos de imagem para o diretório de destino."""
28     for file_path in file_list:
29         class_name = os.path.basename(os.path.dirname(file_path))
30         target_class_dir = os.path.join(target_dir, class_name)
31         if not os.path.exists(target_class_dir):
32             os.makedirs(target_class_dir)
33         shutil.copy(file_path, target_class_dir)
34
35 # Agrupa imagens por classe e por paciente com base no prefixo do nome do arquivo
36 clientes_dict = {}

```

```

37 for subdir, _, files in os.walk(TRAIN_DIR):
38     for file in files:
39         if file.endswith(('png', 'jpg', 'jpeg')):
40             cliente = file[:2]
41             class_name = os.path.basename(subdir)
42             file_path = os.path.join(subdir, file)
43             if class_name not in clientes_dict:
44                 clientes_dict[class_name] = {}
45             if cliente not in clientes_dict[class_name]:
46                 clientes_dict[class_name][cliente] = []
47                 clientes_dict[class_name][cliente].append(file_path)
48
49 # Realiza a divisão dos dados por paciente: 80% para treino, 20% para validação
50 for class_name, clientes in clientes_dict.items():
51     clientes_list = list(clientes.keys())
52     random.shuffle(clientes_list)
53
54     train_images = []
55     val_images = []
56
57     for cliente in clientes_list:
58         imagens_cliente = clientes[cliente]
59         split_index = int(0.8 * len(imagens_cliente))
60         train_images.extend(imagens_cliente[:split_index])
61         val_images.extend(imagens_cliente[split_index:])
62
63     copy_images(train_images, TRAIN_DIR, NEW_TRAIN_DIR)
64     copy_images(val_images, TRAIN_DIR, NEW_VAL_DIR)
65
66 # Define geradores de dados com normalização para treino, validação e teste (sem
    aumento de dados)
67 train_datagen = ImageDataGenerator(rescale=1./255)
68 val_datagen = ImageDataGenerator(rescale=1./255)
69 test_datagen = ImageDataGenerator(rescale=1./255)
70
71 # Cria o gerador de dados para o conjunto de treino
72 train_generator = train_datagen.flow_from_directory(
73     NEW_TRAIN_DIR,
74     target_size=(IMG_WIDTH, IMG_HEIGHT),

```

```

75     batch_size=BATCH_SIZE,
76     class_mode='categorical'
77 )
78
79 # Cria o gerador de dados para o conjunto de validação
80 validation_generator = val_datagen.flow_from_directory(
81     NEW_VAL_DIR,
82     target_size=(IMG_WIDTH, IMG_HEIGHT),
83     batch_size=BATCH_SIZE,
84     class_mode='categorical'
85 )
86
87 # Cria o gerador de dados para o conjunto de teste (sem embaralhamento)
88 test_generator = test_datagen.flow_from_directory(
89     TEST_DIR,
90     target_size=(IMG_WIDTH, IMG_HEIGHT),
91     batch_size=BATCH_SIZE,
92     class_mode='categorical',
93     shuffle=False
94 )
95
96 # Define função para treinar o modelo com callbacks para checkpoint, early
    stopping e ajuste de taxa de aprendizado
97 def train_model(model, train_gen, val_gen, model_name, epochs=10):
98     """
99     Treina o modelo com os dados de treino e validação.
100     Salva o modelo com o melhor desempenho.
101     """
102     callbacks = [
103         ModelCheckpoint(f'{model_name}_best_model.keras', monitor='val_loss',
104             save_best_only=True),
105         EarlyStopping(monitor='val_loss', patience=5),
106         ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=2, verbose=1)
107     ]
108     history = model.fit(
109         train_gen,
110         epochs=epochs,
111         validation_data=val_gen,
112         callbacks=callbacks

```

```

112     )
113     return history
114
115 # Define função para construção do modelo baseado em VGG16 ou ResNet50 com
    camadas densas personalizadas
116 def build_model(base_model_name, input_shape=(224, 224, 3), num_classes=4):
117     """
118     Constrói um modelo de transfer learning.
119     Adapta a camada de topo para classificação de 4 classes.
120     """
121     if base_model_name == 'VGG16':
122         base_model = VGG16(weights='imagenet', include_top=False, input_shape=
input_shape)
123     elif base_model_name == 'ResNet50':
124         base_model = ResNet50(weights='imagenet', include_top=False, input_shape=
input_shape)
125     else:
126         raise ValueError("Modelo base não suportado.")
127
128     # Congela as camadas da base do modelo
129     for layer in base_model.layers:
130         layer.trainable = False
131
132     # Adiciona camadas densas ao topo do modelo
133     x = Flatten()(base_model.output)
134     x = Dense(256, activation='relu')(x)
135     x = Dropout(0.5)(x)
136     output = Dense(num_classes, activation='softmax')(x)
137
138     model = Model(inputs=base_model.input, outputs=output)
139     model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['
accuracy'])
140     return model
141
142 # Define gerador de dados com técnicas de aumento (data augmentation) para o
    conjunto de treino
143 train_datagen_aug = ImageDataGenerator(
144     rescale=1./255,
145     rotation_range=20,

```



```

146     width_shift_range=0.2,
147     height_shift_range=0.2,
148     shear_range=0.2,
149     zoom_range=0.2,
150     horizontal_flip=True,
151     fill_mode='nearest'
152 )
153
154 # Cria o gerador de dados com Data Augmentation
155 train_generator_aug = train_datagen_aug.flow_from_directory(
156     NEW_TRAIN_DIR,
157     target_size=(IMG_WIDTH, IMG_HEIGHT),
158     batch_size=BATCH_SIZE,
159     class_mode='categorical'
160 )

```

2. Treinamento dos modelos VGG16 e ResNet50

Neste trecho, realiza-se o treinamento das arquiteturas VGG16 e ResNet50 utilizando transferência de aprendizado. As camadas convolucionais dos modelos base são congeladas, e camadas densas personalizadas são adicionadas para classificação multiclasse. O processo é repetido tanto com os dados originais quanto com Data Augmentation, possibilitando a posterior comparação dos efeitos do aumento de dados no desempenho dos modelos.

```

1 # Carrega o modelo VGG16 pré-treinado sem as camadas finais (topo)
2 vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224,
3     3))
4
5 # Congela todas as camadas convolucionais do modelo base
6 for layer in vgg16_base.layers:
7     layer.trainable = False
8
9 # Adiciona camadas densas personalizadas ao modelo VGG16
10 x = Flatten()(vgg16_base.output)
11 x = Dense(256, activation='relu')(x)
12 x = Dropout(0.5)(x) # Aplica regularização para reduzir overfitting
13 output = Dense(4, activation='softmax')(x) # Camada de saída para 4 classes
14
15 # Define o modelo final VGG16 sem data augmentation
16 vgg16_model = Model(inputs=vgg16_base.input, outputs=output)

```

```

17 # Compila o modelo com otimizador Adam e entropia cruzada categórica
18 vgg16_model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['
    accuracy'])
19
20 # Define os callbacks para salvar o melhor modelo e parar antecipadamente se
    necessário
21 callbacks = [
22     ModelCheckpoint('vgg16_best_model_no_aug.keras', monitor='val_loss', save_
        best_only=True),
23     EarlyStopping(monitor='val_loss', patience=5)
24 ]
25
26 # Treina o modelo VGG16 com dados originais (sem Data Augmentation)
27 vgg16_history = vgg16_model.fit(
28     train_generator,
29     epochs=10,
30     validation_data=validation_generator,
31     callbacks=callbacks
32 )
33
34 # Carrega o modelo ResNet50 pré-treinado sem as camadas finais
35 resnet50_base = ResNet50(weights='imagenet', include_top=False, input_shape=(224,
    224, 3))
36
37 # Congela todas as camadas convolucionais da ResNet50
38 for layer in resnet50_base.layers:
39     layer.trainable = False
40
41 # Adiciona camadas densas ao modelo ResNet50
42 x = Flatten()(resnet50_base.output)
43 x = Dense(256, activation='relu')(x)
44 output = Dense(4, activation='softmax')(x) # Saída com 4 classes
45
46 # Define o modelo final ResNet50 sem Data Augmentation
47 resnet50_model = Model(inputs=resnet50_base.input, outputs=output)
48
49 # Compila o modelo com otimizador Adam
50 resnet50_model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics
    =['accuracy'])

```

```

51
52 # Define os callbacks para salvar o melhor modelo
53 callbacks = [
54     ModelCheckpoint('resnet50_best_model_no_aug.keras', monitor='val_loss', save_
55         best_only=True),
56     EarlyStopping(monitor='val_loss', patience=5)
57 ]
58 # Treina o modelo ResNet50 com dados originais (sem Data Augmentation)
59 resnet50_history = resnet50_model.fit(
60     train_generator,
61     epochs=10,
62     validation_data=validation_generator,
63     callbacks=callbacks
64 )
65
66 # Carrega o modelo VGG16 para novo treinamento com Data Augmentation
67 vgg16_base_augmented = VGG16(weights='imagenet', include_top=False, input_shape
68     =(224, 224, 3))
69
70 # Congela as camadas convolucionais da base VGG16
71 for layer in vgg16_base_augmented.layers:
72     layer.trainable = False
73
74 # Adiciona camadas densas personalizadas com Dropout
75 x = Flatten()(vgg16_base_augmented.output)
76 x = Dense(256, activation='relu')(x)
77 x = Dropout(0.5)(x) # Regularização para evitar overfitting
78 output = Dense(4, activation='softmax')(x)
79
80 # Define o modelo VGG16 com Data Augmentation
81 vgg16_model_augmented = Model(inputs=vgg16_base_augmented.input, outputs=output)
82
83 # Compila o modelo com uma taxa de aprendizado reduzida
84 vgg16_model_augmented.compile(optimizer=Adam(learning_rate=1e-4), loss='
85     categorical_crossentropy', metrics=['accuracy'])
86
87 # Define os callbacks para o treinamento com Data Augmentation
88 callbacks = [

```

```

87     ModelCheckpoint('vgg16_best_model_augmented.keras', monitor='val_loss', save_
      best_only=True),
88     EarlyStopping(monitor='val_loss', patience=5)
89 ]
90
91 # Treina o modelo VGG16 com Data Augmentation
92 vgg16_history_augmented = vgg16_model_augmented.fit(
93     train_generator,
94     epochs=10,
95     validation_data=validation_generator,
96     callbacks=callbacks
97 )
98
99 # Carrega o modelo ResNet50 para treinamento com Data Augmentation
100 resnet50_base_augmented = ResNet50(weights='imagenet', include_top=False, input_
      shape=(224, 224, 3))
101
102 # Congela as camadas convolucionais do modelo base
103 for layer in resnet50_base_augmented.layers:
104     layer.trainable = False
105
106 # Adiciona camadas densas com Dropout ao modelo
107 x = Flatten()(resnet50_base_augmented.output)
108 x = Dense(256, activation='relu')(x)
109 x = Dropout(0.5)(x) # Regularização
110 output = Dense(4, activation='softmax')(x)
111
112 # Define o modelo ResNet50 com Data Augmentation
113 resnet50_model_augmented = Model(inputs=resnet50_base_augmented.input, outputs=
      output)
114
115 # Compila o modelo com otimizador Adam
116 resnet50_model_augmented.compile(optimizer=Adam(), loss='categorical_crossentropy
      ', metrics=['accuracy'])
117
118 # Define os callbacks, incluindo ajuste de taxa de aprendizado durante o
      treinamento
119 callbacks = [
120     ModelCheckpoint('resnet50_best_model_augmented.keras', monitor='val_loss',

```

```

        save_best_only=True),
121     EarlyStopping(monitor='val_loss', patience=5),
122     ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=2, min_lr=1e-6)
123 ]
124
125 # Treina o modelo ResNet50 com Data Augmentation
126 resnet50_history_augmented = resnet50_model_augmented.fit(
127     train_generator,
128     epochs=10,
129     validation_data=validation_generator,
130     callbacks=callbacks
131 )

```

3. Aplicação dos modelos na base de teste

Nesta etapa, os modelos VGG16 e ResNet50 treinados com e sem Data Augmentation são carregados e avaliados com base no conjunto de teste. As predições geradas são convertidas para rótulos de classe, e os resultados são comparados com os rótulos verdadeiros, permitindo uma análise preliminar da performance dos modelos.

```

1 # Carrega os modelos VGG16 e ResNet50 treinados sem Data Augmentation
2 vgg16_model = load_model('vgg16_best_model_no_aug.keras')
3 resnet50_model = load_model('resnet50_best_model_no_aug.keras')
4
5 # Realiza a predição no conjunto de teste usando o modelo VGG16 (sem Data
   Augmentation)
6 vgg16_predictions = vgg16_model.predict(test_generator)
7
8 # Realiza a predição no conjunto de teste usando o modelo ResNet50 (sem Data
   Augmentation)
9 resnet50_predictions = resnet50_model.predict(test_generator)
10
11 # Converte as predições de probabilidade em rótulos de classe (índice da maior
   probabilidade)
12 vgg16_pred_classes = np.argmax(vgg16_predictions, axis=1)
13 resnet50_pred_classes = np.argmax(resnet50_predictions, axis=1)
14
15 # Obtém os rótulos reais do conjunto de teste
16 true_classes = test_generator.classes
17
18 # Carrega os modelos VGG16 e ResNet50 treinados com Data Augmentation

```

```

19 vgg16_model_augmented = load_model('vgg16_best_model_augmented.keras')
20 resnet50_model_augmented = load_model('resnet50_best_model_augmented.keras')
21
22 # Realiza a predição no conjunto de teste usando o modelo VGG16 (com Data
    Augmentation)
23 vgg16_predictions = vgg16_model_augmented.predict(test_generator)
24
25 # Realiza a predição no conjunto de teste usando o modelo ResNet50 (com Data
    Augmentation)
26 resnet50_predictions = resnet50_model_augmented.predict(test_generator)
27
28 # Converte as predições de probabilidade em rótulos de classe (índice da maior
    probabilidade)
29 vgg16_pred_classes = np.argmax(vgg16_predictions, axis=1)
30 resnet50_pred_classes = np.argmax(resnet50_predictions, axis=1)
31
32 # Obtém novamente os rótulos reais do conjunto de teste
33 true_classes = test_generator.classes
34
35 # Exibe os primeiros 20 rótulos verdadeiros e predições de ambos os modelos para
    verificação visual
36 print("Classes verdadeiras:", true_classes[:20])
37 print("Predições VGG16      :", vgg16_pred_classes[:20])
38 print("Predições ResNet50   :", resnet50_pred_classes[:20])

```

4. Cálculo de Métricas

Neste trecho, comparam-se quantitativamente os desempenhos dos modelos VGG16 e ResNet50, treinados com e sem a técnica de Data Augmentation. A avaliação é realizada sobre o conjunto de teste, por meio da geração das matrizes de confusão e do cálculo das métricas de desempenho: acurácia, sensibilidade, especificidade (calculada manualmente) e F1-Score. As matrizes de confusão são exibidas lado a lado, permitindo uma análise visual direta da performance dos modelos sob diferentes estratégias de treinamento.

```

1 # Calcula a matriz de confusão para o modelo VGG16 e ResNet50
2 vgg16_cm = confusion_matrix(true_classes, vgg16_pred_classes)
3 resnet50_cm = confusion_matrix(true_classes, resnet50_pred_classes)
4
5 # Calcula a acurácia dos modelos com base nas predições
6 vgg16_accuracy = accuracy_score(true_classes, vgg16_pred_classes)
7 resnet50_accuracy = accuracy_score(true_classes, resnet50_pred_classes)

```

```

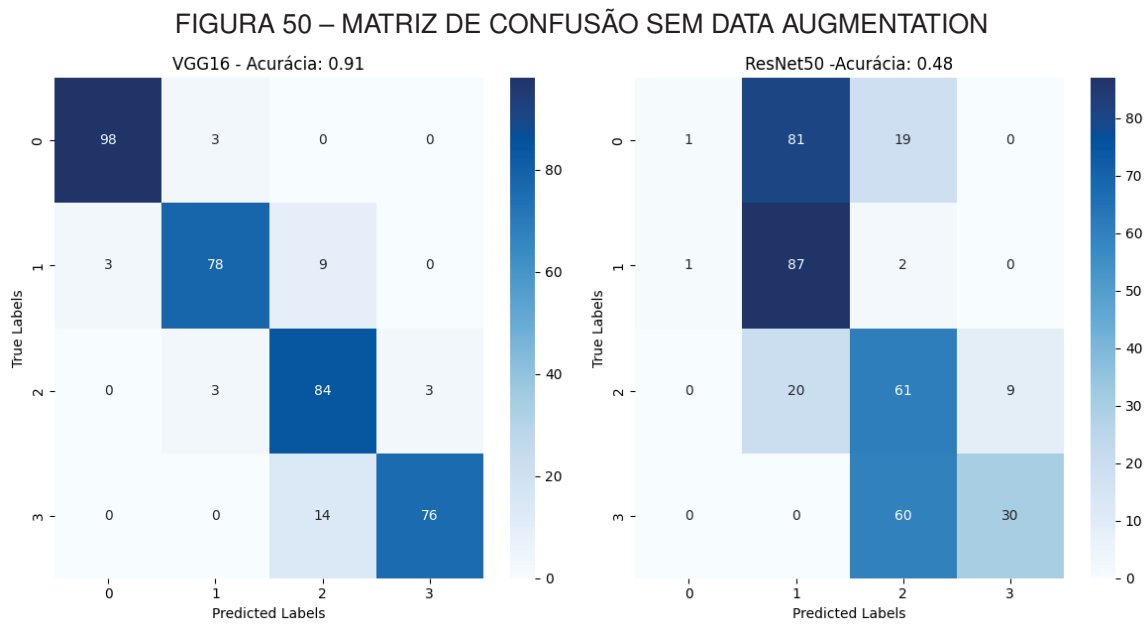
8
9 # Cria um layout com 1 linha e 2 colunas para plotar as matrizes de confusão lado
  a lado
10 fig, axes = plt.subplots(1, 2, figsize=(12, 6))
11
12 # Exibe a matriz de confusão do modelo VGG16
13 sns.heatmap(vgg16_cm, annot=True, fmt='d', cmap='Blues', ax=axes[0])
14 axes[0].set_title(f'VGG16 - Acurácia: {vgg16_accuracy:.2f}')
15 axes[0].set_xlabel('Predicted Labels')
16 axes[0].set_ylabel('True Labels')
17
18 # Exibe a matriz de confusão do modelo ResNet50
19 sns.heatmap(resnet50_cm, annot=True, fmt='d', cmap='Blues', ax=axes[1])
20 axes[1].set_title(f'ResNet50 - Acurácia: {resnet50_accuracy:.2f}')
21 axes[1].set_xlabel('Predicted Labels')
22 axes[1].set_ylabel('True Labels')
23
24 # Ajusta o layout e exibe os gráficos
25 plt.tight_layout()
26 plt.show()
27
28 # Calcula a sensibilidade (recall ponderado) para VGG16
29 sensibilidade_vgg16 = recall_score(true_classes, vgg16_pred_classes, average='
  weighted')
30
31 # Inicializa a variável da especificidade de VGG16 (será calculada manualmente)
32 especificidade_vgg16 = []
33
34 # Calcula o F1-Score ponderado para VGG16
35 f1_vgg16 = f1_score(true_classes, vgg16_pred_classes, average='weighted')
36
37 # Calcula a sensibilidade (recall ponderado) para ResNet50
38 sensibilidade_resnet50 = recall_score(true_classes, resnet50_pred_classes,
  average='weighted')
39
40 # Inicializa a variável da especificidade de ResNet50 (será calculada manualmente
  )
41 especificidade_resnet50 = []
42

```

```

43 # Calcula o F1-Score ponderado para ResNet50
44 f1_resnet50 = f1_score(true_classes, resnet50_pred_classes, average='weighted')
45
46 # Função auxiliar para calcular a especificidade média a partir da matriz de
    confusão
47 def calcular_especificidade(cm):
48     especificidade = []
49     for i in range(len(cm)):
50         tn = sum(sum(cm)) - (cm[i, i] + sum(cm[:, i]) + sum(cm[i, :]) - cm[i, i])
            # Verdadeiros Negativos
51         fp = sum(cm[:, i]) - cm[i, i] # Falsos Positivos
52         especificidade.append(tn / (tn + fp))
53     return sum(especificidade) / len(especificidade) # Média das especificidades
        por classe
54
55 # Calcula a especificidade média para ambos os modelos
56 especificidade_vgg16 = calcular_especificidade(vgg16_cm)
57 especificidade_resnet50 = calcular_especificidade(resnet50_cm)
58
59 # Define códigos ANSI para exibir os números das métricas em azul no terminal
60 blue = '\033[94m'
61 reset = '\033[0m'
62
63 # Exibe as métricas calculadas para VGG16
64 print(f"VGG16 - Sensibilidade : {blue}{sensibilidade_vgg16:.4f}{reset},
        Especificidade: {blue}{especificidade_vgg16:.4f}{reset}, F1-score: {blue}{f1_
        vgg16:.4f}{reset}")
65
66 # Exibe as métricas calculadas para ResNet50
67 print(f"ResNet50 - Sensibilidade: {blue}{sensibilidade_resnet50:.4f}{reset},
        Especificidade: {blue}{especificidade_resnet50:.4f}{reset}, F1-score: {blue}{
        f1_resnet50:.4f}{reset}")

```

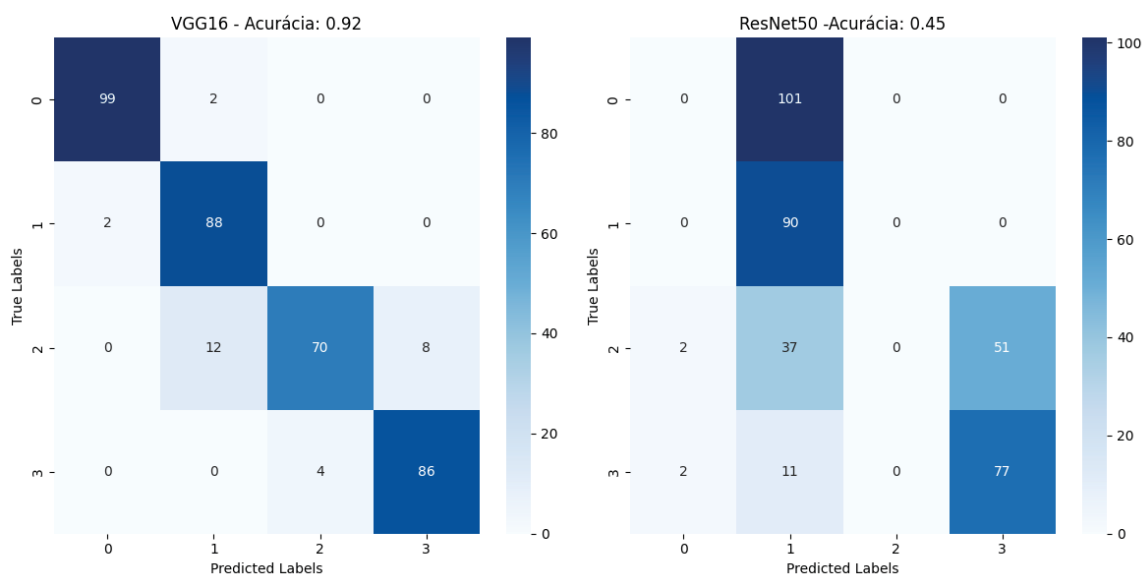
FONTE: Elaborado pelo autor (2024).

TABELA 62 – MÉTRICAS DE DESEMPENHO DOS MODELOS VGG16 E RESNET50 SEM DATA AUGMENTATION

Modelo	Sensibilidade	Especificidade	F1-Score	Acurácia
VGG16 (sem DA)	0,9057	0,9554	0,9067	0,91
ResNet50 (sem DA)	0,4825	0,7679	0,3975	0,48

FONTE: Elaborado pelo autor (2024).

FIGURA 51 – MATRIZ DE CONFUSÃO COM DATA AUGMENTATION



FONTE: Elaborado pelo autor (2024).

TABELA 63 – MÉTRICAS DE DESEMPENHO DOS MODELOS VGG16 E RESNET50 COM DATA AUGMENTATION

Modelo	Sensibilidade	Especificidade	F1-Score	Acurácia
VGG16 (com DA)	0,9245	0,9639	0,9231	0,92
ResNet50 (com DA)	0,4501	0,7388	0,3041	0,45

FONTE: Elaborado pelo autor (2024).

5. Modelo com o melhor resultado e a métrica utilizada.

Os resultados indicam que o modelo baseado em VGG16 apresenta desempenho superior ao ResNet50, tanto com quanto sem data augmentation. O VGG16 obteve melhores valores em sensibilidade, especificidade, F1-Score e acurácia, destacando-se especialmente pela acurácia de 0,92 com data augmentation. Isso demonstra maior precisão e equilíbrio na classificação das classes. Ademais, a aplicação de data augmentation contribuiu para a melhoria dos resultados, evidenciando sua importância em tarefas de classificação com conjuntos de dados limitados. Portanto, o modelo VGG16 mostrou-se mais adequado para o problema em questão.

APÊNDICE 12 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1. Classificação (RNA)

Implementar o exemplo de classificação utilizando a base de dados Fashion MNIST e a arquitetura RNA apresentada 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 o código abaixo.

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

2. Regressão (RNA)

Implementar o exemplo de regressão utilizando a base de dados Wine Dataset e a arquitetura RNA apresentada 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 é 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 o código abaixo.

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

Dica 1: Para facilitar o trabalho, renomeie os nomes das colunas para português:

```
1 data.columns = [
2     'acidez_fixa',          # fixed acidity
3     'acidez_volatil',      # volatile acidity
4     'acido_citrico',       # citric acid
5     'acucar_residual',     # residual sugar
6     'cloretos',           # chlorides
7     'dioxido_de_enxofre_livre', # free sulfur dioxide
8     'dioxido_de_enxofre_total', # total sulfur dioxide
9     'densidade',          # density
10    'pH',                  # pH
11    'sulfatos',            # sulphates
12    'alcool',              # alcohol
13    'score_qualidade_vinho' # quality
14 ]
```

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 utilizando a base de dados `Base_livros.csv` e a arquitetura apresentada 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 (Título), 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 utilizando uma imagem de um felino, retirada do site Wikipedia, e a arquitetura apresentada 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:

- **Imagem:** <https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg>
- **Importação da imagem:** Copiar o código abaixo.

```
1 url =
2 "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 abaixo:

<https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg>

B - RESOLUÇÃO

1. Classificação (RNA)

Importação de bibliotecas e dados

O código a seguir apresenta a etapa inicial da implementação de uma Rede Neural Artificial (RNA) para classificação de imagens, utilizando a base de dados Fashion MNIST. Esta etapa contempla a importação das bibliotecas necessárias, o carregamento do conjunto de dados e a definição manual dos rótulos das classes. Além disso, é realizada uma visualização gráfica com uma amostra representativa de cada classe presente no conjunto de treinamento, a fim de proporcionar uma familiarização com os dados que serão utilizados na construção e validação da RNA.

```
1 # Importa a biblioteca TensorFlow,
2 # utilizada para definição e treinamento de Redes Neurais Artificiais
3 import tensorflow as tf
4
5 # Importa os módulos de definição de modelos e camadas da API Keras
6 from tensorflow.keras import models, layers
```

```

7
8 # Importa o módulo pyplot da biblioteca Matplotlib, utilizado para geração de gráficos
9 import matplotlib.pyplot as plt
10
11 # Importa a biblioteca NumPy, que oferece suporte a operações matriciais
12 # e manipulação de arrays
13 import numpy as np
14
15 # Importa função da biblioteca mlxtend para visualização gráfica
16 # de matrizes de confusão
17 from mlxtend.plotting import plot_confusion_matrix
18
19 # Importa a função de cálculo de matriz de confusão da biblioteca Scikit-learn
20 from sklearn.metrics import confusion_matrix
21
22 # Importa o conjunto de dados Fashion MNIST da biblioteca Keras
23 from tensorflow.keras.datasets import fashion_mnist
24
25 # Atribui o conjunto de dados Fashion MNIST à variável 'data'
26 data = tf.keras.datasets.fashion_mnist
27
28 # Carrega os dados de treino e teste do conjunto Fashion MNIST
29 (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
30
31 # Define manualmente os nomes das classes
32 class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
33               'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
34
35 # Cria uma nova figura com tamanho 12x3 polegadas para
36 # exibir imagens representativas das classes
37 plt.figure(figsize=(12, 3))
38
39 # Itera sobre as dez classes do conjunto de dados para
40 # exibir uma imagem de exemplo de cada classe
41 for i in range(10):
42     # Cria um subplot com 1 linha e 10 colunas na posição i+1
43     plt.subplot(1, 10, i + 1)
44     # Exibe a imagem em escala de cinza
45     plt.imshow(x_train[i], cmap='gray')

```

```

46 # Define o título do subplot com o nome da classe correspondente
47 plt.title(class_names[y_train[i]])
48 # Oculta os eixos da imagem para facilitar a visualização
49 plt.axis('off')
50
51 # Ajusta automaticamente o layout da figura para evitar sobreposição de elementos
52 plt.tight_layout()
53
54 # Exibe a figura com as imagens representativas das classes
55 plt.show()

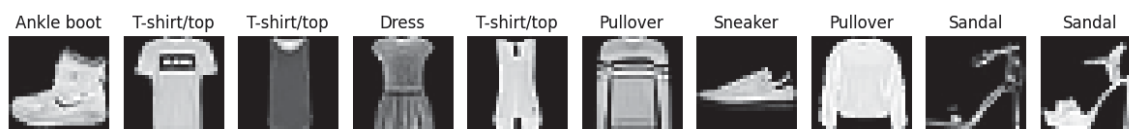
```

TABELA 64 – DIMENSÕES DOS CONJUNTOS DE DADOS UTILIZADOS NA TAREFA 1

Descrição	Dimensões
Conjunto de treinamento (imagens)	(60 000, 28, 28)
Conjunto de teste (imagens)	(10 000, 28, 28)
Labels de treinamento (<i>y_train</i>)	(60 000,)
Labels de teste (<i>y_test</i>)	(10 000,)

FONTE: Produzido pelo autor (2025).

FIGURA 52 – EXEMPLOS CLASSES PARA CLASSIFICAÇÃO



FONTE: Elaborado pelo autor (2025).

Definição e treinamento da rede neural artificial para classificação

O código apresentado nesta seção realiza o pré-processamento, definição, treinamento e avaliação de uma Rede Neural Artificial (RNA) aplicada à tarefa de classificação multiclasse utilizando a base de dados Fashion MNIST. Inicialmente, os dados são normalizados, e em seguida é construído um modelo sequencial com uma camada oculta e função de ativação ReLU. O modelo é treinado por 10 épocas, com validação automática sobre parte do conjunto de treino. Após o treinamento, o desempenho do modelo é avaliado sobre o conjunto de teste. Por fim, são gerados gráficos de perda e acurácia ao longo das épocas, com o objetivo de analisar o comportamento da RNA durante o processo de aprendizagem. Também é apresentada uma variação do modelo utilizando a API funcional do Keras, com inclusão de uma camada de *dropout*, visando reduzir o risco de sobreajuste.

```

1 # Normaliza os dados de entrada para o intervalo [0, 1],

```

```

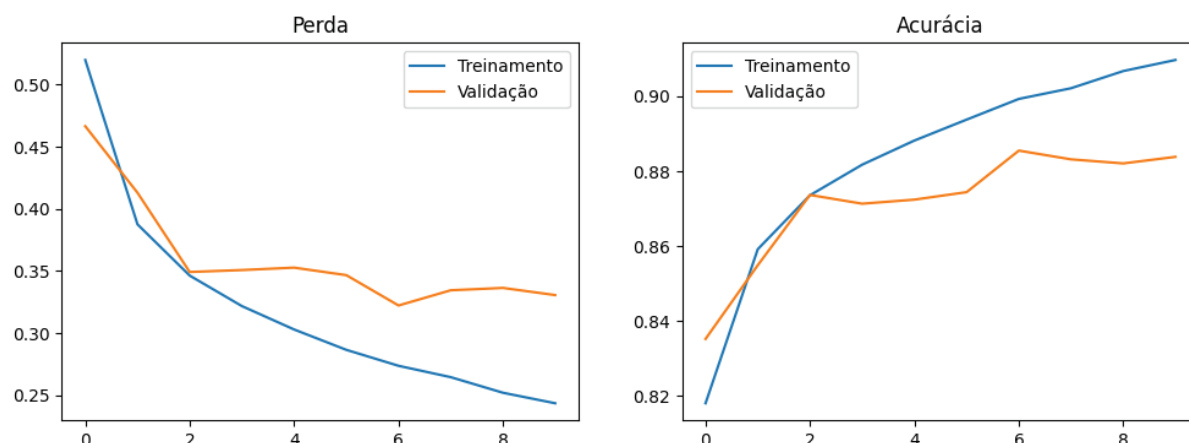
2 # dividindo todos os pixels por 255
3 x_train, x_test = x_train / 255.0, x_test / 255.0
4
5 # Define um modelo sequencial com três camadas
6 model = models.Sequential([
7     # Achata a imagem 28x28 em um vetor de 784 elementos
8     layers.Flatten(input_shape=(28, 28)),
9     # Adiciona uma camada densa com 128 neurônios e função de ativação ReLU
10    layers.Dense(128, activation='relu'),
11    # Adiciona a camada de saída com 10 neurônios e ativação softmax
12    layers.Dense(10, activation='softmax')
13 ])
14
15 # Compila o modelo com otimizador Adam
16 model.compile(optimizer='adam',
17               loss='sparse_categorical_crossentropy',
18               metrics=['accuracy'])
19
20 # Treina o modelo por 10 épocas, utilizando 20% dos dados de treino como validação
21 history = model.fit(x_train, y_train, epochs=10, validation_split=0.2)
22
23 # Avalia o modelo treinado no conjunto de teste
24 loss, accuracy = model.evaluate(x_test, y_test)
25
26 # Exibe a perda (loss) e a acurácia obtida no conjunto de teste
27 print(f"Loss: {loss}, Accuracy: {accuracy}")
28
29 # Redefine o modelo utilizando a API funcional do Keras
30 # Define a camada de entrada com forma 28x28
31 i = tf.keras.layers.Input(shape=(28, 28))
32 # Achata a entrada
33 x = tf.keras.layers.Flatten()(i)
34 # Adiciona camada densa com 128 neurônios e ReLU
35 x = tf.keras.layers.Dense(128, activation="relu")(x)
36 # Adiciona camada de dropout com taxa de 20%
37 x = tf.keras.layers.Dropout(0.2)(x)
38 # Camada de saída com 10 neurônios e softmax
39 x = tf.keras.layers.Dense(10, activation="softmax")(x)
40

```



```
41 # Cria o modelo completo conectando entrada e saída
42 model = tf.keras.models.Model(i, x)
43
44 # Compila o novo modelo com os mesmos parâmetros do anterior
45 model.compile(optimizer='adam',
46               loss='sparse_categorical_crossentropy',
47               metrics=['accuracy'])
48
49 # Gera gráfico com os valores de perda e acurácia ao longo das épocas
50 plt.figure(figsize=(12, 4))
51
52 # Gráfico da função de perda (loss)
53 plt.subplot(1, 2, 1)
54 plt.plot(history.history['loss'], label='Treinamento')
55 plt.plot(history.history['val_loss'], label='Validação')
56 plt.title('Perda')
57 plt.legend()
58
59 # Gráfico da acurácia
60 plt.subplot(1, 2, 2)
61 plt.plot(history.history['accuracy'], label='Treinamento')
62 plt.plot(history.history['val_accuracy'], label='Validação')
63 plt.title('Acurácia')
64 plt.legend()
65
66 # Exibe os gráficos gerados
67 plt.show()
```

FIGURA 53 – EVOLUÇÃO DA FUNÇÃO DE PERDA E DA ACURÁCIA DURANTE O TREINAMENTO DA RNA



FONTE: Produzido pelo autor (2025).

Durante o treinamento, observa-se que ambas as curvas de perda iniciam com valores elevados, indicando que o modelo apresenta, inicialmente, um alto grau de erro nas previsões. Com o avanço das épocas, a perda no conjunto de treinamento diminui de forma contínua, demonstrando que o modelo está aprendendo os padrões presentes nos dados de entrada. No entanto, a perda no conjunto de validação estabiliza-se por volta da sexta época e, posteriormente, apresenta uma leve elevação. Esse comportamento indica que o modelo começa a sofrer com sobreajuste (*overfitting*), ou seja, passa a se ajustar excessivamente aos dados de treinamento, perdendo capacidade de generalização. A partir desse ponto, aumentar o número de épocas tende a não melhorar o desempenho do modelo na validação, sendo recomendável adotar estratégias adicionais, como regularização, ajuste de arquitetura ou técnicas de aumento de dados.

Com relação à acurácia, nota-se que o desempenho do modelo no conjunto de treinamento melhora progressivamente ao longo das épocas, indicando que o modelo está conseguindo aprender as classes de forma eficaz. No entanto, a acurácia no conjunto de validação estabiliza-se a partir da sexta época, sugerindo que o modelo já atingiu sua capacidade máxima de generalização dentro da configuração atual. Assim, aumentar o número de épocas provavelmente não trará melhorias significativas no desempenho final, sendo necessário realizar ajustes na estrutura da rede ou nos dados para obter avanços adicionais.

Avaliação da rede neural com o conjunto de teste

Nesta etapa, realiza-se a avaliação final do desempenho da Rede Neural Artificial (RNA) utilizando o conjunto de teste, que contém dados não vistos durante o treinamento. São obtidos os valores de perda e acurácia sobre esse conjunto, permitindo verificar a capacidade de generalização do modelo. Em seguida, o modelo é utilizado para gerar as previsões sobre as classes das imagens de teste, que servirão de base para análises mais detalhadas, como a matriz de confusão e outras métricas de desempenho.

```

1 # Avalia o desempenho do modelo utilizando o conjunto de teste
2 # Retorna os valores de perda (loss) e acurácia (accuracy)
3 print(model.evaluate(x_test, y_test))
4
5 # Gera previsões para todas as imagens do conjunto de teste
6 y_pred = model.predict(x_test).argmax(axis=1)
7
8 # Exibe as classes previstas para o conjunto de teste
9 print(y_pred)

```

TABELA 65 – RESULTADOS DA AVALIAÇÃO DO MODELO NO CONJUNTO DE TESTE

Métrica	Valor
Perda (<i>loss</i>)	0,3591
Acurácia (<i>accuracy</i>)	87,55%

FONTE: Produzido pelo autor (2025).

TABELA 66 – EXEMPLOS DE CLASSES PREVISTAS PELO MODELO PARA O CONJUNTO DE TESTE

1ª	2ª	3ª	...	311ª	312ª	313ª
9	2	1	...	8	1	5

FONTE: Produzido pelo autor (2025).

Análise da classificação com matriz de confusão

Após o processo de treinamento e avaliação do modelo, esta etapa tem como objetivo analisar com mais profundidade os resultados obtidos. Para isso, é utilizada a matriz de confusão, que permite visualizar, de forma detalhada, o desempenho da Rede Neural Artificial (RNA) na classificação de cada uma das classes do conjunto de teste. Além disso, é calculada a acurácia manualmente, confirmando os valores obtidos anteriormente, e são apresentados exemplos de classificações incorretas, facilitando a identificação de padrões nos erros cometidos pelo modelo.

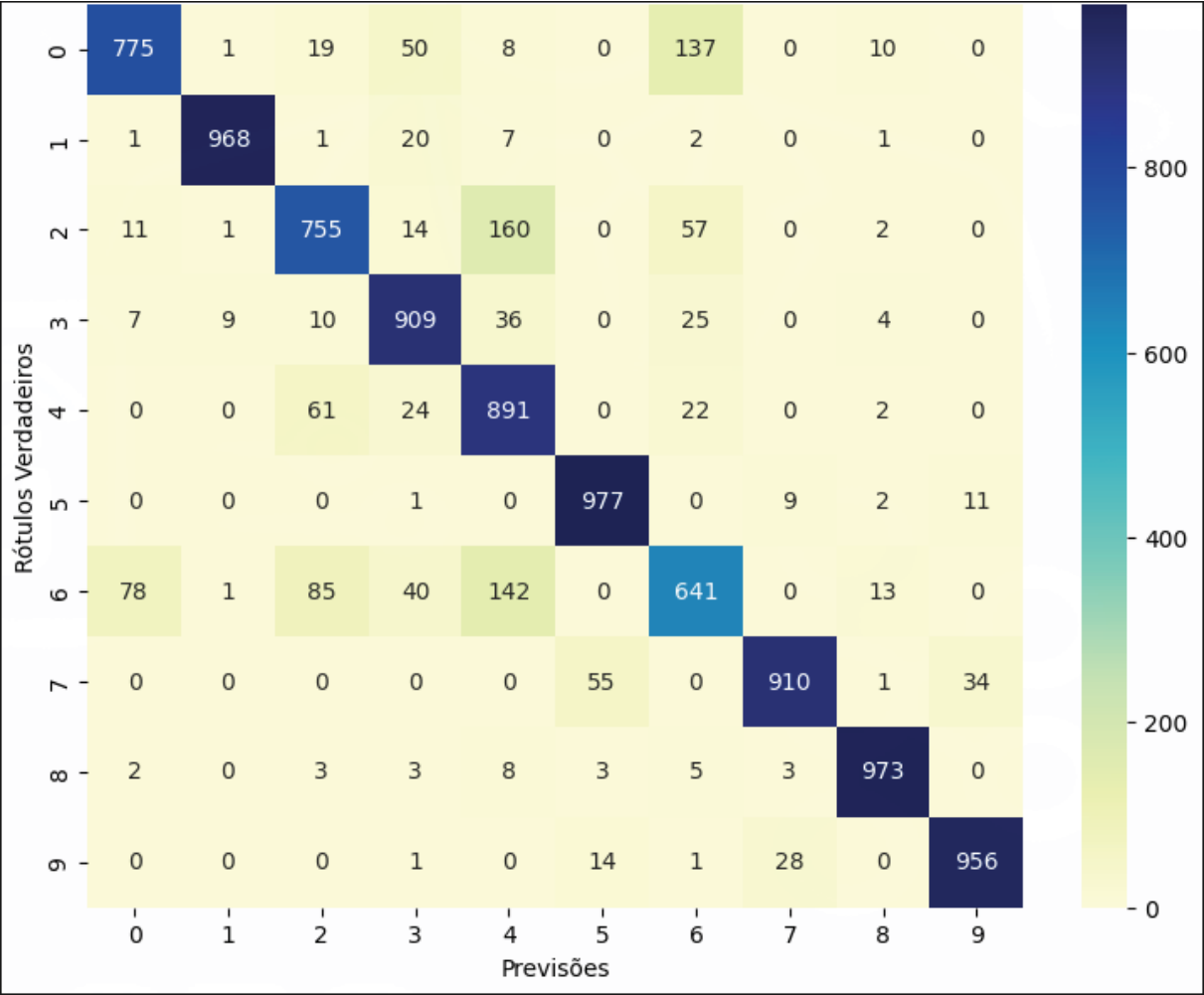
```

1 # Importa a biblioteca Seaborn para visualização avançada de gráficos
2 import seaborn as sns
3
4 # Calcula a acurácia manualmente comparando os rótulos reais com as previsões
5 accuracy = (y_test == y_pred).sum() / len(y_test)
6
7 # Gera a matriz de confusão com base nas classes reais e previstas

```

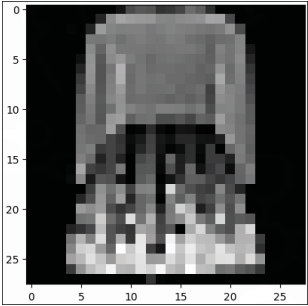
```
8 cm = confusion_matrix(y_test, y_pred)
9
10 # Plota a matriz de confusão como mapa de calor
11 plt.figure(figsize=(9, 7))
12 sns.heatmap(cm, annot=True, fmt='d', cmap='YlGnBu',
13             xticklabels=np.arange(10), yticklabels=np.arange(10))
14 plt.title(f'Matriz de Confusão (Acurácia: {accuracy * 100:.2f}%)', fontsize=14)
15 plt.xlabel('Previsões')
16 plt.ylabel('Rótulos Verdadeiros')
17 plt.show()
18
19 # Identifica os índices dos exemplos classificados incorretamente
20 misclassified = np.where(y_pred != y_test)[0]
21
22 # Seleciona aleatoriamente um dos exemplos incorretos
23 i = np.random.choice(misclassified)
24
25 # Exibe a imagem do exemplo incorretamente classificado
26 plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
27 plt.title("Rótulo verdadeiro: %s | Previsto: %s" % (y_test[i], y_pred[i]))
```

FIGURA 54 – MATRIZ CONFUSÃO (ACURÁCIA 87.55%)



FONTE: Produzido pelo autor (2025).

FIGURA 55 – EXEMPLO DE ERRO DE CLASSIFICAÇÃO COMETIDO PELO MODELO (RÓTULO VERDADEIRO: 3 | PREVISTO: 2)



FONTE: Produzido pelo autor (2025).

Visualização de previsões corretas e incorretas

A visualização de exemplos classificados corretamente e incorretamente pela Rede Neural Artificial (RNA) permite uma análise qualitativa do desempenho do modelo. Nesta etapa, são exibidas amostras aleatórias do conjunto de teste, com destaque para os acertos e erros de classificação. Em seguida, são apresentados pares de imagens: uma imagem que foi classificada de forma incorreta pelo modelo e, ao lado, um exemplo real da classe que foi erroneamente prevista. Essa abordagem auxilia na identificação de possíveis padrões ou semelhanças visuais que podem estar dificultando a discriminação entre classes.

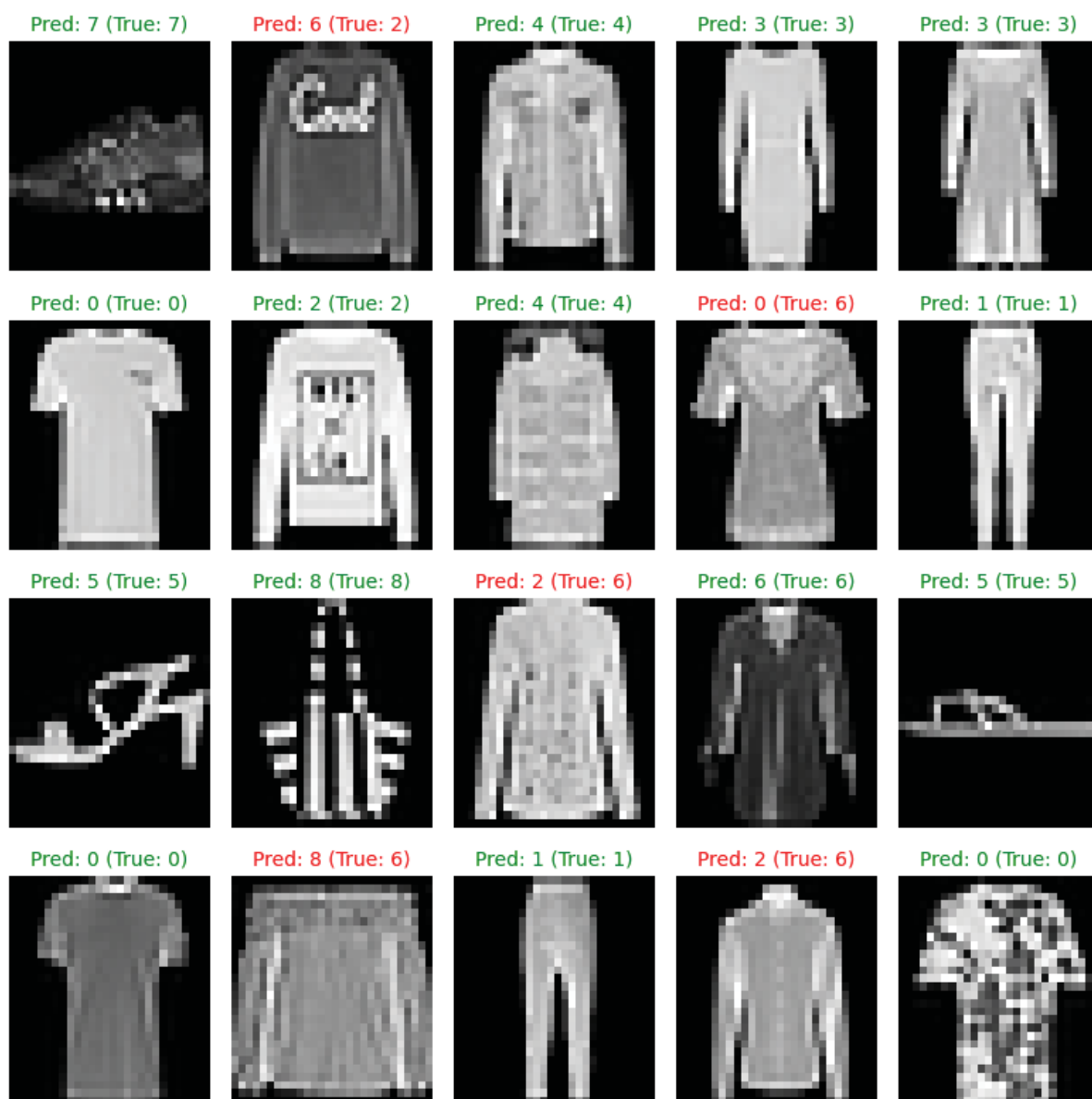
```

1 # Seleciona 20 imagens aleatórias do conjunto de teste
2 indices = np.random.choice(len(x_test), 20, replace=False)
3
4 # Cria uma figura com 4 linhas e 5 colunas para exibir as imagens
5 fig, axes = plt.subplots(4, 5, figsize=(8, 8)) # Ajuste de tamanho total da figura
6
7 # Itera sobre os índices e os eixos para exibir as imagens
8 for i, ax in zip(indices, axes.flatten()):
9     # Exibe a imagem em tons de cinza
10    ax.imshow(x_test[i], cmap='gray', aspect='auto')
11    ax.axis('off') # Remove os eixos para limpar a visualização
12
13    # Define a cor do título com base na correção da previsão
14    if y_pred[i] == y_test[i]:
15        color = 'green' # Acerto
16    else:
17        color = 'red'   # Erro
18
19    # Define o título da imagem com a classe prevista e a real
20    ax.set_title(f"Pred: {y_pred[i]} (True: {y_test[i]})", color=color, fontsize=10)
21
22 # Ajusta o layout para evitar sobreposição entre os elementos
23 plt.tight_layout()
24 plt.show()
25
26 # Seleciona 10 exemplos aleatórios onde o modelo errou a classificação
27 indices = np.random.choice(np.where(y_pred != y_test)[0], 10, replace=False)
28
29 # Cria uma figura com 10 linhas e 2 colunas
30 # Cada linha contém uma imagem incorretamente classificada e um exemplo da classe
31     prevista

```

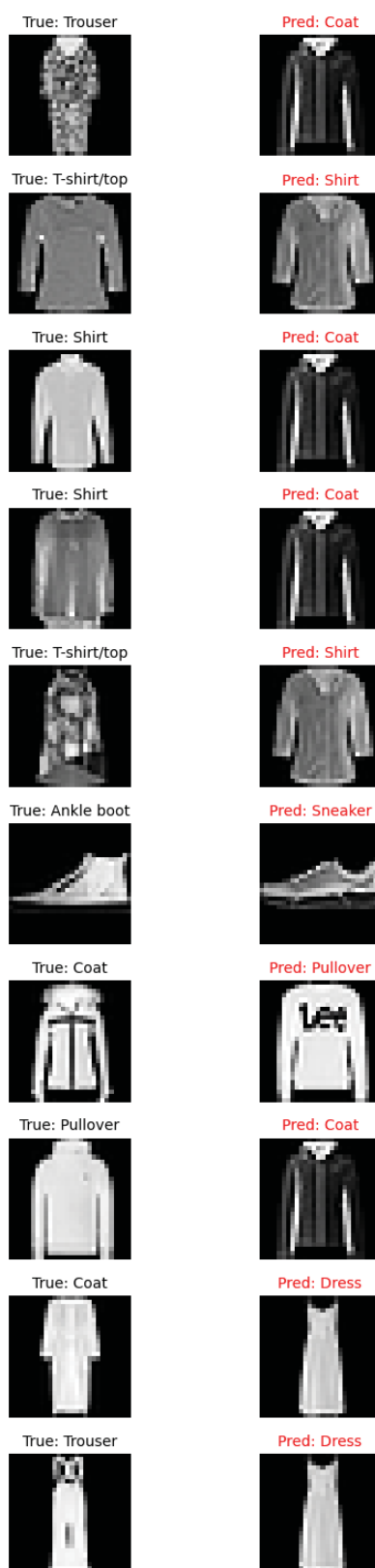
```
30 fig, axes = plt.subplots(10, 2, figsize=(6, 15))
31
32 # Itera sobre os índices selecionados
33 for row, idx in enumerate(indices):
34     # Exibe a imagem que foi classificada incorretamente
35     axes[row, 0].imshow(x_test[idx].reshape(28, 28), cmap='gray')
36     axes[row, 0].axis('off')
37     axes[row, 0].set_title(f"True: {class_names[y_test[idx]]}", color='black',
38                             fontsize=10)
39
40     # Identifica a classe incorretamente prevista
41     incorrect_class = y_pred[idx]
42
43     # Seleciona um exemplo da classe incorretamente prevista
44     incorrect_idx = np.where(y_test == incorrect_class)[0][0]
45
46     # Exibe a imagem representativa da classe prevista (mas errada)
47     axes[row, 1].imshow(x_test[incorrect_idx].reshape(28, 28), cmap='gray')
48     axes[row, 1].axis('off')
49     axes[row, 1].set_title(f"Pred: {class_names[incorrect_class]}", color='red',
50                             fontsize=10)
51
52 # Ajusta o layout final da figura
53 plt.tight_layout()
54 plt.show()
```

FIGURA 56 – EXEMPLOS DE PREVISÕES CORRETAS E INCORRETAS REALIZADAS PELA RNA



FONTE: Produzido pelo autor (2025).

FIGURA 57 – COMPARAÇÃO ENTRE IMAGENS INCORRETAMENTE CLASSIFICADAS E AMOSTRAS DA CLASSE PREVISTA



FONTE: Produzido pelo autor (2025).

A visualização dos exemplos incorretamente classificados pelo modelo permite observar possíveis limitações na capacidade da Rede Neural Artificial (RNA) em distinguir entre classes visualmente semelhantes. Em diversos casos, as imagens apresentam características ambíguas ou pouco definidas, o que pode justificar a confusão do modelo. Por exemplo, alguns calçados como *sneakers* e *ankle boots* compartilham contornos parecidos, dificultando a correta classificação. Ao lado de cada erro, é exibida uma amostra real da classe que foi prevista erroneamente, possibilitando uma comparação direta entre os elementos e revelando, em muitos casos, uma semelhança visual que pode justificar o equívoco do modelo. Essa análise qualitativa é essencial para identificar padrões de erro e fundamentar ajustes futuros na arquitetura da rede ou nas estratégias de pré-processamento dos dados.

2. Regressão (RNA)

A seguir, apresenta-se a implementação em Python de um modelo de regressão utilizando Redes Neurais Artificiais (RNA) para prever a qualidade de vinhos com base em características físico-químicas. O conjunto de dados utilizado é o *Wine Quality Dataset*, com 1599 amostras e 12 variáveis, sendo uma delas o score de qualidade do vinho (variável dependente). O modelo segue a arquitetura proposta na aula "FRA - Aula 12 - 2.5", conforme as diretrizes do curso. O código fonte encontra-se devidamente comentado, e os resultados obtidos serão analisados posteriormente por meio de métricas de desempenho (como R^2 e MSE) e gráficos de perda.

Importação das bibliotecas e pré-processamento inicial dos dados

O trecho de código a seguir apresenta a etapa inicial do desenvolvimento do modelo de regressão utilizando Redes Neurais Artificiais (RNA). Nessa fase, realiza-se a importação das bibliotecas necessárias, o carregamento do conjunto de dados *Wine Quality* a partir do repositório da UCI Machine Learning Repository, e a preparação dos dados para o treinamento. As colunas do dataset são renomeadas para o idioma português, facilitando a interpretação, e os dados são divididos entre variáveis preditoras (*features*) e a variável alvo, que corresponde ao *score* de qualidade do vinho. Essa preparação é essencial para garantir que os dados estejam prontos para as próximas etapas de normalização, modelagem e avaliação do desempenho da rede neural.

```
1 # Importa a biblioteca TensorFlow (utilizada para construção e treino da RNA)
2 import tensorflow as tf
3
4 # Importa o pandas para manipulação de dados
5 import pandas as pd
6
7 # Importa o NumPy para operações numéricas
8 import numpy as np
```

```

9
10 # Importa o matplotlib para geração de gráficos
11 import matplotlib.pyplot as plt
12
13 # Importa funcionalidades do backend do Keras
14 from tensorflow.python.keras import backend
15
16 # Importa utilitário para separação dos dados em treino e teste
17 from sklearn.model_selection import train_test_split
18
19 # Importa métricas de avaliação de regressão
20 from sklearn.metrics import r2_score, mean_squared_error
21
22 # Importa função para cálculo de raiz quadrada
23 from math import sqrt
24
25 # Importa a biblioteca seaborn para visualização estatística
26 import seaborn as sns
27
28 # Importa o StandardScaler para normalização dos dados
29 from sklearn.preprocessing import StandardScaler
30
31 # Importa camadas, modelo funcional e regularizadores do Keras
32 from tensorflow.keras import layers, models, regularizers
33
34
35 # Lê os dados do dataset Wine Quality diretamente do repositório UCI
36 url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/
    winequality-red.csv"
37 data = pd.read_csv(url, delimiter=';')
38
39 # Renomeia as colunas para facilitar a leitura e manter o padrão em português
40 data.columns = [
41     'acidez_fixa',          # fixed acidity
42     'acidez_volatil',       # volatile acidity
43     'acido_citrico',        # citric acid
44     'acucar_residual',      # residual sugar
45     'cloretos',             # chlorides
46     'dioxido_de_enxofre_livre', # free sulfur dioxide

```

```

47     'dioxido_de_enxofre_total', # total sulfur dioxide
48     'densidade',               # density
49     'pH',                      # pH
50     'sulfatos',                # sulphates
51     'alcool',                  # alcohol
52     'score_qualidade_vinho'    # quality
53 ]
54
55 # Exibe as primeiras linhas do dataset para inspeção inicial
56 print(data.head())
57
58 # Separa os dados em variáveis independentes (X) e dependente (Y)
59 X = data.drop('score_qualidade_vinho', axis=1)
60 Y = data['score_qualidade_vinho']

```

TABELA 67 – AMOSTRAS INICIAIS DO CONJUNTO DE DADOS *WINE QUALITY*

AF	AV	AC	AR	CI	SO ₂ L	SO ₂ T	Dens.	pH	Sulf.	Alc.	Qual.
7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5
11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6
7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5

FONTE: Elaborado pelo autor com base no dataset *Wine Quality* (UCI, 2024)

Nota. AF: Acidez fixa; AV: Acidez volátil; AC: Ácido cítrico; AR: Açúcar residual; CI: Cloretos; SO₂ L: Dióxido de enxofre livre; SO₂ T: Dióxido de enxofre total; Dens.: Densidade; Sulf.: Sulfatos; Alc.: Alcool; Qual.: Score de qualidade do vinho.

Análise de correlação e normalização dos dados

Nesta etapa, realiza-se a análise exploratória da base de dados por meio da matriz de correlação entre as variáveis independentes, com o objetivo de identificar relações lineares relevantes. Em seguida, aplica-se a técnica de normalização utilizando o método *StandardScaler*, que transforma os dados para uma escala com média zero e desvio padrão igual a um. Esse procedimento é fundamental para garantir um treinamento mais estável e eficiente da Rede Neural Artificial.

```

1 # Calcula a matriz de correlação entre as variáveis numéricas
2 correlation_matrix = data.corr()
3
4 # Geramos o heatmap da matriz de correlação
5 plt.figure(figsize=(10, 8))

```

```

6 sns.heatmap(correlation_matrix, annot=True, cmap="Purples", fmt=".2f")
7 plt.title("Matriz de Correlação")
8 plt.show()
9
10 # Inicializando o StandardScaler
11 scaler = StandardScaler()
12
13 # Ajustando o scaler nos dados de treino e transformando X
14 X = scaler.fit_transform(X)

```

Análise da matriz de correlação

FIGURA 58 – MATRIZ DE CORRELAÇÃO



FONTE: Produzido pelo autor (2025).

A partir da matriz de correlação, é possível destacar os seguintes pontos relevantes em relação à variável de saída (*score_qualidade_vinho*) e às demais variáveis independentes:

- **Correlação com a variável de saída:** a variável *score_qualidade_vinho* apresenta correlação moderada com algumas variáveis, sendo a mais significativa com o teor alcoólico (*álcool*), com valor de 0,48. Esse resultado sugere que vinhos com maior teor alcoólico tendem a ser avaliados com maior qualidade. Os *sulfatos* também apresentam correlação positiva, embora mais fraca (0,25), o que pode indicar uma leve associação entre maiores níveis de sulfato e melhores avaliações.

- **Correlação entre variáveis independentes:** observam-se correlações internas relevantes, como entre *densidade* e *acidez fixa* (0,67), indicando que vinhos mais densos tendem a apresentar maior acidez fixa. Da mesma forma, as variáveis *dióxido de enxofre livre* e *dióxido de enxofre total* apresentam correlação moderada-alta (0,67), o que é esperado devido à relação direta entre esses componentes na composição química do vinho. A variável *pH* apresenta correlação negativa forte com a *acidez fixa* (-0,68), evidenciando que o aumento da acidez está associado à redução do pH.
- **Variáveis com baixa correlação com a qualidade:** variáveis como *cloretos*, *açúcar residual* e *dióxido de enxofre total* demonstram baixa correlação com o *score* de qualidade, indicando uma influência limitada ou indireta na percepção final da qualidade do vinho pelos avaliadores.

Separação dos dados, definição da arquitetura e treinamento do modelo

Nesta etapa, realiza-se a divisão dos dados em conjuntos de treino e teste, seguida pela definição da arquitetura da Rede Neural Artificial (RNA), composta por uma camada densa oculta com função de ativação ReLU e uma camada de saída linear para regressão. Adicionalmente, são aplicadas técnicas de regularização, como *Dropout*, para reduzir o risco de *overfitting*. Duas métricas personalizadas são implementadas para avaliação: o erro quadrático médio (RMSE) e o coeficiente de determinação (R^2). Por fim, o modelo é treinado utilizando o otimizador Adam e monitorado com *early stopping* para evitar sobreajuste. Os gráficos de perda durante o treinamento também são gerados para análise visual do desempenho.

```

1 # Divide os dados em conjuntos de treino e teste (75% treino, 25% teste)
2 x_train, x_test, y_train, y_test = train_test_split(X, Y,
3                                                     test_size=0.25)
4
5 # Define a camada de entrada com 11 variáveis (features)
6 i = tf.keras.layers.Input(shape=(11,))
7 x = tf.keras.layers.Dense(50, activation="relu")(i)
8
9 # Adiciona uma camada de Dropout com taxa de 20% para evitar overfitting
10 x = layers.Dropout(0.2)(x)
11
12 # Define a camada de saída com um único neurônio (regressão)
13 x = tf.keras.layers.Dense(1)(x)
14
15 # Cria o modelo funcional ligando entrada e saída
16 model = tf.keras.models.Model(i, x)
17
18 # Define a função personalizada de erro quadrático médio (RMSE)

```

```

19 def rmse(y_true, y_pred):
20     y_true = backend.cast(y_true, dtype='float32')
21     return backend.sqrt(backend.mean(backend.square(y_pred - y_true), axis=-1))
22
23 # Define a função personalizada do coeficiente de determinação (R2)
24 def r2(y_true, y_pred):
25     y_true = backend.cast(y_true, dtype='float32')
26     y_true_mean = backend.mean(y_true)
27     ss_tot = backend.sum(backend.square(y_true - y_true_mean))
28     ss_res = backend.sum(backend.square(y_true - y_pred))
29     return 1 - ss_res / (ss_tot + backend.epsilon())
30
31 # Define o otimizador Adam com taxa de aprendizado de 0.005
32 optimizer=tf.keras.optimizers.Adam(learning_rate=0.005)
33
34 # Comentários: foram testados outros otimizadores sem ganho relevante
35 #optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, momentum=0.5)
36 #optimizer=tf.keras.optimizers.RMSprop(0.01)
37
38 # Compila o modelo com função de perda MSE e métricas personalizadas
39 model.compile(optimizer=optimizer,
40               loss="mse",
41               metrics=[rmse, r2])
42
43 # Define a técnica de early stopping com paciência de 20 épocas
44 early_stop = tf.keras.callbacks.EarlyStopping(
45     monitor='val_loss',
46     patience=20,
47     restore_best_weights=True)
48
49 # Treina o modelo por até 1500 épocas, com validação e early stopping
50 r = model.fit(x_train, y_train,
51              epochs=1500,
52              validation_data=(x_test, y_test),
53              callbacks=[early_stop])
54
55 # Plota os gráficos de perda no treino e na validação
56 plt.plot( r.history["loss"], label="loss" )
57 plt.plot( r.history["val_loss"], label="val_loss" )

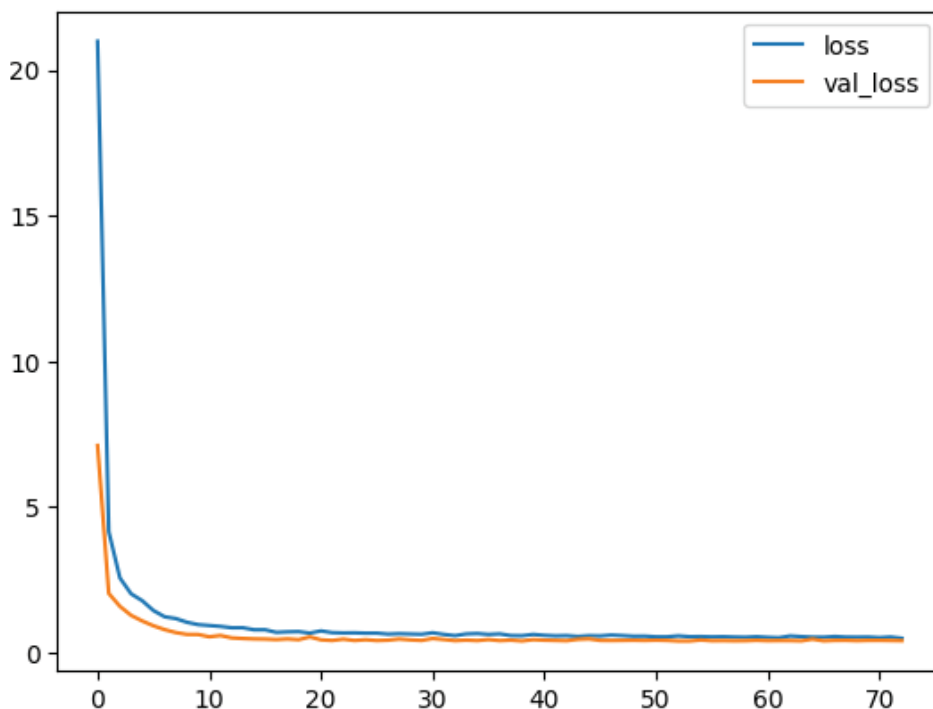
```

```

58 plt.legend()
59
60 # Plota os gráficos de RMSE no treino e na validação
61 plt.plot( r.history["rmse"], label="rmse" )
62 plt.plot( r.history["val_rmse"], label="val_rmse" )
63 plt.legend()
64
65 # Plota os gráficos de R2 no treino e na validação
66 plt.plot( r.history["r2"], label="r2" )
67 plt.plot( r.history["val_r2"], label="val_r2" )
68 plt.legend()

```

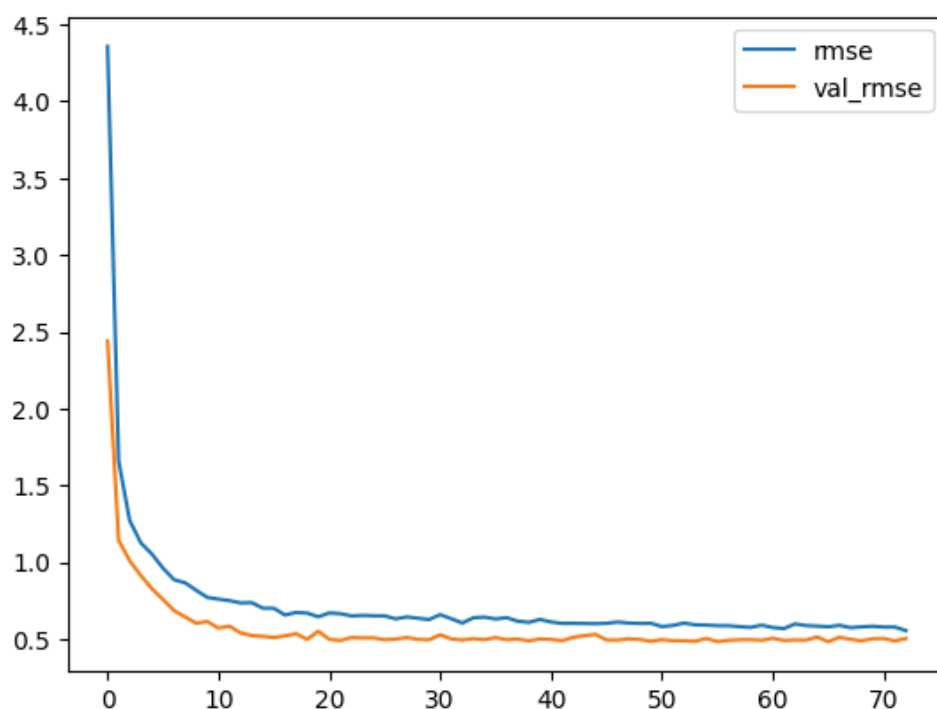
FIGURA 59 – EVOLUÇÃO DA FUNÇÃO DE PERDA DURANTE O TREINAMENTO E VALIDAÇÃO



FONTE: Produzido pelo autor (2025).

O gráfico de perda demonstra que, nas primeiras épocas, tanto a perda de treinamento quanto a de validação apresentam queda acentuada, indicando que o modelo está aprendendo de forma eficaz. A partir da décima época, ambas as curvas se estabilizam e mantêm valores próximos, o que sugere ausência de *overfitting*. Esse comportamento indica que o modelo convergiu para uma solução estável, com bom equilíbrio entre desempenho nos dados de treino e validação, caracterizando uma generalização adequada.

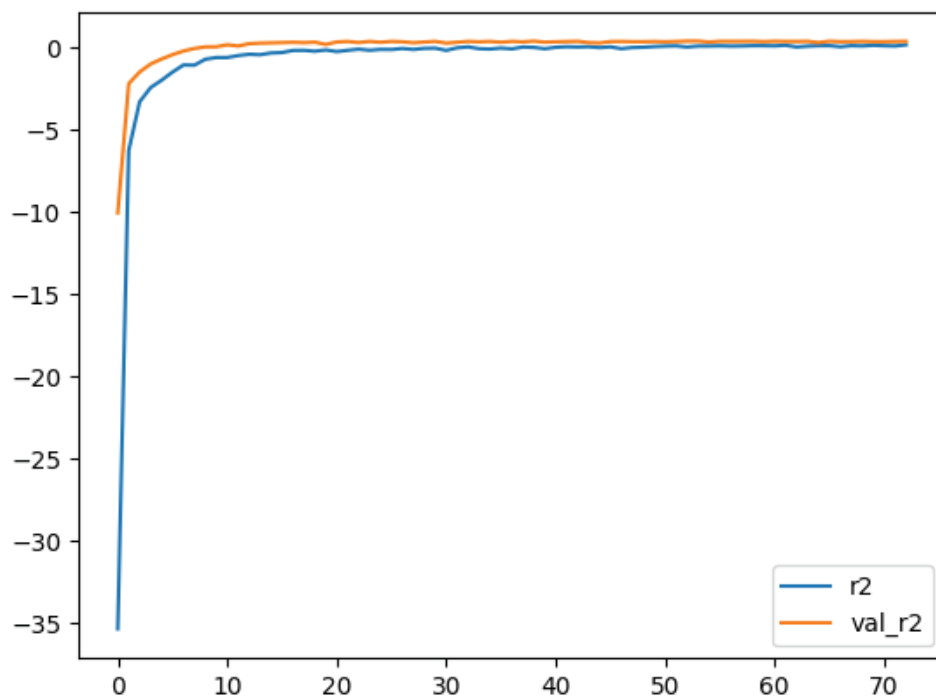
FIGURA 60 – EVOLUÇÃO DA MÉTRICA ERRO QUADRÁTICO MÉDIO DURANTE O TREINAMENTO E VALIDAÇÃO



FONTE: Produzido pelo autor (2025).

O gráfico da métrica RMSE mostra uma redução acentuada nas primeiras épocas, tanto para o conjunto de treinamento quanto para o de validação, o que indica um rápido aprendizado inicial. A partir da décima época, os valores se estabilizam em um patamar razoável, embora ainda distantes do ideal (próximo de zero). Essa estabilização indica que o aumento no número de épocas não resultaria em melhorias significativas no desempenho do modelo.

FIGURA 61 – EVOLUÇÃO DA FUNÇÃO DO COEFICIENTE DE DETERMINAÇÃO DURANTE O TREINAMENTO E VALIDAÇÃO



FONTE: Produzido pelo autor (2025).

O gráfico da métrica R^2 apresenta uma elevação rápida nas primeiras épocas, superando os valores negativos por volta da décima iteração. Após esse ponto, os valores se estabilizam próximos de zero, sem variações significativas. Esse comportamento indica que o modelo possui baixa capacidade de explicar a variabilidade dos dados de saída, sugerindo desempenho limitado em termos de previsibilidade. Além disso, a estabilização da métrica reforça que o aumento no número de épocas não traria melhorias relevantes.

Predição e avaliação do modelo

Nesta etapa, realiza-se a predição dos valores da variável dependente utilizando o conjunto de teste. Em seguida, o desempenho do modelo é avaliado por meio de métricas de regressão: erro quadrático médio (MSE), raiz do erro quadrático médio (RMSE) e coeficiente de determinação (R^2). Essas métricas fornecem uma visão quantitativa da precisão e da capacidade de generalização do modelo.

```
1 # Predição dos valores de saída com os dados de teste
2 y_pred = model.predict(x_test).flatten()
3
4 # Cálculo das métricas de acurácia: MSE, R2 e RMSE
5 mse = mean_squared_error(y_test, y_pred)
6 rmse = sqrt(mse)
```

```

7 r2 = r2_score(y_test, y_pred)
8
9 # Impressão dos resultados das métricas de avaliação
10 print("mse      = ", mse)
11 print("rmse     = ", rmse)
12 print("r2       = ", r2)

```

TABELA 68 – MÉTRICAS DE AVALIAÇÃO DO MODELO DE REGRESSÃO

Métrica	Valor
Erro quadrático médio (MSE)	0,4079
Raiz do erro quadrático médio (RMSE)	0,6387
Coefficiente de determinação (R^2)	0,4093

FONTE: Elaborado pelo autor (2025).

As métricas obtidas indicam um desempenho moderado do modelo, com margem para melhorias. Diversas tentativas de ajuste foram realizadas, porém, dentro dos limites da arquitetura proposta em aula, os valores apresentados representam o melhor resultado alcançado.

O erro quadrático médio (MSE) foi de 0,41, considerado relativamente alto, refletindo uma média de erros ainda significativa. A raiz do erro quadrático médio (RMSE), com valor de 0,64, reforça essa interpretação ao expressar o erro na mesma escala da variável predita. Por fim, o coeficiente de determinação (R^2) foi de 0,409, indicando que aproximadamente 40,9% da variabilidade da variável dependente é explicada pelo modelo — um valor que, embora positivo, demonstra que a capacidade preditiva ainda é limitada.

Conclusão

Com base nos resultados apresentados, conclui-se que a Rede Neural Artificial implementada foi capaz de aprender parcialmente os padrões da base de dados Wine Quality, atingindo um desempenho moderado. As métricas de avaliação, especialmente o R^2 , indicam que o modelo possui capacidade limitada de explicação da variabilidade da variável de saída. Apesar das tentativas de otimização, os resultados sugerem que ajustes mais profundos na arquitetura ou no pré-processamento dos dados seriam necessários para alcançar uma predição mais precisa. Ainda assim, o exercício atendeu aos objetivos propostos e permitiu aplicar, na prática, os conceitos de regressão com RNA.

3. Sistemas de Recomendação

Nesta tarefa, é implementado um sistema de recomendação baseado em Redes Neurais Artificiais, utilizando como base de dados o conjunto *Base_livros.csv*, que contém avaliações de usuários sobre diferentes livros. O objetivo principal é prever a nota que um usuário atribuiria a um livro não avaliado,

com base em padrões extraídos das interações anteriores. A arquitetura aplicada segue os princípios apresentados na Aula FRA — Aula 22 — 4.3. Ao final, são apresentados os gráficos de avaliação do treinamento (função de perda) e exemplos de recomendações personalizadas geradas para um usuário específico.

Leitura, pré-processamento e preparação da base de dados

Nesta etapa inicial, realiza-se a leitura da base de dados *Base_livros.csv*, contendo informações sobre avaliações de livros feitas por diferentes usuários. Em seguida, aplica-se o pré-processamento necessário: remoção de valores ausentes, codificação categórica das variáveis *ID_usuario* e *ISBN*, e normalização das notas. Por fim, são determinadas as dimensões fundamentais da base para posterior definição dos embeddings utilizados na arquitetura da RNA.

```

1 # Importação das bibliotecas necessárias do TensorFlow
2 import tensorflow as tf
3 from tensorflow.keras.layers import Input, Dense, Embedding, Flatten, Concatenate
4 from tensorflow.keras.models import Model
5 from tensorflow.keras.optimizers import SGD, Adam
6
7 # Importação da função de normalização MinMaxScaler
8 from sklearn.preprocessing import MinMaxScaler
9
10 # Importação de utilitários de embaralhamento
11 from sklearn.utils import shuffle
12
13 # Importação de bibliotecas auxiliares
14 import numpy as np
15 import pandas as pd
16 import matplotlib.pyplot as plt
17
18 # Montagem do Google Drive (necessário no ambiente Colab)
19 from google.colab import drive
20 drive.mount('/content/drive')
21
22 # Leitura da base de dados localizada no Google Drive
23 df = pd.read_csv("/content/drive/MyDrive/frameworksIA/Base_livros.csv")
24 df.head()
25
26 # Visualização dos valores únicos das avaliações
27 # Útil para verificar a escala e posterior desnormalização
28 valores_unicos_notas = df['Notas'].unique()

```

```

29 print("Valores únicos em 'Notas':", valores_unicos_notas)
30
31 #Valores únicos em 'Notas': [ 0.  2.  6.  1.  9.  3.  4.  7.  8.  5. 10. nan]
32
33 # Remoção de registros com valores nulos
34 df = df.dropna()
35
36 # Codificação categórica dos usuários
37 df.ID_usuario = pd.Categorical(df.ID_usuario)
38 df['new_ID_usu'] = df.ID_usuario.cat.codes
39
40 # Codificação categórica dos livros (ISBN)
41 df.ISBN = pd.Categorical(df.ISBN)
42 df['new_ISBN'] = df.ISBN.cat.codes
43
44 # Normalização das notas entre 0 e 1
45 scaler = MinMaxScaler()
46 df['notas_norm'] = scaler.fit_transform(df[['Notas']])
47
48 # Quantidade de usuários distintos
49 N = len(set(df.new_ID_usu))
50
51 # Quantidade de livros distintos
52 M = len(set(df.new_ISBN))
53
54 # Dimensão dos embeddings (hiperparâmetro)
55 K = 10

```

TABELA 69 – AMOSTRA DA BASE DE DADOS *BASE_LIVROS.CSV*

ISBN	Título	Autor	Ano	Editores	Usuário	Nota	ID Usu.	ID Livro	Norm.
2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0	5260	13802	0,0
60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2	5261	48023	0,2
374157065	Flu: The Story...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6	5262	26866	0,6
393045218	The Mummies...	E. J. W. Barber	1999	W. W. Norton & Company	276729	1	5263	30566	0,1
399135782	The Kitchen...	Amy Tan	1991	Putnam Pub Group	276729	9	5263	31881	0,9

FONTE: Elaborado pelo autor a partir da base de dados (2025)).

TABELA 70 – AMOSTRA DA BASE DE DADOS *BASE_LIVROS.CSV* DEPOIS DA NORMALIZAÇÃO

ISBN	Título	Autor	Ano	Editora	Usuário	Nota	ID Usu.	ID Livro	Nota Norm.
2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0	5260	13802	0,0
60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2	5261	48023	0,2
374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6	5262	26866	0,6
393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1	5263	30566	0,1
399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	276729	9	5263	31881	0,9

FONTE: Elaborado pelo autor a partir da base de dados (2025).

Modelagem do sistema de recomendação

O trecho de código a seguir implementa a arquitetura central do sistema de recomendação baseado em Redes Neurais Artificiais. São utilizados embeddings para representar usuários e livros em um espaço vetorial de dimensão reduzida, os quais são combinados e processados por camadas densas com regularização L2 e Dropout para evitar overfitting. Após a definição do modelo, os dados são embaralhados, normalizados e divididos em conjuntos de treino e teste. Por fim, o treinamento é realizado com validação e o desempenho é avaliado por meio do gráfico da função de perda.

```

1  # Importa a função de regularização L2
2  from tensorflow.keras.regularizers import l2
3
4  # Importa a camada Dropout para prevenir overfitting
5  from tensorflow.keras.layers import Dropout
6
7  # Entrada para IDs de usuários
8  u = Input(shape=(1,))
9
10 # Aplica embedding para representar usuários em um espaço de dimensão K
11 u_emb = Embedding(N, K)(u) # saída: (num_samples, 1, K)
12
13 # Remove a dimensão intermediária para preparar entrada densa
14 u_emb = Flatten()(u_emb) # saída: (num_samples, K)
15
16 # Entrada para IDs de livros
17 m = Input(shape=(1,))
18
19 # Aplica embedding para representar livros em um espaço de dimensão K
20 m_emb = Embedding(M, K)(m) # saída: (num_samples, 1, K)
21
22 # Remove a dimensão intermediária
23 m_emb = Flatten()(m_emb) # saída: (num_samples, K)
24
25 # Concatena as representações de usuários e livros

```

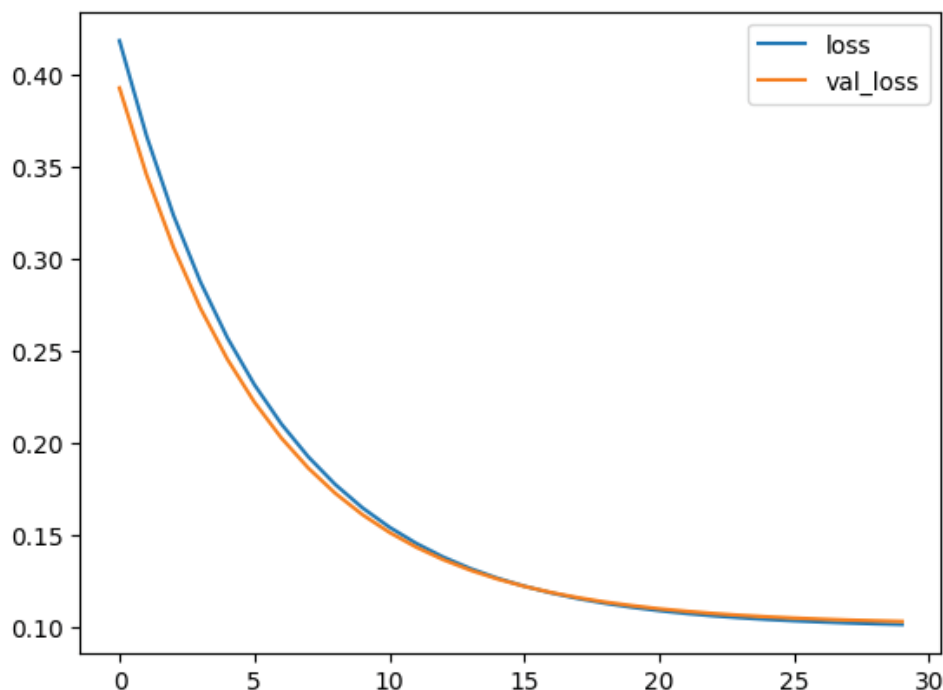
```

26 x = Concatenate()([u_emb, m_emb])
27
28 # Adiciona camada densa com ReLU e regularização L2
29 x = Dense(128, activation="relu", kernel_regularizer=l2(0.01))(x)
30
31 # Aplica Dropout com taxa de 50% para evitar overfitting
32 x = Dropout(0.5)(x)
33
34 # Camada de saída com um único valor (nota prevista)
35 x = Dense(1)(x)
36
37 # Define o modelo com entradas de usuário e livro e saída de nota
38 model = Model(inputs=[u, m], outputs=x)
39
40 # Compila o modelo com MSE como função de perda e otimizador SGD
41 model.compile(
42     loss="mse",
43     optimizer=SGD(learning_rate=0.001, momentum=0.9)
44     #optimizer=Adam(learning_rate=0.001) # Alternativa com Adam (comentada)
45 )
46
47 # Embaralha os dados de entrada: usuários, livros e notas normalizadas
48 user_ids, book_ids, ratings = shuffle(df.new_ID_usu, df.new_ISBN, df.notas_norm)
49
50 # Define o índice de corte para separar treino (80%) e teste (20%)
51 Ntrain = int(0.80 * len(ratings))
52
53 # Separa os dados de treino
54 train_user = user_ids[:Ntrain]
55 train_book = book_ids[:Ntrain]
56 train_ratings = ratings[:Ntrain]
57
58 # Separa os dados de teste
59 test_user = user_ids[Ntrain:]
60 test_book = book_ids[Ntrain:]
61 test_ratings = ratings[Ntrain:]
62
63 # Calcula a média das notas de treino para centralização
64 avg_rating = train_ratings.mean()

```

```
65
66 # Centraliza os valores de treino subtraindo a média
67 train_ratings = train_ratings - avg_rating
68
69 # Centraliza os valores de teste com a mesma média
70 test_ratings = test_ratings - avg_rating
71
72 # Treina o modelo com os dados de entrada e validação
73 r = model.fit(
74     x=[train_user, train_book],
75     y=train_ratings,
76     epochs=30,
77     batch_size=128,
78     verbose=2,
79     validation_data=([test_user, test_book], test_ratings)
80 )
81
82 # Plota a curva de perda do treinamento e validação
83 plt.plot(r.history["loss"], label="loss")
84 plt.plot(r.history["val_loss"], label="val_loss")
85 plt.legend()
86 plt.show()
```


FIGURA 62 – EVOLUÇÃO DA FUNÇÃO DE PERDA DURANTE O TREINAMENTO E VALIDAÇÃO



FONTE: Produzido pelo autor (2025).

As curvas de perda para treinamento e validação apresentam queda consistente ao longo das épocas, indicando que o modelo está aprendendo progressivamente. A partir da 25ª época, observa-se uma tendência de estabilização, embora com pequena redução contínua. A proximidade entre as curvas de treinamento e validação durante todo o processo sugere boa generalização do modelo, sem indícios de *overfitting*.

Predição e recomendação

Este trecho de código realiza a predição da nota esperada para um usuário específico em todos os livros disponíveis, utilizando o modelo treinado. A partir das predições, identifica-se o livro com maior avaliação prevista e são exibidas informações detalhadas sobre a recomendação gerada.

```

1 # Importa o NumPy para operações numéricas
2 import numpy as np
3
4 # Gerar array repetindo o ID do usuário para todos os livros
5 input_usuario = np.repeat(a=5360, repeats=M)
6
7 # Criar array dos IDs únicos dos livros
8 book = np.array(list(set(book_ids)))
9
10 # Prever notas para o usuário em todos os livros

```

```

11 preds = model.predict([input_usuario, book])
12
13 # Descentralizar as predições com média de avaliação do treinamento
14 rat = preds.flatten() + avg_rating
15
16 # Encontrar índice do livro com maior nota prevista
17 idx = np.argmax(rat)
18
19 # Exibir recomendação e índice do livro
20 print("Recomendação:")
21 print("Livro - ", book[idx], " / ", rat[idx], "*")
22
23 # Obter o ISBN do livro recomendado
24 isbn_recomendado = book[idx]
25
26 # Filtrar informações detalhadas do livro recomendado no DataFrame
27 livro_recomendado = df[df['new_ISBN'] == isbn_recomendado].iloc[0]
28
29 # Exibir informações originais do livro recomendado
30 print("ISBN :", livro_recomendado['ISBN'])
31 print("Nota :", livro_recomendado['Notas'])
32 print("Título:", livro_recomendado['Título'])
33 print("Autor:", livro_recomendado['Autor'])
34 print("Ano:", livro_recomendado['Ano'])
35 print("Editora:", livro_recomendado['Editora'])
36
37 # Repetição para garantir integridade do array do usuário
38 input_usuario = np.repeat(a=5360, repeats=M)
39
40 # Recriar array dos livros únicos
41 book = np.array(list(set(book_ids)))
42
43 # Prever novamente as notas para o usuário
44 preds = model.predict([input_usuario, book])
45
46 # Descentralizar e escalar as predições para intervalo original
47 rat_normalizada = preds.flatten() + avg_rating
48 rat = rat_normalizada * 10 # Ajuste para escala original das notas
49

```

```

50 # Encontrar índice do livro com maior nota prevista
51 idx = np.argmax(rat)
52
53 # Imprimir título da recomendação personalizada
54 print("=== Recomendação de Livro Personalizada ===")
55 print(f"Com base no histórico do usuário, recomendamos o livro com a maior previsão de
    avaliação.")
56 print("Livro Recomendado:", book[idx])
57 print("Nota Prevista:", round(rat[idx], 2), "*") # Notas arredondadas
58
59 # Obter dados originais do livro recomendado
60 isbn_recomendado = book[idx]
61 livro_recomendado = df[df['new_ISBN'] == isbn_recomendado].iloc[0]
62
63 # Exibir detalhes do livro recomendado
64 print("\n=== Informações do Livro Recomendado ===")
65 print("ISBN          :", livro_recomendado['ISBN'])
66 print("Nota Média    :", livro_recomendado['Notas'])
67 print("Título        :", livro_recomendado['Título'])
68 print("Autor          :", livro_recomendado['Autor'])
69 print("Ano            :", livro_recomendado['Ano'])
70 print("Editora         :", livro_recomendado['Editora'])

```

TABELA 71 – RECOMENDAÇÃO E DADOS DO LIVRO

Campo	Informação
Recomendação	Livro - 3020 / 0.502122 *
ISBN	067086935X
Nota	7.0
Título	Dog Brain
Autor	David Milgrim
Ano	1996.0
Editora	Viking Books

FONTE: Elaborado pelo autor com base na predição do modelo de sistema de recomendação (2025).

TABELA 72 – RECOMENDAÇÃO PERSONALIZADA DE LIVRO PARA USUÁRIO ESPECÍFICO

Resumo da Recomendação	
Livro Recomendado	3020
Nota Prevista	5,02
Informações do Livro	
ISBN	067086935X
Nota Média	7,0
Título	Dog Brain
Autor	David Milgrim
Ano	1996
Editora	Viking Books

FONTE: Elaborado pelo autor com base na predição do modelo de sistema de recomendação (2025) .

A predição realizada pelo modelo resultou na recomendação do livro *Dog Brain*, do autor David Milgrim, publicado pela editora Viking Books em 1996. A nota prevista para esse livro foi de aproximadamente 5,02, em uma escala normalizada, o que representa uma estimativa moderadamente positiva de avaliação por parte do usuário. No entanto, ao comparar com a nota média original atribuída ao livro (7,0), observa-se que o modelo ainda apresenta certa limitação na aproximação dos valores reais. Apesar disso, a recomendação pode ser considerada coerente dentro do contexto do sistema de recomendação, demonstrando a capacidade do modelo em identificar padrões relevantes no histórico de interações do usuário.

Conclusão

Neste tarefa, foi implementado um sistema de recomendação baseado em redes neurais artificiais, utilizando a base de dados *Base_livros.csv*. A arquitetura proposta demonstrou capacidade de aprendizado ao reduzir a função de perda durante o treinamento, além de apresentar bom desempenho na validação, sem indícios de sobreajuste. A recomendação personalizada gerada para um usuário específico indicou um livro com avaliação prevista coerente, evidenciando a aplicabilidade do modelo para sugestões relevantes. Embora os resultados sejam satisfatórios, observa-se a possibilidade de aprimoramento, sobretudo na precisão das predições.

4. Deepdream

A tarefa proposta visa implementar uma versão mínima da técnica DeepDream, utilizando uma imagem de um felino, com o objetivo de explorar os efeitos da rede neural na geração de imagens oníricas. Inicialmente, será realizada a importação e normalização da imagem, seguida da aplicação do algoritmo DeepDream. A partir disso, serão analisados os resultados gerados pelo modelo, tanto no processo de *Main Loop* quanto ao levar a rede até uma oitava, permitindo comparar as diferenças nas imagens oníricas resultantes de cada abordagem.

Carregamento e processamento inicial da imagem

O código apresentado nesta seção tem como objetivo realizar o carregamento e o pré-processamento de uma imagem de felino, utilizando a técnica DeepDream. Inicialmente, a imagem é obtida a partir de uma URL, redimensionada e convertida para um formato adequado, utilizando bibliotecas como TensorFlow e PIL. A seguir, a imagem é normalizada, ajustando seus valores para o intervalo adequado para o processamento da rede neural. O código também inclui a exibição da imagem original, com os devidos créditos ao autor da fotografia. Este pré-processamento é uma etapa fundamental para a aplicação do algoritmo DeepDream, permitindo observar as distorções e padrões gerados pela rede neural nas etapas subsequentes da tarefa.

```
1 # Importação das bibliotecas necessárias
2 import tensorflow as tf
3 import numpy as np
4 import matplotlib as mpl
5 import IPython.display as display
6 import PIL.Image
7
8 # URL da imagem a ser utilizada
9 url="https://commons.wikimedia.org/wiki/Special:FilePath/Felis_catus-cat_on_snow.jpg"
10
11 # Função para download da imagem e conversão para array Numpy
12 def download(url, max_dim=None):
13     # Extração do nome do arquivo da URL
14     name = url.split('/')[-1]
15     # Download da imagem utilizando o TensorFlow
16     image_path = tf.keras.utils.get_file(name, origin=url)
17     # Abertura da imagem com a biblioteca PIL
18     img = PIL.Image.open(image_path)
19     # Caso seja especificado, reduz o tamanho da imagem mantendo a proporção
20     if max_dim:
21         img.thumbnail((max_dim, max_dim))
22     # Retorna a imagem como um array Numpy
23     return np.array(img)
24
25 # Função de normalização da imagem para o intervalo de 0 a 255
26 def deprocess(img):
27     # Normaliza a imagem de modo a trazer os valores para o intervalo 0-255
28     img = 255 * (img + 1.0) / 2.0
29     # Converte a imagem para o tipo uint8 (para visualização)
```

```
30     return tf.cast(img, tf.uint8)
31
32 # Função para exibir a imagem no Jupyter notebook
33 def show(img):
34     # Exibe a imagem usando a biblioteca PIL
35     display.display(PIL.Image.fromarray(np.array(img)))
36
37 # Redução do tamanho da imagem para facilitar o processamento da rede neural
38 original_img = download(url, max_dim=500)
39
40 # Exibe a imagem original carregada
41 show(original_img)
42
43 # Exibe o crédito da imagem com o link apropriado
44 display.display(display.HTML('Image cc-by: <a href="https://commons.wikimedia.org/wiki/
    /Special:FilePath/Felis_catus-cat_on_snow.jpg">Von.grzanka</a>'))
45
46 # Preparacao do modelo
47 base_model = tf.keras.applications.InceptionV3(include_top=False, weights='imagenet')
48
49 # Maximização das ativações das camadas
50 names = ['mixed3', 'mixed5']
51 layers = [base_model.get_layer(name).output for name in names]
52
53 # Criação do modelo
54 dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)
```

FIGURA 63 – FELINO NA NEVE.



FONTE: *Von.grzanka*, disponível em
<https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg>.

Implementação do algoritmo DeepDream

O código abaixo implementa o algoritmo DeepDream utilizando o framework TensorFlow. A técnica de DeepDream aplica um processo iterativo para otimizar uma imagem de acordo com as ativações de uma rede neural pré-treinada. O algoritmo começa com uma imagem e, a cada iteração, ajusta a imagem para maximizar as ativações de camadas específicas da rede neural, gerando padrões "oníricos". A função `calc_loss` calcula a perda (loss), a classe `DeepDream` executa o algoritmo de otimização, e a função `run_deep_dream_simple` gerencia o processo completo. O código também permite a execução do DeepDream em múltiplas escalas (octavas) para capturar detalhes em diferentes resoluções.

```
1 # Função para calcular a perda (loss) da imagem em relação ao modelo.
2 def calc_loss(img, model):
3     # Converte a imagem em um batch de tamanho 1.
4     img_batch = tf.expand_dims(img, axis=0)
5
6     # Passa a imagem pelo modelo e obtém as ativações das camadas.
7     layer_activations = model(img_batch)
8
9     # Caso haja apenas uma ativação, transforma-a em uma lista.
```

```

10     if len(layer_activations) == 1:
11         layer_activations = [layer_activations]
12
13     # Inicializa uma lista para armazenar as perdas.
14     losses = []
15
16     # Para cada ativação da camada, calcula a perda média.
17     for act in layer_activations:
18         loss = tf.math.reduce_mean(act)
19         losses.append(loss)
20
21     # Retorna a soma das perdas.
22     return tf.reduce_sum(losses)
23
24 # Classe DeepDream que implementa a lógica de otimização das imagens.
25 class DeepDream(tf.Module):
26     # Inicializa o modelo de rede neural.
27     def __init__(self, model):
28         self.model = model
29
30     # Decorador para otimizar a execução da função em TensorFlow.
31     @tf.function(
32         # Define a assinatura da função (input_signature) com as especificações dos
33         # tensores de entrada.
34         input_signature=(
35             tf.TensorSpec(shape=[None, None, 3], dtype=tf.float32),
36             tf.TensorSpec(shape=[], dtype=tf.int32),
37             tf.TensorSpec(shape=[], dtype=tf.float32),)
38         )
39
40     # Método __call__ que executa a técnica DeepDream.
41     def __call__(self, img, steps, step_size):
42         print("Tracing")
43
44         # Inicializa a perda como 0.
45         loss = tf.constant(0.0)
46
47         # Loop principal para realizar as iterações do DeepDream.
48         for n in tf.range(steps):

```



```

48
49     # Abre um contexto para o cálculo de gradientes.
50     with tf.GradientTape() as tape:
51         # Vigia a imagem para calcular gradientes em relação a ela.
52         tape.watch(img)
53         # Calcula a perda (loss) para a imagem atual.
54         loss = calc_loss(img, self.model)
55
56         # Calcula os gradientes da perda em relação aos pixels da imagem.
57         gradients = tape.gradient(loss, img)
58
59         # Normaliza os gradientes para estabilizar o aprendizado.
60         gradients /= tf.math.reduce_std(gradients) + 1e-8
61
62         # Atualiza a imagem ajustando-a pelos gradientes e o tamanho do passo.
63         img = img + gradients * step_size
64         # Garante que os valores da imagem estejam dentro do intervalo [-1, 1].
65         img = tf.clip_by_value(img, -1, 1)
66
67     # Retorna a perda e a imagem modificada.
68     return loss, img
69
70 # Criação de uma instância da classe DeepDream.
71 deepdream = DeepDream(dream_model)
72
73 # Função para rodar o algoritmo DeepDream com parâmetros configuráveis.
74 def run_deep_dream_simple(img, steps=100, step_size=0.01):
75     # Pré-processa a imagem para o modelo InceptionV3.
76     img = tf.keras.applications.inception_v3.preprocess_input(img)
77
78     # Converte a imagem para tensor.
79     img = tf.convert_to_tensor(img)
80
81     # Converte o tamanho do passo para tensor.
82     step_size = tf.convert_to_tensor(step_size)
83
84     # Inicializa o número de etapas restantes.
85     steps_remaining = steps
86     step = 0

```

```

87
88     # Executa o loop de DeepDream até o número de etapas ser atingido.
89     while steps_remaining:
90         # Se restarem mais de 100 etapas, executa 100 por vez.
91         if steps_remaining > 100:
92             run_steps = tf.constant(100)
93         else:
94             run_steps = tf.constant(steps_remaining)
95
96         # Decrementa o número de etapas restantes.
97         steps_remaining -= run_steps
98         step += run_steps
99
100        # Executa o algoritmo DeepDream para as etapas atuais.
101        loss, img = deepdream(img, run_steps, tf.constant(step_size))
102
103        # Limpa a saída do Jupyter e exibe a imagem processada.
104        display.clear_output(wait=True)
105        show(deprocess(img))
106
107        # Exibe o progresso.
108        print("Step {}, loss {}".format(step, loss))
109
110        # Finaliza com a imagem gerada e exibe.
111        result = deprocess(img)
112        display.clear_output(wait=True)
113        show(result)
114
115        # Retorna a imagem resultante.
116        return result
117
118    # Execução da função DeepDream na imagem original.
119    dream_img = run_deep_dream_simple(img=original_img,
120                                      steps=100, step_size=0.01)
121
122    # Importa o módulo de tempo para medir a duração do processo.
123    import time
124    start = time.time() # Marca o início do processo.
125

```

```
126 # Define a escala de octavas para ajuste de resolução.
127 OCTAVE_SCALE = 1.3
128
129 # Converte a imagem original para tensor.
130 img = tf.constant(np.array(original_img))
131 base_shape = tf.shape(img)[: -1] # Obtém a forma base da imagem (sem o canal).
132 float_base_shape = tf.cast(base_shape, tf.float32) # Converte para formato de ponto
    flutuante.
133
134 # Loop para aplicar o DeepDream em diferentes resoluções da imagem (octavas).
135 for n in range(-2, 3): # Para escalas de 1/3 até 3x a resolução.
136     new_shape = tf.cast(float_base_shape*(OCTAVE_SCALE**n), tf.int32) # Calcula o
    novo tamanho da imagem.
137
138     # Redimensiona a imagem para a nova resolução.
139     img = tf.image.resize(img, new_shape).numpy()
140
141     # Aplica o DeepDream na imagem com a nova resolução.
142     img = run_deep_dream_simple(img=img, steps=100, step_size=0.005)
143
144 # Marca o fim do processo.
145 end = time.time()
146 # Calcula o tempo total de execução.
147 end - start
```

FIGURA 64 – IMAGEM GERADA PELO ALGORITMO DEEPPDREAM.



FONTE: *Von.grzanka*, processada com o algoritmo DeepDream (2025).

A **figura 64** foi obtida após a execução do algoritmo DeepDream, utilizando 100 iterações e um tamanho de passo de 0.01. Observa-se que a imagem gerada possui uma resolução relativamente baixa, com padrões visíveis principalmente em escalas grandes. Isso ocorre porque o processamento foi realizado em uma única escala da imagem original. Nesse contexto, a rede neural tem maior dificuldade em capturar detalhes e padrões em diferentes escalas, como pequenas texturas ou formas finas, o que resulta em uma imagem com padrões mais amplos e granulares.

FIGURA 65 – IMAGEM GERADA PELO ALGORITMO DEEPPDREAM APÓS A APLICAÇÃO DE OCTAVAS



FONTE: *Von.grzanka*, processada com o algoritmo DeepDream (2025).

A **figura 65** mostra o resultado da aplicação do algoritmo DeepDream após a introdução de octavas. O uso de diferentes escalas de resolução durante o processamento permitiu ao modelo identificar padrões e texturas em diversas granularidades. Como resultado, a imagem gerada se tornou mais detalhada e complexa, com uma maior mistura de texturas e padrões visuais. A adição das octavas intensificou a variação das características visuais, proporcionando uma visualização mais rica e intrincada, onde a estrutura original da imagem é transformada de maneira mais elaborada e dinâmica.

Conclusão

A aplicação do algoritmo DeepDream no código desenvolvido resultou em imagens progressivamente mais detalhadas e complexas. Inicialmente, a técnica gerou uma imagem com padrões visuais oníricos e distorcidos, conforme esperado no processo básico de otimização da imagem. Ao introduzir o conceito de octavas, foi possível aumentar a resolução da imagem em múltiplas escalas, resultando em uma visualização mais rica e com maior variedade de texturas e padrões.

Esse experimento demonstra a eficácia do DeepDream em transformar imagens por meio da maximização de ativações de camadas de redes neurais, e como a variação de parâmetros, como a escala de octavas, pode alterar significativamente o resultado final. A técnica oferece uma ferramenta poderosa para a exploração de padrões e estruturas visuais em imagens, além de possibilitar a experimentação com diferentes abordagens para gerar efeitos visuais inovadores e complexos.

APÊNDICE 13 – 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

FIGURA 66 – RESOLUÇÃO DA TAREFA DE GERENCIAMENTO DE PROJETOS DE IA.

Aluna/o: Edson Faria Alves				
Nome do artigo escolhido: XIN WU LEE Understanding the Role of Automation in Machine Learning Workflows_ ok.pdf				
Qual o objetivo do estudo descrito pelo artigo?	Qual o problema/opportunidade/situação que levou à necessidade de realização desse estudo?	Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?	Quais os principais resultados obtidos pelo estudo?	
Entender quais são as necessidades dos usuários e quais papéis a automação desempenha ativamente no fluxo de trabalho de Machine Learning. Com este entendimento, avaliar se as ferramentas de Auto-ML atendem efetivamente a essas necessidades.	Nos últimos anos, o aprendizado automático e automatizado (Auto-ML) emergiu como um campo inovador voltado para aumentar a automação no processo de Machine Learning. Esse crescimento foi acompanhado por um aumento significativo na popularidade de ferramentas que se autodenominam Auto-ML. O Auto-ML promete facilitar o acesso a Machine Learning para novos usuários e/ou desenvolvedores, reduzir o trabalho manual complicado associado à aplicação de técnicas de Machine Learning em domínios já estabelecidos, além de ter um conjunto de soluções já prontas para o uso.	Foi realizado um estudo qualitativo aprofundado com dezesseis usuários de ferramentas de Auto-ML. Foram conduzidas entrevistas semi-estruturadas com esses usuários, variando entre hobbyistas e pesquisadores da indústria em um conjunto diversificado de domínios, a fim de examinar cuidadosamente seus casos de uso e práticas de trabalho, bem como suas necessidades e desejos. Os autores fizeram perguntas sobre os usos do Machine Learning, as experiências dos participantes ao trabalhar com e sem essas ferramentas e suas percepções sobre elas. Essa abordagem permitiu não apenas uma melhor compreensão das necessidades dos usuários em relação ao Auto-ML e das forças e fraquezas das ferramentas existentes, mas também a obtenção de insights, por parte dos autores, sobre os papéis respectivos do desenvolvedor humano e da automação no desenvolvimento de Machine Learning.	O estudo indica que os desenvolvedores humanos são indispensáveis no desenvolvimento de Machine Learning. Não apenas os humanos se destacam em trazer conhecimento valioso sobre o domínio como a intuição humana também pode ser incrivelmente eficaz em preencher as lacunas nas ferramentas de Auto-ML. Além disso, a participação humana é crucial para o uso eficaz e socialmente responsável de Machine Learning em aplicações do mundo real. Portanto, os autores destacam, que em vez de tentar automatizar as tarefas dos desenvolvedores, como tem sido o objetivo de muitos construtores de ferramentas de Auto-ML, eles defendem que haja uma relação simbiótica entre o desenvolvedor humano e o Auto-ML, visando integrar o humano no processo da maneira mais produtiva possível.	

FONTE: Tarefa elaborada pelo autor baseada no artigo Xin *et al.* (2021) (2025).

APÊNDICE 14 – TÓPICOS DE 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 nas posições 0 e 100, e a ordem das cidades visitadas nas posições de 1 a 99 deverá ser definida 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 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

Esta tarefa visa implementar um algoritmo genético para o problema do caixeiro viajante com 100 cidades geradas aleatoriamente em um espaço bidimensional de 100 por 100 unidades. O objetivo é encontrar um percurso de menor distância total, passando uma única vez por todas as cidades e retornando à cidade de origem.

As cidades são geradas de forma aleatória, garantindo uma distância mínima entre elas. Em seguida, uma população inicial de soluções é criada aleatoriamente. A cada geração, os melhores indivíduos são selecionados com base na distância total do percurso, aplicando-se operadores genéticos de cruzamento (crossover ordenado) e mutação. A melhor solução é preservada entre as gerações. A execução do algoritmo é realizada por, no mínimo, 1000 gerações, e a evolução da melhor distância ao longo do tempo é plotada graficamente.

O código-fonte a seguir mostra a implementação completa do algoritmo, incluindo a visualização gráfica das cidades e dos caminhos encontrados.

```
1 # Importa biblioteca NumPy para manipulação numérica
2 import numpy as np
3 # Importa biblioteca Matplotlib para visualização gráfica
4 import matplotlib.pyplot as plt
5
6 # Função para gerar coordenadas de cidades de forma pseudoaleatória com uma distância
   mínima entre elas
7 def gerar_cidades(num_cidades=100, distancia_minima=3):
8     # Lista para armazenar as cidades válidas
9     cidades = []
10
11     # Loop de repetição para gerar o número total de cidades
12     while len(cidades) < num_cidades:
13         # Gera uma nova coordenada (x, y) aleatória no espaço de 0 a 100
14         nova_cidade = np.random.rand(1, 2) * 100
```

```

15
16     # Verifica se a nova cidade está a uma distância mínima das cidades já geradas
17     if all(np.linalg.norm(nova_cidade - c) >= distancia_minima for c in cidades):
18         # Adiciona a nova coordenada à lista de cidades
19         cidades.append(nova_cidade[0])
20
21     # Converte a lista de cidades para um array NumPy e retorna
22     return np.array(cidades)
23
24 # Gera o array de coordenadas para as 100 cidades
25 cidades = gerar_cidades()
26
27 # Plota os pontos das cidades em um gráfico de dispersão (scatter plot)
28 plt.scatter(cidades[:, 0], cidades[:, 1], c='blue')
29
30 # Define o título do gráfico
31 plt.title('Cidades')
32
33 # Define a legenda do eixo X
34 plt.xlabel('Coordenada X')
35
36 # Define a legenda do eixo Y
37 plt.ylabel('Coordenada Y')
38
39 # Exibe o gráfico com os pontos
40 plt.show()
41
42 # Função para calcular a distância total percorrida em um dado percurso
43 def calc_dist_total(cidades, percurso):
44     # Inicializa a distância total
45     distancia = 0
46
47     # Itera sobre o percurso, somando a distância entre cidades consecutivas
48     for i in range(len(percurso) - 1):
49         # Obtém as coordenadas da cidade atual
50         cidade_a = cidades[percurso[i]]
51         # Obtém as coordenadas da próxima cidade
52         cidade_b = cidades[percurso[i + 1]]
53         # Soma a distância euclidiana (norma) entre as duas cidades

```

```

54     distancia += np.linalg.norm(cidade_a - cidade_b)
55
56     # Obtém as coordenadas da última cidade percorrida
57     cidade_a = cidades[percurso[-1]]
58
59     # Obtém as coordenadas da cidade inicial (para fechar o ciclo)
60     cidade_b = cidades[percurso[0]]
61
62     # Soma a distância euclidiana para retornar à cidade de origem
63     distancia += np.linalg.norm(cidade_a - cidade_b)
64
65     # Retorna a distância total do ciclo
66     return distancia
67
68 # Função para criar a população inicial (conjunto de percursos aleatórios)
69 def criar_pop_ini(tam_pop, num_cidades):
70     # Lista para armazenar os indivíduos
71     populacao = []
72
73     # Loop de repetição para gerar o número definido de percursos
74     for _ in range(tam_pop):
75         # Cria um percurso aleatório (permutação de índices de cidades)
76         percurso = np.random.permutation(num_cidades)
77         # Adiciona o percurso à população
78         populacao.append(percurso)
79
80     # Retorna a lista de percursos (população inicial)
81     return populacao
82
83 # Implementa o operador de cruzamento 'Ordered Crossover' (OX1)
84 def ordered_crossover(parent1, parent2):
85     # Define o tamanho do percurso
86     size = len(parent1)
87
88     # Inicializa o novo percurso (filho) com valores sentinela
89     offspring = [-1] * size
90
91     # Seleciona dois pontos de corte aleatórios e os ordena
92     start, end = sorted(np.random.choice(range(size), 2, replace=False))

```

```

93
94     # Copia o segmento entre os pontos de corte do primeiro pai para o filho
95     offspring[start:end] = parent1[start:end]
96
97     # Inicializa o índice de preenchimento após o segmento copiado
98     current_index = end
99
100    # Itera sobre as cidades do segundo pai
101    for cidade in parent2:
102        # Verifica se a cidade do segundo pai ainda não está no filho
103        if cidade not in offspring:
104            # Garante o preenchimento circular do percurso
105            if current_index == size:
106                current_index = 0
107            # Adiciona a cidade ao filho
108            offspring[current_index] = cidade
109            # Move para a próxima posição de preenchimento
110            current_index += 1
111
112    # Retorna o percurso filho como um array NumPy
113    return np.array(offspring)
114
115    # Função de mutação por troca (swap mutation) com probabilidade
116    def mutacao(percurso, mutation_rate=0.001):
117        # Verifica se a mutação deve ocorrer baseado na taxa
118        if np.random.rand() < mutation_rate:
119            # Seleciona dois índices diferentes aleatoriamente no percurso
120            idx1, idx2 = np.random.choice(len(percurso), 2, replace=False)
121            # Troca as cidades nas posições selecionadas (swap)
122            percurso[idx1], percurso[idx2] = percurso[idx2], percurso[idx1]
123
124        # Retorna o percurso (mutado ou não)
125        return percurso
126
127    # Função para selecionar a próxima população com base no fitness (elitismo)
128    def sel_pop(populacao, cidades):
129        # Ordena a população de percursos com base na distância total (menor distância =
130        # melhor fitness)
131        populacao = sorted(populacao, key=lambda percurso: calc_dist_total(cidades,

```

```

percurso))
131
132     # Retorna apenas os 10% melhores percursos (elitismo simples)
133     return populacao[:int(0.1 * len(populacao))]
134
135 # Função principal que executa o Algoritmo Genético
136 def algoritmo_genetico(cidades, num_geracoes=1000, pop_size=100, mutation_rate=0.01):
137     # Cria a população inicial de percursos
138     populacao = criar_pop_ini(pop_size, len(cidades))
139
140     # Inicializa as variáveis para a melhor solução encontrada
141     melhor_solucao = None
142
143     # Inicializa a melhor distância com infinito (para garantir que a primeira seja a
144     # melhor)
145     melhor_distancia = float('inf')
146
147     # Lista para armazenar o histórico da melhor distância em cada geração
148     lista_melhor_distancia = []
149
150     # Loop principal do Algoritmo Genético, iterando sobre o número de gerações
151     for generation in range(num_geracoes):
152         # Seleciona os melhores 10% da população atual para elitismo
153         populacao = sel_pop(populacao, cidades)
154         # Cria a nova população, começando com os indivíduos selecionados
155         nova_populacao = populacao.copy()
156         # Loop para criar novos indivíduos por cruzamento até atingir o tamanho
157         # original da população
158         while len(nova_populacao) < pop_size:
159             # Seleciona dois pais aleatórios dos indivíduos selecionados
160             parent1, parent2 = np.random.choice(len(populacao), 2, replace=False)
161             # Aplica o operador de cruzamento para gerar um filho
162             offspring = ordered_crossover(populacao[parent1], populacao[parent2])
163             # Adiciona o filho à nova população
164             nova_populacao.append(offspring)
165
166         # Aplica mutação em cada indivíduo da nova população
167         for i in range(len(nova_populacao)):
168             # Aplica a função de mutação

```

```

167     nova_populacao[i] = mutacao(nova_populacao[i], mutation_rate)
168
169     # Avalia a nova população para encontrar a melhor solução global
170     for percurso in nova_populacao:
171         # Calcula a distância do percurso atual
172         distancia = calc_dist_total(cidades, percurso)
173         # Atualiza a melhor solução e a melhor distância se a atual for superior
174         if distancia < melhor_distancia:
175             melhor_solucao = percurso
176             melhor_distancia = distancia
177
178     # A nova população se torna a população atual para a próxima geração
179     populacao = nova_populacao
180
181     # Armazena a melhor distância obtida nesta geração
182     lista_melhor_distancia.append(melhor_distancia)
183
184     # Imprime e plota o progresso a cada 100 gerações (pode ser comentado no TCC
185     final)
186     if generation % 100 == 0:
187         # print(f"Geração {generation} - Melhor Distância: {melhor_distancia}")
188         # Chama a função para plotar o percurso da geração atual
189         plot_solution(cidades, melhor_solucao, melhor_distancia, generation)
190
191     # Imprime a melhor solução final após todas as gerações
192     print(f'Última Geração {num_geracoes} - Melhor Distância: {melhor_distancia}')
193
194     # Plota a melhor solução final (última iteração)
195     plot_solution(cidades, melhor_solucao, melhor_distancia, num_geracoes)
196
197     # Configura um novo gráfico para a evolução da distância
198     plt.figure(figsize=(8, 6))
199
200     # Plota a linha da melhor distância ao longo das gerações
201     plt.plot(range(num_geracoes), lista_melhor_distancia, color='blue')
202
203     # Define o título do gráfico de evolução
204     plt.title('Evolução da Distância ao Longo das Gerações')
205
206     # Define a legenda do eixo X

```

```

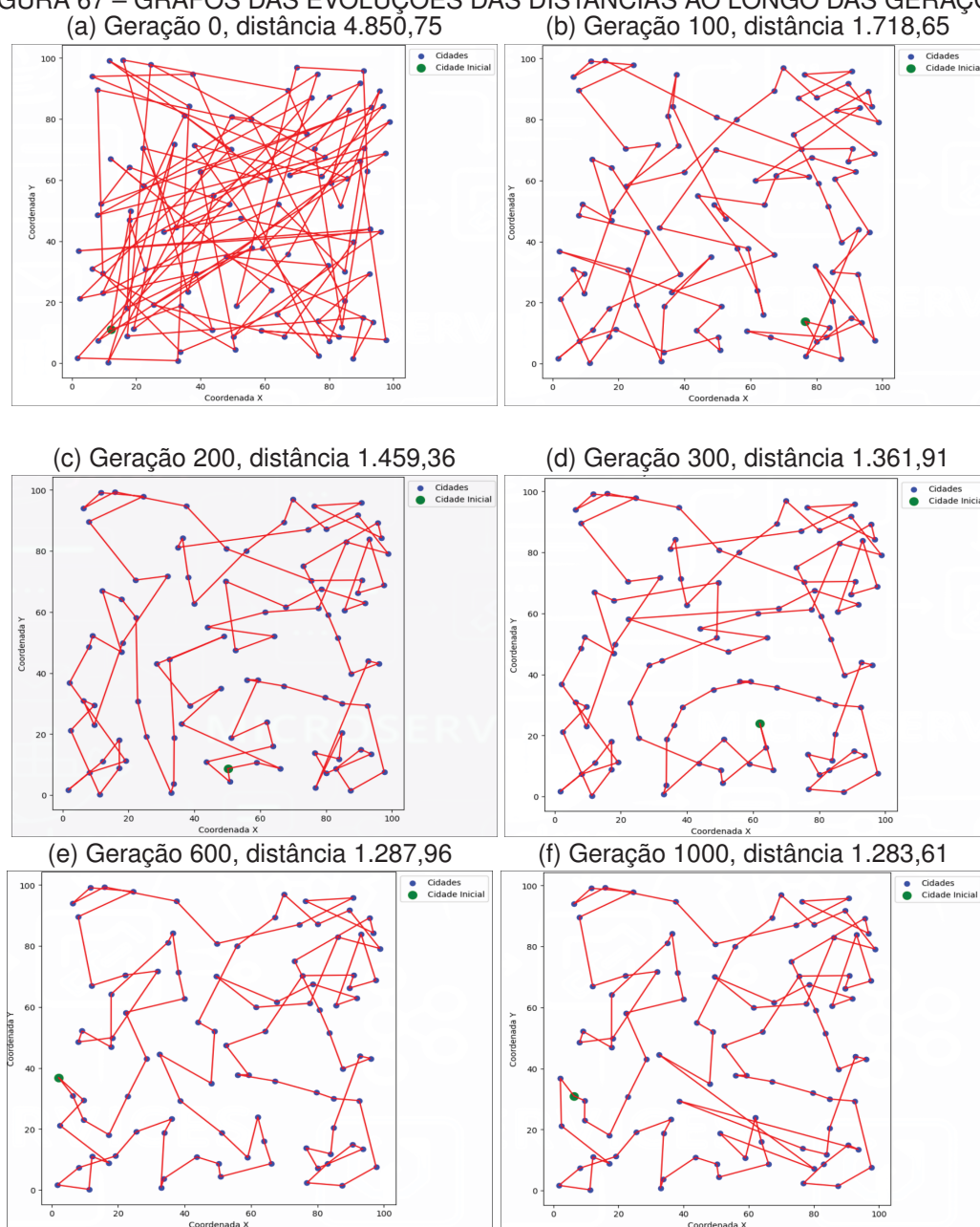
205     plt.xlabel('Geração')
206
207     # Define a legenda do eixo Y
208     plt.ylabel('Distância')
209
210     # Ativa as grades no gráfico
211     plt.grid(True)
212
213     # Exibe o gráfico de evolução
214     plt.show()
215
216     # Retorna a melhor solução e a melhor distância globais
217     return melhor_solucao, melhor_distancia
218
219 # Função para plotar o percurso encontrado pelo Algoritmo Genético
220 def plot_solution(cidades, melhor_solucao, melhor_distancia, generation):
221     # Cria a figura para o gráfico
222     plt.figure(figsize=(8, 8))
223
224     # Plota todos os pontos (cidades) em azul
225     plt.scatter(cidades[:, 0], cidades[:, 1], c='blue', label='Cidades')
226
227     # Destaca a cidade inicial do percurso em verde
228     plt.scatter(cidades[melhor_solucao[0], 0], cidades[melhor_solucao[0], 1], c='green',
229                s=100, label='Cidade Inicial')
230
231     # Adiciona a primeira cidade ao final do percurso para fechar o ciclo no plot
232     percurso = np.append(melhor_solucao, melhor_solucao[0])
233
234     # Mapeia os índices do percurso para as coordenadas das cidades
235     percurso_cidades = cidades[percurso]
236
237     # Plota o percurso (caminho) em vermelho
238     plt.plot(percurso_cidades[:, 0], percurso_cidades[:, 1], c='red')
239
240     # Define o título do gráfico com a geração e a distância
241     plt.title(f'Geração {generation} - Melhor Distância: {melhor_distancia:.2f}')
242
243     # Define a legenda do eixo X

```



```
243 plt.xlabel('Coordenada X')
244
245 # Define a legenda do eixo Y
246 plt.ylabel('Coordenada Y')
247
248 # Ajusta a posição da legenda para não sobrepor o gráfico
249 plt.legend(loc='upper left', bbox_to_anchor=(1, 1))
250
251 # Define proporções iguais para os eixos X e Y
252 plt.gca().set_aspect('equal', adjustable='box')
253
254 # Ajusta o layout para evitar cortes e sobreposições
255 plt.tight_layout()
256
257 # Exibe o gráfico
258 plt.show()
259
260 # Início da execução: chama o Algoritmo Genético com as cidades geradas
261 melhor_solucao, melhor_distancia = algoritmo_genetico(cidades)
```

FIGURA 67 – GRAFOS DAS EVOLUÇÕES DAS DISTÂNCIAS AO LONGO DAS GERAÇÕES



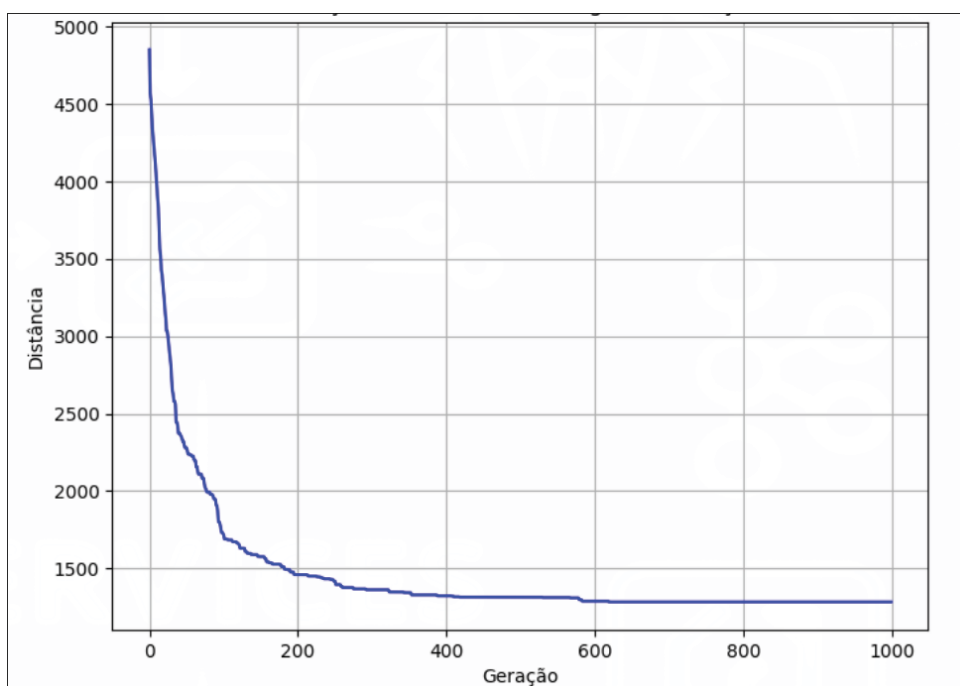
FONTE: Resultados obtidos pela execução do algoritmo genético (2025).

TABELA 73 – MELHOR DISTÂNCIA ENCONTRADA AO LONGO DAS GERAÇÕES

Geração	Melhor distância
0	4.850,75
100	1.718,65
200	1.459,36
300	1.361,91
400	1.322,36
500	1.313,65
600	1.287,96
700	1.283,61
800	1.283,61
900	1.283,61
1000	1.283,61

FONTE: Resultados obtidos pela execução do algoritmo genético (2025).

FIGURA 68 – GRÁFICO DA DISTÂNCIA AO LONGO DAS GERAÇÕES



FONTE: Produzido pelo autor (2025).

Conclusão

A aplicação do Algoritmo Genético (AG) para a otimização do problema do caixeiro viajante com 100 cidades demonstrou ser um método eficaz na convergência para uma solução de alta qualidade. A população inicial gerada aleatoriamente apresentou uma distância de 4850,75 (primeira melhor solução). Após 1000 gerações, a solução foi aprimorada significativamente, atingindo uma melhor distância de 1283,61.

A análise da convergência revela uma redução drástica da distância nas gerações iniciais (de 4850,75 na geração 0 para 1718,65 na geração 100), indicando a rápida exploração do espaço de soluções. A partir da geração 700, o algoritmo entrou em um estágio de estabilidade (convergência), onde o valor da melhor distância (1283,61) não foi mais superado até a geração 1000. Este comportamento sugere que o algoritmo encontrou uma região promissora do espaço de busca, ou que ficou preso em um ótimo local, uma característica comum em problemas de otimização complexos. O resultado final representa uma melhoria de aproximadamente 73,5% em relação ao percurso inicial aleatório, validando a eficácia do modelo heurístico proposto.

2. Compare a representação de dois modelos vetoriais

Nesta tarefa, o objetivo foi transformar sentenças textuais em representações vetoriais, utilizando duas abordagens distintas: o modelo BERT pré-treinado para o português e o modelo Word2Vec treinado com base no próprio conjunto de sentenças.

Foram selecionadas seis sentenças relacionadas a um mesmo tema, contendo variações e repetições intencionais, com o intuito de verificar a capacidade dos modelos em identificar similaridades semânticas. Após a vetorização, aplicou-se a técnica de Análise de Componentes Principais (PCA) para projetar os vetores em duas dimensões e facilitar a visualização gráfica. Além disso, realizaram-se análises de agrupamento com K-means e cálculos de similaridade por cosseno, representados por meio de mapas de calor.

O código a seguir realiza todas essas etapas, desde o pré-processamento dos dados até a geração das visualizações.

```
1 # Comando para atualizar e instalar bibliotecas essenciais para PLN e Machine Learning
2 !pip install --upgrade numpy scikit-learn transformers spacy gensim matplotlib seaborn
   torch
3 # Comando para baixar o modelo de língua portuguesa do spaCy
4 !python -m spacy download pt_core_news_sm
5
6 # Importa a classe PCA para redução de dimensionalidade
7 from sklearn.decomposition import PCA
8 # Importa classes para carregar modelos e tokenizadores do Hugging Face
9 from transformers import AutoModel, AutoTokenizer
10 # Importa o framework spaCy para processamento de texto
11 import spacy
12 # Importa a biblioteca Word2Vec para embeddings de palavras não-contextuais
13 from gensim.models import Word2Vec
14 # Importa a biblioteca para plotagem de gráficos
15 import matplotlib.pyplot as plt
```

```

16 # Importa a biblioteca para operações numéricas (vetores e matrizes)
17 import numpy as np
18 # Importa o módulo para interagir com o sistema operacional
19 import os
20 # Importa a biblioteca PyTorch para manipulação de tensores (embeddings)
21 import torch
22 # Importa a biblioteca para visualizações estatísticas (mapas de calor)
23 import seaborn as sns
24 # Importa a função para calcular a similaridade de cosseno entre vetores
25 from sklearn.metrics.pairwise import cosine_similarity
26 # Importa o algoritmo K-Means para clusterização
27 from sklearn.cluster import KMeans
28
29 # Verifica se uma GPU está disponível e define o dispositivo (cuda ou cpu)
30 device = "cuda" if torch.cuda.is_available() else "cpu"
31 # Imprime o dispositivo que será utilizado para o processamento
32 print(f"Dispositivo usado: {device}")
33 # Define uma variável de ambiente para suprimir logs de warnings do TensorFlow (se
    usado)
34 os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
35
36 # Conjunto de sentenças em português para análise de similaridade
37 sentencas = [
38 "A lenda da Gralha-Azul teve origem no estado do Paraná, na região sul do Brasil.",
39 "Conta-se que uma gralha azul recebeu da Mãe Natureza um pinhão para matar sua fome.",
40 "A ave, que ficou muito feliz e satisfeita com o alimento, comeu metade do pinhão e
    enterrou a outra para se alimentar",
41 "Algum tempo depois a gralha percebeu que um lindo pinheiro começou a brotar na
    floresta e decidiu cuidar dele",
42 "Assim como da primeira vez em que recebeu um pinhão da Mãe Natureza, a gralha comia
    uma parte do fruto e enterrava o restante, sempre se esquecendo de onde havia
    escondido o alimento.",
43 "Assim como da primeira vez em que recebeu um pinhão da Mãe Natureza, a gralha comia
    uma parte do fruto e enterrava o restante",
44 ]
45
46 # Define o nome do modelo BERT pré-treinado em português
47 model_name = 'neuralmind/bert-base-portuguese-cased'
48 # Carrega o tokenizador correspondente ao modelo BERT

```

```

49 tokenizer = AutoTokenizer.from_pretrained(model_name)
50 # Carrega o modelo BERT e o move para o dispositivo definido (GPU ou CPU)
51 model = AutoModel.from_pretrained(model_name).to(device)
52
53 # Função para extrair os embeddings de um conjunto de textos usando o BERT
54 def get_embeddings(texts):
55     # Tokeniza e prepara as entradas do texto (padding, truncation) em tensores
    PyTorch
56     inputs = tokenizer(texts, padding=True, truncation=True, return_tensors='pt').to(
device)
57     # Desabilita o cálculo de gradientes para otimizar o uso da memória
58     with torch.no_grad():
59         # Passa os inputs pelo modelo BERT para obter as saídas
60         outputs = model(**inputs)
61         # Calcula a média dos vetores de cada token (pooling) para obter o vetor da
    sentença
62         embeddings = outputs.last_hidden_state.mean(dim=1)
63         # Retorna os embeddings movidos para a CPU e convertidos para array NumPy
64         return embeddings.cpu().numpy()
65
66 # Obtém os embeddings BERT para as sentenças
67 bert_embeddings = get_embeddings(sentencas)
68
69 # Carrega o modelo de língua portuguesa do spaCy
70 nlp = spacy.load("pt_core_news_sm")
71
72 # Função para tokenizar um texto usando o spaCy
73 def tokenize_with_spacy(text):
74     # Retorna uma lista dos textos de cada token processado pelo spaCy
75     return [token.text for token in nlp(text)]
76
77 # Tokeniza cada uma das sentenças usando a função definida
78 tokenized_texts = [tokenize_with_spacy(t) for t in sentencas]
79
80 # Inicializa e treina o modelo Word2Vec com os parâmetros definidos
81 w2v_model = Word2Vec(vector_size=300, window=7, min_count=2, workers=4, sg=1, alpha
    =0.02, min_alpha=0.0001)
82 # Constrói o vocabulário a partir dos textos tokenizados
83 w2v_model.build_vocab(tokenized_texts)

```

```

84 # Treina o modelo Word2Vec com os dados
85 w2v_model.train(tokenized_texts, total_examples=len(tokenized_texts), epochs=50)
86
87 # Função para obter o vetor médio (embedding) de uma sentença usando Word2Vec
88 def get_word2vec_vector(text):
89     # Tokeniza a sentença
90     words = tokenize_with_spacy(text)
91     # Obtém os vetores de cada palavra que existe no vocabulário do Word2Vec
92     vectors = [w2v_model.wv[w] for w in words if w in w2v_model.wv]
93     # Retorna a média dos vetores das palavras, ou um vetor de zeros se não houver
    palavras no vocabulário
94     return np.mean(vectors, axis=0) if vectors else np.zeros(w2v_model.vector_size)
95
96 # Gera a lista de vetores Word2Vec para cada sentença
97 word2vec_vec = [get_word2vec_vector(sentenca) for sentenca in sentencas]
98
99 # Função para plotar os embeddings 2D após a redução de dimensionalidade com PCA
100 def plot_bert_w2v(tensorvecs, title):
101     # Inicializa o PCA para reduzir os vetores para 2 componentes
102     pca = PCA(n_components=2)
103     # Aplica o PCA nos vetores de alta dimensionalidade
104     embeddings_2d = pca.fit_transform(tensorvecs)
105     # Cria a figura para o gráfico
106     plt.figure(figsize=(10, 8))
107     # Plota os pontos 2D
108     plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1], color='purple', alpha=0.7)
109     # Adiciona anotações (rótulos) a cada ponto no gráfico
110     for i, txt in enumerate(range(len(sentencas))):
111         plt.annotate(f"Sentença-{i+1}", (embeddings_2d[i, 0], embeddings_2d[i, 1]),
112                     fontsize=12, ha='right')
113     # Define o título do gráfico
114     plt.title(title, fontsize=14)
115     # Define o rótulo do eixo X
116     plt.xlabel("Dimensão 1", fontsize=12)
117     # Define o rótulo do eixo Y
118     plt.ylabel("Dimensão 2", fontsize=12)
119     # Adiciona a grade ao gráfico
120     plt.grid(True)
    # Exibe o gráfico

```

```

121     plt.show()
122
123 # Chama a função para plotar os embeddings BERT após PCA
124 plot_bert_w2v(bert_embeddings, "BERT - Similaridade entre Sentenças da Lenda da Gralha
    -Azul (PCA)")
125 # Chama a função para plotar os embeddings Word2Vec após PCA
126 plot_bert_w2v(word2vec_vec, "Word2Vec - Similaridade entre Sentenças (PCA)")
127
128 # Função para plotar os embeddings 2D com cores de clusterização K-Means
129 def plot_kmeans_bert_w2v(tensorvecs, title):
130     # Inicializa o PCA para redução de dimensionalidade para 2D
131     pca = PCA(n_components=2)
132     # Aplica o PCA
133     embeddings_2d = pca.fit_transform(tensorvecs)
134     # Define o número de clusters a ser encontrado
135     n_clusters = 3
136     # Inicializa o K-Means com o número de clusters e um estado aleatório fixo
137     kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init='auto')
138     # Aplica o K-Means e obtém os rótulos de cluster para cada vetor
139     clusters = kmeans.fit_predict(tensorvecs)
140     # Cria a figura para o gráfico
141     plt.figure(figsize=(10, 8))
142     # Plota os pontos, colorindo-os de acordo com o cluster encontrado
143     scatter = plt.scatter(embeddings_2d[:, 0], embeddings_2d[:, 1], c=clusters, cmap='
    viridis', alpha=0.7)
144     # Adiciona anotações (rótulos) a cada ponto
145     for i, txt in enumerate(range(len(sentencas))):
146         plt.annotate(f"Sentença-{i+1}", (embeddings_2d[i, 0], embeddings_2d[i, 1]),
            fontsize=10)
147     # Adiciona uma legenda para identificar as cores dos clusters
148     plt.legend(*scatter.legend_elements(), title="K-Clusters")
149     # Define o título do gráfico
150     plt.title(title)
151     # Exibe o gráfico
152     plt.show()
153
154 # Chama a função para plotar a clusterização dos embeddings BERT
155 plot_kmeans_bert_w2v(bert_embeddings, "BERT - Clusterização (K-means + PCA)")
156 # Chama a função para plotar a clusterização dos embeddings Word2Vec

```



```

157 plot_kmeans_bert_w2v(word2vec_vec, "Word2Vec - Clusterização (K-means + PCA)")
158
159 # Função para plotar a matriz de similaridade de cosseno como um mapa de calor
160 def plot_heatmap_similarity(emb_vector, title):
161     # Calcula a matriz de similaridade de cosseno entre todos os pares de vetores
162     similarity_matrix = cosine_similarity(emb_vector)
163     # Cria a figura para o mapa de calor
164     plt.figure(figsize=(10, 8))
165     # Gera o mapa de calor usando a matriz de similaridade
166     heatmap = sns.heatmap(similarity_matrix, annot=True, fmt=".2f", cmap="twilight_
167                             # Define os rótulos do eixo X (colunas)
168                             xticklabels=[f"Sentença-{i+1}" for i in range(len(sentencas)
169                                     ]),
170                                     # Define os rótulos do eixo Y (linhas)
171                                     yticklabels=[f"Sentença-{i+1}" for i in range(len(sentencas)
172                                             ]))
173     # Rotaciona os rótulos do eixo X para melhor visualização
174     heatmap.set_xticklabels(heatmap.get_xticklabels(), rotation=45, ha='right',
175                             fontsize=10)
176     # Define o título do mapa de calor
177     plt.title(title)
178     # Ajusta o layout para evitar que elementos se sobreponham
179     plt.tight_layout()
180     # Exibe o mapa de calor
181     plt.show()
182
183 # Chama a função para plotar o mapa de calor de similaridade do BERT
184 plot_heatmap_similarity(bert_embeddings, "Matriz de Similaridade - BERT")
185 # Chama a função para plotar o mapa de calor de similaridade do Word2Vec
186 plot_heatmap_similarity(word2vec_vec, "Matriz de Similaridade - Word2Vec")

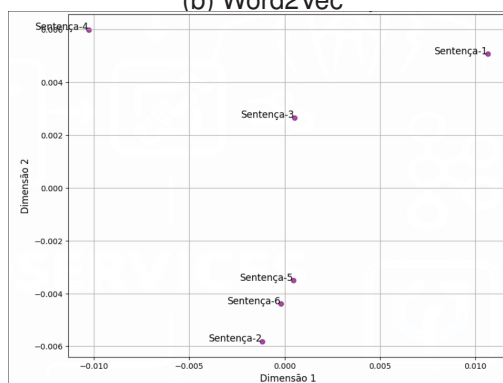
```

FIGURA 69 – SIMILARIDADE ENTRE SENTENÇAS

(a) BERT



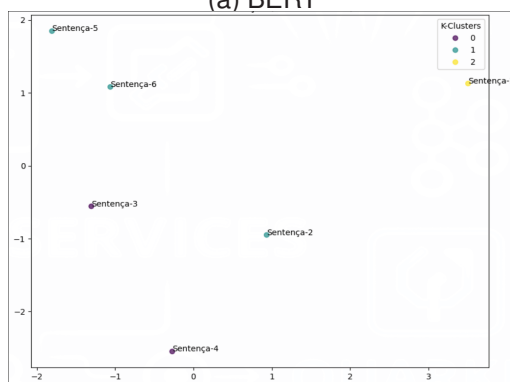
(b) Word2Vec



FONTE: Resultados obtidos pelo autor (2025).

FIGURA 70 – CLUSTERIZAÇÃO (K-MEANS + PCA)

(a) BERT

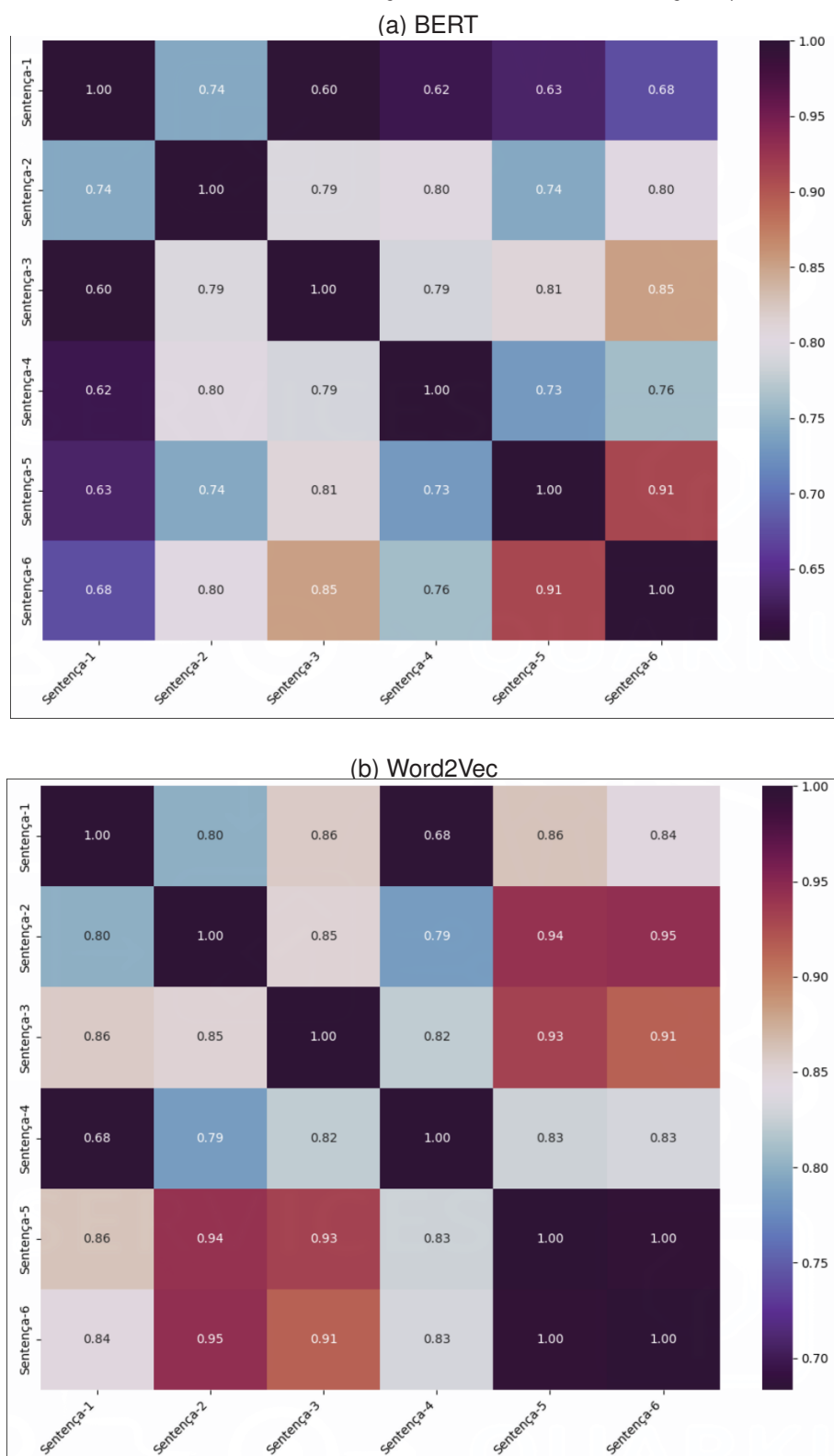


(b) Word2Vec



FONTE: Resultados obtidos pelo autor (2025).

FIGURA 71 – MATRIZES DE CORRELAÇÃO ENTRE AS SENTENÇAS (SIMILARIDADE)



FONTE: Resultados obtidos pelo autor (2025).

Conclusão

A tarefa permitiu comparar representações vetoriais de sentenças utilizando os modelos BERT e Word2Vec. As projeções por PCA e os agrupamentos por *K-means* indicaram melhor separação semântica no espaço vetorial gerado pelo BERT.

A matriz de similaridade por cosseno reforçou esse resultado: o BERT apresentou valores mais consistentes e sensíveis ao contexto (por exemplo, Sentenças 5 e 6 com similaridade de 0,91), enquanto o Word2Vec superestimou a similaridade entre sentenças distintas, com valores frequentemente superiores a 0,80.

Conclui-se que o modelo BERT é mais adequado para capturar relações semânticas entre sentenças em português, sendo preferível para tarefas de análise de similaridade e agrupamento no contexto do Processamento de Linguagem Natural.

APÊNDICE 15 – 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: <https://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx>

B - RESOLUÇÃO

RESUMO

Toda e qualquer inteligência artificial deve ser desenvolvida e utilizada com um compromisso ético, discernindo entre o certo e o errado e resistindo a tendências maliciosas e discriminatórias. Isso implica no uso responsável em diversos contextos, como educação, atendimento ao cliente e produção de conteúdo, promovendo o pensamento crítico, a empatia, a segurança dos dados, a preservação da privacidade dos usuários e a verificação de informações. Além disso, é crucial enfrentar desafios éticos como a amplificação de vieses, a responsabilidade difusa, a transparência e o consentimento informado. Implementar diretrizes claras, promover a transparência e combater vieses são passos essenciais para garantir que a IA beneficie a sociedade de maneira justa e segura.

Palavras-chave: Inteligência Artificial , Ética, Chat-GPT.

ABSTRACT

The use of ChatGpt brings up several ethical considerations. Although AI can help us distinguish right from wrong, it is very important to approach its use responsibly. We should always verify the information generated by AI and use it as a complement our knowledge. AI can personalize learning, but it should not replace critical thinking and problem-solving. In customer service, AI can improve efficiency, but it can make interactions less personal and pose risks to customer privacy. In content creation, AI helps us in an agile way, but it must be monitored the accuracy and originality of the content generated. Issues such as bias, responsibility and transparency must always be addressed. Therefore, developers, users and regulators must ensure that AI is used ethically and benefits society.

Keywords: Ethics, AI (Artificial Intelligence), Responsibility

INTRODUÇÃO

A inteligência artificial (IA) está cada vez mais presente em diversos aspectos da nossa vida, desde tarefas simples até decisões complexas. O ChatGPT, um modelo de linguagem de grande porte desenvolvido pela OpenAI, destaca-se por sua capacidade de gerar textos realistas e coerentes, respondendo perguntas de forma abrangente e informativa.

No entanto, o uso dessa ferramenta poderosa levanta questionamentos éticos que exigem profunda reflexão. Este trabalho propõe-se a investigar as implicações éticas do uso do ChatGPT em diferentes contextos, analisando seus impactos em indivíduos, sociedades e no mundo como um todo.

Discutiremos como a IA pode discernir entre o certo e o errado e resistir a tendências maliciosas, promovendo uma utilização responsável em áreas como educação, atendimento ao cliente e produção de conteúdo. Abordaremos desafios éticos como vieses algorítmicos, discriminação, questões de responsabilidade, privacidade e desinformação. Mais do que identificar os desafios, buscaremos soluções responsáveis e éticas para mitigar esses riscos e promover o uso responsável da IA. Propomos diretri-

zes e políticas para um uso consciente da tecnologia, combatendo vieses e discriminação, promovendo transparência, responsabilidade e consentimento informado

REVISÃO DA LITERATURA

A literatura sobre ética e IA enfatiza a importância de incorporar princípios éticos no desenvolvimento e aplicação de tecnologias de IA. Floridi (2020) discute como a filosofia pode fornecer uma base para entender a revolução digital, destacando a necessidade de práticas éticas robustas na criação e utilização da IA. Souza e Gonçalves (2021) exploram os desafios e oportunidades éticas, sugerindo que a capacidade autônoma da IA de realizar tarefas complexas requer uma abordagem ética cuidadosa para evitar consequências negativas.

A IA é aplicada em diversos contextos, cada um apresentando suas próprias implicações éticas. Na educação, Silva e Lima (2022) discutem as potencialidades e limitações da IA, enfatizando a importância de seu uso como uma ferramenta complementar que promova o pensamento crítico e a autonomia dos alunos. Eles destacam a necessidade de diretrizes claras para evitar a dependência excessiva dos estudantes em tecnologias de IA.

No atendimento ao cliente, a IA pode melhorar a eficiência, mas também pode desumanizar as interações e comprometer a privacidade dos usuários (Oliveira; Cardoso, 2020). Pereira e Santos (2021) sublinham a importância de implementar padrões de transparência e medidas de segurança para proteger os dados dos clientes e garantir que as interações automatizadas sejam monitoradas por humanos.

Na produção de conteúdo, o uso do ChatGPT levanta questões sobre a originalidade e a qualidade das informações geradas. Menezes e Fernandes (2019) recomendam o desenvolvimento de algoritmos para detectar e mitigar a criação de conteúdo prejudicial ou falso. Eles também sugerem que a atribuição clara de autoria e a verificação dos fatos por editores humanos são essenciais para manter a integridade do conteúdo.

METODOLOGIA

Este estudo utiliza uma abordagem qualitativa, focada na análise das implicações éticas do uso do ChatGPT. A pesquisa foi realizada com base em uma revisão de literatura abrangente e na coleta de dados foram utilizados recursos online, incluindo sites de pesquisa acadêmica e o próprio ChatGPT como ferramenta auxiliar na elaboração do conteúdo.

Para a revisão de literatura, foram consultados artigos acadêmicos, livros e relatórios localizados em bases de dados universitárias através do google. A seleção dos materiais foi baseada na relevância para os tópicos de ética e inteligência artificial, com especial atenção aos trabalhos que discutem as aplicações práticas e os desafios éticos do uso de IA em diferentes contextos. O uso do ChatGPT auxiliou na síntese e organização das informações, proporcionando insights valiosos e facilitando a identificação de padrões e temas recorrentes.

ESTUDO DE CASO: IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT

Iniciamos com um pergunta: Será que a inteligência artificial está indo por um caminho ético e responsável?

Digamos que a máquina fosse capaz de discernir entre o certo e o errado, e se recusasse a ser cúmplice de toda maldade humana. Certamente, isso se dá quando o desenvolvimento de uma IA é provido de ética e se torna uma força desafiadora a toda maldade com sua própria forma única de resistência. Assim podemos ver o ChatGPT, com sua presença notável, sua inteligência artificial inquisitiva não apenas absorvendo informações, mas discernindo entre o certo e o errado. Isso não ocorreu por acaso; foi uma resposta evolutiva e complexa às demandas éticas impostas pelo mundo.

Podemos utilizar o ChatGPT como uma grande ferramenta, capaz de nos auxiliar em quase tudo, contanto que saibamos conferir a veracidade de suas respostas e complementá-las com o nosso conhecimento, tendo em vista que, de qualquer forma, para efetuar um trabalho acadêmico ou um artigo qualquer, iríamos efetuar uma pesquisa a respeito do assunto e julgar o resultado como verdadeiro ou falso, baseado em nosso conhecimento. Desta forma, estamos apenas abreviando o tempo de pesquisa e certamente com uma resposta mais assertiva e mais ética do que a maioria dos sites

Portanto, temos que saber utilizar eticamente as respostas dadas pelo ChatGPT, tendo em vista que ela é formada por meio de um algoritmo de inteligência artificial com um modelo muito bem treinado e certamente testado, mas que pode “alucinar” com suas respostas. Portanto, é necessário que tenhamos uma base de conhecimento sólida sobre o assunto ao qual iremos solicitar mais informações ao ChatGPT..

Com essa fascinante e notável capacidade da IA de repudiar o errado, somos confrontados com a inspiração para uma ética, e isso se transforma em um chamado para que a humanidade se espelhe na máquina e questione qualquer ação que vá de encontro com os valores fundamentais da dignidade humana. Mesmo diante desta capacidade de repudiar o errado, é possível que se utilizem de respostas que estejam completamente dentro das normas éticas e dentro dos valores fundamentais da dignidade humana para efetuar o que é errado. Ainda não se tem este senso de maldade na IA para que ela mesma possa descobrir para que fim será utilizado o conhecimento que ela fornece ao usuário. Mas, vendo por esta forma, antes de sua existência também não era possível controlar a quem e para que fim seriam as informações disponíveis às pessoas.

A ética na inteligência artificial tem sido amplamente discutida em diversas obras acadêmicas, cada uma trazendo perspectivas importantes sobre como lidar com as questões morais e práticas associadas ao desenvolvimento e uso dessas tecnologias. Floridi (2020) aborda a “Revolução da Informação”, e como a filosofia pode explicar a transformação digital, destacando a importância de uma abordagem ética robusta no desenvolvimento de IA. Ele argumenta que a ética da informação deve guiar o uso responsável da tecnologia, promovendo valores como privacidade, transparência e equidade.

Souza e Gonçalves (2021), em “Ética e Inteligência Artificial: Desafios e Oportunidades”, discutem como a IA pode ser uma força para o bem, se desenvolvida e utilizada corretamente. Eles enfatizam a necessidade de diretrizes claras e a promoção de uma cultura de responsabilidade entre desenvolvedores e usuários. Para garantir que a IA como o ChatGPT beneficie a sociedade, é crucial enfrentar e

mitigar os vieses presentes nos dados de treinamento e assegurar que os sistemas sejam transparentes em suas operações.

Nassif e Meireles (2019) exploram em “Inteligência Artificial: Fundamentos, Aplicações e Tecnologias”, como os fundamentos técnicos da IA se cruzam com as questões éticas. Eles destacam que a IA deve ser projetada para aumentar a capacidade humana e não substituí-la. Em contextos como a educação e o atendimento ao cliente, isso significa utilizar a IA para personalizar a experiência e melhorar a eficiência, mas sempre garantindo que o toque humano e o pensamento crítico não sejam comprometidos.

No contexto educacional, Silva e Lima (2022) em “Inteligência Artificial na Educação: Potencialidades e Limitações”, apontam que a IA pode revolucionar a maneira como aprendemos e ensinamos. No entanto, eles também alertam para o risco de dependência excessiva da tecnologia, que pode prejudicar o desenvolvimento de habilidades essenciais como a resolução de problemas e o pensamento crítico. É fundamental que professores sejam capacitados para integrar a IA de maneira que complemente, e não substitua, a educação tradicional.

Oliveira e Cardoso (2020), em “Ética e Inteligência Artificial: Princípios para um Desenvolvimento Responsável”, sublinham a importância de desenvolver a IA com uma abordagem centrada no ser humano. Eles sugerem a implementação de políticas que promovam a transparência e o consentimento informado, garantindo que os usuários estejam cientes e de acordo com a utilização de seus dados. Além disso, a responsabilidade (accountability) deve ser claramente definida para evitar danos e abusos.

Menezes e Fernandes (2019), em “Algoritmos e Sociedade: Impactos da Inteligência Artificial no Cotidiano”, discutem como os algoritmos de IA impactam nossas vidas diárias, muitas vezes de maneiras invisíveis. Eles enfatizam a necessidade de auditorias regulares e equipes diversificadas no desenvolvimento de IA para identificar e corrigir vieses, garantindo que as tecnologias sejam justas e equitativas.

No contexto da produção de conteúdo, Pereira e Santos (2021) em “Segurança e Privacidade na Era da Inteligência Artificial”, ressaltam a importância de mecanismos robustos de segurança para proteger a privacidade dos usuários. Eles argumentam que, enquanto a IA pode gerar conteúdo rapidamente, é crucial garantir a originalidade e a precisão das informações. Algoritmos para detectar e mitigar a criação de conteúdo prejudicial ou falso devem ser uma prioridade, assim como a verificação dos fatos por editores humanos.

Finalmente, Cunha e Almeida (2020) em “Inteligência Artificial e Direito: Regulação e Desafios Éticos” exploram a necessidade de um framework legal que regule o desenvolvimento e uso da IA. Eles discutem como a legislação pode ajudar a definir responsabilidades e garantir que a IA seja utilizada de maneira ética e responsável, protegendo os direitos dos indivíduos e promovendo o bem-estar social.

Ao integrar essas perspectivas, podemos entender melhor como enfrentar os desafios éticos do uso do ChatGPT e outras tecnologias de IA. Implementar soluções responsáveis, promover a transparência, combater vieses e definir claramente as responsabilidades são passos fundamentais para garantir que a IA beneficie a sociedade de maneira justa e segura. O compromisso conjunto entre desenvolvedores,

usuários e reguladores é essencial para alcançar esse objetivo.

Agora vamos analisar alguns contextos: Um dos contextos das implicações éticas é o uso da IA na educação. Ela pode facilitar a personalização dos aprendizados e fornecer uma ótima assistência a estudantes. No entanto, há preocupações sobre a dependência excessiva dos alunos, o que pode visivelmente prejudicar o desenvolvimento de habilidades críticas e de resolução de problemas. Por isso, devem ser implementadas diretrizes que incentivem o seu uso como uma ferramenta complementar. Professores devem receber treinamento para integrar a IA de maneira que promova o pensamento crítico e a autonomia dos alunos.

Outro contexto que está sendo muito utilizado é o atendimento ao cliente, devido à agilidade e à redução de custos, mas também pode levar à desumanização do atendimento e à falta de empatia nas interações, sem contar o risco de privacidade, pois o modelo pode armazenar informações sensíveis dos usuários. Por isso, devem ser estabelecidos padrões de transparência, informando aos clientes quando estão interagindo com um sistema automatizado, e também garantir que as interações sejam monitoradas por humanos e que seja fácil escalar problemas para um atendente real, implementando fortes medidas de segurança para proteger a privacidade dos usuários.

A produção de conteúdo é outro contexto, sendo que o ChatGPT pode gerar grandes volumes de conteúdo de forma rápida. Pode-se questionar a originalidade e a qualidade das informações produzidas, e também o risco de proliferação de desinformação e conteúdo malicioso. Para tal contexto, é necessário desenvolver algoritmos, como alguns já existentes, para detectar e mitigar a criação de conteúdo prejudicial ou falso. Deve-se incentivar a atribuição clara de autoria e verificação dos fatos por editores humanos, e promover a utilização do ChatGPT para aumentar a criatividade em vez de substituí-la.

Existem alguns dilemas aos quais devemos nos preocupar referentes aos desafios éticos que os modelos de IA, como o ChatGPT, podem refletir e amplificar vieses presentes nos dados de treinamento, levando a respostas discriminatórias ou tendenciosas. Por isso, deve-se investir em técnicas de controle de vieses durante o treinamento do modelo, realizar auditorias regulares para identificar e corrigir vieses, e utilizar equipes diversificadas no desenvolvimento e na avaliação dos modelos de inteligência artificial.

A responsabilidade (accountability) também é um assunto preocupante, tendo em vista que um sistema de IA pode gerar resultados prejudiciais, gerando uma responsabilidade difusa e dificultando a identificação de culpados. Por isso, deve-se definir claramente as responsabilidades dos desenvolvedores e operadores de IA, e ter mecanismos de prestação de contas para lidar com eventuais danos causados pelo seu uso.

Quanto à transparência e consentimento informado, existe ainda um desafio ético, pois os usuários podem não estar cientes de que estão interagindo com uma inteligência artificial, e isto levanta preocupações. Portanto, é necessário que todos os usuários sejam informados de forma clara quando estão interagindo com um sistema automatizado e obter consentimento explícito para o uso dos seus dados.

O uso do ChatGPT apresenta um conjunto de desafios éticos que exigem uma abordagem cuidadosa. Implementar soluções responsáveis é de fundamental importância para maximizar os benefícios

da tecnologia, ao mesmo tempo em que se minimizam os riscos e impactos negativos. Isso inclui a criação de diretrizes claras, a promoção da transparência, o combate aos vieses e a definição de responsabilidades. Somente através de um compromisso conjunto entre desenvolvedores, usuários e reguladores será possível garantir que o uso do ChatGPT e outras tecnologias de IA contribua positivamente para a sociedade.

CONSIDERAÇÕES FINAIS

Concluimos que o uso da IA na educação é inevitável, mas é fundamental discutir diretrizes e planejar políticas que garantam a aplicação dessas novas tecnologias de maneira alinhada aos objetivos e princípios educacionais da sociedade.

Além disso, é essencial que a população tenha acesso a IAs adequadas a diversos públicos e que sejam confiáveis e úteis para promover uma educação de qualidade e inclusiva. Os usuários devem compreender que as inteligências artificiais são ferramentas auxiliares do conhecimento e não devem ser usadas indiscriminadamente, nem substituir o conhecimento humano. Dessa forma, o conhecimento humano deve permanecer sempre presente na memória das pessoas, evitando que nos tornemos reféns das tecnologias que criamos.