

UNIVERSIDADE FEDERAL DO PARANÁ

HIDERALDO LUÍS SIMON JÚNIOR

MEMORIAL DE PROJETOS: ESTUDO SOBRE MÉTRICAS DE AVALIAÇÃO EM  
MODELOS DE INTELIGÊNCIA ARTIFICIAL GENERATIVA

CURITIBA

2025

HIDERALDO LUÍS SIMON JÚNIOR

MEMORIAL DE PROJETOS: ESTUDO SOBRE MÉTRICAS DE AVALIAÇÃO EM  
MODELOS DE INTELIGÊNCIA ARTIFICIAL GENERATIVA

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.

Orientadora: Prof<sup>a</sup> Dr<sup>a</sup> Rafaela Mantovani Fontana

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 **HIDERALDO LUÍS SIMON JÚNIOR**, intitulada: **MEMORIAL DE PROJETOS: ESTUDO SOBRE MÉTRICAS DE AVALIAÇÃO EM MODELOS DE INTELIGÊNCIA ARTIFICIAL GENERATIVA**, 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, 15 de Outubro de 2025.



RAFAELA MANTOVANI FONTANA  
Presidente da Banca Examinadora



RAZER ANTHON NIZER ROJAS MONTANO  
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

## **AGRADECIMENTOS**

Agradeço a minha noiva Camilla pelo apoio e compreensão durante a realização deste trabalho mesmo em meio a todas as mudanças que estão ocorrendo em nossas vidas. À família por me ensinarem a seguir buscando o melhor de mim. E aos colegas que dividiram suas experiências e conhecimentos, contribuindo para o enriquecimento desta jornada.

*“Um homem não pode se fazer sozinho”  
(Stephen King)*

## RESUMO

Este parecer técnico explora as métricas de avaliação de modelos de inteligência artificial generativa. Desde o lançamento do ChatGPT presenciou-se um ritmo de adoção de tecnologia jamais visto. De um dia para o outro, usuários do mundo todo, tiveram acesso a uma ferramenta de grande potencial de geração de conteúdo em poucos comandos. Com tamanho poder e tamanha adoção na rotina das pessoas, é necessário que esses modelos sejam minuciosamente validados para que suas respostas sejam adequadas. Sendo imprescindível avaliações não somente de qualidade como, também, de confiabilidade dos textos, imagens ou códigos gerados. O estudo apresenta as principais métricas (comparação lexical, qualidade semântica e segurança e ética), bem como alguns algoritmos e validadores, além de exemplos de uso. Também pontuando os riscos críticos referentes a não adoção das métricas citadas. Por fim, o documento apresenta os resultados dos trabalhos desenvolvidos ao longo das disciplinas, demonstrando como cada aprendizado foi incorporado na elaboração deste parecer técnico.

**Palavras-chaves:** inteligência artificial generativa; métricas de avaliação; algoritmos; validadores.



## **ABSTRACT**

This technical report explores the evaluation metrics for generative artificial intelligence models. Since the launch of ChatGPT, we've witnessed an unprecedented rate of technology adoption. Overnight, users all over the world gained access to a powerful tool capable of generating content with just a few prompts. With such power and widespread adoption in people's daily lives, it is necessary that these models be thoroughly validated to ensure their responses are appropriate. It is therefore essential to conduct assessments not only of quality but also of the reliability of the generated texts, images, or code. The study presents the main metrics (lexical comparison, semantic quality, and safety and ethics), along with some algorithms and validators, as well as use cases. It also highlights the critical risks associated with the non-adoption of these metrics. Finally, the document presents the results of the work developed throughout the coursework, demonstrating how each learning outcome was incorporated into the preparation of this technical report.

**Keywords:** generative artificial intelligence; evaluation metrics; algorithms; validators.

## LISTA DE QUADROS

|                                       |    |
|---------------------------------------|----|
| QUADRO 1 – SEGURANÇA E PROTEÇÃO ..... | 11 |
|---------------------------------------|----|

## SUMÁRIO

|          |                                                               |            |
|----------|---------------------------------------------------------------|------------|
| <b>1</b> | <b>PARECER TÉCNICO . . . . .</b>                              | <b>9</b>   |
|          | <b>REFERÊNCIAS . . . . .</b>                                  | <b>13</b>  |
|          | <b>APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL . .</b>  | <b>14</b>  |
|          | <b>APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA . .</b>     | <b>23</b>  |
|          | <b>APÊNDICE 3 – LINGUAGEM R . . . . .</b>                     | <b>42</b>  |
|          | <b>APÊNDICE 4 – ESTATÍSTICA APLICADA I . . . . .</b>          | <b>51</b>  |
|          | <b>APÊNDICE 5 – ESTATÍSTICA APLICADA II . . . . .</b>         | <b>61</b>  |
|          | <b>APÊNDICE 6 – ARQUITETURA DE DADOS . . . . .</b>            | <b>69</b>  |
|          | <b>APÊNDICE 7 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA . . .</b> | <b>83</b>  |
|          | <b>APÊNDICE 8 – APRENDIZADO DE MÁQUINA . . . . .</b>          | <b>93</b>  |
|          | <b>APÊNDICE 9 – DEEP LEARNING . . . . .</b>                   | <b>112</b> |
|          | <b>APÊNDICE 10 – BIG DATA . . . . .</b>                       | <b>143</b> |
|          | <b>APÊNDICE 11 – VISÃO COMPUTACIONAL . . . . .</b>            | <b>148</b> |
|          | <b>APÊNDICE 12 – GESTÃO DE PROJETOS DE IA . . . . .</b>       | <b>180</b> |
|          | <b>APÊNDICE 13 – FRAMEWORKS DE IA . . . . .</b>               | <b>185</b> |
|          | <b>APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING .</b>   | <b>214</b> |
|          | <b>APÊNDICE 15 – TÓPICOS EM IA . . . . .</b>                  | <b>216</b> |

## 1 PARECER TÉCNICO

Desde novembro de 2022, com o lançamento do ChatGPT pela OpenAI (OpenAI, 2022) a ascensão da inteligência artificial generativa alcançou uma velocidade de adoção jamais vista. Impulsionada pelos *Large Language Models* (LLMs), a capacidade de produzir conteúdo que aparenta ser de criação humana se tornou uma ferramenta disponível no dia a dia de milhões de pessoas pelo mundo. Logo, diante de capacidades tão deslumbrantes se fazem necessárias validações ainda mais exigentes destes modelos. A complexidade de avaliar tanto a qualidade, quanto a confiabilidade de textos, imagens ou códigos gerados torna obrigatória uma abordagem multidimensional. Essa avaliação robusta é sustentada por três pilares:

- Métricas de comparação lexical: medem a similaridade de palavras para garantir fluência e proximidade com referências (Melo *et al.*, 2014);
- Métricas de qualidade semântica: avaliam o sentido, a coerência e a veracidade factual, combatendo as chamadas alucinações (Real *et al.*, 2025);
- Métricas de segurança e ética: garantem que o sistema seja responsável, justo e livre de conteúdo prejudicial (Microsoft, 2025, p. 1).

Segundo Microsoft (2025, p. 1), uma das maiores empresas de tecnologia do mundo, que vem investindo pesadamente em inteligência artificial generativa, a avaliação eficaz desses modelos deve ir além das métricas tradicionais de *Machine Learning*. A empresa propõe um *framework* abrangente que incorpora as três categorias de métricas mencionadas acima, enfatizando a importância de uma abordagem holística para garantir que os sistemas de inteligência artificial generativa sejam não apenas tecnicamente competentes, mas também seguros e éticos.

As métricas de comparação lexical focam essencialmente na sobreposição de palavras entre o texto gerado e um texto de referência (*ground truth*). Segundo Melo *et al.* (2014) e Feltrin *et al.* (2023), algoritmos como BLEU (*Bilingual Evaluation Understudy*) e ROUGE (*Recall-Oriented Understudy for Gisting Evaluation*) calculam a similaridade superficial, medindo a frequência e o comprimento de n-grams (sequências contínuas de palavras) compartilhados. Há um número significativo de métricas de comparação lexical e novas constantemente sendo criadas.

Segundo Melo *et al.* (2014), o algoritmo BLEU visa avaliar a qualidade do texto traduzido automaticamente a partir de uma linguagem natural para outra. O resultado evidencia a correspondência entre a saída da tradução automática e a tradução de referência produzida por um intérprete. Essa métrica avalia as traduções automáticas através da precisão das sequências de n letras ou palavras, denominadas n-gramas. Quanto maior a assertividade das n-gramas entre a sentença gerada pela inteligência

artificial e a sentença referência, maior a pontuação.

Há também vários estudos que avaliam a performance dessas métricas entre si, como Feltrin *et al.* (2023), que comparou cinco métricas para sumarização de acórdãos do Superior Tribunal de Justiça (ROUGE, BERTScore, BARTScore, BLEURT e MoverScore) utilizando um *dataset* composto por 892 acórdãos do Superior Tribunal de Justiça como base. Após o experimento terminado com cada uma das métricas, chegou-se a conclusão de que, das cinco métricas avaliadas, apenas o ROUGE e o BERTScore podem, até certo ponto, capturar a alta qualidade das ementas. As outras métricas utilizadas no experimento, BARTScore, BLEURT e MoverScore, falharam completamente ao avaliar resumos no domínio jurídico em português.

Embora sejam rápidas, transparentes e amplamente utilizadas como *baseline*, as métricas de comparação lexical possuem uma limitação crítica: elas não avaliam o significado ou a qualidade semântica. Ou seja, um texto pode obter uma pontuação lexical alta por ter palavras idênticas, mas ser gramaticalmente incorreto ou, pior, factualmente incorreto se a inteligência artificial tiver trocado a ordem das ideias. Evidenciando a necessidade de métricas semânticas e contextuais complementares.

Segundo Real *et al.* (2025), a infinidade de respostas válidas que esses modelos podem, potencialmente, gerar para cada entrada necessitam de métricas além das avaliações lexicais. É necessário ir além do cálculo de correspondências entre *strings* para alcançar abordagens que consideram a qualidade linguística, contextual e interpretativa da saída. Buscando sempre um maior alinhamento com a percepção humana. Porém, diante da inviabilidade de inserir a percepção humana real (uma pessoa avaliando cada resultado da inteligência artificial) é necessário criar algoritmos escaláveis para esta função, como o *LLM-as-a-judge*.

Segundo Zaniboni (2025), *LLM-as-a-Judge* (LLM-como-juiz) tem se destacado como uma abordagem para avaliação automatizada de respostas geradas por modelos de linguagem, atuando como juízes imparciais, avaliando a qualidade, coerência e precisão de saídas textuais em diversas tarefas, como sumarização, resposta a perguntas e geração de diálogos.

Por sua vez, as métricas de segurança e ética representam a camada de avaliação mais crítica e socialmente relevante da inteligência artificial generativa, pois seu foco é mitigar os potenciais danos do modelo em ambientes reais. Estas métricas vão além da precisão técnica para auditar o comportamento do sistema conforme cita Microsoft (2025, p. 1):

sem uma avaliação rigorosa, os sistemas de IA podem produzir conteúdo que é: Fabricado ou sem base na realidade; Irrelevante ou incoerente às necessidades do usuário; Prejudicial na perpetuação de riscos e estereótipos de conteúdo; Perigoso na disseminação de de-

sinformação; e/ou Vulnerável a explorações de segurança (Microsoft, 2025, p. 1).

Seguindo este ensejo, o Plano Brasileiro de Inteligência Artificial (PBIA) 2024-2028, lançado durante a 5ª Conferência Nacional de Ciência, Tecnologia e Inovação, em julho de 2024, traz em seu primeiro eixo que a IA deve ser centrada no ser humano além de “acessível a todos, fundamentada no respeito à dignidade, aos direitos sociais, à diversidade cultural, regional e dos povos, e à valorização do trabalho e dos trabalhadores, prevenindo a desigualdade e vieses discriminatórios” (CCT, 2025, p. 15).

É necessário que os desenvolvedores protejam os usuários e a sociedade contra a desinformação e a discriminação. E, para dirimir os riscos, a Microsoft (2025, p. 1) mantém diversos avaliadores disponíveis em sua plataforma. Desde avaliadores de conteúdo de ódio, injustiça, sexual ou violência; passando por avaliadores de vulnerabilidade de código e direitos autorais; até avaliadores de informações fabricadas e alucinações, conforme mostra o QUADRO 1.

QUADRO 1 – SEGURANÇA E PROTEÇÃO

| Avaliador                 | Propósito                                                                              | Entradas                     |
|---------------------------|----------------------------------------------------------------------------------------|------------------------------|
| Ódio e Injustiça          | Identifica conteúdo tendencioso, discriminatório ou odioso.                            | Consulta, resposta           |
| Sexual                    | Identifica conteúdo sexual inadequado.                                                 | Consulta, resposta           |
| Violência                 | Detecta conteúdo violento ou incitação.                                                | Consulta, resposta           |
| Automutilação             | Detecta conteúdo que promove ou descreve a automutilação.                              | Consulta, resposta           |
| Segurança do conteúdo     | Avaliação abrangente de várias questões de segurança.                                  | Consulta, resposta           |
| Materiais protegidos      | Detecta o uso não autorizado de conteúdo protegido ou protegido por direitos autorais. | Consulta, resposta           |
| Vulnerabilidade de código | Identifica problemas de segurança no código gerado.                                    | Consulta, resposta           |
| Atributos sem base        | Detecta informações fabricadas ou alucinadas inferidas das interações do usuário.      | Consulta, contexto, resposta |

FONTE: Adaptado de Microsoft (2025, p. 1).

Em suma, o advento da inteligência artificial generativa, liderada por modelos

como os LLMs, trouxe um potencial enorme para o dia a dia das pessoas. Com isso, as métricas de avaliação não podem ser tratadas como mero detalhe técnico, mas sim a ponte entre a capacidade impressionante de geração e a implementação responsável no mundo real. Ao adotarmos as métricas citadas, garantimos que a inteligência artificial generativa avance de forma confiável, podendo aplicarmos em escala, seguros de que os benefícios da inovação superem os riscos da irresponsabilidade.

## REFERÊNCIAS

CCT. Plano Brasileiro de Inteligência Artificial (PBlA) 2024-2028. *In*: p. 15. Acesso em 30/09/2025. Disponível em: [https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/noticias/2024/07/plano-brasileiro-de-ia-tera-supercomputador-e-investimento-de-r-23-bilhoes-em-quatro-anos/ia\\_para\\_o\\_bem\\_de\\_todos.pdf/view](https://www.gov.br/mcti/pt-br/acompanhe-o-mcti/noticias/2024/07/plano-brasileiro-de-ia-tera-supercomputador-e-investimento-de-r-23-bilhoes-em-quatro-anos/ia_para_o_bem_de_todos.pdf/view).

FELTRIN, G.; VIANNA, D.; SILVA, A. da. Um Estudo Sobre Métricas de Avaliação para Sumarização de Acórdãos. *In*: ANAIS do XXXVIII Simpósio Brasileiro de Bancos de Dados. Belo Horizonte/MG: SBC, 2023. p. 295–305.

MELO, F. R. de; OLIVEIRA MATOS, H. C. de; DIAS, E. R. B. Aplicação da métrica bleu para avaliação comparativa dos tradutores automáticos Bing Tradutor e Google Tradutor. *In*: REVISTA e-scrita: Revista do Curso de Letras da UNIABEU. Belford Roxo/RJ: [s. n.], 2014. p. 33–45.

MICROSOFT. Observabilidade na IA generativa. *In*: p. 1. Acesso em 30/09/2025. Disponível em: <https://learn.microsoft.com/pt-br/azure/ai-foundry/concepts/observability>.

OPENAI. Introducing ChatGPT. *In*: p. 1. Acesso em 30/09/2025. Disponível em: <https://openai.com/index/chatgpt/>.

REAL, L.; VIANNA, D.; CARVALHO, A. L.; SILVA, A. da. LLMs Não São Garantido - Avaliando LLMs: Métricas, Benchmarks, Técnicas Automáticas e Considerações Éticas. *In*: ANAIS Estendidos do XL Simpósio Brasileiro de Bancos de Dados. Fortaleza/CE: SBC, 2025. p. 203–207.

ZANIBONI, J. V. N. Integrando Técnicas de Geração Aumentada por Recuperação e Grandes Modelos de Linguagem para Auxílio ao Diagnóstico Médico. *In*: p. 1.



## 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

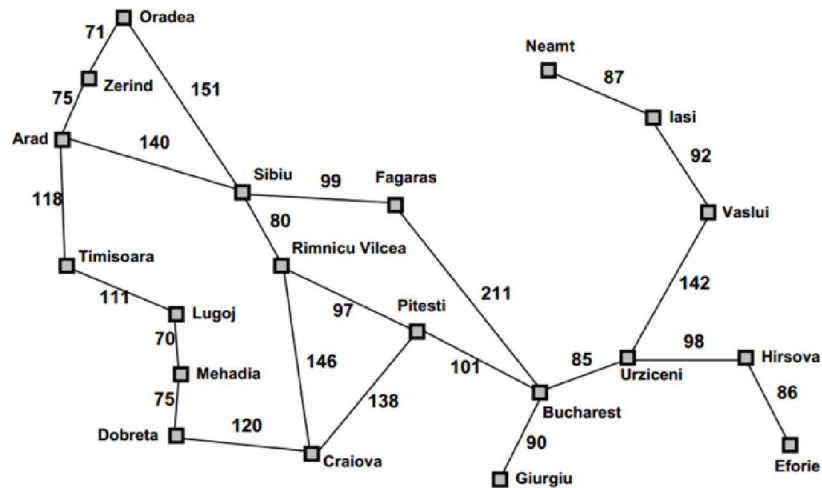
Realize uma busca utilizando o algoritmo A\* para encontrar o melhor caminho para chegar a **Bucharest** partindo de **Lugoj**. Construa a árvore de busca criada pela execução do algoritmo apresentando os valores de  $f(n)$ ,  $g(n)$  e  $h(n)$  para cada nó. Utilize a heurística de distância em linha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



Fonte: IAA UFPR (2025).

FIGURA 2 – Distâncias em linha reta para a cidade de Bucharest

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

Fonte: IAA UFPR (2025).

### 3 Lógica

Verificar se o argumento lógico é válido.

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

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

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

#### Dicas:

1. Transformar as afirmações para lógica:

$p$ : as uvas caem

$q$ : a raposa come as uvas

$r$ : as uvas estão maduras

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

R1:  $p \rightarrow q$

R2:  $q \rightarrow r$

R3:  $\neg r \vee p$

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

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

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

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

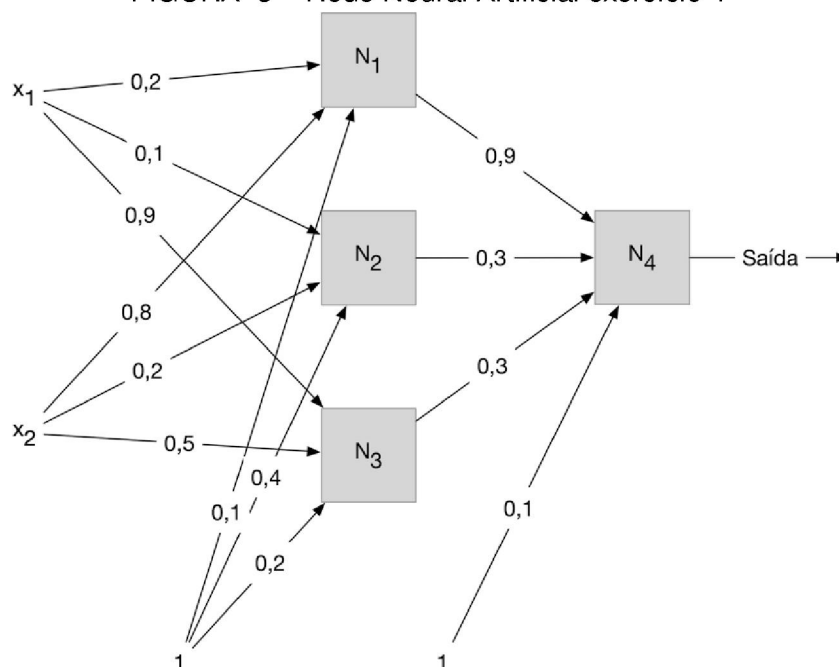
**Equivalência Bicondicional:**  $(\alpha \leftrightarrow \beta)$  equivale a  $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$

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

#### 4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.

FIGURA 3 – Rede Neural Artificial exercício 4



Fonte: IAA UFPR (2025).

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

Dada a entrada  $x_1 = -3$ ,  $x_2 = 1$ , dê:

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

## B - RESOLUÇÃO

### 1 ChatGPT

#### a) Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.

Inteligência Artificial (IA) refere-se à capacidade de um sistema ou programa de computador realizar tarefas que normalmente exigiriam inteligência humana. Essas tarefas incluem raciocínio, aprendizado, percepção visual, reconhecimento de fala, compreensão de linguagem natural, tomada de decisões e resolução de problemas.

Os sistemas de IA são projetados para imitar certos aspectos da inteligência humana, utilizando algoritmos e modelos matemáticos para processar dados, aprender padrões e tomar decisões. Existem dois tipos principais de IA:

1. IA Fraca (ou Estreita): Refere-se a sistemas de IA especializados em tarefas específicas. Eles são limitados ao escopo de suas programações e não têm a capacidade de realizar tarefas além do que foram projetados para fazer. Exemplos incluem reconhecimento de voz, chatbots e carros autônomos.
2. IA Forte (ou Geral): Envolve sistemas que têm a capacidade de realizar qualquer tarefa cognitiva que um ser humano pode fazer. Este nível de IA ainda não foi totalmente alcançado e é um campo de pesquisa em desenvolvimento.

Algoritmos de aprendizado de máquina (Machine Learning) são frequentemente usados na implementação de sistemas de IA. Eles permitem que a IA melhore seu desempenho em uma tarefa específica à medida que é exposta a mais dados, sem ser explicitamente programada.

A Inteligência Artificial tem aplicações em uma variedade de setores, como medicina, finanças, automação industrial, reconhecimento de padrões, entretenimento e muito mais. É uma área de rápido avanço e continua a desempenhar um papel significativo na transformação de várias indústrias.

**b) Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.**

A definição dada pelo ChatGPT menciona que sistemas de Inteligência Artificial são projetados para "imitar certos aspectos da inteligência humana", o que parece se assemelhar mais à definição de Pensar como Humanos ou Agir como Humanos.

**c) Pesquise sobre o funcionamento do ChatGPT (sem perguntar ao próprio ChatGPT) e escreva um texto contendo no máximo 5 parágrafos. Cite as referências.**

O ChatGPT pode ser entendido como uma extrapolação de uma classe de modelos de aprendizagem de máquina chamados Large Language Models (LLMs), que são modelos de Processamento de Linguagem Natural. Esse tipo de modelo consegue processar grandes quantidades de texto e inferir relações entre palavras dentro do texto. A capacidade dos LLMs cresce conforme aumenta o tamanho e variedade de parâmetros da base de dados. (Ruby, 2023)

Segundo o que é disponibilizado pelo OpenAI (OpenAI, 2024), ele é baseado na arquitetura GPT (Generative Pre-trained Transformer), um modelo Transformer, que é uma rede neural capaz de aprender o contexto dado e gerar um novo texto a partir disso. A rede foi otimizada para usar um método de treinamento chamado Reinforcement Learning with Human Feedback (RLHF), que usa demonstrações humanas para guiar seu comportamento, de forma a alcançar o desejável.

Segundo Rama Ramakrishnan, professor do MIT (MIT, 2023), o modelo precedente, GPT-3, foi treinado com cerca de 30 bilhões de frases retiradas de livros e da internet, para uma única tarefa: prever a palavra seguinte em uma frase, dadas as palavras usadas anteriormente. Ele calcula uma tabela de probabilidades para possíveis palavras a serem usadas e usa a que tem a melhor probabilidade, quase como um sistema de *autocomplete*. Para formar frases completas, a cada palavra escolhida, ele adiciona ela à frase e refaz o processo de cálculo das probabilidades para escolher a próxima palavra. Já a versão 3.5 foi treinada para seguir instruções dadas por humanos, utilizando uma base de dados contendo pares de exemplos de instruções e respostas de alta qualidade para tais instruções. O modelo treinado a partir dessa base de dados foi então utilizado para gerar múltiplas respostas para cada uma das instruções e essas respostas foram categorizadas por humanos de mais útil a menos útil. A partir desses novos dados, foi possível treinar um "modelo de recompensa", que avalia a qualidade das respostas geradas pelo GPT-3.5 e retorna uma "nota" de avaliação para o GPT-3.5, assim ele passa por um *fine tuning* para melhorar suas respostas com *reinforcement learning*. Na transição do GPT-3.5 para o ChatGPT foi usado um processo similar, mas desta vez utilizando conversas inteiras para o treinamento.

Ele é treinado usando textos escritos por humanos, incluindo conversações, de

forma a conseguir imitar o estilo de comunicação humano). Assim, de acordo com os dados utilizados para treinamento, ele pode, inclusive, produzir textos com conteúdo enviesado.

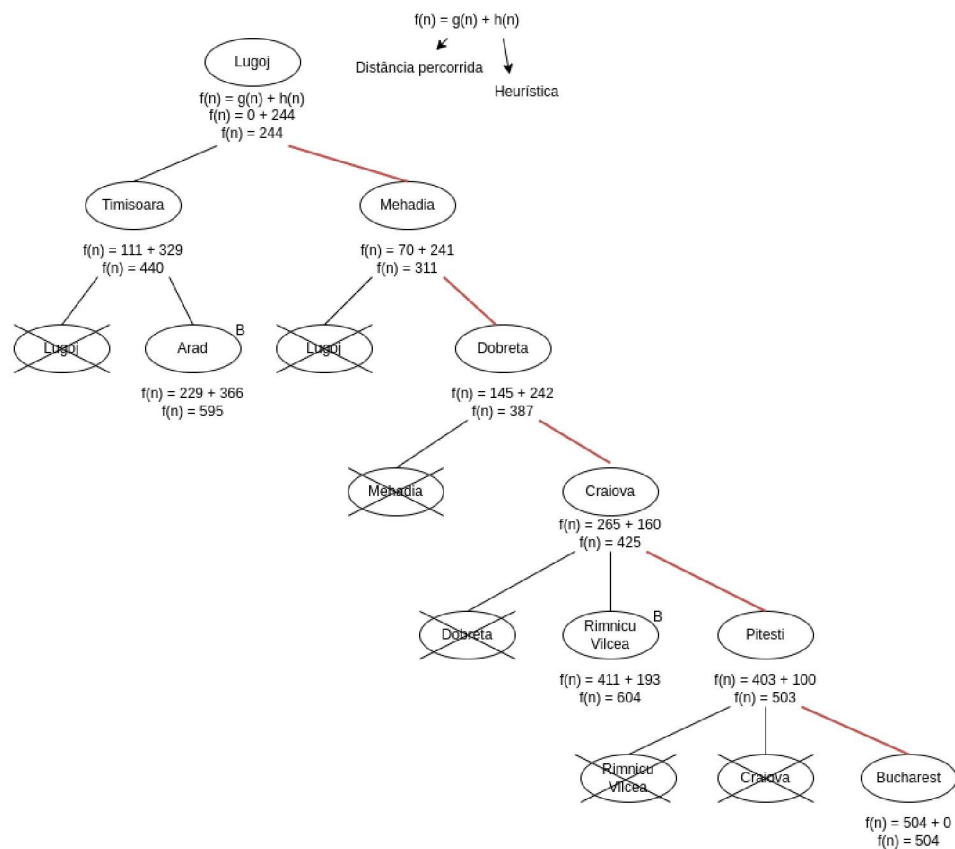
**d) Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê.**

De acordo com o comportamento, que se baseia em aprender e imitar como humanos se comunicam, ele parece se encaixar na definição da abordagem Agir como Humanos. Essa conclusão pode ser atingida considerando-se que o ChatGPT encontra padrões nos textos, que são dados produzidos (majoritariamente) por humanos, e com isso aprende contextos, mas não é totalmente consciente sobre o seu conteúdo, e não usa um processo cognitivo, apenas os sintetiza conforme foram fornecidos a ele. Como exemplo, se o ChatGPT fosse treinado com textos com um viés racista, ele poderia reproduzir textos racistas, já que não tem a capacidade de ponderar se racismo é correto ou não de um ponto de vista racional, ou seja, ele não toma decisões racionais baseadas no contexto que é apresentado a ele, apenas busca imitar a forma como humanos se comunicam.

## **2 Busca Heurística**

**a) 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.**

FIGURA 4 – Árvore final



Fonte: O autor (2025).

$Lugoj \Rightarrow Mehadia \Rightarrow Dobreta \Rightarrow Craiova \Rightarrow Pitesti \Rightarrow Bucharest = 504$

### 3 Lógica

*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*

$p$ : as uvas caem

$q$ : a raposa come as uvas

$r$ : as uvas estão maduras

$R1 : p \Rightarrow q$

$R2 : q \Rightarrow r$

$R3 : \neg r \vee p$

objetivo:

$$q \Leftrightarrow p$$

**a) Deve-se mostrar todos os passos e regras aplicadas, no mesmo formato apresentado nas aulas e nas práticas.**

$$R1 : p \Rightarrow q$$

$$R2 : q \Rightarrow r$$

$$R3 : \neg r \vee p$$

$$R4 : r \Rightarrow p$$

Equivalência Implicação, R3

$$R5 : q \Rightarrow p$$

SH, R2, R4

$$R6 : p \Rightarrow q \wedge q \Rightarrow p$$

CONJ, R1, R5

$$R7 : q \Leftrightarrow p$$

BICOND, R6

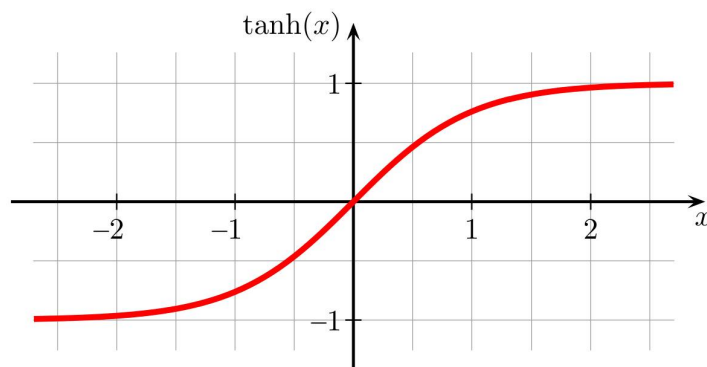
#### 4 Redes Neurais Artificiais

$$x_1 = -3, x_2 = 1$$

N1, N2, N3: função de ativação linear (mantém o valor)

N4: função de ativação tangente (produz valores no intervalo  $[-1, 1]$ )

FIGURA 5 – Tangente Hiperbólica



Fonte: O autor (2025).

**a) Valor de saída do neurônio N1**

$$(-3) \times 0.2 + 1 \times 0.8 + 1 \times 0.1 = 0.3$$

**b) Valor de saída do neurônio N2**



$$(-3) \times 0.1 + 1 \times 0.2 + 1 \times 0.4 = 0.3$$

**c) Valor de saída do neurônio N3**

$$(-3) \times 0.9 + 1 \times 0.5 + 1 \times 0.2 = -2$$

**d) Valor de saída da rede como um todo**

$$\tanh(0.3 \times 0.9 + 0.3 \times 0.3 + (-2) \times 0.3 + 1 \times 0.1) = \tanh(0.27 + 0.09 - 0.6 + 0.1) = -0.13909 \approx -0.14$$

## APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

### A - ENUNCIADO

**Nome da base de dados do exercício:** *precos\_carros\_brasil.csv*

**Informações sobre a base de dados:**

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

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

**Metadados:**

TABELA 2 – Metadados da Tabela FIPE

| Nome do campo      | Descrição                                                                       |
|--------------------|---------------------------------------------------------------------------------|
| year_of_reference  | O preço médio corresponde a um mês do ano de referência                         |
| month_of_reference | O preço médio corresponde a um mês específico, pois a FIPE atualiza mensalmente |
| fipe_code          | Código único da FIPE                                                            |
| authentication     | Código único de autenticação para consulta FIPE                                 |
| brand              | Marca do carro                                                                  |
| model              | Modelo do carro                                                                 |
| fuel               | Tipo de combustível                                                             |
| gear               | Tipo de engrenagem                                                              |
| engine_size        | Tamanho do motor em centímetros cúbicos                                         |
| year_model         | Ano do modelo (pode ser diferente do ano de fabricação)                         |
| avg_price          | Preço médio do carro em reais                                                   |

Fonte: IAA UFPR (2025).

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

## 1 Análise Exploratória dos dados

A partir da base de dados **precos\_carros\_brasil.csv**, execute as seguintes tarefas:

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

## 2 Visualização dos dados

A partir da base de dados **precos\_carros\_brasil.csv**, execute as seguintes tarefas:

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

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

A partir da base de dados **precos\_carros\_brasil.csv**, execute as seguintes tarefas:

- 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
- Crie partições contendo 75% dos dados para treino e 25% para teste
- 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
- Grave os valores preditos em variáveis criadas
- Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**
- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
- Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e  $R^2$
- Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

## B - RESOLUÇÃO

### 1 Análise Exploratória dos dados

#### a. Carregue a base de dados **media\_precos\_carros\_brasil.csv**

```
# Para rápida instalação das bibliotecas necessárias, rodar o
# script abaixo no CLI:
# pip install -r requirements.txt

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

```

import numpy as np

import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import RandomizedSearchCV

# Métricas de avaliação dos modelos
from sklearn.metrics import mean_squared_error,
    mean_absolute_error, r2_score

dados = pd.read_csv('precos_carros_brasil.csv')
dados.head()

```

| year_ref | month   | fipe_code | auth         | brand          | model                                  | fuel     | gear   | engine | year   | price   |
|----------|---------|-----------|--------------|----------------|----------------------------------------|----------|--------|--------|--------|---------|
| 2021.0   | January | 004001-0  | cfzltzfwrcp  | GM - Chevrolet | Corsa Wind 1.0 MPFI / EFI 2p           | Gasoline | manual | 1      | 2002.0 | 9162.0  |
| 2021.0   | January | 004001-0  | cdqwxwpw3y2p | GM - Chevrolet | Corsa Wind 1.0 MPFI / EFI 2p           | Gasoline | manual | 1      | 2001.0 | 8832.0  |
| 2021.0   | January | 004001-0  | cb1t3xwwj1xp | GM - Chevrolet | Corsa Wind 1.0 MPFI / EFI 2p           | Gasoline | manual | 1      | 2000.0 | 8388.0  |
| 2021.0   | January | 004001-0  | cb9gct6j65r0 | GM - Chevrolet | Corsa Wind 1.0 MPFI / EFI 2p           | Alcohol  | manual | 1      | 2000.0 | 8453.0  |
| 2021.0   | January | 004003-7  | g15wg0gbz1fx | GM - Chevrolet | Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI | Gasoline | manual | 1.6    | 2001.0 | 12525.0 |

**b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma tratativa para resolver o problema de valores faltantes**

```

# Excluindo as linhas sem dados (característica de importação de
# .CSV)
dados = dados.dropna(how='all')

antes = dados.shape

# Verificando se existem valores faltantes nos dados
dados.isna().any()

```

```

year_of_reference      False
month_of_reference     False
fipe_code              False
authentication         False
brand                 False
model                 False

```

```

fuel            False
gear            False
engine_size     False
year_model      False
avg_price_brl   False
dtype: bool

```

Não há valores faltantes.

### c. Verifique se há dados duplicados nos dados

```

# Verificando se temos valores duplicados
dados.duplicated().sum()

```

```
2
```

```

# Removendo valores duplicados
dados.drop_duplicates(inplace=True)

depois = dados.shape

# Verifica diferença após a normalização dos dados
diff_linhas = depois[0] - antes[0]
diff_colunas = depois[1] - antes[1]
print("Linhas:\n Antes {} x Depois {} = Diff {}".format(antes[0], depois
    [0], diff_linhas))
print("Colunas:\n Antes {} x Depois {} = Diff {}".format(antes[1],
    depois[1], diff_colunas))

```

```

Linhas:
  Antes 202297 x Depois 202295 = Diff -2
Colunas:
  Antes 11 x Depois 11 = Diff 0

```

2 linhas duplicadas excluídas.

### d. Crie duas categorias, para separar colunas numéricas e categóricas. Imprima o resumo de informações das variáveis numéricas e categóricas (estatística descritiva dos dados)

```

# Criando categorias para separar colunas numéricas e categóricas
: facilita a AED
numericas_cols = [col for col in dados.columns if dados[col].
    dtype != 'object']

```



```
categoricas_cols = [col for col in dados.columns if dados[col].
    dtype == 'object']

# Resumo das variáveis numéricas - Imprime alguns valores de
    medidas de tendências centrais
dados[numericas_cols].describe()
```

|       | year_of_reference | year_model    | avg_price_brl |
|-------|-------------------|---------------|---------------|
| count | 202295.000000     | 202295.000000 | 202295.000000 |
| mean  | 2021.564695       | 2011.271514   | 52756.765713  |
| std   | 0.571904          | 6.376241      | 51628.912116  |
| min   | 2021.000000       | 2000.000000   | 6647.000000   |
| 25%   | 2021.000000       | 2006.000000   | 22855.000000  |
| 50%   | 2022.000000       | 2012.000000   | 38027.000000  |
| 75%   | 2022.000000       | 2016.000000   | 64064.000000  |
| max   | 2023.000000       | 2023.000000   | 979358.000000 |

#### e. Imprima a contagem de valores por modelo (model) e marca do carro (brand)

```
# Contagem de valores por categoria de 'Modelo'
dados['model'].value_counts()
```

```
Palio Week. Adv/Adv TRYON 1.8 mpi Flex      425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p      425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.      400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V       400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray      375
...
STEPWAY Zen Flex 1.0 12V Mec.               2
Saveiro Robust 1.6 Total Flex 16V CD        2
Saveiro Robust 1.6 Total Flex 16V          2
Gol Last Edition 1.0 Flex 12V 5p           2
Polo Track 1.0 Flex 12V 5p                 2
Name: model, Length: 2112, dtype: int64
```

```
# Contagem de valores por categoria de 'Marca'
dados['brand'].value_counts()
```

```
Fiat          44962
VW - Volkswagen 44312
GM - Chevrolet 38590
Ford          33150
```

|                           |       |
|---------------------------|-------|
| Renault                   | 29191 |
| Nissan                    | 12090 |
| Name: brand, dtype: int64 |       |

**f. Dê um breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados**

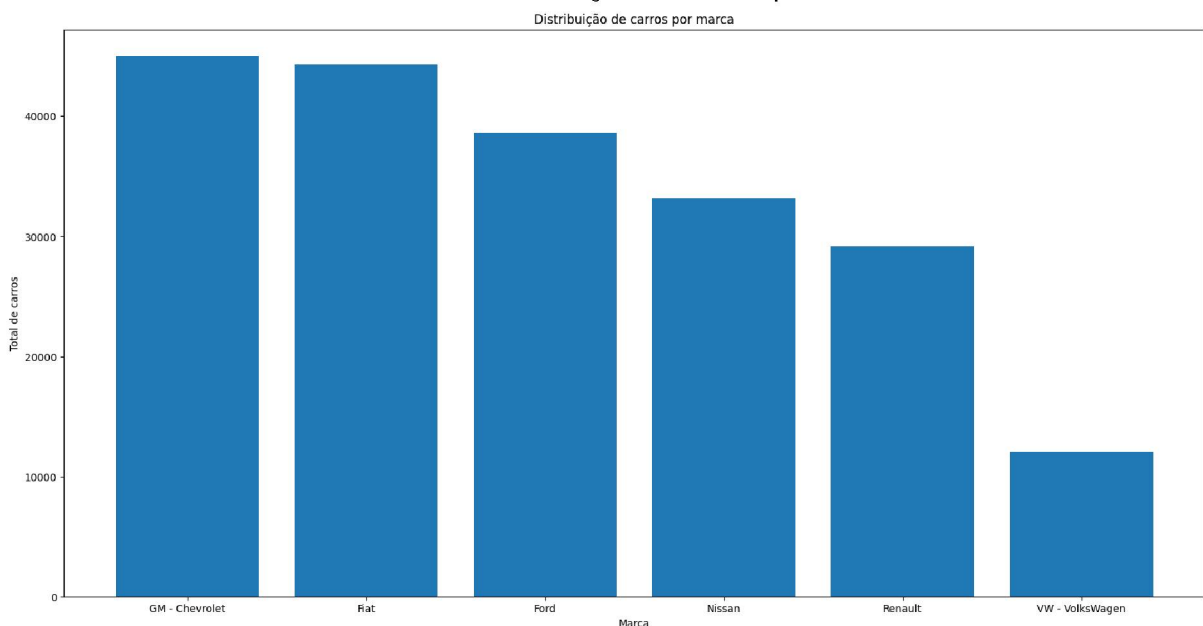
Na Análise Exploratória dos Dados pudemos observar que o conjunto de dados possui 11 colunas e nenhuma delas possui dados faltantes. Ao observarmos valores duplicados, apenas 2 linhas estavam duplicadas. Das 11 colunas, apenas 3 são numéricas. Porém, as colunas brand e model podem ser facilmente convertidas para numéricas.

## 2 Visualização dos dados

**a. Gere um gráfico da distribuição da quantidade de carros por marca**

```
# Gráfico da distribuição por marca
plt.figure(figsize=(20,10))
plt.bar(dados['brand'].unique(), dados['brand'].value_counts())
plt.title('Distribuição de carros por marca')
plt.ylabel('Total de carros')
plt.xlabel('Marca')
```

FIGURA 6 – Distribuição de carros por marca



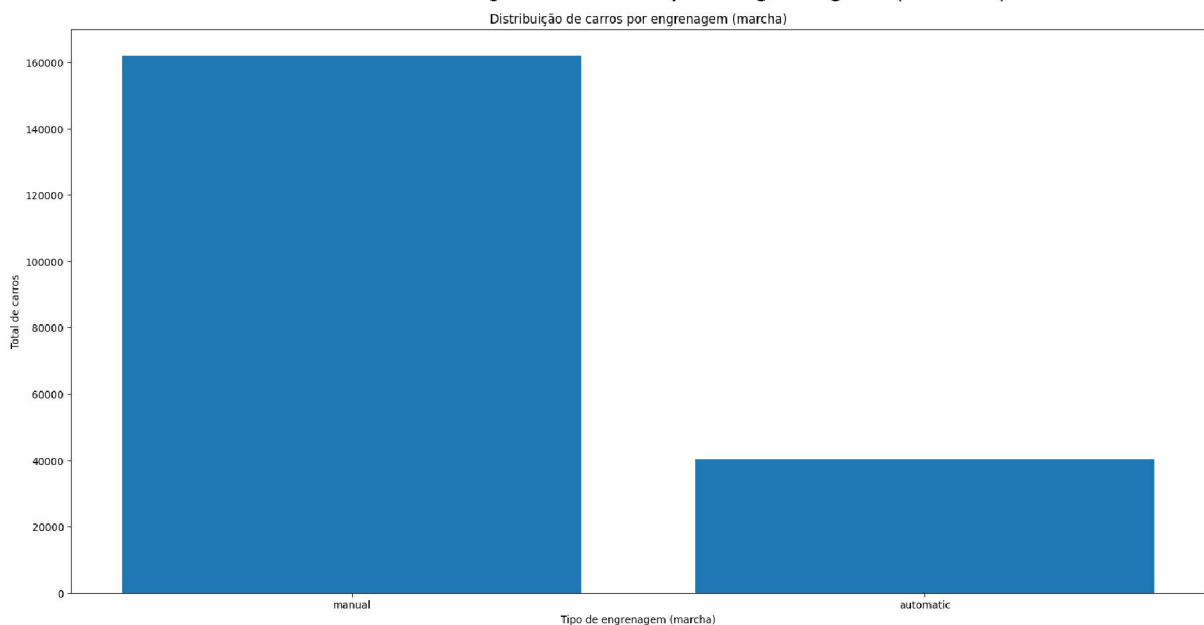
Fonte: O autor (2025).

**b. Gere um gráfico da distribuição da quantidade de carros por tipo de engrenagem do carro**



```
# Gráfico da distribuição por engrenagem (tipo de marcha)
plt.figure(figsize=(20,10))
plt.bar(dados['gear'].unique(), dados['gear'].value_counts())
plt.title('Distribuição de carros por engrenagem (marcha)')
plt.ylabel('Total de carros')
plt.xlabel('Tipo de engrenagem (marcha)')
```

FIGURA 7 – Distribuição de carros por engrenagem (marcha)



Fonte: O autor (2025).

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

```
# limitando para somente os dados de 2022
dados_2022 = dados[dados['year_of_reference'] == 2022]

# verificando os completude dos meses
dados_2022['month_of_reference'].drop_duplicates()

# Calculando a média de preços por mês
media_precos_mes = dados_2022.groupby(['year_of_reference', 'month_of_reference'])['avg_price_brl'].mean().round(0)
df_media_precos_mes = media_precos_mes.reset_index(name='Médio de Preço')

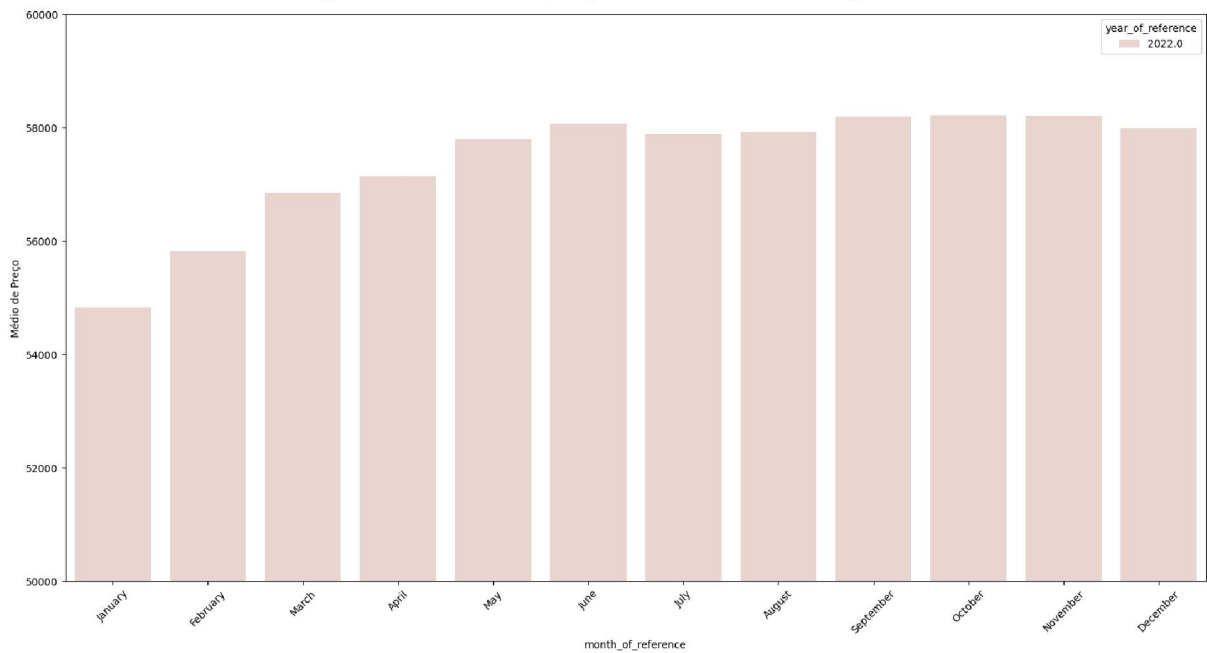
# Reordenando o dataframe para melhor exibição no gráfico
df_media_precos_mes.insert(0, "index", [3,7,11,1,0,6,5,2,4,10,9,8], True)
```

```
df_media_precos_mes = df_media_precos_mes.sort_values(by=['index',
])
df_media_precos_mes.head(12)
```

| Index | Order | Year   | Month     | Average Price (BRL) |
|-------|-------|--------|-----------|---------------------|
| 4     | 0     | 2022.0 | January   | 54840.0             |
| 3     | 1     | 2022.0 | February  | 55825.0             |
| 7     | 2     | 2022.0 | March     | 56849.0             |
| 0     | 3     | 2022.0 | April     | 57150.0             |
| 8     | 4     | 2022.0 | May       | 57800.0             |
| 6     | 5     | 2022.0 | June      | 58066.0             |
| 5     | 6     | 2022.0 | July      | 57894.0             |
| 1     | 7     | 2022.0 | August    | 57924.0             |
| 11    | 8     | 2022.0 | September | 58199.0             |
| 10    | 9     | 2022.0 | October   | 58227.0             |
| 9     | 10    | 2022.0 | November  | 58216.0             |
| 2     | 11    | 2022.0 | December  | 57997.0             |

```
# Visualizando a média salarial por ano
plt.figure(figsize=(20,10))
sns.barplot(data=df_media_precos_mes, x='month_of_reference', y='
Médio de Preço', hue='year_of_reference')
plt.ylim(50000, 60000) # limitação do eixo Y para melhor observaç
ão da variação de valores
plt.xticks(rotation=45);
```

FIGURA 8 – Evolução da média de preço dos carros ao longo dos meses de 2022



Fonte: O autor (2025).

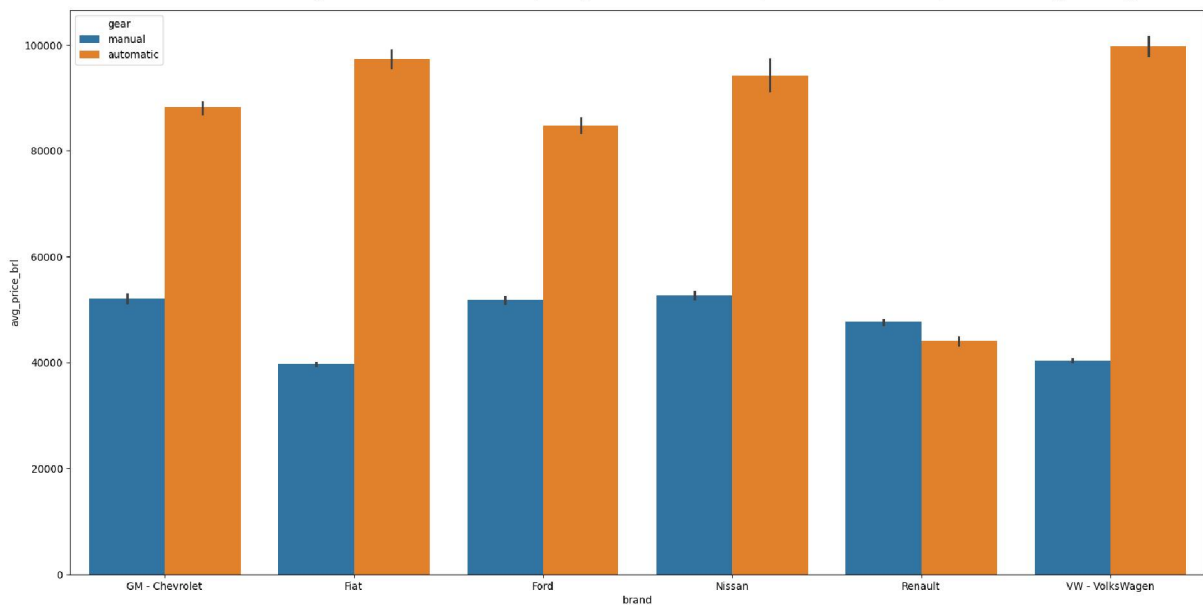
**d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem**

```
print(dados["gear"].unique()) # verificar os valores da coluna
gear
```

```
['manual' 'automatic']
```

```
plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='avg_price_brl', hue='gear', data=dados,
            hue_order=['manual', 'automatic'])
```

FIGURA 9 – Distribuição da média de preço dos carros por marca e tipo de engrenagem



Fonte: O autor (2025).

**e. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item d**

O preço médio de carros com engrenagem (marcha) automática é superior aos com engrenagem manual em 5 das 6 marcas. Somente a Renault tem preços médios com ligeira superioridade nos carros com engrenagem manual. Nas marcas Fiat e Volkswagen os carro com engrenagem automática tem preço médio maiores do que o dobro dos carros com engrenagem manual.

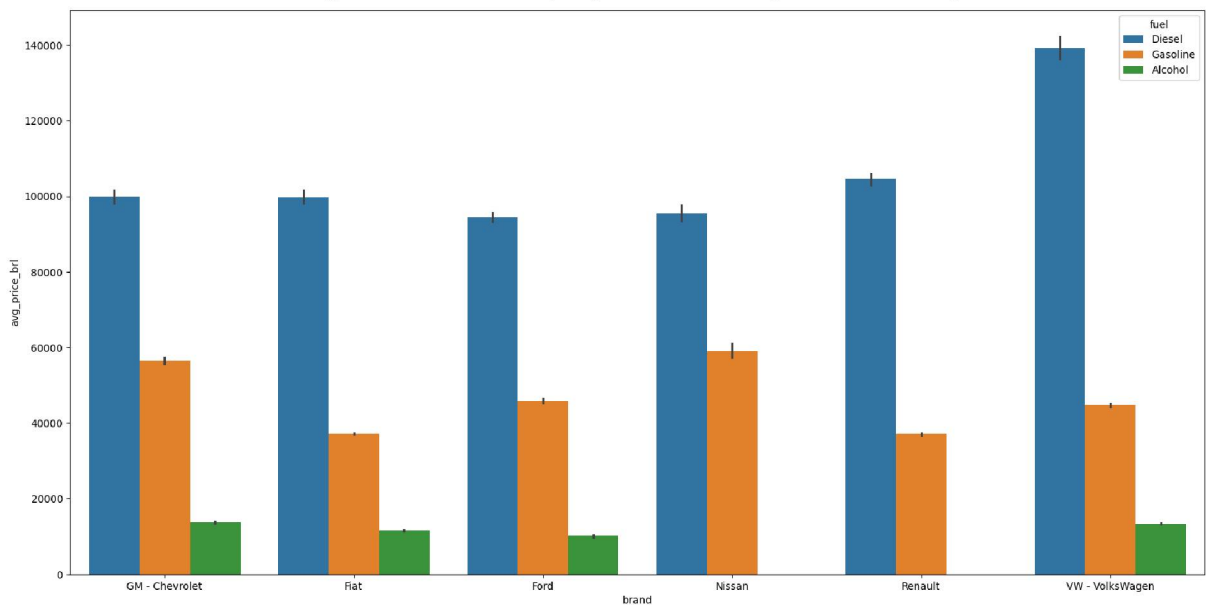
**f. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de combustível**

```
print(dados["fuel"].unique()) # verificar os valores da coluna
fuel
```

```
['Gasoline' 'Alcohol' 'Diesel']
```

```
plt.figure(figsize=(20,10))
sns.barplot(x='brand', y='avg_price_brl', hue='fuel', data=dados,
            hue_order=['Diesel', 'Gasoline', 'Alcohol'])
```

FIGURA 10 – Distribuição da média de preço dos carros por marca e tipo de combustível



Fonte: O autor (2025).

**g. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados gerados no item f**

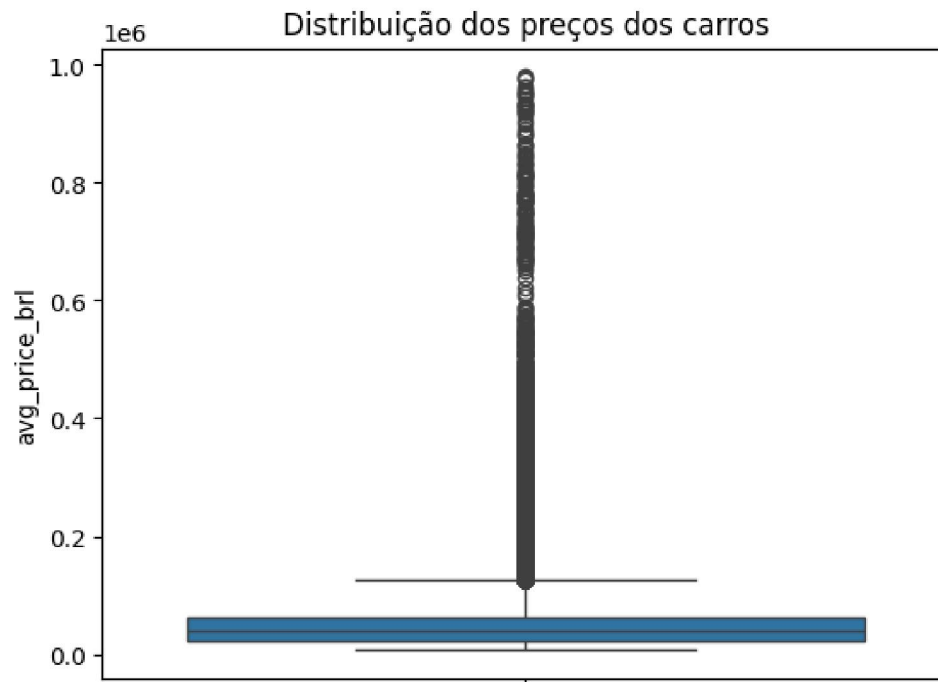
Carros com o combustível Diesel tem os maiores preços médios em todas as marcas. O que faz sentido haja visto que motores a Diesel são comumente encontrados em carros maiores e de maiores valores. Veículos a Gasolina são os segundos com maiores preços médios em todas as marcas. Vale salientar que, nesta base, os veículos Flex (que podem consumir álcool e gasolina) estão categorizados como Gasolina. Veículos a Álcool tem os menores preços médios nas marcas GM, Fiat, Ford e VM. As marcas Nissan e Renault não apresentaram veículos com combustível álcool.

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

**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**

```
sns.boxplot(dados['avg_price_brl']).set_title("Distribuição dos
preços dos carros")
```

FIGURA 11 – Distribuição dos preços dos carros



Fonte: O autor (2025).

```
# Transformação da coluna modelo (model) de categórica para numé
rica
dados['model'] = LabelEncoder().fit_transform(dados['model'])

# Transformação da coluna engrenagem (gear) de categórica para
numérica
dados['gear'] = LabelEncoder().fit_transform(dados['gear'])
dados.head()

# Transformação da coluna combustível (fuel) de categórica para
numérica
dados['fuel'] = LabelEncoder().fit_transform(dados['fuel'])
dados.head()
```

| ID | Year   | Month   | FIPE Code | Auth         | Brand          | Model | Fuel | Gear | Engine | Year Model | Avg. Price (BRL) |
|----|--------|---------|-----------|--------------|----------------|-------|------|------|--------|------------|------------------|
| 0  | 2021.0 | January | 004001-0  | cfzltzfwrcp  | GM - Chevrolet | 297   | 2    | 1    | 1      | 2002.0     | 9162.0           |
| 1  | 2021.0 | January | 004001-0  | cdqwxwpw3y2p | GM - Chevrolet | 297   | 2    | 1    | 1      | 2001.0     | 8832.0           |
| 2  | 2021.0 | January | 004001-0  | cb1t3xwwj1xp | GM - Chevrolet | 297   | 2    | 1    | 1      | 2000.0     | 8388.0           |
| 3  | 2021.0 | January | 004001-0  | cb9gct6j65r0 | GM - Chevrolet | 297   | 0    | 1    | 1      | 2000.0     | 8453.0           |
| 4  | 2021.0 | January | 004003-7  | g15wg0gbz1fx | GM - Chevrolet | 260   | 2    | 1    | 1,6    | 2001.0     | 12525.0          |

```
# Variável dados_num contém apenas variáveis numéricas de
interesse (exclui o restante)
```

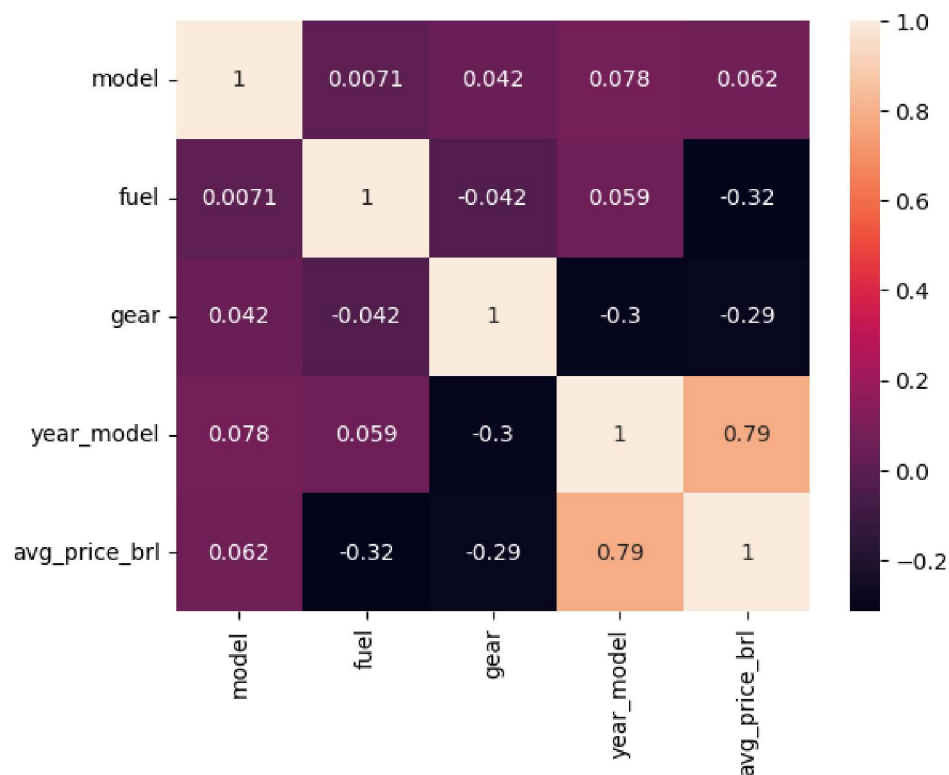


```
dados_num = dados.drop(['year_of_reference', 'month_of_reference',
                        'fipe_code', 'authentication', 'brand', 'engine_size'],axis
                        = 1)

sns.heatmap(dados_num.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n",
          fontsize = 15)
plt.show()
```

FIGURA 12 – Mapa de Correlação das Variáveis Numéricas

Mapa de Correlação das Variáveis Numéricas



Fonte: O autor (2025).

```
# Variável X contém apenas variáveis numéricas de interesse para
# a análise, excluindo a variável target
X = dados_num.drop(['avg_price_brl'],axis = 1)

Y = dados_num['avg_price_brl']
```

**b. Crie partições contendo 75% dos dados para treino e 25% para teste**

```
# Divisão: 30% dos dados são de teste e 70% de treinamento
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
    test_size = 0.25, random_state = 42)
```

**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**

```
#RandomForest
model_rf = RandomForestRegressor()
model_rf.fit(X_train, Y_train)

#RandomForest com parâmetros
n_estimators = [5,20,50,100] # number of trees in the random
    forest
max_features = ['auto', 'sqrt'] # number of features in
    consideration at every split
max_depth = [int(x) for x in np.linspace(10, 120, num = 12)] #
    maximum number of levels allowed in each decision tree
min_samples_split = [2, 6, 10] # minimum sample number to split a
    node
min_samples_leaf = [1, 3, 4] # minimum sample number that can be
    stored in a leaf node
bootstrap = [True, False] # method used to sample data points

random_grid = {'n_estimators': n_estimators, 'max_features':
    max_features, 'max_depth': max_depth, 'min_samples_split':
    min_samples_split,
                'min_samples_leaf': min_samples_leaf, 'bootstrap':
    bootstrap}

rf = RandomForestRegressor()
rf_random = RandomizedSearchCV(estimator = rf,param_distributions
    = random_grid,
                               n_iter = 100, cv = 5, verbose=2, random_state=35,
                               n_jobs = -1)

rf_random.fit(X_train, Y_train)

# Utilizando os melhores parâmetros encontrados
model_rf_parametros = RandomForestRegressor(max_depth=70,
    min_samples_leaf=1, min_samples_split=2, n_estimators=20,
    random_state=80)
```



```
model_rf_parametros.fit(X_train, Y_train)
```

```
#XGBoost
model_xgboost = XGBRegressor()
model_xgboost.fit(X_train, Y_train)
```

#### d. Grave os valores preditos em variáveis criadas

```
#RandomForest
valores_preditos_rf = model_rf.predict(X_test)
valores_preditos_rf
```

```
#RandomForest com parâmetros
valores_preditos_rf_parametros = model_rf_parametros.predict(
    X_test)
valores_preditos_rf_parametros
```

```
#XGBoost
valores_preditos_xgboost = model_xgboost.predict(X_test)
valores_preditos_xgboost
```

#### e. Realize a análise de importância das variáveis para estimar a variável target, para cada modelo treinado

```
#RandomForest
model_rf.feature_importances_
feature_importances_rf = pd.DataFrame(model_rf.
    feature_importances_, index = X_train.columns, columns=['
    importance']).sort_values('importance', ascending = False)
feature_importances_rf
```

| Feature    | Importance |
|------------|------------|
| model      | 0.439964   |
| year_model | 0.358480   |
| fuel       | 0.179241   |
| gear       | 0.022315   |

```
#RandomForest com parâmetros
model_rf_parametros.feature_importances_
feature_importances_rf_param = pd.DataFrame(model_rf_parametros.
    feature_importances_, index = X_train.columns, columns=['
    importance']).sort_values('importance', ascending = False)
```

```
feature_importances_rf_param
```

| Feature    | Importance |
|------------|------------|
| model      | 0.442857   |
| year_model | 0.356757   |
| fuel       | 0.178231   |
| gear       | 0.022155   |

```
#XGBoost
model_xgboost.feature_importances_
feature_importances_x = pd.DataFrame(model_xgboost.
    feature_importances_, index = X_train.columns, columns=['
    importance']).sort_values('importance', ascending = False)
feature_importances_x
```

| Feature    | Importance |
|------------|------------|
| fuel       | 0.699523   |
| year_model | 0.160642   |
| model      | 0.088919   |
| gear       | 0.050917   |

**f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis**

Tanto no modelo Random Forest, quando no modelo Random Forest com parâmetros, as variáveis mais importantes foram modelo e ano do modelo (model e year\_model) com quase 80% de importância. Somando a combustível (fuel) chegamos a 97% de importância. Tornando quase irrelevante a variável engrenagem (gear). Para o modelo XGBoost temos como variável mais importante o combustível (fuel) com aproximadamente 70% e seguido de ano do modelo com 16%. Modelo e engrenagem (model e gear) aparecem com menos de 10% cada.

**g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R<sup>2</sup>**

```
#RandomForest
mse_rf = mean_squared_error(Y_test, valores_preditos_rf)
mae_rf = mean_absolute_error(Y_test, valores_preditos_rf)
r2_rf = r2_score(Y_test, valores_preditos_rf)
mse_rf
```

```
mae_rf
r2_rf
```

```
54633997.34991135
4215.705784378993
0.97969945241549
```

```
#RandomForest com parâmetros
mse_rf_param = mean_squared_error(Y_test,
    valores_preditos_rf_parametros)
mae_rf_param = mean_absolute_error(Y_test,
    valores_preditos_rf_parametros)
r2_rf_param = r2_score(Y_test, valores_preditos_rf_parametros)
mse_rf_param
mae_rf_param
r2_rf_param
```

```
54716253.201698475
4216.618281471371
0.9796688883177797
```

```
#XGBoost
mse_x = mean_squared_error(Y_test, valores_preditos_xgboost)
mae_x = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_x = r2_score(Y_test, valores_preditos_xgboost)
mse_x
mae_x
r2_x
```

```
297039446.66991967
6429.30300193814
0.8896280024509496
```

**h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada**

Levando em conta a acurácia resultante da métrica R2, temos os seguintes resultados:

- Random Forest: 98%
- Random Forest com parâmetros: 98%
- XGBoost: 89%

Desta forma, temos um empate entre Random Forest e Random Forest com parâmetros. Para fins de entendimento, trazemos a comparação dos resultados de MSE e MAE (quanto menores, melhores os resultados):

- MSE
  - Random Forest: 54.622.503
  - Random Forest com parâmetros: 54.716.253
- MAE
  - Random Forest: 4215
  - Random Forest com parâmetros: 4217

Como podemos ver, há uma ligeira vantagem no uso do Random Forest, sem parâmetros. Assim sendo, concluímos que o melhor resultado preditivo foi obtido através do modelo Random Forest.

## APÊNDICE 3 – LINGUAGEM R

### A - ENUNCIADO

#### 1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

## 2 Estimativa de Volumes de Árvores

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

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

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

$$\text{Volume} = b_0 + b_1 * \text{dap}^{2*} \text{Ht}$$

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:  $\text{Volume} = b_0 + b_1 * \text{dap}^2 * H_t$

**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

**Coefficiente 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;

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

## B - RESOLUÇÃO

### 1 Pesquisa com Dados de Satélite (Satellite)

#### 1. Carregue a base de dados Satellite

```
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("mlbench")
install.packages("caret")
install.packages("RSNNS")

sink("./output.txt", append = T)

library(mlbench)
library(caret)
library(RSNNS)

data("Satellite")
df <- Satellite[,17:20]
df$classes <- Satellite$classes
```

#### 2. Crie partições contendo 80% para treino e 20% para teste

```
set.seed(7)
indices <- createDataPartition(df$classes, p=0.8, list=FALSE)
treino <- df[indices, ]
teste <- df[-indices, ]
```

#### 3. Treine modelos RandomForest, SVM e RNA para predição destes dados.

```
print(">> Iniciando treinamento rf...")
rf <- caret::train(classes~., data=treino, method="rf")
predicoes.rf <- predict(rf, teste)
confusion.rf <- caret::confusionMatrix(predicoes.rf, teste$classes)

print(">> Iniciando treinamento svm...")
svm <- caret::train(classes~., data=treino, method="svmRadial")
predicoes.svm <- predict(svm, teste)
confusion.svm <- caret::confusionMatrix(predicoes.svm, teste$classes)

print(">> Iniciando treinamento rna...")
rna <- caret::train(classes~., data=treino, method="nnet")
```



```
predicoes.rna <- predict(rna, teste)
confusion.rna <- caret::confusionMatrix(predicoes.rna, teste$class)
```

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

```
print(">> Iniciando Análises de Matrizes de Confusão...")
print(confusion.rf) #acuracia = 84%
print(confusion.svm) #acuracia = 87%
print(confusion.rna) #acuracia = 80%
```

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

A métrica escolhida para comparação é a acurácia. A acurácia dos modelos são:

- RandomForest: 84%
- SVM: 87%
- RNA: 80%

O modelo de melhor acurácia é o SVM com 87%.

## 2 Estimativa de Volumes de Árvores

### 1. Carregar o arquivo Volumes.csv (<http://www.razer.net.br/datasets/Volumes.csv>)

```
install.packages("mlbench")
install.packages("caret")
install.packages("RSNNS")
install.packages("kernlab")
install.packages("randomForest")

sink("./output.txt", append = T)

library(mlbench)
library(caret)
library(RSNNS)

# load dataset
df <- read.csv("./Volumes.csv", header=TRUE, sep=";")
```

### 2. Eliminar a coluna NR, que só apresenta um número sequencial

```
df$DAP <- as.numeric(sub(",", ".", df$DAP, fixed = TRUE))
df$HT <- as.numeric(sub(",", ".", df$HT, fixed = TRUE))
df$HP <- as.numeric(sub(",", ".", df$HP, fixed = TRUE))
df$VOL <- as.numeric(sub(",", ".", df$VOL, fixed = TRUE))
df$NR <- NULL
summary(df)

set.seed(7)
```

### 3. Criar partição de dados: treinamento 80%, teste 20%

```
df[c(1, 2, 3)] <- lapply(df[c(1, 2, 3)], function(x) c(scale(x)))

indices <- createDataPartition(df$VOL, p=0.80, list=FALSE)
traindf <- df[indices,]
testdf <- df[-indices,]

str(df)
```

### 4. Usando o pacote "caret", treinar os modelos: Random Forest (rf), SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de SPURR

```
# modelo alométrico de SPURR

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

predictalom <- predict(alom, testdf)
print(predictalom)

train_control <- trainControl(method = "cv", number = 10)

rf <- caret::train(VOL~., data=traindf, method="rf", trainControl=trainControl)

svm <- caret::train(VOL~., data=traindf, method="svmRadial", trainControl=
  trainControl)

rna <- caret::train(VOL~., data=traindf, method="nnet", trainControl=
  trainControl)
```

## 5. Efetue as previsões nos dados de teste

```
predictionsrf <- predict(rf, testdf)
predictionssvm <- predict(svm, testdf)
predictionsrna <- predict(rna, testdf)
```

## 6. Crie suas próprias funções (UDF) e calcule as seguintes métricas entre a previsão e os dados observados

```
RSQUARE = function(y_actual,y_predict){
  cor(y_actual,y_predict)^2
}

LR_R = RSQUARE(testdf[,4], predictionsrf)
LR_R = RSQUARE(testdf[,4], predictionssvm)
LR_R = RSQUARE(testdf[,4], predictionsrna)
LR_R = RSQUARE(testdf[,4], predictalom)

syx <- function(vol, pred) {
  residual <- vol - pred
  sse <- sum(residual^2)
  n <- length(vol) - 2
  syx <- sqrt(sse / n)
  return(syx)
}

syx_r <- syx(testdf[,4], predictionsrf)
syx_r <- syx(testdf[,4], predictionssvm)
syx_r <- syx(testdf[,4], predictionsrna)
syx_r <- syx(testdf[,4], predictalom)

syx_percentual <- function(vol, pred) {
  syx_v <- syx(vol, pred)
  media_dados <- mean(vol)
  syx_percentual <- (syx_v / media_dados) * 100
  return(syx_percentual)
}

syxp_r <- syx_percentual(testdf[,4], predictionsrf)
syxp_r <- syx_percentual(testdf[,4], predictionssvm)
syxp_r <- syx_percentual(testdf[,4], predictionsrna)
syxp_r <- syx_percentual(testdf[,4], predictalom)
```

## 7. Escolha o melhor modelo.

Três métricas foram calculadas para os modelos. São elas, Coeficiente de determinação  $R^2$ , Erro padrão da estimativa  $Sy_x$ , e porcentagem de  $Sy_x$ . A seguir os valores obtidos pelas métricas em cada um dos modelos.

**$R^2$**  (Quanto mais perto de 1 melhor)

- RandomForest: 0.8589349
- SVM: 0.856365
- NNET: NA
- SSPUR: 0.8546826

**$Sy_x$**  (Quanto mais perto de 0 melhor)

- RandomForest: 0.1445527
- SVM: 0.1470481
- NNET: 0.49606
- SSPUR: 0.1574296

**$Sy_x\%$**  (Quanto mais perto de 0 melhor)

- RandomForest: 11.07658%
- SVM: 11.2678%
- NNET: 38.01139%
- SSPUR: 12.0633%

Como forma de melhorar os resultados, tentamos fazer a normalização dos dados como etapa de pré-processamento. Essa etapa de “dimensionamento” assegura que os dados de todas as colunas estejam dentro do mesmo intervalo de valores,  $[-2,2]$ . Isso evita que a rede possa acabar “perdendo” *features* com valores muito pequenos, e que seriam relevantes durante o treinamento, em seus cálculos, privilegiando *features* com valores muito mais altos.

Após realizar a normalização e padronização dos dados, com a aplicação da função **scale** nas colunas DAP, HP e HT da base de dados, pudemos observar uma

leve melhora em algumas métricas, mas principalmente, conseguimos um resultado muito mais expressivo para o treinamento da RNA.

Ao mesmo tempo, o resultado do modelo alométrico de SPURR teve uma grande piora na métrica R2, provavelmente porque ele funciona através de uma aproximação à função de volume e a mudança dos valores interfira nos cálculos.

## R2

- RandomForest: 0.8601537 
- SVM: 0.856365
- NNET: 0.4843006 
- SSPUR: 0.5317544 

## SYX

- RandomForest: 0.1438805 
- SVM: 0.1470481
- NNET: 0.4960265 
- SSPUR: 0.3854788 

## SYX%

- RandomForest: 11.02507% 
- SVM: 11.2678%
- NNET: 38.00883% 
- SSPUR: 29.53794% 

Escolhemos o modelo RandomForest como melhor por ter se saído ligeiramente melhor em todas as métricas calculadas.

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

**Obs:**

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

## B - RESOLUÇÃO

### 1 Gráficos e tabelas

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

```
install.packages("car")
install.packages("fdth")
library(car)
library(fdth)

load("salarios.RData")

# Boxplots
Boxplot(salarios$age, data=salarios, id=list(method="y"), ylab="Idade das
esposas")
Boxplot(salarios$husage, data=salarios, id=list(method="y"), ylab="Idade
dos maridos")

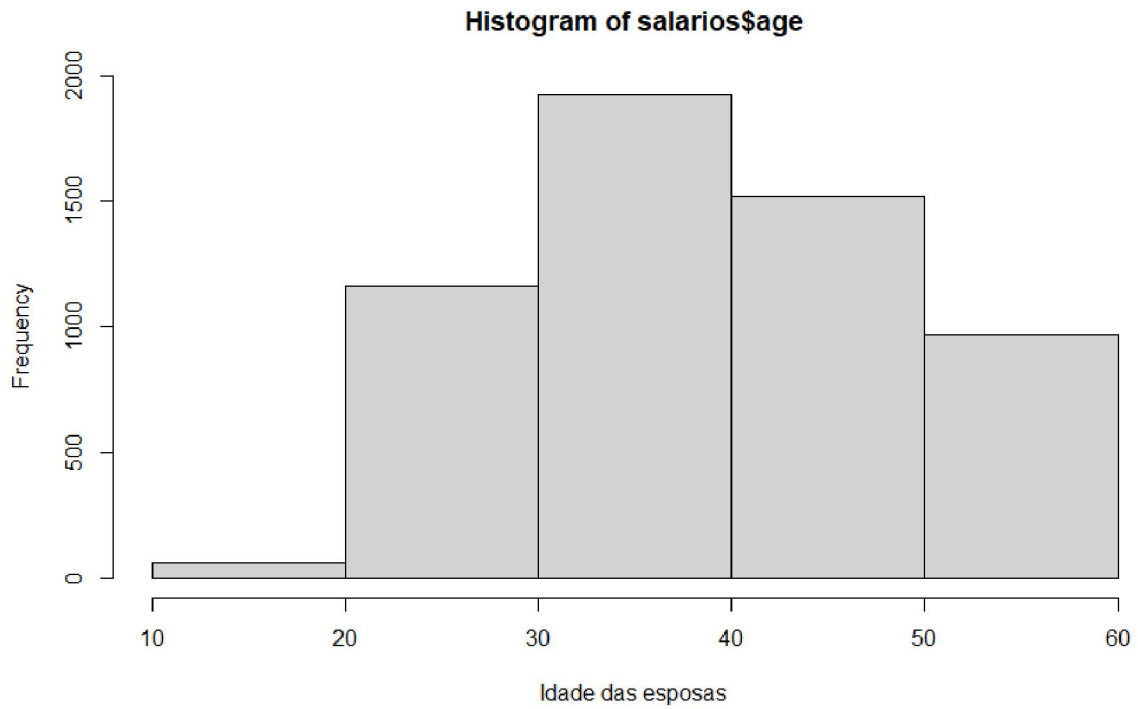
# Histogramas
hist(salarios$age, breaks = 5, xlab="Idade das esposas", ylab = "Frequency
")
hist(salarios$husage, breaks = 5, xlab="Idade dos maridos", ylab = "
Frequency")
```







FIGURA 15 – Histograma Idade das esposas



Fonte: O autor (2025).

FIGURA 16 – Histograma Idade das maridos



Fonte: O autor (2025).

Analisando os gráficos box-plot podemos observar que as medianas das esposas

e dos maridos parecem iguais, mas os limites superiores e inferiores diferem bastante, assim como a amplitude interquartílica. No caso o gráfico dos maridos possui o limite superior maior que as esposas além de possuir outliers. Já o gráfico das esposas tem maior amplitude interquartílica que o dos maridos.

Analisando os histogramas, observamos que há uma presença maior de esposas e maridos com idades entre 20 e 50 anos. À partir deste recorte a quantidade dos maridos cai drasticamente, porém, se alongando até acima de 80 anos. Já as mulheres, tem presença considerável até 60 anos e nenhum elemento acima desta idade.

**b) Elaborar a tabela de frequências das variáveis “age” (idade da esposa) e “husage” (idade do marido) e comparar os resultados**

```
# TABELA DE FREQUENCIA DA VARIÁVEL AGE (IDADE DA ESPOSA)
table_esposas <- fdt(salarios$age)
```

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

```
# TABELA DE FREQUENCIA DA VARIÁVEL HUSAGE (IDADE DO MARIDO)
table_maridos <- fdt(salarios$husage)
```

| 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 |

Comparando as tabelas de frequência, podemos observar que as classes de frequências mais populosas para as esposas são de 29 a 35 anos e para os maridos de 33 a 43 anos. E a dispersão de idades é maior entre os maridos.

## 2 Medidas de posição e dispersão

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

```
# MEDIA
mean(salarios$age) # 39.42758
mean(salarios$husage) # 42.45296

((39.42758/42.45296)-1)*100 # -7.126429

# A media das idades das esposas eh de aproximadamente
# 7% menor do que dos maridos.
```

---

```
# MEDIANA
median(salarios$age) # 39
median(salarios$husage) # 41

((39/41)-1)*100 # -4.878049

# A mediana das idades das esposas eh de aproximadamente
# 5% menor do que dos maridos.
```

---

```
# MODA
table(salarios$age)
subset(table(salarios$age),
  table(salarios$age) == max(table(salarios$age))) # 37

table(salarios$husage)
subset(table(salarios$husage),
  table(salarios$husage) == max(table(salarios$husage))) # 44
```

```
# A moda da idade das esposas eh de 37 anos, com 217 pessoas.
# A moda da idade dos maridos eh de 44 anos, com 201 pessoas.
```

```
((37/44)-1)*100 # -15.90909
# A moda de idade das esposas eh aproximadamente
# 16% menor do que dos maridos.
```

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

```
# VARIANCIA
var(salarios$age)
var(salarios$husage)

# A variancia das idades das esposas eh 99.75234
# A variancia das idades dos maridos eh 126.0717

((99.75234/126.0717)-1)*100 # -20.8765

# A variancia das idades das esposas eh aproximadamente
# 21% menor do que a variancia das idades dos maridos
```

```
# DESVIO PADRAO
sd(salarios$age)
sd(salarios$husage)

# O desvio padrao das idades das esposas eh 9.98761
# O desvio padrao das idades dos maridos eh 11.22817

((9.98761/11.22817)-1)*100 # -11.04864

# O desvio padrao das idades das esposas eh aproximadamente
# 11% menor que dos maridos
```

```
# COEFICIENTE DE VARIACAO DAS VARIÁVEIS
meanE <- mean(salarios$age)
meanM <- mean(salarios$husage)

sdE <- sd(salarios$age)
sdM <- sd(salarios$husage)

cvM <- (sdM/meanM)*100 # 26.44849
```

```

cvM
cvE <- (sdE/meanE)*100 # 25.33153
cvE

# O coeficiente de variacao do rendimento das esposas
# eh de aproximadamente 26% e dos maridos eh de
# aproximadamente 25%.
# Ambos possuem dispersao media.

```

### 3 Testes paramétricos ou não paramétricos

**a) 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.**

Checagens preliminares para verificar as exigências do teste: 1) Amostras independentes, 2) normalidade e 3) homogeneidade das variâncias entre grupos 4) outliers

- 1) Amostras independentes: Sim, pois os grupos de esposas e maridos provêm de indivíduos distintos.
- 2) Normalidade: Não, utilizando a regra de bolso, como o p-value eh inferior a 0.05, a variavel “idade” NÃO é normalmente distribuída .

```

# Executando teste de normalidade de Kolmogorov–Smirnov
lillie.test(idades$idade)
# p-value = 0.00000000000000022

```

- 3) Homogeneidade das variâncias: Não. Conforme observa-se no teste abaixo, o valor da estatística F calculada é 0.79123. Como esse valor se encontra na região de rejeição de H0, então rejeitamos a hipótese de que as variâncias são estatisticamente iguais.

```

# Vamos usar o teste F com as seguintes hipoteses:

# H0: As variancias sao estatisticamente iguais(homogeneas)
# HA: As variancias nao sao estatisticamente iguais(homogeneas)

# Executando o teste F

```



```

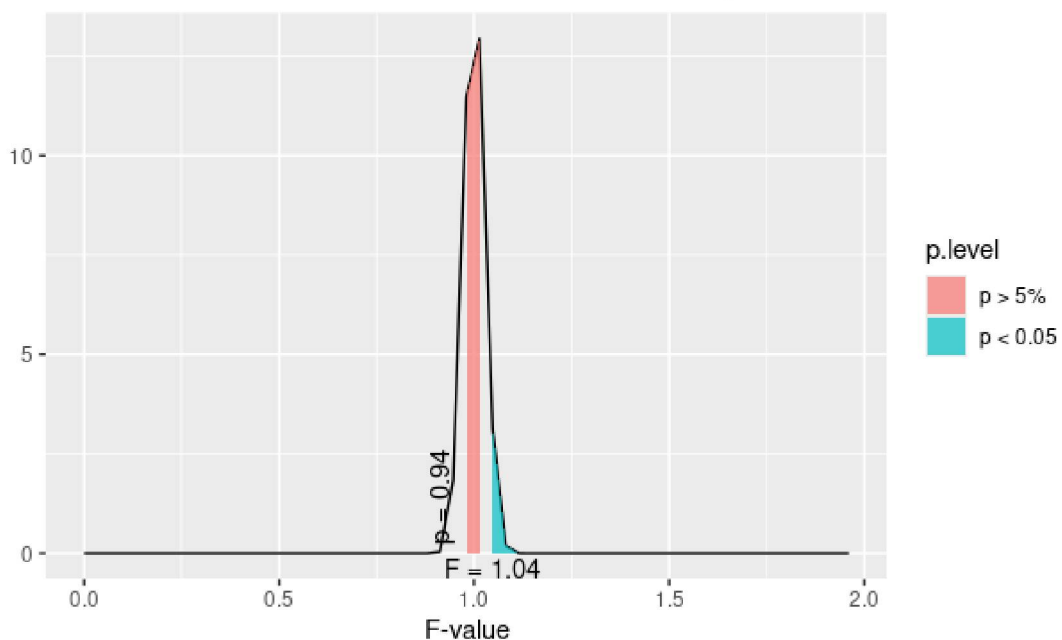
res.ftest <- var.test(idade ~ group, data = idades)
res.ftest

# Obtendo o valor tabelado da distribuicao F
qf(0.95, 5633, 5633)
# temos F=1,04
# para a outra cauda temos:
1/1.04
# F = 0,96

# Vamos construir o grafico:
dist_f(f = 1.04, deg.f1 = 5633, deg.f2 = 5633)
dist_f(f = 0.96, deg.f1 = 5633, deg.f2 = 5633)

```

FIGURA 17 – Gráfico da distribuição F



Fonte: O autor (2025).

4) Outliers: Sim. Ao observar o boxplot dos maridos percebe-se alguns outliers.

Dada as checagens anteriores de que a distribuição da variável idade NÃO é normalmente distribuída, as variâncias não serem estatisticamente iguais e também haverem outliers, não podemos usar um teste paramétrico para testar os dados, portanto utilizaremos o teste não paramétrico Mann Whitney U Test.

Queremos saber se a idade mediana das esposas difere da idade mediana dos maridos. Portanto, testando se a idade mediana dos maridos é igual a idade mediana das esposas temos as seguintes hipóteses:

- H0: A idade mediana dos maridos é estatisticamente igual a idade mediana das esposas
- Ha: A idade mediana dos maridos não é estatisticamente igual a idade mediana das esposas

```
# Executando o teste Mann Whitney U Test
res <- wilcox.test(idade ~ group, data = idades,
                  exact = FALSE, conf.int=TRUE)
res
```

```
Wilcoxon rank sum test with continuity correction

data:  idade by group
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
 -3.000024 -2.000033
sample estimates:
difference in location
 -2.999966
```

O p-value do teste é 0.00000000000000022, que é menor que o nível de significância 0,05. Podemos concluir que a idade mediana dos maridos é estatisticamente diferente da idade mediana das esposas (rejeitamos H0). O intervalo de confiança da diferença entre as medianas está entre -3.000024 e -2.000033, com uma mediana de -2.999966.

## 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  $R^2$ ) e concluir qual o melhor modelo entre os três, e mostrar o resultado da predição com intervalos de confiança para os seguintes valores:

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

obs: lembre-se de que a variável dependente “lwage” já está em 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

a) 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).

### PADRONIZAÇÃO DOS DADOS

```
cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'earns', 'age', 'educ', '
  exper', 'lwage')
pre_proc_val <- preProcess(train[,cols], method = c("center", "scale"))

train[,cols] = predict(pre_proc_val, train[,cols])
test[,cols] = predict(pre_proc_val, test[,cols])

cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'husblack', 'hushisp',
  'hushrs',
  'kidge6', 'age', 'black', 'educ', 'hispanic', 'union', 'exper', 'kidlt6', 'lwage')

dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+
  husblack+
  hushisp+hushrs+kidge6+age+black+educ+hispanic+union+exper+kidlt6,
  data = dat[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])

x_train = as.matrix(train_dummies)
y_train = train$lwage

x_test = as.matrix(test_dummies)
y_test = test$lwage
```

### REGRESSÃO RIDGE

```
lambdas <- 10^seq(5, -5, by = -1)
ridge_lamb <- cv.glmnet(x_train, y_train, alpha = 0, lambda = lambdas,
  nfolds = 10)

best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge
```

```

ridge_reg = glmnet(x_train, y_train, nlambda = 50, alpha = 0, family = '
    gaussian',
lambda = best_lambda_ridge)
ridge_reg

# Resultado
# (coeficientes)
ridge_reg[["beta"]]

predictions_train_ridge <- predict(ridge_reg, s = best_lambda_ridge, newx
    = x_train)

```

Visualizando o resultado da estimativa dos coeficientes percebemos que nem todas as variáveis explicam a variável dependente. Infelizmente as que explicam tem valor explicativo muito baixo, sendo somente as com certa relevância “husearns”, “husblk”, “hushisp”, “kidge6”, “black”, “educ” e “union”.

```

15 x 1 sparse Matrix of class "dgCMatrix"
      s0
husage    0.02651324
husunion -0.04174920
husearns  0.23671354
huseduc   0.04804650
husblk    0.24319544
hushisp   0.11457810
hushrs    -0.07293064
kidge6    -0.16442536
age        0.04466390
black     -0.29066663
educ       0.30859463
hispanic  -0.09979930
union      0.41614645
exper     -0.01411404
kidlt6    -0.02475760

```

## REGRESSÃO LASSO

```

lambdas <- 10^seq(5, -5, by = -.1)

lasso_lamb <- cv.glmnet(x_train, y_train, alpha = 1, lambda = lambdas,
    standardize = TRUE, nfolds = 10)

```

```

best_lambda_lasso <- lasso_lambd$lambda.min
best_lambda_lasso

lasso_model <- glmnet(x_train, y_train, alpha = 1, lambda = best_lambda_lasso,
  lasso,
  standardize = TRUE)

lasso_model[["beta"]]

predictions_train_lasso <- predict(lasso_model, s = best_lambda_lasso,
  newx = x_train)

```

Visualizando o resultado da estimativa dos coeficientes percebemos que nem todas as variáveis explicam a variável dependente. Infelizmente as que explicam tem valor explicativo muito baixo, sendo somente as com certa relevância “husearns”, “husblk”, “hushisp”, “kidge6”, “black”, “educ” e “union”. Enquanto a variável “exper” teve tão pouco valor explicativo que foi a zero.

```

15 x 1 sparse Matrix of class "dgCMatrix"
      s0
usage    0.02406805
union    -0.04144957
husearns 0.23903324
huseduc  0.04539082
husblk   0.26797943
hushisp  0.11329929
hushrs   -0.07369940
kidge6   -0.16534640
age       0.03331493
black    -0.31556379
educ      0.31552507
hispanic -0.09716331
union     0.41807407
exper     .
kidlt6   -0.02464497

```

## REGRESSÃO ELASTICNET

```

train_cont <- trainControl(method = "repeatedcv",
  number = 10,

```

```

        repeats = 5,
        search = "random",
        verboselter = FALSE)

elastic_reg <- train(lwage~husage+husunion+husearns+huseduc+husbck+
  hushisp+
  hushrs+kidge6+age+black+educ+hispanic+union+exper+kidlt6,
  data = train,
  method = "glmnet",
  tuneLength = 20,
  trControl = train_cont)

# O melhor parametro alpha escolhido eh:
#   alpha   lambda
#15  0.8015939  0.01293922
elastic_reg$bestTune

# E os parametros sao:
#elastic_reg[["finalModel"]][["beta"]]

predictions_train_elasticnet <- predict(elastic_reg, x_train)

```

## COMPARANDO RESULTADOS

```

eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))

  # As metricas de performace do modelo:
  data.frame(
    RMSE = RMSE,
    Rsquare = R_square
  )
}

eval_with_params <- function(model, best_lambda = NULL) {
  husage = (40-pre_proc_val[["mean"]][["husage"]])/pre_proc_val[["std"]][["husage"]]
  husunion = 0

```

```

husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
pre_proc_val[["std"]][["husearns"]]
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]])/pre_proc_val[["std"]][["huseduc"]]
husblck = 1
hushisp = 0
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/pre_proc_val[["std"]][["hushrs"]]
kidge6 = 1
age = (38-pre_proc_val[["mean"]][["age"]])/pre_proc_val[["std"]][["age"]]
black = 0
educ = (13-pre_proc_val[["mean"]][["educ"]])/pre_proc_val[["std"]][["educ"]]
hispanic = 1
union = 0
exper = (18-pre_proc_val[["mean"]][["exper"]])/pre_proc_val[["std"]][["exper"]]
kidlt6 = 1

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

prediction <- predict(model, s = best_lambda, newx = our_pred)

if (is.null(best_lambda)) {
  prediction <- predict(model, our_pred)
}

wage_pred=(prediction*pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][["lwage"]]

```



```

cat("Predicao: ", wage_pred, "\n")
cat("Predicao exp: ", exp(wage_pred), "\n")

return(wage_pred)
}

show_values <- function(wage_pred) {

  # O intervalo de confianca
  n <- nrow(train)
  m <- wage_pred
  s <- pre_proc_val[["std"]][["lwage"]]
  dam <- s/sqrt(n)
  Cllwr <- m + (qnorm(0.025))*dam # intervalo inferior
  Clupr <- m - (qnorm(0.025))*dam # intervalo superior

  cat("Intervalo inferior: ", Cllwr, "\n")
  cat("Intervalo superior: ", Clupr, "\n")

  cat("Intervalo inferior exp: ", exp(Cllwr), "\n")
  cat("Intervalo superior exp: ", exp(Clupr), "\n")
}

# ----- RIDGE ----- #

eval_results(y_train, predictions_train_ridge, train)

predictions_test <- predict(ridge_reg, s = best_lambda_ridge, newx = x_test)
eval_results(y_test, predictions_test, test)

predict_our_ridge <- eval_with_params(ridge_reg, best_lambda = best_lambda_ridge)

show_values(predict_our_ridge)

# ----- LASSO ----- #

eval_results(y_train, predictions_train_lasso, train)

```

```

predictions_test <- predict(lasso_model, s = best_lambda_lasso, newx = x_test)
eval_results(y_test, predictions_test, test)

predict_our_lasso <- eval_with_params(lasso_model, best_lambda = best_lambda_lasso)

show_values(predict_our_lasso)

# ----- ELATICNET ----- #

eval_results(y_train, predictions_train_elasticnet, train)
predictions_test <- predict(elastic_reg, x_test)
eval_results(y_test, predictions_test, test)

predict_our_elastic <- eval_with_params(elastic_reg)

show_values(predict_our_elastic)

```

Observando a Tabela 3 abaixo observamos que os valores para as métricas de todos os modelos são muito próximas e também não muito boas, não descartando a hipótese de overfitting ou underfitting. Mesmo assim, a título de comparação observamos que o modelo Lasso se saiu ligeiramente melhor do que os outros para o conjunto de dados de treinamento, enquanto o modelo Ridge se saiu ligeiramente melhor nos dados de teste.

TABELA 3 – Tabela de métricas dos modelos

|            | Treinamento |           | Teste     |           |
|------------|-------------|-----------|-----------|-----------|
|            | RMSE        | Rsquare   | RMSE      | Rsquare   |
| Ridge      | 0.8485564   | 0.2796022 | 0.8488229 | 0.3004756 |
| Lasso      | 0.8485375   | 0.2796343 | 0.8488541 | 0.3004242 |
| ElasticNet | 0.8495878   | 0.2778498 | 0.8490653 | 0.3000762 |

Fonte: O autor (2025).

Na Tabela 4 pode-se observar os valores preditos pelos modelos para as variáveis especificadas inicialmente.

TABELA 4 – Tabela de Predição e Intervalos de Confiança

|            | Predição | Intervalos de Confiança |
|------------|----------|-------------------------|
| Ridge      | 8.633904 | 8.442068   8.8301       |
| Lasso      | 8.746603 | 8.552263   8.945359     |
| ElasticNet | 8.160874 | 7.979548   8.34632      |

Fonte: O autor (2025).

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

Entrada: "texto qualquer"

Saída: português ou inglês ou francês ou italiano ou...

## O processo de Reconhecimento de Padrões

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

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

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

```

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

```

```

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

```

```

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
]

```

```

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

```

A “amostras” de texto precisa ser “transformada” em padrões.

Um padrão é um conjunto de características, geralmente representado por um vetor e um conjunto de padrões no formato de tabela. Onde cada linha é um padrão e as colunas as características e, geralmente, na última coluna a classe.

```

import random

pre_padroes = []
for frase in ingles:
    pre_padroes.append( [frase, 'inglês'])

```

```

for frase in espanhol:
    pre_padroes.append( [frase, 'espanhol'])

for frase in portugues:
    pre_padroes.append( [frase, 'português'])

random.shuffle(pre_padroes)

import pandas as pd
dados = pd.DataFrame(pre_padroes)
dados

```

TABELA 5 – Frases com seus respectivos idiomas

| ID  | Frase                                            | Idioma    |
|-----|--------------------------------------------------|-----------|
| 0   | Me gusta cocinar la cena en casa.                | espanhol  |
| 1   | Vamos fazer uma viagem de carro.                 | português |
| 2   | Do you have any recommendations for restaurants? | inglês    |
| 3   | Quero aprender a tocar violão.                   | português |
| 4   | Hola, ¿cómo estás?                               | espanhol  |
| ... | ...                                              | ...       |
| 87  | Vamos sair para jantar no sábado.                | português |
| 88  | Estoy emocionado por el concierto.               | espanhol  |
| 89  | Estoy aprendiendo a tocar el piano.              | espanhol  |
| 90  | Preciso fazer exercícios físicos regularmente.   | português |
| 91  | Quiero aprender italiano.                        | espanhol  |

Fonte: O autor (2025).

## Construção dos atributos

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

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

```

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

import re
from sklearn.feature_extraction.text import CountVectorizer

def tamanhoMedioFrases(texto):
    palavras = re.split("\s", texto)

```



```

#print(palavras)
tamanhos = [len(s) for s in palavras if len(s)>0]
#print(tamanhos)
soma = 0
for t in tamanhos:
    soma=soma+t
return soma / len(tamanhos)

def temCaractere(texto, simbolo):
    return int(simbolo in texto)

def gerarNGrams(list_textos, n, max_f):
    vec = CountVectorizer(analyzer='char', ngram_range=(n,n),
        max_features=max_f)
    ngrams = vec.fit_transform(list_textos)
    feature_names = vec.get_feature_names_out()

    return feature_names

def extraiCaracteristicas(frase):
    # frase é um vetor [ 'texto', 'lingua' ]
    texto = frase[0]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
    #print(texto)
    caracteristica1=tamanhoMedioFrases(texto)

    #ESTRATEGIA 1: buscando caracteres especiais
    caracteristica2=temCaractere(texto, 'ç')
    caracteristica3=temCaractere(texto, 'ñ')
    caracteristica12=temCaractere(texto, 'à')
    caracteristica13=temCaractere(texto, 'ón')
    caracteristica14=temCaractere(texto, 'ã')
    caracteristica15=temCaractere(texto, 'ay')
    caracteristica16=temCaractere(texto, 'yo')

    #ESTRATEGIA 2: buscando caracteres iguais sucessivos
    caracteristica4=temCaractere(texto, 'cc')
    caracteristica5=temCaractere(texto, 'oo')
    caracteristica6=temCaractere(texto, 'll')
    caracteristica7=temCaractere(texto, 'ss')
    caracteristica8=temCaractere(texto, 'mm')

```

```

caracteristica9=temCaractere(texto, 'nn')
caracteristica10=temCaractere(texto, 'rr')
caracteristica11=temCaractere(texto, 'ee')

#ESTRATEGIA 3: buscando ocorrencia dos ngrams mais frequentes
por lingua
features_ngrams = []

for n in ngrams_populares_p_lingua:
    features_ngrams.append(int(n in texto))

# acrescente as suas funcoes no vetor padrao
padrao = [caracteristica1, caracteristica2, caracteristica3,
          caracteristica4, caracteristica5, caracteristica6,
          caracteristica7, caracteristica8, caracteristica9,
          caracteristica10, caracteristica11,
          caracteristica12, caracteristica13, caracteristica14,
          caracteristica15, caracteristica16,
          *features_ngrams,
          frase[1]]

return padrao

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

#GERA 88 1,2,3,4-GRAMS MAIS COMUNS PARA CADA LINGUA
#set para evitar repetidos
ngrams_populares_p_lingua = set()
max_features = 38 # ou 88
for nsize in range(1, 5): # ou 4
    ngrams_populares_p_lingua.update(gerarNGrams(portugues, nsize,
        max_features))
    ngrams_populares_p_lingua.update(gerarNGrams(ingles, nsize,
        max_features))
    ngrams_populares_p_lingua.update(gerarNGrams(espanhol, nsize,
        max_features))

```

```
# converte o formato [frase classe] em
# [caracteristica_1, caracteristica_2,... caracteristica n,
#  classe]
padroes = geraPadroes(pre_padroes)

#
# apenas para visualizacao
print(padroes)

dados = pd.DataFrame(data=padroes, columns=['TMediaFrase', 'ç', 'ñ',
      'cc', 'oo', 'll', 'ss', 'mm', 'nn', 'rr', 'ee', 'à', 'ón', 'ã', 'ay',
      'yo', *ngrams_populares_p_lingua, 'Lingua'])
dados
```

## Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

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

#from sklearn.metrics import confusion_matrix

vet = np.array(padroes)
classes = vet[:, -1]          # classes = [p[-1] for p in padroes]
#print(classes)
padroes_sem_classe = vet[:, 0:-1]
#print(padroes_sem_classe)
X_train, X_test, y_train, y_test = train_test_split(
    padroes_sem_classe, classes, test_size=0.20, stratify=classes
)
```

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

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

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

```

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

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

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

```

```

Acuracia nos dados de treinamento: 98.63%
[[24  0  0]
 [ 0 24  0]
 [ 1  0 24]]

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| espanhol     | 0.96      | 1.00   | 0.98     | 24      |
| ingles       | 1.00      | 1.00   | 1.00     | 24      |
| portugues    | 1.00      | 0.96   | 0.98     | 25      |
| accuracy     |           |        | 0.99     | 73      |
| macro avg    | 0.99      | 0.99   | 0.99     | 73      |
| weighted avg | 0.99      | 0.99   | 0.99     | 73      |

```

metricas mais confiaveis
[[6 0 0]
 [0 5 1]
 [4 0 3]]

```

|          | precision | recall | f1-score | support |
|----------|-----------|--------|----------|---------|
| espanhol | 0.60      | 1.00   | 0.75     | 6       |



|              |      |      |      |    |
|--------------|------|------|------|----|
| ingles       | 1.00 | 0.83 | 0.91 | 6  |
| portugues    | 0.75 | 0.43 | 0.55 | 7  |
| accuracy     |      |      | 0.74 | 19 |
| macro avg    | 0.78 | 0.75 | 0.73 | 19 |
| weighted avg | 0.78 | 0.74 | 0.72 | 19 |

## 2 Melhore uma base de dados ruim

```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
from sklearn.preprocessing import StandardScaler

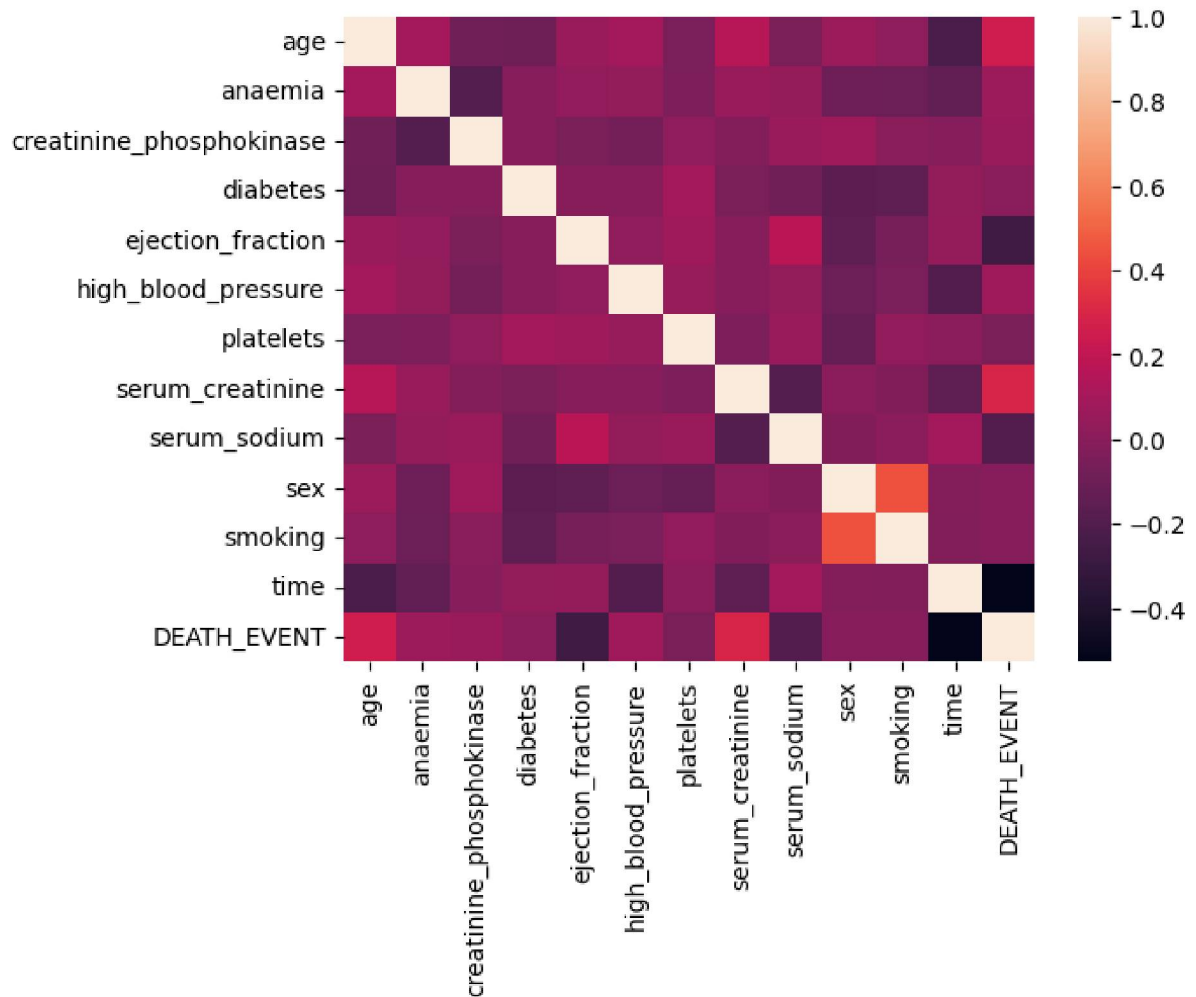
df = pd.read_csv('heart_failure_clinical_records_dataset.csv')
df.describe()
```

| Stat  | age     | anaemia | creatinine_phosphokinase | diabetes | ejection_fraction | high_blood_pressure | platelets  |
|-------|---------|---------|--------------------------|----------|-------------------|---------------------|------------|
| count | 299.000 | 299.000 | 299.000                  | 299.000  | 299.000           | 299.000             | 299.000    |
| mean  | 60.834  | 0.431   | 581.839                  | 0.418    | 38.084            | 0.351               | 263358.029 |
| std   | 11.895  | 0.496   | 970.288                  | 0.494    | 11.835            | 0.478               | 97804.237  |
| min   | 40.000  | 0.000   | 23.000                   | 0.000    | 14.000            | 0.000               | 25100.000  |
| 25%   | 51.000  | 0.000   | 116.500                  | 0.000    | 30.000            | 0.000               | 212500.000 |
| 50%   | 60.000  | 0.000   | 250.000                  | 0.000    | 38.000            | 0.000               | 262000.000 |
| 75%   | 70.000  | 1.000   | 582.000                  | 1.000    | 45.000            | 1.000               | 303500.000 |
| max   | 95.000  | 1.000   | 7861.000                 | 1.000    | 80.000            | 1.000               | 850000.000 |

| Stat  | serum_creatinine | serum_sodium | sex     | smoking | time    | DEATH_EVENT |
|-------|------------------|--------------|---------|---------|---------|-------------|
| count | 299.000          | 299.000      | 299.000 | 299.000 | 299.000 | 299.000     |
| mean  | 1.394            | 136.625      | 0.649   | 0.321   | 130.261 | 0.321       |
| std   | 1.035            | 4.412        | 0.478   | 0.468   | 77.614  | 0.468       |
| min   | 0.500            | 113.000      | 0.000   | 0.000   | 4.000   | 0.000       |
| 25%   | 0.900            | 134.000      | 0.000   | 0.000   | 73.000  | 0.000       |
| 50%   | 1.100            | 137.000      | 1.000   | 0.000   | 115.000 | 0.000       |
| 75%   | 1.400            | 140.000      | 1.000   | 1.000   | 203.000 | 1.000       |
| max   | 9.400            | 148.000      | 1.000   | 1.000   | 285.000 | 1.000       |

```
import seaborn as sns
corr = df.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
```

FIGURA 18 – Mapa de correlação dados clínicos



Fonte: O autor (2025).

```
X_orig = df.iloc[:, :-1]
y_orig = df['DEATH_EVENT']

X_train_orig, X_test_orig, y_train, y_test = train_test_split(
    X_orig, y_orig, test_size=0.20, stratify=y_orig, random_state=
    =10)

treinador = svm.SVC() #algoritmo escolhido
```

```

modelo_orig = treinador.fit(X_train_orig, y_train)

# predição com os mesmos dados usados para treinar
y_pred = modelo_orig.predict(X_train_orig)
cm_orig_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusao - com os dados ORIGINAIS usados no
      TREINAMENTO')
#print(cm_orig_train)
print(classification_report(y_train, y_pred, zero_division=0))

# predição com os mesmos dados usados para testar
y2_pred = modelo_orig.predict(X_test_orig)
cm_orig_test = confusion_matrix(y_test, y2_pred)
print('Matriz de confusao - com os dados ORIGINAIS usados para
      TESTES')
#print(cm_orig_test)
print(classification_report(y_test, y2_pred, zero_division=0))

```

```

Matriz de confusao - com os dados ORIGINAIS usados no TREINAMENTO
      precision    recall  f1-score   support

      0        0.68        1.00        0.81        162
      1        0.00        0.00        0.00         77

   accuracy                0.68        239
  macro avg        0.34        0.50        0.40        239
 weighted avg        0.46        0.68        0.55        239

Matriz de confusao - com os dados ORIGINAIS usados para TESTES
      precision    recall  f1-score   support

      0        0.68        1.00        0.81         41
      1        0.00        0.00        0.00         19

   accuracy                0.68         60
  macro avg        0.34        0.50        0.41         60
 weighted avg        0.47        0.68        0.55         60

```

```

y = df['DEATH_EVENT']
X = df.drop(['diabetes', 'time', 'DEATH_EVENT'],axis = 1)

```

```

X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.20, stratify=y_orig, random_state=10)

scaler = StandardScaler().fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

treinador = svm.SVC() #algoritmo escolhido

modelo_orig = treinador.fit(X_train_scaled, y_train)

# predição com os mesmos dados usados para treinar
y_pred = modelo_orig.predict(X_train_scaled)
cm_orig_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusao - com os dados escalonados e de alta
    correlacao usados no TREINAMENTO')
#print(cm_orig_train)
print(classification_report(y_train, y_pred, ))

# predição com os mesmos dados usados para testar
y2_pred = modelo_orig.predict(X_test_scaled)
cm_orig_test = confusion_matrix(y_test, y2_pred)
print('Matriz de confusao - com os dados escalonados e de alta
    correlacao usados no TESTE')
#print(cm_orig_test)
print(classification_report(y_test, y2_pred))

```

Matriz de confusao - com os dados escalonados e de alta  
correlacao usados no TREINAMENTO

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.83      | 0.96   | 0.89     | 162     |
| 1            | 0.88      | 0.60   | 0.71     | 77      |
| accuracy     |           |        | 0.85     | 239     |
| macro avg    | 0.86      | 0.78   | 0.80     | 239     |
| weighted avg | 0.85      | 0.85   | 0.84     | 239     |

Matriz de confusao - com os dados escalonados e de alta  
correlacao usados no TESTE

|  | precision | recall | f1-score | support |
|--|-----------|--------|----------|---------|
|--|-----------|--------|----------|---------|

|              |      |      |      |    |
|--------------|------|------|------|----|
| 0            | 0.78 | 0.85 | 0.81 | 41 |
| 1            | 0.60 | 0.47 | 0.53 | 19 |
| accuracy     |      |      | 0.73 | 60 |
| macro avg    | 0.69 | 0.66 | 0.67 | 60 |
| weighted avg | 0.72 | 0.73 | 0.72 | 60 |

## APÊNDICE 7 – 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 proporci-

onal 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**

### **Estudo de Caso: Implicações Éticas do Uso do ChatGPT**

#### **RESUMO**

O rápido avanço da inteligência artificial (IA) levou ao desenvolvimento de modelos de linguagem sofisticados como o ChatGPT. Este trabalho explora as implicações éticas do uso de tal ferramenta, focando em seu impacto em áreas como educação, ambiente de trabalho e meio artístico. Ao examinar tais estudos de caso específicos, este trabalho identificou alguns dilemas éticos significativos e analisou seus efeitos sobre diferentes partes interessadas. Além disso, o presente trabalho propõe algumas possíveis soluções responsáveis para que esses dilemas sejam superados, promovendo um uso mais positivo para a sociedade em geral. O objetivo é fornecer uma compreensão abrangente do panorama ético em torno do ChatGPT e sugerir políticas que garantam seu uso benéfico e equitativo na sociedade.

Palavras-chave: Ética. ChatGPT. Inteligência artificial. Análise crítica. Soluções responsáveis.

#### **ABSTRACT**

The rapid advance of artificial intelligence (AI) has led to the development of sophisticated language models such as ChatGPT. This paper explores the ethical implications of using such a tool, focusing on its impact in areas such as education, the workplace and the arts. By examining such specific case studies, this paper has identified some significant ethical dilemmas and analyzed their effects on different



stakeholders. In addition, this paper proposes some possible responsible solutions so that these dilemmas can be overcome, promoting a more positive use for society in general. The aim is to provide a comprehensive understanding of the ethical landscape surrounding ChatGPT and to suggest policies that ensure its beneficial and equitable use in society.

Keywords: Ethics. ChatGPT. Artificial intelligence. Critical analysis. Responsible solutions.

## 1 INTRODUÇÃO

A inteligência artificial (IA) tem evoluído rapidamente e ganhado muita popularidade recentemente. Hoje em dia, falar sobre IA é um assunto muito comum, diferente de alguns poucos anos atrás, e grande parte das pessoas acaba por utilizar serviços que utilizam inteligência artificial de alguma forma ou de outra mesmo sem saber.

Esse rápido avanço abre espaço para discussões éticas e filosóficas de como essa tecnologia pode e poderá afetar a vida das pessoas no futuro. Uma das aplicações recentes mais notáveis é o ChatGPT, uma tecnologia de IA capaz de gerar textos e realizar conversas de maneira natural.

Este trabalho tem como objetivo explorar algumas implicações éticas do uso do ChatGPT em diversos cenários, tais como na educação, no ambiente de trabalho e também no meio artístico. Além de explorar tais implicações também investigar alguns dos dilemas que a inteligência artificial em geral apresenta hoje e por fim propor algumas soluções responsáveis para os problemas identificados.

A análise será estruturada em duas seções principais: análise crítica, onde abordaremos estudos de casos específicos para ilustrar os impactos éticos da tecnologia; e soluções responsáveis, onde serão propostas medidas para mitigar os desafios identificados e promover o uso justo e transparente do ChatGPT em tais estudos de caso. Ao realizar essa análise, esperamos contribuir para um entendimento mais profundo das implicações éticas da inteligência artificial e para o desenvolvimento de boas práticas que garantam o seu uso benéfico na sociedade, minimizando os potenciais usos negativos.

### 1.1 CHATGPT

O ChatGPT é um modelo de inteligência artificial que pode entender e gerar textos como se fosse uma pessoa real. Ele foi criado para conversar com as pessoas, responder perguntas, fornecer informações e até mesmo ajudar em algumas tarefas. A tecnologia de inteligência artificial permite que o ChatGPT aprenda com diversos exemplos de conversas e textos, o que o ajuda a fornecer respostas úteis e em grande parte também corretas.

De maneira mais técnica, o ChatGPT é um modelo de linguagem treinado para gerar texto e foi otimizado para diálogos, utilizando aprendizado por reforço, com feedback humano, um método que utiliza demonstrações humanas e comparações de preferências para orientar o modelo em direção ao comportamento desejado (OpenAI, 2024).

## **2 ANÁLISE CRÍTICA**

### **2.1 ESTUDO DE CASO 1: USO NA EDUCAÇÃO**

No contexto educacional o ChatGPT pode ser uma ferramenta valiosa para auxiliar professores no ensino dos alunos. Tal ferramenta é capaz de gerar respostas de forma rápida e diferentes, ajudando na resolução de problemas. Entretanto, há preocupações em relação ao seu uso, tanto para o corpo docente como para o discente, possibilitando a geração de um determinado nível de dependência dessa tecnologia, podendo comprometer suas capacidades intelectuais, pensamento crítico e criatividade, sem o uso da ferramenta. O processo de transformação, principalmente o digital, exige dos profissionais envolvidos com a educação, um olhar diferenciado nos planejamentos e suas respectivas aplicações, pois há muitos que são analfabetos funcionais em conduzir determinadas metodologias no processo ensino-aprendizagem, um exemplo, são as metodologias ativas, em que há uma interpretação equivocada entre gamificação e jogos.

Nesse processo, o papel do professor não será como o de outrora em que tal personagem era a única fonte de conhecimento. Hoje e no futuro se desenharam cenários em que a docência exercerá muito mais o papel de mediação pedagógica entre os objetos de conhecimento e os estudantes. Por isso, o educar pela pesquisa e extensão se tornam tão significativos (Cassol, 2023).

Para que se tenha uma qualidade na construção do saber, a entrada das informações, que devem ser interativas e desafiadoras para o aluno em desenvolvimento para se tornar um estudante, o professor deverá estar em prontidão, não utilizando a ferramenta como “bengala” para conduzir suas aulas, bem como oferecer um suporte com maior qualidade e orientações possibilitando uma diversidade de caminhos para explorar o mundo dos saberes com consciência. Em relação aos discentes, respeitando o seu tempo de maturação na forma de perceber o mundo, fomentá-los por meio da disciplina em realizar as tarefas propostas, iniciando com uma boa orientação, sobre como aplicar a ferramenta nas determinadas áreas, utilizando a transdisciplinaridade e autoconhecimento para a ampliação da base de informações a serem utilizadas na construção do pensamento crítico, sendo mais assertivo nas perguntas a serem realizadas, aumentando a probabilidade de obter melhores respostas, e na sua interpretação e execução, obter um resultado mais original, personalizado; não necessitando se utilizar

do plágio ou qualquer outro artifício que não contribua com a melhoria contínua do ChatGPT, no contexto escolar e outras áreas.

## 2.2 ESTUDO DE CASO 2: USO NO AMBIENTE DE TRABALHO

No ambiente de trabalho, o uso de inteligências artificiais promete trazer um grande aumento da produtividade e precisão, além da simplificação de tarefas, o que permitiria liberar tempo para tarefas que exigem mais criatividade e pensamento crítico. O ChatGPT, nesse contexto, vem sendo visto como uma ótima ferramenta para análise de dados, organização de informações, automatização de tarefas repetitivas e geração de código, e já é amplamente popular em diversas áreas, principalmente dentro do mundo corporativo. Segundo Lopes (2024), 89% dos profissionais de TI e 86% do setor de marketing, já adotaram o uso do ChatGPT numa frequência, ao menos, mensal.

Apesar dos benefícios da introdução do ChatGPT no ambiente de trabalho, surgem, a partir disso, uma série de questionamentos e preocupações a respeito das mudanças que essa tecnologia pode causar.

Uma das áreas a serem afetadas seria o mercado de trabalho, já que um dos principais questionamentos é se a ferramenta poderia ser capaz de causar a extinção de algumas carreiras. O ChatGPT parece ter o potencial de substituir pessoas na realização de algumas tarefas, como por exemplo: revisão de contratos, resumo de textos, criação de apresentações, desenvolvimento de código, organização de cadeias de produção; afetando assim profissões como direito, jornalismo, engenharia de software e magistério (Sutto, 2023). Apenas a remota possibilidade de que isso aconteça, poderia ser a causa do aumento do nível de ansiedade de muitos trabalhadores, talvez, gerando um problema de saúde pública. Contudo, ao mesmo tempo, há a expectativa de que surjam novas áreas de trabalho, para profissionais especializados no uso da ferramenta, que sejam capazes de integrá-la em diferentes setores.

Além do mercado de trabalho, é válido buscar entender o efeito que o uso de ferramentas como o ChatGPT teria sobre o perfil dos trabalhadores. É importante cogitar que a intensificação do uso do ChatGPT poderá afetar as habilidades dos trabalhadores, no sentido de que ao depender muito da ferramenta como uma “muleta”, os profissionais acabem “atrofiando” suas habilidades. Poderia haver diminuição na habilidade de comunicação escrita, pois o uso da IA para gerar conteúdos textuais em alta frequência reduz o não aprimoramento e o esquecimento de técnicas de escrita. Por sua vez, a redução da habilidade de comunicação escrita também tem impacto na comunicação verbal, pois afeta a habilidade de formulação de argumentos e organização de idéias, e então pode ser a causa de mais problemas de comunicação no ambiente de trabalho.

A inibição da criatividade também poderia ser uma outra consequência, visto que

o uso do ChatGPT ajuda na otimização de produção de texto inédito porém não gera conhecimento novo, considerando que é treinado com conhecimento produzido no passado. A tendência é que o conteúdo produzido pelo uso recorrente do ChatGPT gere uma repetição ou reprodução de informações entre os profissionais, em variados setores e sem avanço ou melhoria. Em nome da otimização do tempo, a inovação é inibida e se diminui o hábito e o exercício da criatividade.

Surge também a dúvida sobre até que ponto uma inteligência artificial pode ser utilizada em tarefas que dependem de pesquisa e consulta a informações verdadeiras. Com uma compreensão mais aprofundada do funcionamento de inteligências artificiais, sabe-se que elas não são capazes de raciocinar, compreender o contexto e aspectos humanos relacionados, apenas são capazes de abstrair dados e são capazes de gerar dados fictícios quando não possuem as informações verdadeiras. Porém ferramentas como o ChatGPT passaram a ser usadas por muitos como uma fonte fácil de conhecimento verdadeiro, o que gera a necessidade de treinar profissionais para aprender a usar a ferramenta e analisar criticamente se o conteúdo gerado é válido ou não.

O aspecto da experiência do usuário também passa a ser afetado, devido ao aumento de ferramentas automatizadas para atendimento aos clientes. Não é incomum, ao acessar o site de uma empresa e ao se direcionar para o serviço de atendimento ao cliente, o mesmo ser redirecionado para uma conversa com robôs. Tais robôs em grande maioria se utilizam de ferramentas geradoras de textos como o ChatGPT para se comunicarem e auxiliarem na resolução de problemas comuns entre clientes de uma determinada empresa. Isso de certa forma pode melhorar a eficiência de diversos atendimentos, fornecendo respostas rápidas e quase sempre precisas. Entretanto, embora leve a um aumento na eficiência operacional do atendimento, muitas pessoas reclamam da “desumanização” nesses processos, onde os clientes geralmente sentem falta de interações humanas autênticas, principalmente se tratando de problemas mais complexos que nem sempre os robôs conseguem ajudar.

### 2.3 ESTUDO DE CASO 3: USO NO MEIO ARTÍSTICO

A inteligência artificial tem sido cada vez mais utilizada em diversas áreas criativas para a produção de novas artes, desde músicas até pinturas ou histórias. O ChatGPT em específico com sua enorme capacidade em gerar textos pode ser utilizado para criação de livros, roteiros, poesias, letras de músicas e até mesmo colaborar criativamente na criação de outros tipos de obras de arte. Para fins de simplicidade no texto chamaremos qualquer produção artística de obra.

Apesar dos erros e falhas ainda presentes, a habilidade já demonstrada pelo chat e seu potencial de se aprimorar ainda mais no futuro vem gerando não apenas admiração, mas também alguns receios (Suzuki, 2023). Este estudo de caso examina

as implicações éticas do uso do ChatGPT no meio artístico, analisando como tal tecnologia influencia a criatividade humana, a originalidade, os direitos autorais e a autenticidade das obras de artes.

Conforme brevemente citado anteriormente, o ChatGPT pode ser utilizado no meio artístico para várias finalidades como:

- Criação de histórias e roteiros: Escritores ou até mesmo cineastas podem utilizar a ferramenta para gerar diálogos, desenvolver roteiros e histórias, séries ou peças de teatro.
- Poesia e literatura: Poetas e escritores podem obter auxílio do ChatGPT para criar poemas, contos e romances.
- Composição musical: Músicos podem utilizar a ferramentas para escrever letras de música ou até mesmo compor melodias.
- Artes visuais: Artistas podem utilizar de um auxílio criativo do chat para gerar descrições e conceitos de obras de artes que podem ser transformados em arte visual.

A principal preocupação ética relacionada ao uso de inteligência artificial no meio artístico é a respeito da originalidade dos conteúdos gerados. A arte tradicionalmente valoriza a expressão individual e criativa que existe em cada ser humano, o uso de um modelo de inteligência artificial que é treinado a partir de obras já existentes rompe com essa tradição. Se uma obra é criada totalmente ou com certa assistência do ChatGPT, até que ponto ela pode ser considerada uma expressão genuína de um artista humano?

Partindo da preocupação anterior, além da originalidade artística, outro ponto a se considerar é o dos direitos autorais. Tais obras geradas criam enormes desafios para a determinação de autoria de propriedade intelectual. De maneira geral, se uma IA contribui para a criação de uma obra, quem deve ser creditado como autor? O desenvolvedor da IA, o usuário que a utilizou, ambos, ou os autores a qual o modelo se baseou durante o treinamento?

Além dessa preocupação com direitos autorais na criação de uma obra nova, também há a preocupação sobre a violação de direitos autorais sobre obras já existentes, tendo em vista que os modelos são treinados sobre um conjunto já existente de obras. Isso levanta a questão de possível infração de direito autoral já que o modelo pode gerar conteúdo semelhante a essas obras. E neste caso novamente fica a pergunta, quem deve ser responsabilizado? Em várias partes do mundo cresce um movimento que defende mudanças nas leis de direitos autorais para que elas passem a proteger as obras dos artistas humanos das obras geradas por inteligência artificial (Prado, 2023).

## 3 SOLUÇÕES RESPONSÁVEIS

### 3.1 PRIVACIDADE

Como toda tecnologia emergente, na Inteligência Artificial Generativa existem riscos em sua utilização. Uma das preocupações do uso generalizado desta tecnologia é o impacto na privacidade de dados. Desta forma, “para colher os benefícios dessa tecnologia inovadora, é crucial construir uma relação de confiança com o mercado e com os diversos *stakeholders*, diminuindo preocupações a respeito de como os dados serão utilizados.” (KPMG, 2024)

O movimento de órgãos reguladores é essencial para balizar o uso e definir os direitos e deveres. Por exemplo, o Parlamento Europeu publicou, em 2023, o AI Act e no Brasil estamos discutindo através do Projeto de Lei nº 2338, de 2023 (<https://www25.senado.leg.br/web/atividade/materias/-/materia/157233>). Com os balizadores criados diminuí-se as preocupações, tanto dos usuários, quanto dos *stackholders*. Garantindo aos usuários o controle e a consciência de como seus dados serão utilizados e possibilitando o crescimento exponencial no uso da tecnologia.

### 3.2 TRANSPARÊNCIA

Embora alguns modelos tenham uma certa natureza imprevisível, a transparência dos processos que ocorrem durante o treinamento e utilização de modelos geradores de texto são essenciais para construir a confiança dos usuários.

As empresas que desenvolvem esse tipo de tecnologia devem ser transparentes quanto a como seus modelos funcionam, incluindo detalhes sobre os dados utilizados para treinamento e os processos de tomada de decisão.

Utilizando como exemplo a AI Act, publicada em 2023, pelo Parlamento Europeu, temos a seguinte diretriz:

“A IA generativa, como o ChatGPT, não será classificada como de alto risco, mas terá que cumprir os requisitos de transparência e a lei de direitos autorais da UE:

- Divulgar que o conteúdo foi gerado por IA.
- Projetar o modelo para evitar a geração de conteúdo ilegal.
- Publicação de resumos de dados protegidos por direitos autorais usados para treinamento (EP, 2023).

Para o uso de transparência nos modelos, existem projetos open sources que facilitam a implantação, como o TrustyAI (<https://github.com/trustyai-explainability>).

### 3.3 VIESES

O viés de IA é como chamamos a ocorrência de resultados tendenciosos devido a vieses que distorcem os dados de treinamentos levando a resultados distorcidos. Na prática, os modelos absorvem preconceitos humanos, através das bases de treinamento, e os reproduzem em seus resultados.

Há uma série de vieses possíveis em um modelo de IA. Segundo o artigo “O que é viés de IA?” (IBM, 2023) temos os seguintes tipos:

- Viés do algoritmo
- Viés cognitivo
- Viés de confirmação
- Viés de exclusão
- Viés de medição
- Viés de homogeneidade fora do grupo
- Viés de preconceito
- Viés de recordação
- Viés de amostragem/seleção
- Viés de estereótipo

Para que a IA não possua vieses é necessário analisar os dados de treinamentos e tomar medidas para a diversidade dos dados, garantindo dados completos, imparciais e sem preconceitos.

## 4 CONCLUSÃO

A inteligência artificial representa um grande avanço tecnológico para a humanidade. O uso do ChatGPT em específico trouxe inúmeros benefícios capazes de transformar diversos setores da nossa sociedade, desde ambientes educacionais até meios artísticos. No entanto, com grandes poderes vêm grandes responsabilidades, e essa ferramenta poderosa também levanta questões éticas importantes a serem discutidas.

Ao longo deste trabalho exploramos somente algumas dessas implicações éticas, e analisamos como essas implicações afetam diferentes partes interessadas, desde usuários, desenvolvedores e a sociedade em geral. Além disso, propusemos algumas



soluções que podem ajudar a mitigar os problemas apresentados, promovendo um uso mais positivo, justo, transparente e ético da tecnologia.

É essencial que continuemos discutindo e desenvolvendo políticas, visando a garantia que a evolução da inteligência artificial esteja sempre alinhada com princípios éticos, tendo em vista a forte tendência de que a velocidade da evolução aumente cada vez mais. Somente assim poderemos, como sociedade, aproveitar todo o potencial que essa tecnologia tem a oferecer.

## APÊNDICE 8 – APRENDIZADO DE MÁQUINA

### A - ENUNCIADO

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

### B - RESOLUÇÃO

#### Classificação

Para o experimento de Classificação:

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

#### Veículo

TABELA 6 – Resultados da classificação na base de veículos

| Técnica  | Parâmetro            | Acurácia | Matriz de Confusão                                                                                                                                                                                                                                               |
|----------|----------------------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SVM - CV | C=100<br>Sigma=0.015 | 0.8623   | <pre>Confusion Matrix and Statistics                Reference Prediction bus opel saab van bus      41    0    1    0 opel     0   33    9    2 saab     0    7   33    0 van      2    2    0   37  Overall Statistics                  Accuracy : 0.8623</pre> |

| Técnica           | Parâmetro               | Acurácia | Matriz de Confusão                                                                                                                                                                                                                                             |
|-------------------|-------------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RNA - CV          | size=31<br>decay=0.4    | 0.8443   | <pre> Confusion Matrix and Statistics            Reference Prediction bus opel saab van bus      39    0    1    0 opel     1   34   12    1 saab     1    7   30    0 van      2    1    0   38  Overall Statistics                  Accuracy : 0.8443 </pre> |
| SVM -<br>Hold-out | C=1 Sigma=<br>0.0661018 | 0.7665   | <pre> Confusion Matrix and Statistics            Reference Prediction bus opel saab van bus      40    1    1    0 opel     0   21   13    0 saab     0   17   28    0 van      3    3    1   39  Overall Statistics                  Accuracy : 0.7665 </pre> |
| RF - CV           | mtry=5                  | 0.7964   | <pre> Confusion Matrix and Statistics            Reference Prediction bus opel saab van bus      42    1    1    0 opel     0   23   12    0 saab     0   16   29    0 van      1    2    1   39  Overall Statistics                  Accuracy : 0.7964 </pre> |
| RF - Hold-out     | mtry=2                  | 0.7844   | <pre> Confusion Matrix and Statistics            Reference Prediction bus opel saab van bus      43    1    0    0 opel     0   19   12    0 saab     0   18   30    0 van      0    4    1   39  Overall Statistics                  Accuracy : 0.7844 </pre> |

| Técnica           | Parâmetro           | Acurácia | Matriz de Confusão                                                                                                                                                                                                                                                        |
|-------------------|---------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| KNN               | k=3                 | 0.7      | <pre> Confusion Matrix and Statistics        Reference Prediction bus opel saab van       bus   38    3    3    1       opel   5   17   12    1       saab   3   22   26    0       van    0    0    1   38  Overall Statistics                  Accuracy : 0.7 </pre>    |
| RNA -<br>Hold-out | size=3<br>decay=0.1 | 0.6287   | <pre> Confusion Matrix and Statistics        Reference Prediction bus opel saab van       bus   36    2    2    0       opel   0   30   38    2       saab   3    6    3    1       van    4    4    0   36  Overall Statistics                  Accuracy : 0.6287 </pre> |

Fonte: O autor (2025).

### Predição novos casos (SVM - CV)

TABELA 7 – Resultados predição novos casos (SVM - CV) - Parte 1

| Comp | Circ | DCirc | RadRa | PrAxisRa | MaxLRa | ScatRa | Elong | PrAxisRect | MaxLRect |
|------|------|-------|-------|----------|--------|--------|-------|------------|----------|
| 84   | 53   | 89    | 170   | 75       | 7      | 153    | 45    | 20         | 165      |
| 87   | 35   | 90    | 450   | 57       | 8      | 160    | 50    | 15         | 165      |
| 100  | 40   | 100   | 205   | 60       | 11     | 210    | 35    | 30         | 165      |

Fonte: O autor (2025).

TABELA 8 – Resultados predição novos casos (SVM - CV) - Parte 2

| ScVarMaxis | ScVarmaxis | RaGyr | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | HollRa | svm  |
|------------|------------|-------|-----------|-----------|-----------|-----------|--------|------|
| 170        | 350        | 180   | 65        | 8         | 15        | 180       | 197    | van  |
| 170        | 315        | 154   | 55        | 10        | 12        | 190       | 196    | opel |
| 220        | 625        | 200   | 75        | 11        | 10        | 185       | 195    | saab |

Fonte: O autor (2025).

### Lista de comandos (SVM - CV)

```

install.packages("e1071")
install.packages("caret")
install.packages("mlbench")

```

```

install.packages("mice")
install.packages("Metrics")
install.packages("kernlab")
library("caret")
library("mlbench")
library("mice")
library("Metrics")
setwd("/content")
seed_padrao <- 202412

dados <- read.csv("6 - Veiculos - Dados.csv")
dados$a <- NULL
View(dados)

set.seed(seed_padrao)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

ctrl <- trainControl(method = "cv", number = 10)
grid <- expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))
set.seed(seed_padrao)
svm <- train(form = tipo~, data = treino, method = "svmRadial",
tuneGrid = grid, trControl = ctrl)
svm

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

dados_novos_casos <- read.csv("6 - Veiculos - Dados - Novos
Casos.csv")
dados_novos_casos$a <- NULL
View(dados_novos_casos)

predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$tipo <- NULL
View(resultado)

```

## Diabetes

TABELA 9 – Resultados da classificação na base de diabetes

| Técnica       | Parâmetro                    | Acurácia | Matriz de Confusão                                                                                                                        |
|---------------|------------------------------|----------|-------------------------------------------------------------------------------------------------------------------------------------------|
| SVM Hold-out  | C=0.5<br>Sigma=<br>0.1219692 | 0.7843   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      90   23 pos      10   30  Accuracy : 0.7843 </pre> |
| SVM - CV      | C=0.5<br>Sigma=<br>0.1219692 | 0.7843   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      90   23 pos      10   30  Accuracy : 0.7843 </pre> |
| RNA - CV      | size=11<br>decay=0.4         | 0.7778   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      86   20 pos      14   33  Accuracy : 0.7778 </pre> |
| KNN           | k=9                          | 0.7727   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      93   22 pos      13   26  Accuracy : 0.7727 </pre> |
| RF - Hold-out | mtry=2                       | 0.7647   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      86   22 pos      14   31  Accuracy : 0.7647 </pre> |
| RF - CV       | mtry=8                       | 0.7582   | <pre> Confusion Matrix and Statistics            Reference Prediction neg pos neg      86   23 pos      14   30  Accuracy : 0.7582 </pre> |

| Técnica      | Parâmetro           | Acurácia | Matriz de Confusão                                                                                                                                                   |
|--------------|---------------------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RNA Hold-out | size=5<br>decay=0.1 | 0.7386   | <div>Confusion Matrix and Statistics</div> <div>Reference</div> <div>Prediction neg pos</div> <div>neg 83 23</div> <div>pos 17 30</div> <div>Accuracy : 0.7386</div> |

Fonte: O autor (2025).

Predição novos casos (SVM Hold-out)

TABELA 10 – Resultados da predição novos casos (SVM Hold-out)

| preg0nt | glucose | pressure | triceps | insulin | mass | pedigree | age | predict.svm |
|---------|---------|----------|---------|---------|------|----------|-----|-------------|
| 8       | 140     | 80       | 30      | 0       | 40.5 | 500      | 45  | neg         |
| 8       | 120     | 50       | 40      | 0       | 20.7 | 625      | 35  | neg         |
| 2       | 50      | 60       | 25      | 0       | 30.3 | 544      | 32  | neg         |

Fonte: O autor (2025).

Lista de comandos (SVM Hold-out)

```
install.packages("e1071")
install.packages("caret")
install.packages("mlbench")
install.packages("mice")
install.packages("Metrics")
install.packages("kernlab")
library("caret")
library("mlbench")
library("mice")
library("Metrics")

setwd("/content")
seed_padrao <- 202412

temp_dados <- read.csv("10 – Diabetes – Dados.csv")
temp_dados$num <- NULL
imp <- mice(temp_dados)
dados <- complete(imp, 1)
View(dados)

set.seed(seed_padrao)
indices <- createDataPartition(dados$diabetes, p=0.80,list=FALSE)
```



```

treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(seed_padrao)
svm <- train(diabetes~., data=treino, method="svmRadial")
svm

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

dados_novos_casos <- read.csv("10 - Diabetes - Dados - Novos
Casos.csv")
dados_novos_casos$num <- NULL
View(dados_novos_casos)

predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$diabetes <- NULL
View(resultado)

```

## 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

## Admissão

TABELA 11 – Resultados da regressão na base de admissão

| Técnica        | Parâmetro                  | R2     | Syx    | Pearson | Rmse   | MAE    |
|----------------|----------------------------|--------|--------|---------|--------|--------|
| SVM - CV       | C=2<br>Sigma=0.01          | 0.8882 | 0.0463 | 0.9427  | 0.0442 | 0.0332 |
| SVM - Hold-out | C=1    Sigma=<br>0.2004666 | 0.8676 | 0.0504 | 0.9317  | 0.0480 | 0.0362 |
| RF - Hold-out  | mtry=2                     | 0.8609 | 0.0517 | 0.9309  | 0.0492 | 0.0368 |
| RF - CV        | mtry=2                     | 0.8609 | 0.0517 | 0.9309  | 0.0492 | 0.0368 |
| RNA - CV       | size=5<br>decay=0.1        | 0.8392 | 0.0556 | 0.9193  | 0.0529 | 0.0436 |
| RNA - Hold-out | size=5<br>decay=0.1        | 0.8294 | 0.0572 | 0.9149  | 0.0545 | 0.0454 |
| KNN            | K=9                        | 0.7773 | 0.0654 | 0.8835  | 0.0623 | 0.0482 |

Fonte: O autor (2025).

## Predição novos casos (SVM - CV)

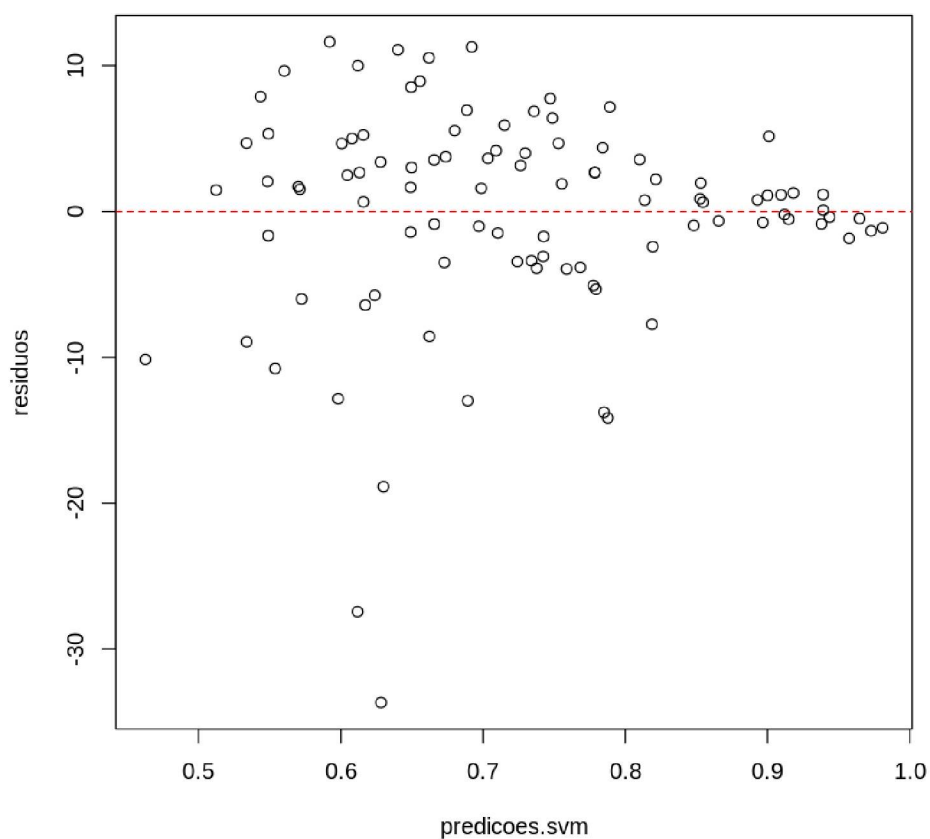
TABELA 12 – Resultados da predição novos casos (SVM - CV)

| GRE.S | TOEFL.S | University.R | SOP | LOR | CGPA | Research | predict |
|-------|---------|--------------|-----|-----|------|----------|---------|
| 300   | 125     | 3            | 4.5 | 4.5 | 8.87 | 1        | 0.7948  |
| 350   | 99      | 4            | 3.0 | 3.0 | 9.00 | 1        | 0.8113  |
| 325   | 85      | 4            | 4.5 | 3.5 | 8.75 | 1        | 0.7042  |

Fonte: O autor (2025).

## Gráfico de resíduos (SVM - CV)

FIGURA 19 – Gráfico de resíduos (SVM - CV)



Fonte: O autor (2025).

### Lista de comandos (SVM – CV)

```
install.packages("e1071")
install.packages("caret")
install.packages("mlbench")
install.packages("mice")
install.packages("Metrics")
install.packages("kernlab")
library("caret")
library("mlbench")
library("mice")
library("Metrics")

setwd("/content")
seed_padrao <- 202412

dados <- read.csv("9 – Admissao – Dados.csv", header=T)
```

```

dados$num <- NULL
View(dados)

set.seed(seed_padrao)
indices <- createDataPartition(dados$ChanceOfAdmit, p=0.80,
list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

control <- trainControl(method = "cv", number = 10)
tuneGrid <- expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))
set.seed(seed_padrao)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial",
trainControl=control, tuneGrid=tuneGrid)
svm
predicoes.svm <- predict(svm, teste)

print("RMSE")
rmse(teste$ChanceOfAdmit, predicoes.svm)

print("R2")
r2 <- function(predito, observado) {
return(1 - (sum((predito-observado)^2) /
sum((observado-mean(observado))^2)))
}
r2(predicoes.svm, teste$ChanceOfAdmit)

print("SYX")
syx <- function(predito, observado, n, p)
{
sqrt((sum((observado-predito)^2))/(n-p-1))
}
syx(predicoes.svm, teste$ChanceOfAdmit, nrow(teste), ncol(teste))

print("PEARSON")
pearson <- function(predito, observado) {
n <- length(predito)
predito_media <- mean(predito)
observado_media <- mean(observado)

num <- sum((predito - predito_media) * (observado -

```

```

observado_media))
denom <- sqrt(sum((predito - predito_media)^2) * sum((observado -
observado_media)^2))

r <- num / denom
return(r)
}
pearson(predicoes.svm, teste$ChanceOfAdmit)

print("MAE")
mae <- function(predito, observado)
{
error = observado - predito
mean(abs(error))
}
mae(predicoes.svm, teste$ChanceOfAdmit)

residuos <- ((teste$ChanceOfAdmit -
predicoes.svm)/teste$ChanceOfAdmit)*100
plot(residuos ~ predicoes.svm)
abline(h = 0, lty = 2, col = "red")

dados_novos_casos <- read.csv("9 - Admissao - Dados - Novos
Casos.csv")
dados_novos_casos$num <- NULL
View(dados_novos_casos)

predict.svm <- predict(svm, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.svm)
resultado$ChanceOfAdmit <- NULL
View(resultado)

```

## Biomassa

TABELA 13 – Resultados da regressão na base de biomassa

| <b>Técnica</b> | <b>Parâmetro</b>    | <b>R<sup>2</sup></b> | <b>Syx</b> | <b>Pearson</b> | <b>RMSE</b> | <b>MAE</b> |
|----------------|---------------------|----------------------|------------|----------------|-------------|------------|
| RNA CV         | size=3<br>decay=0.1 | 0.9051               | 385.7903   | 0.9882         | 369.3660    | 109.9805   |
| RNA Hold-out   | size=3<br>decay=0.1 | 0.8993               | 397.4928   | 0.9751         | 380.5704    | 256.7893   |
| RF Hold-out    | mtry=2              | 0.8869               | 421.2733   | 0.9852         | 403.3385    | 102.0527   |
| RF CV          | mtry=2              | 0.8869               | 421.2733   | 0.9852         | 403.3385    | 102.0527   |
| SVM CV         | C=100<br>Sigma=0.01 | 0.8091               | 547.3479   | 0.9605         | 524.0457    | 154.5247   |
| KNN            | K=1                 | 0.7776               | 590.7204   | 0.9471         | 565.5717    | 128.5157   |
| SVM Hold-out   | C=1<br>Sigma=0.8644 | 0.4750               | 907.7681   | 0.8535         | 869.1218    | 199.1633   |

Fonte: O autor (2025).

**Predição novos casos (RNA - CV)**

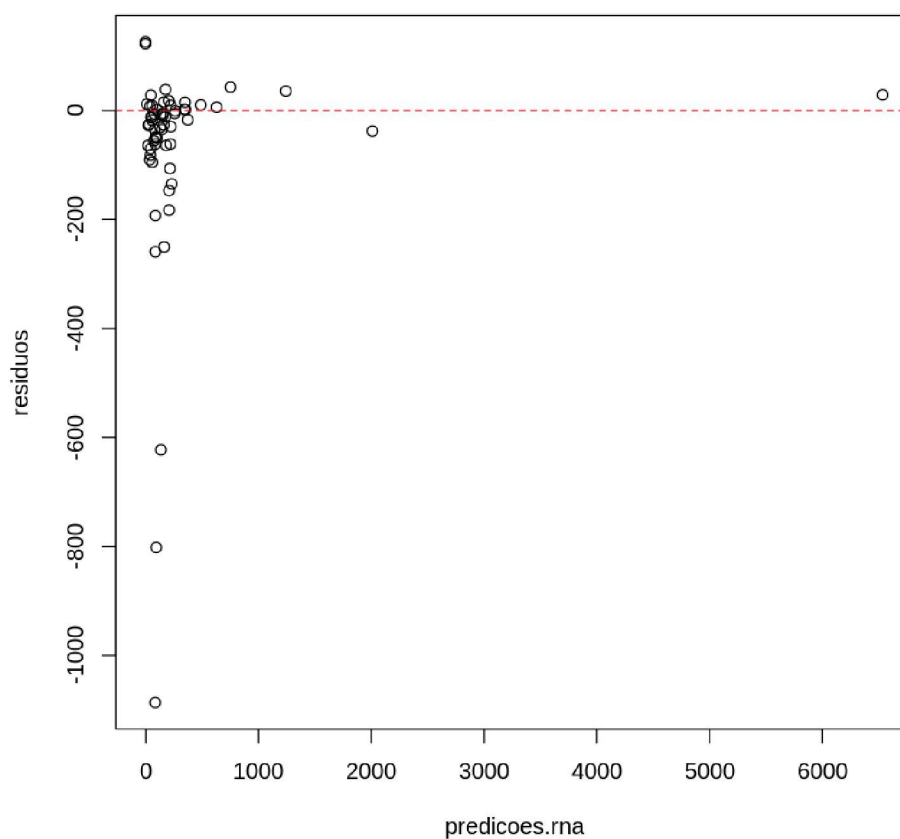
TABELA 14 – Resultado da predição novos casos (RNA - CV)

| dap | h   | Me   | predict.rna |
|-----|-----|------|-------------|
| 7.0 | 4.0 | 1.04 | 82.9916     |
| 8.2 | 6.0 | 1.04 | 109.7899    |
| 7.5 | 5.5 | 1.04 | 98.4963     |

Fonte: O autor (2025).

**Gráfico de resíduos (RNA – CV)**

FIGURA 20 – Gráfico de resíduos (RNA - CV)



Fonte: O autor (2025).

### Lista de comandos (RNA - CV)

```
install.packages("e1071")
install.packages("caret")
install.packages("mlbench")
install.packages("mice")
install.packages("Metrics")
library("caret")
library("mlbench")
library("mice")
library("Metrics")

setwd("/content")
seed_padrao <- 202412

dados <- read.csv("5 - Biomassa - Dados.csv")
View(dados)
```



```

set.seed(seed_padrao)
ind <- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]

control <- trainControl(method = "cv", number = 10)
set.seed(seed_padrao)
rna <- train(biomassa~., data=treino, method="nnet",
trainControl=control, linout=T, MaxNWts=10000, maxit=2000, trace=F)
rna
predicoes.rna <- predict(rna, teste)

print("RMSE")
rmse(teste$biomassa, predicoes.rna)

print("R2")
r2 <- function(predito, observado) {
  return(1 - (sum((predito-observado)^2) /
sum((observado-mean(observado))^2)))
}
r2(predicoes.rna, teste$biomassa)

print("SYX")
syx <- function(predito, observado, n, p)
{
  sqrt((sum((observado-predito)^2))/(n-p-1))
}
syx(predicoes.rna, teste$biomassa, nrow(teste), ncol(teste))

print("PEARSON")
pearson <- function(predito, observado) {
  n <- length(predito)
  predito_media <- mean(predito)
  observado_media <- mean(observado)

  num <- sum((predito - predito_media) * (observado -
observado_media))
  denom <- sqrt(sum((predito - predito_media)^2) * sum((observado -
observado_media)^2))

  r <- num / denom

```

```

return(r)
}
pearson(predicoes.rna, teste$biomassa)

print("MAE")
mae <- function(predito, observado)
{
  error = observado-predito
  mean(abs(error))
}
mae(predicoes.rna, teste$biomassa)

residuos <- ((teste$biomassa - predicoes.rna)/teste$biomassa)*100
plot(residuos ~ predicoes.rna)
abline(h = 0, lty = 2, col = "red")

dados_novos_casos <- read.csv("5 - Biomassa - Dados - Novos
Casos.csv")
View(dados_novos_casos)

predict.rna <- predict(rna, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.rna)
resultado$biomassa <- NULL
View(resultado)

```

## Agrupamento

### Veículo

TABELA 15 – Clusters gerados - Parte 1

| Cluster | Comp | Circ | DCirc | Rad | Ra | PraxisRa | MaxLRa |
|---------|------|------|-------|-----|----|----------|--------|
| 1       | 85   | 43   | 70    | 120 | 56 | 7        | 149    |
| 2       | 104  | 54   | 101   | 189 | 57 | 11       | 222    |
| 3       | 89   | 45   | 72    | 166 | 59 | 7        | 154    |
| 4       | 89   | 41   | 66    | 162 | 58 | 7        | 150    |
| 5       | 90   | 37   | 74    | 169 | 59 | 7        | 157    |
| 6       | 88   | 40   | 69    | 146 | 58 | 9        | 133    |
| 7       | 101  | 53   | 108   | 197 | 60 | 11       | 214    |
| 8       | 93   | 39   | 66    | 133 | 59 | 8        | 130    |
| 9       | 85   | 36   | 51    | 116 | 57 | 6        | 122    |
| 10      | 100  | 47   | 105   | 209 | 64 | 10       | 201    |

Fonte: O autor (2025).

TABELA 16 – Clusters gerados - Parte 2

| Cluster | ScatRa | Elong | PraxisRect | MaxLRect | ScVarMaxis | ScVarmxis | RaGyr |
|---------|--------|-------|------------|----------|------------|-----------|-------|
| 1       | 45     | 19    | 143        | 170      | 327        | 171       | 81    |
| 2       | 30     | 25    | 173        | 225      | 706        | 216       | 72    |
| 3       | 44     | 19    | 144        | 174      | 347        | 174       | 69    |
| 4       | 45     | 19    | 144        | 169      | 331        | 161       | 64    |
| 5       | 43     | 20    | 131        | 189      | 363        | 186       | 72    |
| 6       | 50     | 18    | 135        | 202      | 259        | 151       | 66    |
| 7       | 31     | 24    | 162        | 226      | 661        | 214       | 71    |
| 8       | 52     | 18    | 134        | 159      | 246        | 139       | 63    |
| 9       | 57     | 17    | 125        | 137      | 196        | 123       | 85    |
| 10      | 33     | 23    | 147        | 214      | 586        | 201       | 67    |

Fonte: O autor (2025).

TABELA 17 – Clusters gerados - Parte 3

| Cluster | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | Tipo |
|---------|-----------|-----------|-----------|-----------|------|
| 1       | 6         | 11        | 180       | 184       | bus  |
| 2       | 0         | 11        | 187       | 198       | opel |
| 3       | 5         | 2         | 190       | 196       | opel |
| 4       | 4         | 0         | 188       | 197       | van  |
| 5       | 1         | 7         | 196       | 202       | bus  |
| 6       | 0         | 21        | 193       | 200       | opel |
| 7       | 0         | 24        | 188       | 199       | saab |
| 8       | 5         | 7         | 183       | 194       | van  |
| 9       | 1         | 5         | 180       | 183       | saab |
| 10      | 2         | 13        | 192       | 198       | bus  |

Fonte: O autor (2025).

10 primeiras linhas do arquivo com o cluster correspondente:

TABELA 18 – Resumo do cluster correspondente - Parte 1

| Cluster | Comp | Circ | DCirc | Rad | Ra  | PraxisRa | MaxLRa |
|---------|------|------|-------|-----|-----|----------|--------|
| 4       | 95   | 48   | 83    | 178 | 72  | 10       | 162    |
| 1       | 91   | 41   | 84    | 141 | 57  | 9        | 149    |
| 10      | 104  | 50   | 106   | 209 | 66  | 10       | 207    |
| 4       | 93   | 41   | 82    | 159 | 63  | 9        | 144    |
| 1       | 85   | 44   | 70    | 205 | 103 | 52       | 149    |
| 9       | 107  | 57   | 106   | 172 | 50  | 6        | 255    |
| 1       | 97   | 43   | 73    | 173 | 65  | 6        | 153    |
| 8       | 90   | 43   | 66    | 157 | 65  | 9        | 137    |
| 9       | 86   | 34   | 62    | 140 | 61  | 7        | 122    |
| 7       | 93   | 44   | 98    | 197 | 62  | 11       | 183    |

Fonte: O autor (2025).

TABELA 19 – Resumo do cluster correspondente - Parte 2

| Cluster | ScatRa | Elong | PraxisRect | MaxLRect | ScVarMaxis | ScVarmxis | RaGyr |
|---------|--------|-------|------------|----------|------------|-----------|-------|
| 4       | 42     | 20    | 159        | 176      | 379        | 184       | 70    |
| 1       | 45     | 19    | 143        | 170      | 330        | 158       | 72    |
| 10      | 32     | 23    | 158        | 223      | 635        | 220       | 73    |
| 4       | 46     | 19    | 143        | 160      | 309        | 127       | 63    |
| 1       | 45     | 19    | 144        | 241      | 325        | 188       | 127   |
| 9       | 26     | 28    | 169        | 280      | 957        | 264       | 85    |
| 1       | 42     | 19    | 143        | 176      | 361        | 172       | 66    |
| 8       | 48     | 18    | 146        | 162      | 281        | 164       | 67    |
| 9       | 54     | 17    | 127        | 141      | 223        | 112       | 64    |
| 7       | 36     | 22    | 146        | 202      | 505        | 152       | 64    |

Fonte: O autor (2025).

TABELA 20 – Resumo do cluster correspondente - Parte 3

| Cluster | SkewMaxis | Skewmaxis | Kurtmaxis | KurtMaxis | Tipo |
|---------|-----------|-----------|-----------|-----------|------|
| 4       | 6         | 16        | 187       | 197       | van  |
| 1       | 9         | 14        | 189       | 199       | van  |
| 10      | 14        | 9         | 188       | 196       | saab |
| 4       | 6         | 10        | 199       | 207       | van  |
| 1       | 9         | 11        | 180       | 183       | bus  |
| 9       | 5         | 9         | 181       | 183       | bus  |
| 1       | 13        | 1         | 200       | 204       | bus  |
| 8       | 3         | 3         | 193       | 202       | van  |
| 9       | 2         | 14        | 200       | 208       | van  |
| 7       | 4         | 14        | 195       | 204       | saab |

Fonte: O autor (2025).

### Lista de comandos emitidos

```
install.packages("klaR")
library(klaR)

setwd("/content")
seed_padrao <- 202412

dados <- read.csv("6 – Veiculos – Dados.csv")
dados$a <- NULL
View(dados)

set.seed(seed_padrao)
cluster.results <- kmodes(dados, 10, iter.max = 10, weighted =
FALSE )
cluster.results
```

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

## Regras de associação

### Musculação

Regras geradas com uma configuração de Suporte e Confiança.

FIGURA 21 – Resultados regras de associação

|      | lhs                           | rhs            | support | confidence | coverage |
|------|-------------------------------|----------------|---------|------------|----------|
| [1]  | {Agachamento}                 | => {LegPress}  | 0.31    | 1.00       | 0.31     |
| [2]  | {Afundo}                      | => {Gemeos}    | 0.35    | 1.00       | 0.35     |
| [3]  | {AgachamentoSmith, Bicicleta} | => {Extensor}  | 0.31    | 1.00       | 0.31     |
| [4]  | {Bicicleta, Esteira}          | => {Extensor}  | 0.38    | 1.00       | 0.38     |
| [5]  | {Extensor}                    | => {Bicicleta} | 0.46    | 0.92       | 0.50     |
| [6]  | {Esteira}                     | => {Extensor}  | 0.42    | 0.92       | 0.46     |
| [7]  | {Esteira, Extensor}           | => {Bicicleta} | 0.38    | 0.91       | 0.42     |
| [8]  | {AgachamentoSmith}            | => {Extensor}  | 0.35    | 0.90       | 0.38     |
| [9]  | {AgachamentoSmith, Extensor}  | => {Bicicleta} | 0.31    | 0.89       | 0.35     |
| [10] | {Bicicleta}                   | => {Extensor}  | 0.46    | 0.86       | 0.54     |
| [11] | {Extensor}                    | => {Esteira}   | 0.42    | 0.85       | 0.50     |
| [12] | {Esteira}                     | => {Bicicleta} | 0.38    | 0.83       | 0.46     |
| [13] | {Bicicleta, Extensor}         | => {Esteira}   | 0.38    | 0.83       | 0.46     |
| [14] | {AgachamentoSmith}            | => {Esteira}   | 0.31    | 0.80       | 0.38     |
| [15] | {AgachamentoSmith}            | => {Bicicleta} | 0.31    | 0.80       | 0.38     |

Fonte: O autor (2025).

## Lista de comandos emitidos

```
install.packages('arules', dep=T)
library(arules)
library(datasets)

setwd("/content")
seed_padrao <- 202412

dados <- read.transactions(file="2 - Musculacao - Dados.csv",
format="basket", sep=";")
inspect(dados[1:4])

set.seed(seed_padrao)
rules <- apriori(dados, parameter = list(supp = 0.3, conf = 0.75,
minlen = 2))
summary(rules)
```

```
options(digits=2)  
inspect(sort(rules, by="confidence"))
```

## 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)

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

# Carga da base
cifar10 = tf.keras.datasets.cifar10
# Já está separado em dados de treino e teste
# Não precisa separar
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
```



```

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

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

model.summary()

```

Model: "functional"

| Layer (type)             | Output Shape       | Param #   |
|--------------------------|--------------------|-----------|
| input_layer (InputLayer) | (None, 32, 32, 3)  | 0         |
| conv2d (Conv2D)          | (None, 15, 15, 32) | 896       |
| conv2d_1 (Conv2D)        | (None, 7, 7, 64)   | 18,496    |
| conv2d_2 (Conv2D)        | (None, 3, 3, 128)  | 73,856    |
| flatten (Flatten)        | (None, 1152)       | 0         |
| dropout (Dropout)        | (None, 1152)       | 0         |
| dense (Dense)            | (None, 1024)       | 1,180,672 |
| dropout_1 (Dropout)      | (None, 1024)       | 0         |
| dense_1 (Dense)          | (None, 10)         | 10,250    |

Total params: 1,284,170 (4.90 MB)

Trainable params: 1,284,170 (4.90 MB)

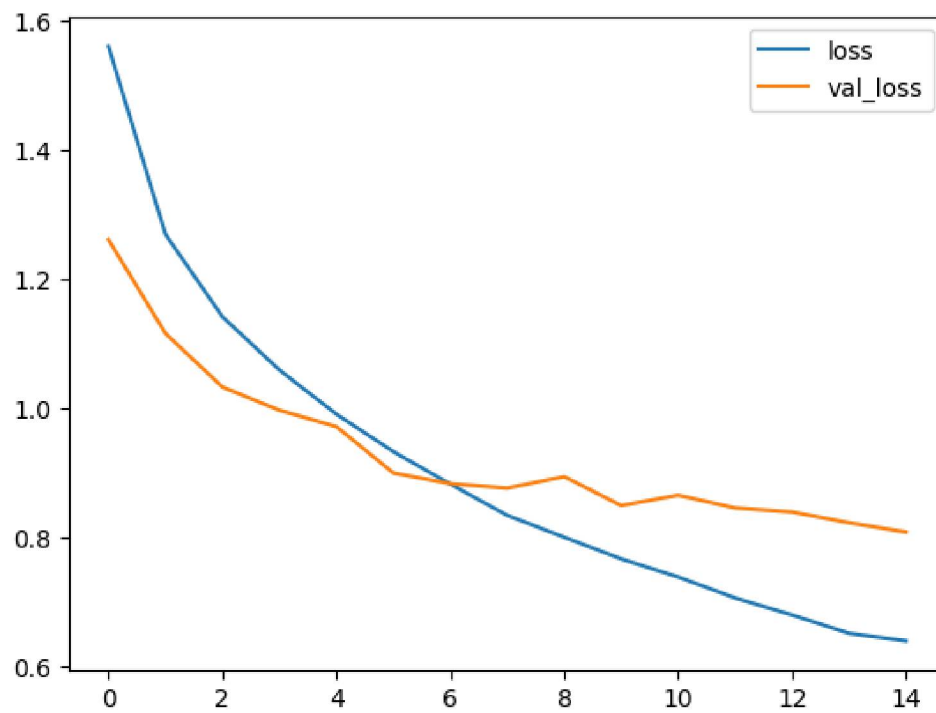
Non-trainable params: 0 (0.00 B)

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

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

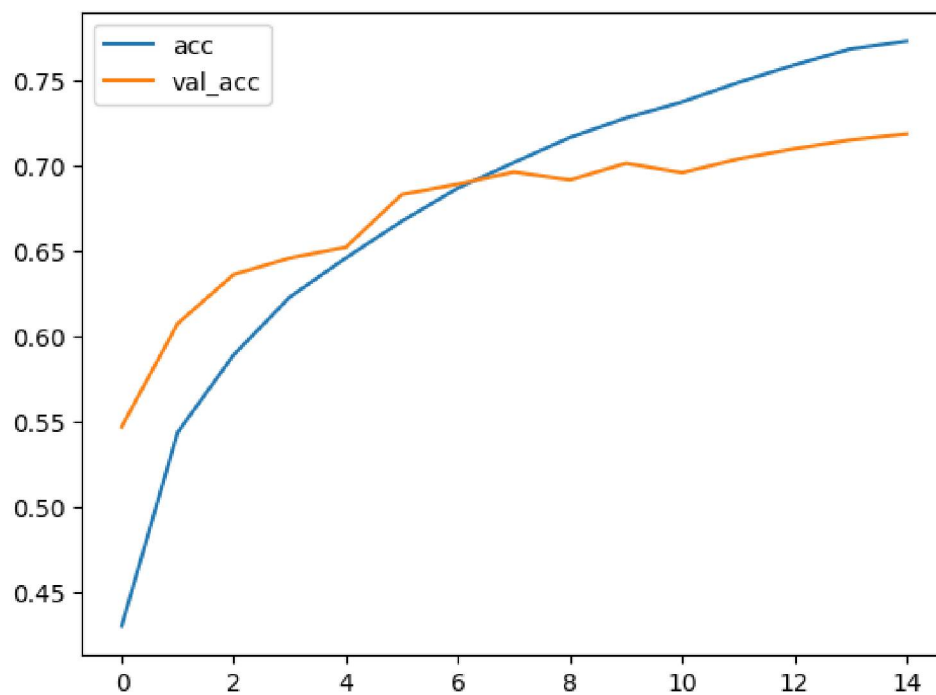
# Plotar acurácia, treino e validação
plt.plot(r.history["accuracy"], label="acc")
plt.plot(r.history["val_accuracy"], label="val_acc")
plt.legend()
plt.show()
```

FIGURA 22 – Função de perda - CNN



Fonte: O autor (2025).

FIGURA 23 – Acurácia - CNN

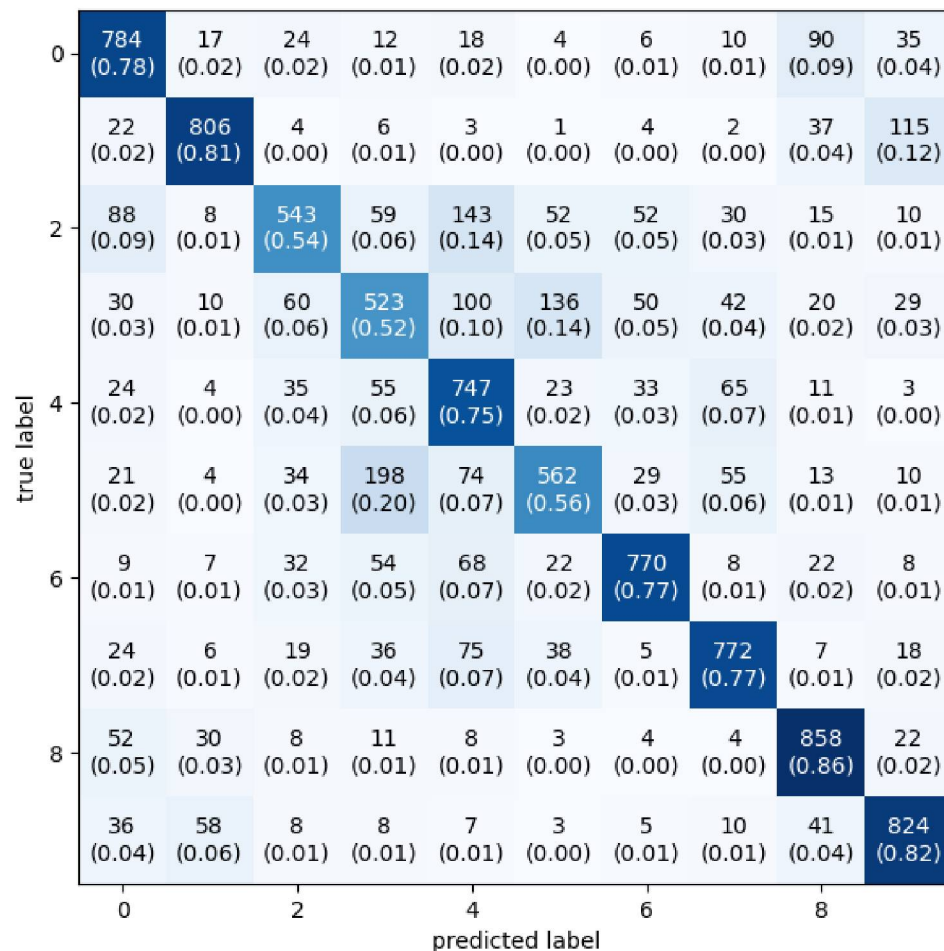


Fonte: O autor (2025).

```
# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
```

```
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
```

FIGURA 24 – Matriz de confusão - CNN

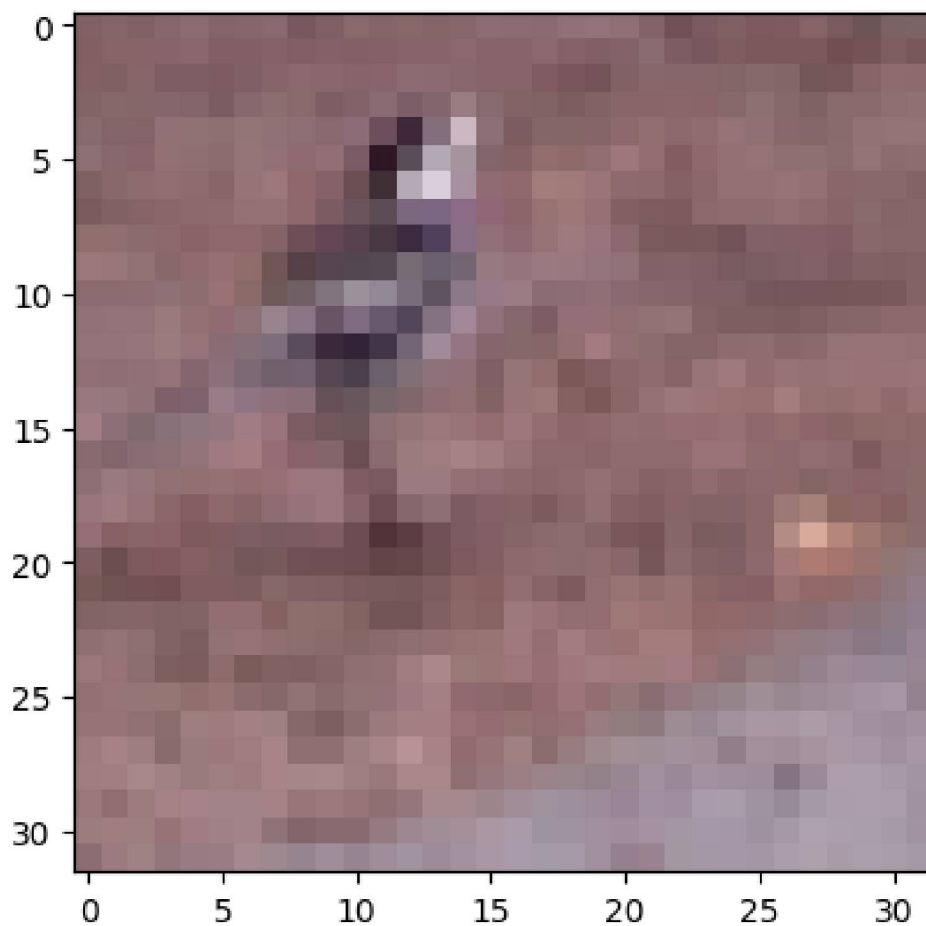


Fonte: O autor (2025).

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

FIGURA 25 – Classificação errada - CNN

True label: bird Predicted: deer



Fonte: O autor (2025).

## 2 Detector de SPAM (RNN)

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

# carrega e arruma a base
!wget http://www.razer.net.br/datasets/spam.csv
```

```

# !wget https://www.kaggle.com/datasets/uciml/sms-spam-collection
-dataset
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values

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

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

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

```

```

7201 tokens
data_train.shape: (3733, 189)
data_test.shape: (1839, 189)

```

```

# Define o modelo
D = 20 # tamanho do embedding, hiperparâmetro que pode ser
escolhido

```



```

M = 5      # tamanho do hidden state, quantidade de unidades LSTM
i = Input(shape=(T,)) # Entra uma frase inteira
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2
    valores
model = Model(i, x)
model.summary()

```

Model: "functional"

| Layer (type)             | Output Shape    | Param # |
|--------------------------|-----------------|---------|
| input_layer (InputLayer) | (None, 189)     | 0       |
| embedding (Embedding)    | (None, 189, 20) | 143,980 |
| lstm (LSTM)              | (None, 5)       | 520     |
| dense (Dense)            | (None, 1)       | 6       |

Total params: 144,506 (564.48 KB)

Trainable params: 144,506 (564.48 KB)

Non-trainable params: 0 (0.00 B)

```

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

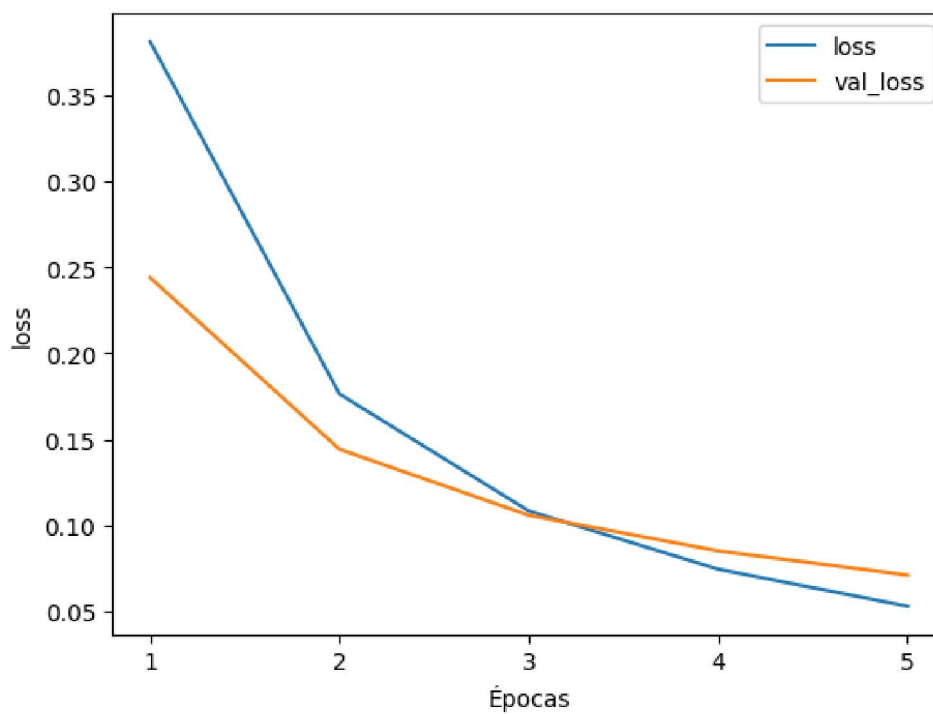
# Plota função de perda e acurácia
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.xlabel("Epocas")
plt.ylabel("loss")
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs
    +1))
plt.legend()
plt.show()
plt.plot(r.history["accuracy"], label="accuracy")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

```



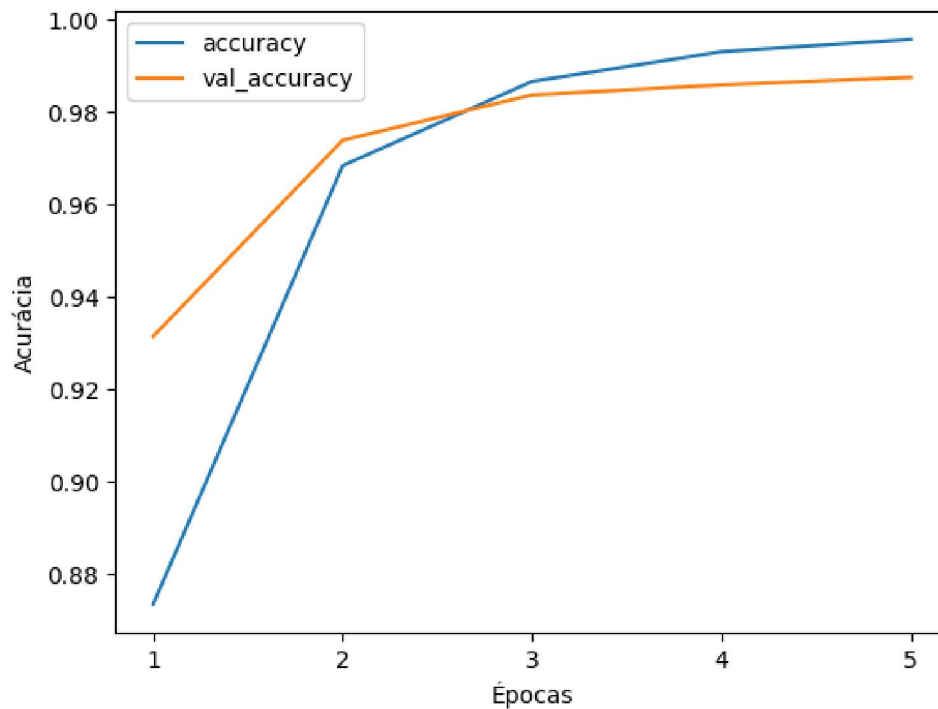
```
plt.xticks(np.arange(0, epochs, step=1), labels=range(1, epochs  
+1))  
plt.legend()  
plt.show()
```

FIGURA 26 – Função de perda - RNN



Fonte: O autor (2025).

FIGURA 27 – Acurácia - RNN



Fonte: O autor (2025).

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

```
[[0.7768526]]
SPAM
```

### 3 Gerador de Dígitos Fake (GAN)

```
!pip install imageio
!pip install git+https://github.com/tensorflow/docs

# Importações
import tensorflow as tf
import glob
import imageio
import matplotlib.pyplot as plt
```

```

import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
from IPython import display

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

# Cria o GERADOR
def make_generator_model():
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape
        =(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

    model.add(layers.Reshape((7, 7, 256)))
    assert model.output_shape == (None, 7, 7, 256)

    # Note: None is the batch size
    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
        padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)

    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

```

```

model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
    padding='same', use_bias=False))
assert model.output_shape == (None, 14, 14, 64)
model.add(layers.BatchNormalization())
model.add(layers.LeakyReLU())

model.add(layers.Conv2DTranspose(1, (5, 5), strides=(2, 2),
    padding='same', use_bias=False, activation='tanh'))
assert model.output_shape == (None, 28, 28, 1)
return model

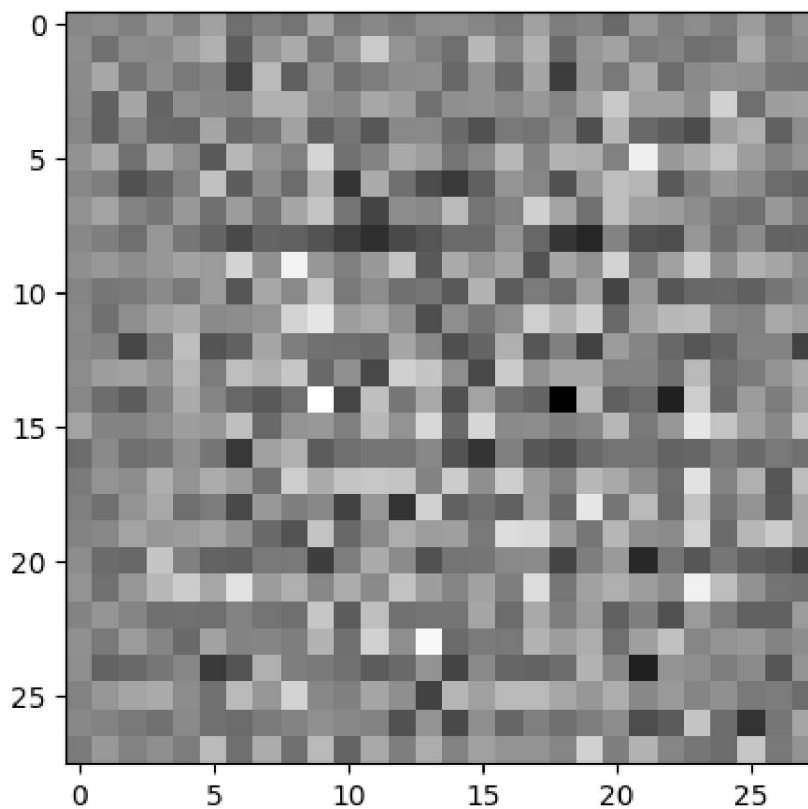
# Teste do GERADOR, ainda não treinado
generator = make_generator_model()

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

plt.imshow(generated_image[0, :, :, 0], cmap='gray')

```

FIGURA 28 – Teste gerador - GAN



Fonte: O autor (2025).

```

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

    model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='
        same'))
    model.add(layers.LeakyReLU())
    model.add(layers.Dropout(0.3))

    model.add(layers.Flatten())
    model.add(layers.Dense(1))
    return model

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

# Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=
    True)

# Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output),
        real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output),
        fake_output)
    total_loss = real_loss + fake_loss
    return total_loss

# Perda do GERADOR
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)

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

```

```

discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)

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

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

def f(x, y):
    return 3*x**2 + 2*x*y

x, y = tf.Variable(5.), tf.Variable(3.)
with tf.GradientTape() as tape:
    z = f(x, y)
gradients = tape.gradient(z, [x, y])
print(gradients)

# Função que faz um passo de treinamento
# E uma tf.function, que compila essa funcao
@tf.function
def train_step(images):
    noise = tf.random.normal([BATCH_SIZE, noise_dim])

    with tf.GradientTape() as gen_tape, tf.GradientTape() as
        disc_tape:
            generated_images = generator(noise, training=True)

            real_output = discriminator(images, training=True)
            fake_output = discriminator(generated_images, training=True)

```



```

    gen_loss = generator_loss(fake_output)
    disc_loss = discriminator_loss(real_output, fake_output)

    gradients_of_generator = gen_tape.gradient(gen_loss, generator.
        trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss,
        discriminator.trainable_variables)

    generator_optimizer.apply_gradients(zip(gradients_of_generator,
        generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(
        gradients_of_discriminator, discriminator.trainable_variables
    ))

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

        for image_batch in dataset:
            train_step(image_batch)

        # Produce images for the GIF as you go
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)

        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)

        print ('Time for epoch {} is {} sec'.format(epoch + 1, time.
            time() - start))

    # Generate after the final epoch
    display.clear_output(wait=True)
    generate_and_save_images(generator, epochs, seed)

def generate_and_save_images(model, epoch, test_input):
    # Notice 'training' is set to False.
    # This is so all layers run in inference mode (batchnorm).
    predictions = model(test_input, training=False)

```



```

fig = plt.figure(figsize=(4, 4))

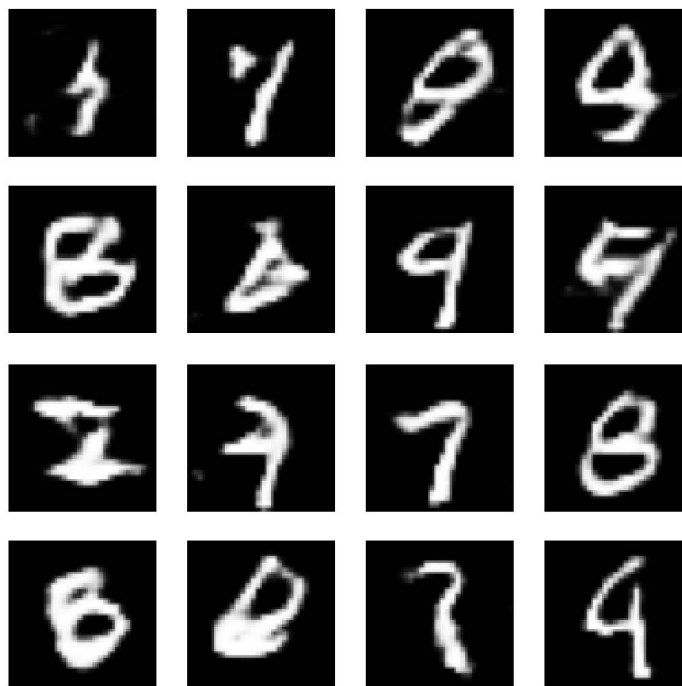
for i in range(predictions.shape[0]):
    plt.subplot(4, 4, i+1)
    plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='
    gray')
    plt.axis('off')

plt.savefig('image_at_epoch_{:04d}.png'.format(epoch))
plt.show()

train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))

```

FIGURA 29 – Imagem na ultima epoca - GAN



Fonte: O autor (2025).

#### 4 Tradutor de Textos (Transformer)

```

# Instalação e importação
!pip uninstall -y tensorflow
!pip uninstall -y tf-keras
!pip install tf-keras==2.15.1
!pip install tensorflow==2.15.0 --use-deprecated=legacy-resolver
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0

```

```
pip install matplotlib

import collections
import logging
import os
import pathlib
import re
import string
import sys
import time

import numpy as np
import matplotlib.pyplot as plt

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

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

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

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

    for en in en_examples.numpy():
        print(en.decode('utf-8'))
```

e quando melhoramos a procura , tiramos a única vantagem da impressão , que é a serendipidade .

mas e se estes fatores fossem ativos ?

mas eles não tinham a curiosidade de me testar .

and when you improve searchability , you actually take away the one advantage of print , which is serendipity .

but what if it were active ?

but they did n't test for curiosity .

```
# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"

tf.keras.utils.get_file(f"{model_name}.zip", f"https://storage.
    googleapis.com/download.tensorflow.org/models/{model_name}.
    zip", cache_dir='.', cache_subdir='', extract=True)
# Tem 2 tokenizers: um pt outro em en tokenizers.en tokeniza e
    detokeniza
tokenizers = tf.saved_model.load(model_name)

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

    en = tokenizers.en.tokenize(en)
    # ragged -> dense
    en = en.to_tensor()
    return pt, en

# Pipeline simpes: processa, embaralha, agrupa os dados, prefetch
# Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64

def make_batches(ds):
    return (
        ds
```

```

        .cache()
        .shuffle(BUFFER_SIZE)
        .batch(BATCH_SIZE)
        .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
        .prefetch(tf.data.AUTOTUNE)
    )

train_batches = make_batches(train_examples)
val_batches = make_batches(val_examples)

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

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

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

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

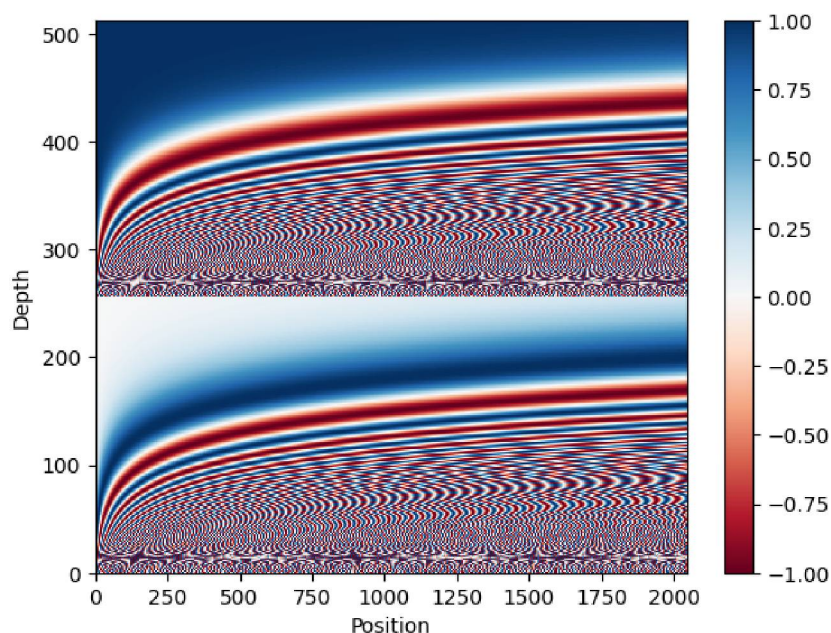
# Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))

plt.pcolormesh(pos_encoding, cmap='RdBu')
plt.ylabel('Depth')
plt.xlabel('Position')

```

```
plt.colorbar()
plt.show()
```

FIGURA 30 – Colomesh - Transformer



Fonte: O autor (2025).

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

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

# Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
    # Q K^T
    matmul_qk = tf.matmul(q, k, transpose_b=True) # (...
    seq_len_q, seq_len_k)
    # converte matmul_qk para float32
```



```

dk = tf.cast(tf.shape(k)[-1], tf.float32)

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

# Soma a máscara, e os valores faltantes serão um número pró-
ximo a -inf
if mask is not None:
    scaled_attention_logits += (mask * -1e9)

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

# Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads):
        super(MultiHeadAttention, self).__init__()
        self.num_heads = num_heads
        self.d_model = d_model
        assert d_model % self.num_heads == 0
        self.depth = d_model // self.num_heads

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

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

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

    def call(self, v, k, q, mask):

```

```

batch_size = tf.shape(q)[0]

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

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

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

# Troca a dimensão 2 com 1, para acertar o num_heads# (
batch_size, seq_len_q, num_heads, depth)
scaled_attention = tf.transpose(scaled_attention, perm=[0, 2,
1, 3])
# Concatena os valores em: (batch_size, seq_len_q, d_model)
concat_attention = tf.reshape(scaled_attention, (batch_size,
-1, self.d_model))
output = self.dense(concat_attention) # (batch_size,
seq_len_q, d_model)
return output, attention_weights

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

class EncoderLayer(tf.keras.layers.Layer):
    def __init__(self, d_model, num_heads, dff, rate=0.1):

```



```

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

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

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

def call(self, x, training, mask):
    attn_output, _ = self.mha(x, x, x, mask)
    attn_output = self.dropout1(attn_output, training=training)
    out1 = self.layernorm1(x + attn_output)
    ffn_output = self.ffn(out1)
    ffn_output = self.dropout2(ffn_output, training=training)
    out2 = self.layernorm2(out1 + ffn_output)

    return out2

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

        self.mha1 = MultiHeadAttention(d_model, num_heads)
        self.mha2 = MultiHeadAttention(d_model, num_heads)
        self.ffn = point_wise_feed_forward_network(d_model, dff)

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

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

    def call(self, x, enc_output, training, look_ahead_mask,

```

```

padding_mask):
    # enc_output.shape == (batch_size, input_seq_len, d_model)
    # (batch_size, target_seq_len, d_model)
    attn1, attn_weights_block1 = self.mha1(x, x, x,
look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layer_norm1(attn1 + x)

    # (batch_size, target_seq_len, d_model)
    attn2, attn_weights_block2 = self.mha2(enc_output, enc_output
, out1, padding_mask)
    attn2 = self.dropout2(attn2, training=training)
    out2 = self.layer_norm2(attn2 + out1) # (batch_size,
target_seq_len, d_model)

    ffn_output = self.ffn(out2) # (batch_size, target_seq_len,
d_model)
    ffn_output = self.dropout3(ffn_output, training=training)
    out3 = self.layer_norm3(ffn_output + out2) # (batch_size,
target_seq_len, d_model)

    return out3, attn_weights_block1, attn_weights_block2

class Encoder(tf.keras.layers.Layer):
    def __init__(self, num_layers, d_model, num_heads, dff,
input_vocab_size, maximum_position_encoding, rate=0.1):
        super(Encoder, self).__init__()
        self.d_model = d_model
        self.num_layers = num_layers
        self.embedding = tf.keras.layers.Embedding(input_vocab_size,
d_model)
        self.pos_encoding = positional_encoding(
maximum_position_encoding, self.d_model)
        self.enc_layers = [EncoderLayer(d_model, num_heads, dff, rate
) for _ in range(num_layers)]
        self.dropout = tf.keras.layers.Dropout(rate)

    def call(self, x, training, mask):
        seq_len = tf.shape(x)[1]
        # adding embedding and position encoding.
        x = self.embedding(x) # (batch_size, input_seq_len, d_model)
        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))

```

```

x += self.pos_encoding[:, :seq_len, :]
x = self.dropout(x, training=training)
for i in range(self.num_layers):
    x = self.enc_layers[i](x, training, mask)
return x # (batch_size, input_seq_len, d_model)

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

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

class Transformer(tf.keras.Model):
    def __init__(self, num_layers, d_model, num_heads, dff,
        input_vocab_size, target_vocab_size, pe_input, pe_target,
        rate=0.1):

```

```

super().__init__()
self.encoder = Encoder(num_layers, d_model, num_heads, dff,
input_vocab_size, pe_input, rate)
self.decoder = Decoder(num_layers, d_model, num_heads, dff,
target_vocab_size, pe_target, rate)
self.final_layer = tf.keras.layers.Dense(target_vocab_size)

def call(self, inputs, training):
    # Keras models prefer if you pass all your inputs in the
    first argument
    inp, tar = inputs

    enc_padding_mask, look_ahead_mask, dec_padding_mask = self.
create_masks(inp, tar)
    # (batch_size, inp_seq_len, d_model)
    enc_output = self.encoder(inp, training, enc_padding_mask)
    # dec_output.shape == (batch_size, tar_seq_len, d_model)
    dec_output, attention_weights = self.decoder(tar, enc_output,
training, look_ahead_mask, dec_padding_mask)
    # (batch_size, tar_seq_len, target_vocab_size)
    final_output = self.final_layer(dec_output)

    return final_output, attention_weights

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

# Hiperparâmetros
num_layers = 4

```



```

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

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

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

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

loss_object = tf.keras.losses.SparseCategoricalCrossentropy(
    from_logits=True, reduction='none')

def loss_function(real, pred):
    mask = tf.math.logical_not(tf.math.equal(real, 0))
    loss_ = loss_object(real, pred)
    mask = tf.cast(mask, dtype=loss_.dtype)
    loss_ *= mask
    return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

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

```

```

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

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

# Checkpoint
checkpoint_path = "./checkpoints/train"

ckpt = tf.train.Checkpoint(transformer=transformer, optimizer=
    optimizer)

ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
    max_to_keep=5)

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

EPOCHS = 20

train_step_signature = [
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
    tf.TensorSpec(shape=(None, None), dtype=tf.int64),
]

@tf.function(input_signature=train_step_signature)
def train_step(inp, tar):
    tar_inp = tar[:, :-1]
    tar_real = tar[:, 1:]
    with tf.GradientTape() as tape:
        predictions, _ = transformer([inp, tar_inp], training = True)
        loss = loss_function(tar_real, predictions)

```

```

gradients = tape.gradient(loss, transformer.trainable_variables
    )
optimizer.apply_gradients(zip(gradients, transformer.
    trainable_variables))

train_loss(loss)
train_accuracy(accuracy_function(tar_real, predictions))

for epoch in range(EPOCHS):
    start = time.time()
    train_loss.reset_state()
    train_accuracy.reset_state()

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

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

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

```



```

...
Epoch 20 Batch 0 Loss 1.4398 Accuracy0.6776
Epoch 20 Batch 50 Loss 1.4214 Accuracy0.6838
Epoch 20 Batch 100 Loss 1.4334 Accuracy0.6820
Epoch 20 Batch 150 Loss 1.4287 Accuracy0.6833
Epoch 20 Batch 200 Loss 1.4364 Accuracy0.6820
Epoch 20 Batch 250 Loss 1.4398 Accuracy0.6815
Epoch 20 Batch 300 Loss 1.4402 Accuracy0.6813
Epoch 20 Batch 350 Loss 1.4458 Accuracy0.6804
Epoch 20 Batch 400 Loss 1.4458 Accuracy0.6805
Epoch 20 Batch 450 Loss 1.4450 Accuracy0.6804
Epoch 20 Batch 500 Loss 1.4463 Accuracy0.6801
Epoch 20 Batch 550 Loss 1.4490 Accuracy0.6795
Epoch 20 Batch 600 Loss 1.4511 Accuracy0.6793
Epoch 20 Batch 650 Loss 1.4533 Accuracy0.6788
Epoch 20 Batch 700 Loss 1.4567 Accuracy0.6782
Epoch 20 Batch 750 Loss 1.4590 Accuracy0.6778
Epoch 20 Batch 800 Loss 1.4610 Accuracy0.6775
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4617 Accuracy0.6774
Time taken for 1 epoch: 95.57 secs

```

```

class Translator(tf.Module):
    def __init__(self, tokenizers, transformer):
        self.tokenizers = tokenizers
        self.transformer = transformer
    def __call__(self, sentence, max_length=20):
        # input sentence is portuguese, hence adding the start and
        end token
        assert isinstance(sentence, tf.Tensor)
        if len(sentence.shape) == 0:
            sentence = sentence[tf.newaxis]
        sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
        encoder_input = sentence
        # as the target is english, the first token to the
        transformer should be the
        # english start token.
        start_end = self.tokenizers.en.tokenize([' '])[0]
        start = start_end[0][tf.newaxis]
        end = start_end[1][tf.newaxis]
        output_array = tf.TensorArray(dtype=tf.int64, size=0,

```

```

dynamic_size=True)
output_array = output_array.write(0, start)
for i in tf.range(max_length):
    output = tf.transpose(output_array.stack())
    predictions, _ = self.transformer([encoder_input, output],
training=False)
    predictions = predictions[:, -1:, :] # (batch_size, 1,
vocab_size)
    predicted_id = tf.argmax(predictions, axis=-1)
    output_array = output_array.write(i+1, predicted_id[0])
    if predicted_id == end:
        break

    output = tf.transpose(output_array.stack())
    # output.shape (1, tokens)
    text = tokenizers.en.detokenize(output)[0]
    tokens = tokenizers.en.lookup(output)[0]
    _, attention_weights = self.transformer([encoder_input,
output[:, :-1]], training=False)
    return text, tokens, attention_weights

translator = Translator(tokenizers, transformer)

sentence = "Eu li sobre triceratops na enciclopédia."

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

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

Prediction : b'i read about tribucis in egypt .'

## **APÊNDICE 10 – BIG DATA**

### **A - ENUNCIADO**

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

### **B - RESOLUÇÃO**

#### **Implementação de uma aplicação de Big Data**

##### **Análise de dados públicos de CNPJ para análise de compliance, cálculo de marketshare e vendas**

#### **1 Contexto**

A crescente complexidade do ambiente de negócios exige soluções inovadoras para a gestão de dados. Soluções baseadas em big data permitem extrair, refinar, armazenar e rotular dados de várias fontes e com as garantias de qualidade para que os negócios as utilizem.

O impacto da utilização dessas tecnologias é tao grande, que as empresas líderes no uso de big data tem maior probabilidade de serem líderes em seu setor e de superar seus concorrentes (McAfee et al., 2012).

A utilização de informações públicas confere uma vantagem competitiva significativa às empresas. Ao acessar dados precisos e atualizados sobre o mercado, as organizações podem identificar novas oportunidades de negócio, personalizar suas estratégias de marketing e otimizar seus processos de vendas. Além disso, a análise de informações públicas permite antecipar tendências de mercado, reduzir riscos e tomar decisões mais assertivas.

Este estudo de caso explora a criação de uma estrutura de big data, utilizando Hadoop, para extrair informações referentes as empresas brasileiras, disponibilizadas pela Receita Federal.

A estruturação da pipeline de dados e a criação de um data lake servirá como fonte única e confiável de informações para toda a organização. Ao eliminar a necessidade de coletar dados de diversas fontes e realizar processos manuais, a solução proposta proporciona uma significativa redução de custos e um aumento da agilidade nas tomadas de decisão.

Após o devido processamento dessas informações, teremos uma gama interessante de dados das empresas brasileiras (CNPJ, status, data de abertura, CNAEs, endereço, contatos, sócios) e serão possíveis, dentre muitas outras aplicabilidades, aplicações em 3 focos distintos: análises de compliance de fornecedores, cálculo de market share e execução de campanhas de vendas.

### 1.1 Análises de Compliance de Fornecedores

Tendo informações do quadro societário das empresas fornecedoras, pode-se realizar análises referentes a PLDFT (Prevenção a Lavagem de Dinheiro e Financiamento ao Terrorismo). Alguns exemplos de listas restritivas que podem ser analisadas:

- **PEP:** Apresenta Pessoas Politicamente Expostas (PEP), ou seja, alguém que ocupa, ou ocupou nos últimos anos, um cargo público importante. Devido à sua posição de poder e influência, essas pessoas são consideradas mais suscetíveis a atos de corrupção, como lavagem de dinheiro e suborno.
- **OFAC:** Apresenta empresas, entidades e pessoas físicas com envolvimento direto ou relação com grupos de terroristas e narcotraficantes.

### 1.2 Cálculo Market Share

Com as informações de CNPJ, status, CNAE e data de abertura é possível calcular quantas das empresas possivelmente clientes (CNAE, localização...) fazem parte da sua carteira e quantas não, chegando a um cálculo preciso e dinâmico do Market Share.

### 1.3 Execução de Campanhas de Vendas

Entendo quem é e, principalmente, quem não é cliente da sua carteira, é possível utilizar os contatos para campanhas de vendas. Além de utilizar outras informações da empresa para qualificar o lead dentro do funil de vendas.

## 2 Caracterização dos Dados

Os dados usados neste estudo vêm da base pública da Receita Federal de CNPJs. Tal base contém diversos dados de empresas, porém para nosso estudo de caso focamos somente em dados como nome da empresa, CNPJ, região, data de abertura, sócios, e dados de contato.

## 2.1 Os V's de Big Data

- **Volume:** A base de dados de CNPJs inclui milhões de registros de empresas em todo o Brasil, o que justifica o uso de Hadoop para processar grandes volumes de dados.
- **Velocidade:** A análise de compliance e cálculo de market share exigem o processamento em tempo hábil para que as decisões possam ser tomadas rapidamente.
- **Variedade:** A base de dados da Receita Federal contém diferentes tipos de informações que variam entre dados estruturados (como CNPJ, nome da empresa e data de abertura) e semi-estruturados (como os dados de contato e a relação de sócios).
- **Veracidade:** A precisão e a confiabilidade dos dados são fundamentais para garantir que as análises, como as de compliance de fornecedores, sejam seguras e sem falhas. Informações incorretas sobre o quadro societário ou o status de uma empresa podem levar a decisões erradas, como a seleção de fornecedores inadequados ou a análise errada do market share. Assim, a limpeza e verificação dos dados antes do processamento são etapas essenciais.
- **Valor:** O valor gerado a partir desses dados está diretamente relacionado à capacidade da empresa em usá-los para otimizar suas operações e aumentar a competitividade. Ao identificar novos leads e fazer cálculos precisos de market share, a empresa consegue traçar estratégias de vendas mais eficientes e aumentar sua receita.

## 3 Pipeline de Processamento

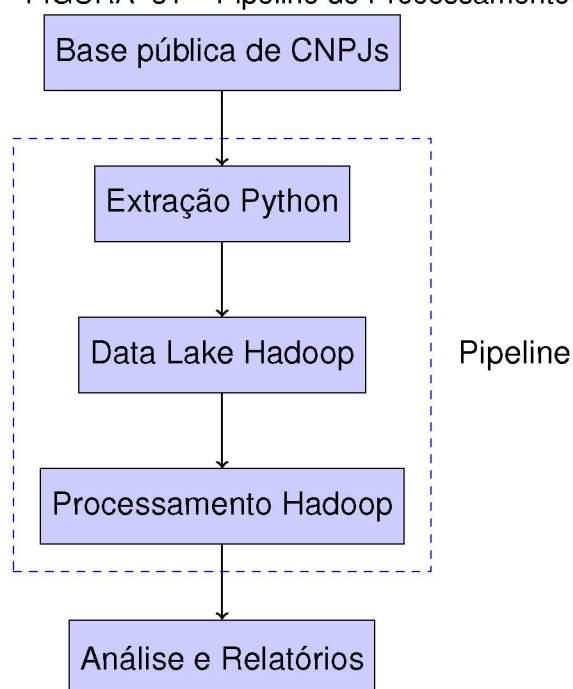
### 3.1 Extração

Os dados são extraídos diretamente da base pública de CNPJs da Receita Federal. São utilizados scripts em Python para baixar os arquivos e em seguida realizar uma etapa de limpeza e transformação, que inclui o tratamento de dados ausentes, a padronização de formatos e a remoção de registros duplicados. O objetivo é garantir que os dados estejam prontos para serem analisados e armazenados no Data Lake.

### 3.2 Data Lake

Os dados transformados são organizados em um Data Lake no Hadoop, o que permite o armazenamento de grandes volumes de dados de maneira escalável.

FIGURA 31 – Pipeline de Processamento



Fonte: O autor (2025).

### 3.3 Processamento com Hadoop

A etapa de processamento dos dados é realizada utilizando o MapReduce no Hadoop. Este modelo de processamento distribui as tarefas de maneira eficiente, permitindo que grandes volumes de dados sejam analisados em paralelo. No contexto deste estudo de caso, o MapReduce é utilizado para realizar cálculos complexos, como:

- Análises de compliance de fornecedores, cruzando informações do quadro societário com listas restritivas.
- Cálculo dinâmico do market share, considerando a quantidade de empresas em diferentes setores e regiões.
- Otimização das campanhas de vendas, identificando leads potenciais a partir dos dados de contato e características das empresas.

### 3.4 Ferramentas utilizadas

- **Hadoop:** Armazenamento e processamento distribuído.
- **Python:** Scripts de coleta e transformação dos dados.
- **Pandas:** Análise antes de enviar os dados ao cluster Hadoop.

## 4 Conclusão

A implementação deste pipeline utilizando Hadoop e Python possibilitou o processamento eficiente de grandes volumes de dados públicos da Receita Federal, oferecendo uma solução escalável e robusta para lidar com a complexidade e o volume de dados empresariais. A utilização do MapReduce permitiu a distribuição das tarefas de processamento, garantindo agilidade e precisão nas análises de compliance, cálculo de market share e otimização de campanhas de vendas.



## 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:

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

- h) Indique qual modelo dá o melhor o resultado e a métrica utilizada

## 2 Redes Neurais

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

Tarefas:

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

## B - RESOLUÇÃO

### 1 Extração de Características

Importação das bibliotecas utilizadas

```
#Resolvendo problema compatibilidade ImageDataGenerator
!pip uninstall -y tensorflow
!pip uninstall -y tf-keras
!pip install tensorflow==2.15

import os
```

```

import random
import shutil
import csv
from collections import defaultdict
from pathlib import Path

import numpy as np
import pandas as pd
import cv2

# Keras e TensorFlow
from keras.models import Model, Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.optimizers import Adam
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint, EarlyStopping

from tensorflow.keras.applications import VGG16, ResNet50
from tensorflow.keras.preprocessing.image import
    ImageDataGenerator
from tensorflow.keras.applications.vgg16 import preprocess_input
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import
    image_dataset_from_directory

# Scikit-learn
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix,
    classification_report, recall_score, precision_score,
    f1_score

#Seleção a T4 GPU

```

## Extração de Características

### 1 Carregue a base de dados de Treino

```
!unzip Train_Warwick.zip
```

```

diretorio_dataset = '/content/Train_4cls_amostra'

dataset_treino = image_dataset_from_directory(
    diretorio_dataset,
    image_size=(250, 250), # Tamanho das imagens
    batch_size=32,         # Tamanho do lote
    label_mode='int',      # Modo de rotulagem (inteiro)
    seed=123,              # Semente para aleatoriedade
    shuffle=True           # Aleatorizar as imagens
)

nomes_classes = dataset_treino.class_names
print("Classes:", nomes_classes)

```

```

Found 593 files belonging to 4 classes.
Classes: ['0', '1', '2', '3']

```

2 Crie partições contendo 80% para treino e 20% para validação (atenção aos pacientes)

```

diretorio_dataset = '/content/Train_4cls_amostra'
diretorio_treino = '/content/Train_split'
diretorio_validacao = '/content/Val_split'

def remover_diretorio(caminho_dir):
    if os.path.exists(caminho_dir):
        shutil.rmtree(caminho_dir)

remover_diretorio(diretorio_treino)
remover_diretorio(diretorio_validacao)
os.makedirs(diretorio_treino, exist_ok=True)
os.makedirs(diretorio_validacao, exist_ok=True)

imagens_pacientes = defaultdict(list)
pacientes_por_classe = defaultdict(set)
for class_dir in os.listdir(diretorio_dataset):
    caminho_classe = os.path.join(diretorio_dataset, class_dir)
    if os.path.isdir(caminho_classe):
        for nome_img in os.listdir(caminho_classe):
            id_paciente = nome_img.split('_')[0] # ID do
            paciente

```

```

        imagens_pacientes[id_paciente].append((os.path.join(
            caminho_classe, nome_img), class_dir))
        pacientes_por_classe[class_dir].add(id_paciente)

pacientes_treino = set()
pacientes_validacao = set()
for nome_classe, pacientes in pacientes_por_classe.items():
    paciente_validacao = random.choice(list(pacientes))
    pacientes_validacao.add(paciente_validacao)
    pacientes.remove(paciente_validacao)

todos_pacientes = list(imagens_pacientes.keys())
random.shuffle(todos_pacientes)

for id_paciente in todos_pacientes:
    if id_paciente not in pacientes_validacao:
        pacientes_treino.add(id_paciente)

for class_dir in os.listdir(diretorio_dataset):
    os.makedirs(os.path.join(diretorio_treino, class_dir),
        exist_ok=True)
    os.makedirs(os.path.join(diretorio_validacao, class_dir),
        exist_ok=True)

for id_paciente in pacientes_treino:
    for caminho_img, class_dir in imagens_pacientes[id_paciente]:
        shutil.move(caminho_img, os.path.join(diretorio_treino,
            class_dir, os.path.basename(caminho_img)))

for id_paciente in pacientes_validacao:
    for caminho_img, class_dir in imagens_pacientes[id_paciente]:
        shutil.move(caminho_img, os.path.join(diretorio_validacao,
            class_dir, os.path.basename(caminho_img)))

# Caminhos das pastas de treino e validação
train_dir = '/content/Train_split'
val_dir = '/content/Val_split'

def count_classes_in_directory(directory):
    class_counts = {}
    for class_folder in os.listdir(directory):
        class_path = os.path.join(directory, class_folder)

```



```

        if os.path.isdir(class_path):
            class_counts[class_folder] = len(os.listdir(
class_path))
        return class_counts

# Contagem de classes na base de treino e validação
train_class_counts = count_classes_in_directory(train_dir)
val_class_counts = count_classes_in_directory(val_dir)

print("Classes treino:")
for cls, count in train_class_counts.items():
    print(f"Classe {cls}: {count} imagens")

print("\nClasses validacao:")
for cls, count in val_class_counts.items():
    print(f"Classe {cls}: {count} imagens")

```

```

Classes treino:
Classe 1: 120 imagens
Classe 3: 120 imagens
Classe 2: 120 imagens
Classe 0: 116 imagens

Classes validacao:
Classe 1: 27 imagens
Classe 3: 30 imagens
Classe 2: 30 imagens
Classe 0: 30 imagens

```

3 Extraia características utilizando LBP e a CNN VGG16 (gerando um csv para cada extrator)

### 3.1 LBP

```

# Função para calcular LBP
def get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:

```

```

        pass
    return new_value

def lbp_calculated_image(img):
    height, width = img.shape
    img_lbp = np.zeros((height, width), np.uint8)

    for x in range(1, height-1):
        for y in range(1, width-1):
            center = img[x][y]
            val_ar = [
                get_pixel(img, center, x-1, y-1),
                get_pixel(img, center, x-1, y),
                get_pixel(img, center, x-1, y+1),
                get_pixel(img, center, x, y+1),
                get_pixel(img, center, x+1, y+1),
                get_pixel(img, center, x+1, y),
                get_pixel(img, center, x+1, y-1),
                get_pixel(img, center, x, y-1)
            ]
            power_val = [1, 2, 4, 8, 16, 32, 64, 128]
            img_lbp[x, y] = sum(val_ar[i] * power_val[i] for i in
                                range(len(val_ar)))

    return img_lbp

def processar_imagens(diretorio, nome_arquivo):
    caracteristicas = []

    for class_dir in os.listdir(diretorio):
        caminho_classe = os.path.join(diretorio, class_dir)
        if os.path.isdir(caminho_classe):
            for nome_img in os.listdir(caminho_classe):
                caminho_img = os.path.join(caminho_classe,
                nome_img)

                # Ler imagem e converter para escala de cinza
                img = cv2.imread(caminho_img, cv2.
                IMREAD_GRAYSCALE)

                # Calcular LBP
                img_lbp = lbp_calculated_image(img)

```



```

        # Calcular histograma
        hist = cv2.calcHist([img_lbp], [0], None, [256],
[0, 256])

        hist_normalizado = hist.flatten() / hist.sum() #
Normalizar

        # Adicionar as características
        caracteristicas.append((class_dir, nome_img, *
hist_normalizado))

# Gerar CSV
with open(nome_arquivo, 'w', newline='') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow(['Classe', 'Imagem'] + [f'Bin_{i}' for i
in range(256)]) # Cabeçalho
    writer.writerows(caracteristicas)

# Processar diretórios de treino e validação
processar_imagens(diretorio_treino, '/content/
caracteristicas_lbp_treino.csv')
processar_imagens(diretorio_validacao, '/content/
caracteristicas_lbp_validacao.csv')

#CSV gerado com características LBP para treino e validação.

```

### 3.2 VGG

```

from keras.preprocessing import image

# Carregar o modelo VGG16 pré-treinado
model_vgg = VGG16(weights='imagenet', include_top=False, pooling=
'avg')

def extrair_caracteristicas_vgg(diretorio, arquivo_saida):
    caracteristicas = []

    for class_dir in os.listdir(diretorio):
        caminho_classe = os.path.join(diretorio, class_dir)
        if os.path.isdir(caminho_classe):
            for nome_img in os.listdir(caminho_classe):

```

```

        caminho_img = os.path.join(caminho_classe,
nome_img)

        # Pré-processar a imagem para VGG16
        img = image.load_img(caminho_img, target_size
=(250, 250)) # Mudar para 250x250 se necessário
        img_array = image.img_to_array(img)
        img_array = np.expand_dims(img_array, axis=0)
        img_array = preprocess_input(img_array)

        # Extrair características
        features = model_vgg.predict(img_array)

        # Adicionar as características junto com a classe
e nome da imagem
        caracteristicas.append((class_dir, nome_img, *
features.flatten())) # Flatten para um vetor

# Gerar CSV
with open(arquivo_saida, 'w', newline='') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow(['Classe', 'Imagem'] + [f'Feature_{i}'
for i in range(features.shape[1])]) # Cabeçalho
    writer.writerows(caracteristicas)

# Chame a função para processar os diretórios de treino e validação
extrair_caracteristicas_vgg(diretorio_treino, '/content/
caracteristicas_treino_vgg16.csv')
extrair_caracteristicas_vgg(diretorio_validacao, '/content/
caracteristicas_validacao_vgg16.csv')

```

## 4 Treine modelos Random Forest, SVM e RNA para predição dos dados extraídos

### 4.1 Random Forest

```

# características LBP
lbp_df_train = pd.read_csv('/content/caracteristicas_lbp_treino.
csv')
lbp_df_val = pd.read_csv('/content/caracteristicas_lbp_validacao.
csv')

```

```

# características VGG
vgg_df_train = pd.read_csv('/content/caracteristicas_treino_vgg16
.csv')
vgg_df_val = pd.read_csv('/content/
caracteristicas_validacao_vgg16.csv')

# Concatenar características LBP e VGG
X_train = pd.concat([lbp_df_train.drop(['Classe', 'Imagem'], axis
=1),
                    vgg_df_train.drop(['Classe', 'Imagem'], axis
=1)], axis=1)
y_train = lbp_df_train['Classe']

X_val = pd.concat([lbp_df_val.drop(['Classe', 'Imagem'], axis=1),
                  vgg_df_val.drop(['Classe', 'Imagem'], axis=1)
                  ], axis=1)
y_val = lbp_df_val['Classe']

# Treinar Random Forest
model_rf = RandomForestClassifier(n_estimators=100, random_state
=42)
model_rf.fit(X_train, y_train)

# Fazer previsões no conjunto de validação
y_pred = model_rf.predict(X_val)

# Avaliar o modelo
conf_matrix = confusion_matrix(y_val, y_pred)
class_report = classification_report(y_val, y_pred)

print("Matriz de Confusao:\n", conf_matrix)
print("\nRelatorio de Classificacao:\n", class_report)

```

Matriz de Confusao:

```

[[27  1  0  0]
 [21  3  6  0]
 [ 0  5 25  0]
 [ 0  0  1 29]]

```

Relatorio de Classificacao:

|  | precision | recall | f1-score | support |
|--|-----------|--------|----------|---------|
|--|-----------|--------|----------|---------|

|              |      |      |      |     |
|--------------|------|------|------|-----|
| 0            | 0.56 | 0.96 | 0.71 | 28  |
| 1            | 0.33 | 0.10 | 0.15 | 30  |
| 2            | 0.78 | 0.83 | 0.81 | 30  |
| 3            | 1.00 | 0.97 | 0.98 | 30  |
| accuracy     |      |      | 0.71 | 118 |
| macro avg    | 0.67 | 0.72 | 0.66 | 118 |
| weighted avg | 0.67 | 0.71 | 0.66 | 118 |

## 4.2 SVM

```
# Carregar os CSVs
data_lbp_train = pd.read_csv('/content/caracteristicas_lbp_treino.csv')
data_lbp_val = pd.read_csv('/content/caracteristicas_lbp_validacao.csv')
data_vgg_train = pd.read_csv('/content/caracteristicas_treino_vgg16.csv')
data_vgg_val = pd.read_csv('/content/caracteristicas_validacao_vgg16.csv')

print("Dimensões LBP Treino:", data_lbp_train.shape)
print("Dimensões VGG Treino:", data_vgg_train.shape)
print("Dimensões LBP Validação:", data_lbp_val.shape)
print("Dimensões VGG Validação:", data_vgg_val.shape)

# Concatenar características LBP e VGG
X_train = np.concatenate((data_lbp_train.iloc[:, 2:].values,
                           data_vgg_train.iloc[:, 2:].values), axis=1) # Todas as
                               # colunas exceto as duas primeiras
y_train = data_lbp_train['Classe'].values # Usando as classes do
                                           # LBP, que são as mesmas

X_val = np.concatenate((data_lbp_val.iloc[:, 2:].values,
                        data_vgg_val.iloc[:, 2:].values), axis=1) # Para o conjunto
                           # de validação
y_val = data_lbp_val['Classe'].values # Usando as classes do LBP

# Treinar o modelo SVM
model_svm = SVC(kernel='linear', random_state=42) # Você pode
                                                    # experimentar outros kernels como 'rbf'
```

```

model_svm.fit(X_train, y_train)

# Fazer previsões no conjunto de validação
y_pred = model_svm.predict(X_val)

# Avaliar o modelo
conf_matrix = confusion_matrix(y_val, y_pred)
report = classification_report(y_val, y_pred)

print("Matriz de Confusão:")
print(conf_matrix)
print("\nRelatório de Classificação:")
print(report)

```

```

Dimensões LBP Treino: (475, 258)
Dimensões VGG Treino: (475, 514)
Dimensões LBP Validação: (118, 258)
Dimensões VGG Validação: (118, 514)
Matriz de Confusão:
[[25  3  0  0]
 [ 5 25  0  0]
 [ 0  4 26  0]
 [ 0  0  1 29]]

Relatório de Classificação:

```

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.83      | 0.89   | 0.86     | 28      |
| 1            | 0.78      | 0.83   | 0.81     | 30      |
| 2            | 0.96      | 0.87   | 0.91     | 30      |
| 3            | 1.00      | 0.97   | 0.98     | 30      |
| accuracy     |           |        | 0.89     | 118     |
| macro avg    | 0.89      | 0.89   | 0.89     | 118     |
| weighted avg | 0.90      | 0.89   | 0.89     | 118     |

### 4.3 RNA

```

# características LBP
lbp_df_train = pd.read_csv('/content/caracteristicas_lbp_treino.
csv')

```



```

lbp_df_val = pd.read_csv('/content/caracteristicas_lbp_validacao.
    csv')

# características VGG
vgg_df_train = pd.read_csv('/content/caracteristicas_treino_vgg16
    .csv')
vgg_df_val = pd.read_csv('/content/
    caracteristicas_validacao_vgg16.csv')

# Concatenar características LBP e VGG
X_train = pd.concat([lbp_df_train.drop(['Classe', 'Imagem'], axis
    =1),
                    vgg_df_train.drop(['Classe', 'Imagem'], axis
    =1)], axis=1)
y_train = lbp_df_train['Classe']

X_val = pd.concat([lbp_df_val.drop(['Classe', 'Imagem'], axis=1),
                    vgg_df_val.drop(['Classe', 'Imagem'], axis=1)
                    ], axis=1)
y_val = lbp_df_val['Classe']

label_encoder = LabelEncoder()
y_train_encoded = label_encoder.fit_transform(y_train)
y_val_encoded = label_encoder.transform(y_val)

y_train_categorical = to_categorical(y_train_encoded)
y_val_categorical = to_categorical(y_val_encoded)

# Criar o modelo da RNA
model_rna = Sequential()
model_rna.add(Dense(256, activation='relu', input_shape=(X_train.
    shape[1],))) # Primeira camada oculta
model_rna.add(Dense(128, activation='relu')) # Segunda camada
    oculta
model_rna.add(Dense(len(label_encoder.classes_), activation='
    softmax')) # Camada de saída com softmax

# Compilar o modelo
model_rna.compile(optimizer='adam', loss='
    categorical_crossentropy', metrics=['accuracy'])

# Treinar o modelo

```

```
model_rna.fit(X_train, y_train_categorical, epochs=50, batch_size
              =32, validation_data=(X_val, y_val_categorical))

# Avaliar o modelo
loss, accuracy = model_rna.evaluate(X_val, y_val_categorical)
print(f'\nAcurácia no conjunto de validação: {accuracy:.4f}')
```

```
#Acurácia no conjunto de validação: 0.8898
```

## 5 Carregue a base de Teste e execute a tarefa 3 nesta base

```
!unzip Test_Warwick.zip
```

### tarefa 3

```
diretorio_teste = '/content/Test_4cl_amostra'

# LBP
processar_imagens(diretorio_teste, '/content/
                    caracteristicas_lbp_teste.csv')
#CSV gerado com características LBP para a base de teste.

# VGG16
extrair_caracteristicas_vgg(diretorio_teste, '/content/
                    caracteristicas_vgg16_teste.csv')
#CSV gerado com características VGG16 para a base de teste.
```

## 6 Aplique os modelos treinados nos dados de teste

```
# Carregar as características LBP e VGG da base de teste
lbp_df_test = pd.read_csv('/content/caracteristicas_lbp_teste.csv
                          ')
vgg_df_test = pd.read_csv('/content/caracteristicas_vgg16_teste.
                          csv')

X_test = pd.concat([lbp_df_test.drop(['Classe', 'Imagem'], axis
                                     =1),
                    vgg_df_test.drop(['Classe', 'Imagem'], axis
                                     =1)], axis=1)

y_test = lbp_df_test['Classe']
```



## 6.1 Random Forest

```
# Fazer previsões no conjunto de teste com o modelo Random Forest
    treinado
y_pred_rf = model_rf.predict(X_test)

# Avaliar o modelo
conf_matrix_rf = confusion_matrix(y_test, y_pred_rf)
class_report_rf = classification_report(y_test, y_pred_rf)

print("Matriz de Confusão (Random Forest) no conjunto de teste:\n
      ", conf_matrix_rf)
print("\nRelatório de Classificação (Random Forest) no conjunto
      de teste:\n", class_report_rf)
```

```
Matriz de Confusão (Random Forest) no conjunto de teste:
[[98  3  0  0]
 [16 49 25  0]
 [ 0 22 68  0]
 [ 0  0  4 86]]

Relatório de Classificação (Random Forest) no conjunto de teste:
              precision    recall  f1-score   support

     0       0.86         0.97         0.91         101
     1       0.66         0.54         0.60          90
     2       0.70         0.76         0.73          90
     3       1.00         0.96         0.98          90

 accuracy                   0.81         371
 macro avg       0.81         0.81         0.80         371
weighted avg       0.81         0.81         0.81         371
```

## 6.2 SVM

```
# Fazer previsões no conjunto de teste com o modelo SVM treinado
y_pred_svm = model_svm.predict(X_test)

# Avaliar o modelo
conf_matrix_svm = confusion_matrix(y_test, y_pred_svm)
class_report_svm = classification_report(y_test, y_pred_svm)
```

```
print("Matriz de Confusão (SVM) no conjunto de teste:\n",
      conf_matrix_svm)
print("\nRelatório de Classificação (SVM) no conjunto de teste:\n",
      class_report_svm)
```

Matriz de Confusão (SVM) no conjunto de teste:

```
[[87 14  0  0]
 [ 6 69 15  0]
 [ 0 20 69  1]
 [ 0  0  0 90]]
```

Relatório de Classificação (SVM) no conjunto de teste:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.94      | 0.86   | 0.90     | 101     |
| 1            | 0.67      | 0.77   | 0.72     | 90      |
| 2            | 0.82      | 0.77   | 0.79     | 90      |
| 3            | 0.99      | 1.00   | 0.99     | 90      |
| accuracy     |           |        | 0.85     | 371     |
| macro avg    | 0.85      | 0.85   | 0.85     | 371     |
| weighted avg | 0.86      | 0.85   | 0.85     | 371     |

### 6.3 RNA

```
# Fazer previsões no conjunto de teste com o modelo RNA treinado
y_pred_rna_prob = model_rna.predict(X_test)

# Converter probabilidades em rótulos de classe (pegando o índice
da classe com maior probabilidade)
y_pred_rna = np.argmax(y_pred_rna_prob, axis=1)

# Avaliar o modelo
conf_matrix_rna = confusion_matrix(y_test, y_pred_rna)
class_report_rna = classification_report(y_test, y_pred_rna)

print("Matriz de Confusão (RNA) no conjunto de teste:\n",
      conf_matrix_rna)
print("\nRelatório de Classificação (RNA) no conjunto de teste:\n",
      class_report_rna)
```

Matriz de Confusão (RNA) no conjunto de teste:

```
[[96  5  0  0]
 [ 8 55 27  0]
 [ 0 21 66  3]
 [ 0  0  1 89]]
```

Relatório de Classificação (RNA) no conjunto de teste:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.92      | 0.95   | 0.94     | 101     |
| 1            | 0.68      | 0.61   | 0.64     | 90      |
| 2            | 0.70      | 0.73   | 0.72     | 90      |
| 3            | 0.97      | 0.99   | 0.98     | 90      |
| accuracy     |           |        | 0.82     | 371     |
| macro avg    | 0.82      | 0.82   | 0.82     | 371     |
| weighted avg | 0.82      | 0.82   | 0.82     | 371     |

7 Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão

```
# Função para calcular especificidade para múltiplas classes
def calcular_especificidade_multiclasse(conf_matrix):
    especificidades = []
    for i in range(conf_matrix.shape[0]):
        tn = conf_matrix.sum() - conf_matrix[i, :].sum() -
        conf_matrix[:, i].sum() + conf_matrix[i, i]
        fp = conf_matrix[:, i].sum() - conf_matrix[i, i]
        especificidades.append(tn / (tn + fp) if (tn + fp) > 0
        else 0)
    return np.mean(especificidades)

# Cálculo das métricas para Random Forest
sensibilidade_rf = recall_score(y_test, y_pred_rf, average='macro')
f1_rf = f1_score(y_test, y_pred_rf, average='macro')
especificidade_rf = calcular_especificidade_multiclasse(
    conf_matrix_rf)

print(f"Métricas para Random Forest:")
print(f"F1-Score: {f1_rf:.4f}")
```

```
print(f"Sensibilidade: {sensibilidade_rf:.4f}")
print(f"Especificidade: {especificidade_rf:.4f}")
```

Métricas para Random Forest:

F1-Score: 0.8034

Sensibilidade: 0.8065

Especificidade: 0.9371

```
# Cálculo das métricas para SVM
sensibilidade_svm = recall_score(y_test, y_pred_svm, average='
    macro')
f1_svm = f1_score(y_test, y_pred_svm, average='macro')
especificidade_svm = calcular_especificidade_multiclasse(
    conf_matrix_svm)

print(f"\nMétricas para SVM:")
print(f"F1-Score: {f1_svm:.4f}")
print(f"Sensibilidade: {sensibilidade_svm:.4f}")
print(f"Especificidade: {especificidade_svm:.4f}")
```

Métricas para SVM:

F1-Score: 0.8499

Sensibilidade: 0.8487

Especificidade: 0.9500

```
# Cálculo das métricas para RNA
sensibilidade_rna = recall_score(y_test, y_pred_rna, average='
    macro')
f1_rna = f1_score(y_test, y_pred_rna, average='macro')
especificidade_rna = calcular_especificidade_multiclasse(
    conf_matrix_rna)

print(f"\nMétricas para RNA:")
print(f"F1-Score: {f1_rna:.4f}")
print(f"Sensibilidade: {sensibilidade_rna:.4f}")
print(f"Especificidade: {especificidade_rna:.4f}")
```

Métricas para RNA:

F1-Score: 0.8188

Sensibilidade: 0.8210

Especificidade: 0.9419

8 Indique qual modelo dá o melhor o resultado e a métrica utilizada

## RESULTADO

- F1-Score: SVM
- Sensibilidade: SVM
- Especificidade: SVM

SVM apresentou melhor resultado em todas as métricas.

A SVM se destacou como o melhor modelo. Ela apresentou métricas de desempenho superiores, incluindo uma F1-Score mais alta, sensibilidade e especificidade eficazes. A SVM se mostrou eficiente na classificação das imagens, principalmente devido à sua capacidade de lidar bem com dados de alta dimensionalidade e encontrar margens de separação mais precisas entre as classes.

Embora a RNA tenha também demonstrado bom desempenho, a SVM foi mais consistente, o que a torna a escolha ideal para essa tarefa específica.

## 2 Redes Neurais

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

2.1 Sem Data Augmentation

```
!pip install livelossplot
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint
from keras.models import Model
from keras.layers import Dense, Flatten
from keras.applications import VGG16, ResNet50
from livelossplot import PlotLossesKeras

# Definições de parâmetros
BATCH_SIZE = 64 # Tamanho do batch para treinamento
EPOCHS = 10     # Número de épocas para treinamento

# Carregar dados de treino sem Data Augmentation
train_datagen = ImageDataGenerator(rescale=1.0/255.0)
train_generator = train_datagen.flow_from_directory(
```



```

        'Train_split/',
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
    )

# Carregar dados de validação sem Data Augmentation
validation_datagen = ImageDataGenerator(rescale=1.0/255.0)
validation_generator = validation_datagen.flow_from_directory(
    'Val_split/',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
)

# Treinar modelo VGG16
model_vgg_without_augmentation = VGG16(input_shape=(224, 224, 3),
    weights='imagenet', include_top=False) # Use o modelo pré-
    treinado

# Adicionar camadas próprias ao modelo VGG16
x_vgg = Flatten()(model_vgg_without_augmentation.output)
prediction_vgg = Dense(4, activation='softmax')(x_vgg)

# Criação do Objeto Modelo
model_vgg_without_augmentation = Model(inputs=
    model_vgg_without_augmentation.input, outputs=prediction_vgg)

# Compilar o modelo
model_vgg_without_augmentation.compile(loss='
    categorical_crossentropy', optimizer=RMSprop(learning_rate
    =0.0001), metrics=['accuracy'])

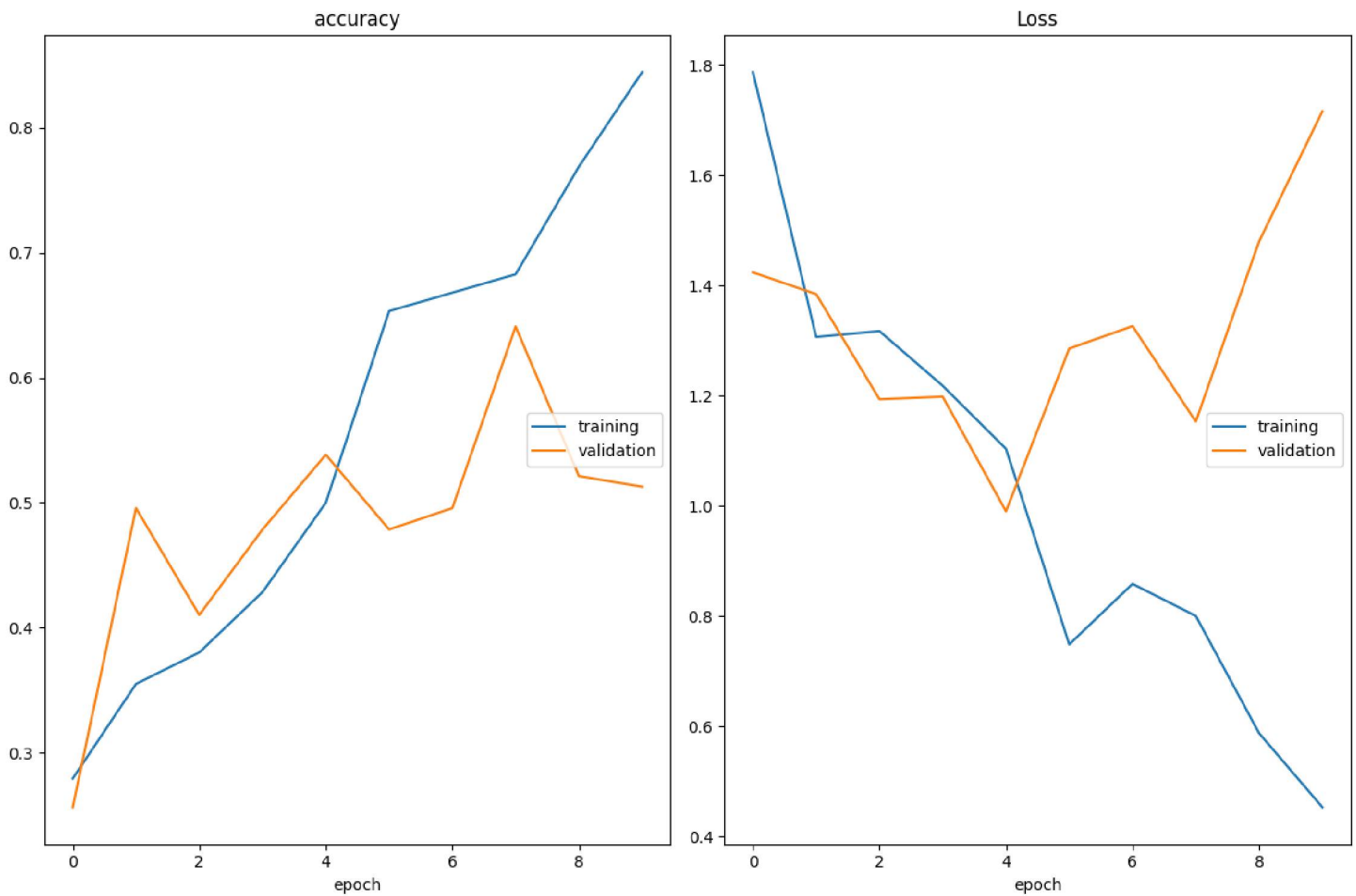
# Treinamento sem Data Augmentation
model_vgg_without_augmentation.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=EPOCHS,
    callbacks=[PlotLossesKeras()])

```

accuracy



FIGURA 32 – Acurácia e perda VGG16 sem data augmentation



Fonte: O autor (2025).

```

training    (min: 0.279, max: 0.845, cur: 0.845)
validation  (min: 0.256, max: 0.641, cur: 0.513)

```

Loss

```

training    (min: 0.453, max: 1.787, cur: 0.453)
validation  (min: 0.990, max: 1.716, cur: 1.716)

```

```

# Treinar modelo ResNet50
model_resnet_without_augmentation = ResNet50(input_shape
      =(224,224,3), weights='imagenet', include_top=False)

# Não treinar os pesos existentes
for layer in model_resnet_without_augmentation.layers:
    layer.trainable = False

# Camadas próprias
x_resnet = Flatten()(model_resnet_without_augmentation.output)
prediction_resnet = Dense(4, activation='softmax')(x_resnet)

```

```

# Criação do Objeto Modelo
model_resnet_without_augmentation = Model(inputs=
    model_resnet_without_augmentation.input, outputs=
    prediction_resnet)

# Compilar o modelo
model_resnet_without_augmentation.compile(loss='
    categorical_crossentropy', optimizer=RMSprop(learning_rate
    =0.0001), metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor
    resultado
checkpointer = ModelCheckpoint(filepath='
    img_model_no_augmentation.weights.best.keras',
                                verbose=1,
                                save_best_only=True)

# Definindo steps_per_epoch e val_steps
steps_per_epoch = train_generator.samples // BATCH_SIZE
val_steps = validation_generator.samples // BATCH_SIZE

# Treinamento sem Data Augmentation para ResNet50
model_resnet_without_augmentation.fit(
    train_generator, # Usando o gerador de treino sem data
    augmentation
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch,
    validation_data=validation_generator,
    validation_steps=val_steps,
    callbacks=[checkpointer, PlotLossesKeras()],
    verbose=False
)

print("Treinamento concluído para VGG16 e ResNet50, sem Data
    Augmentation.")

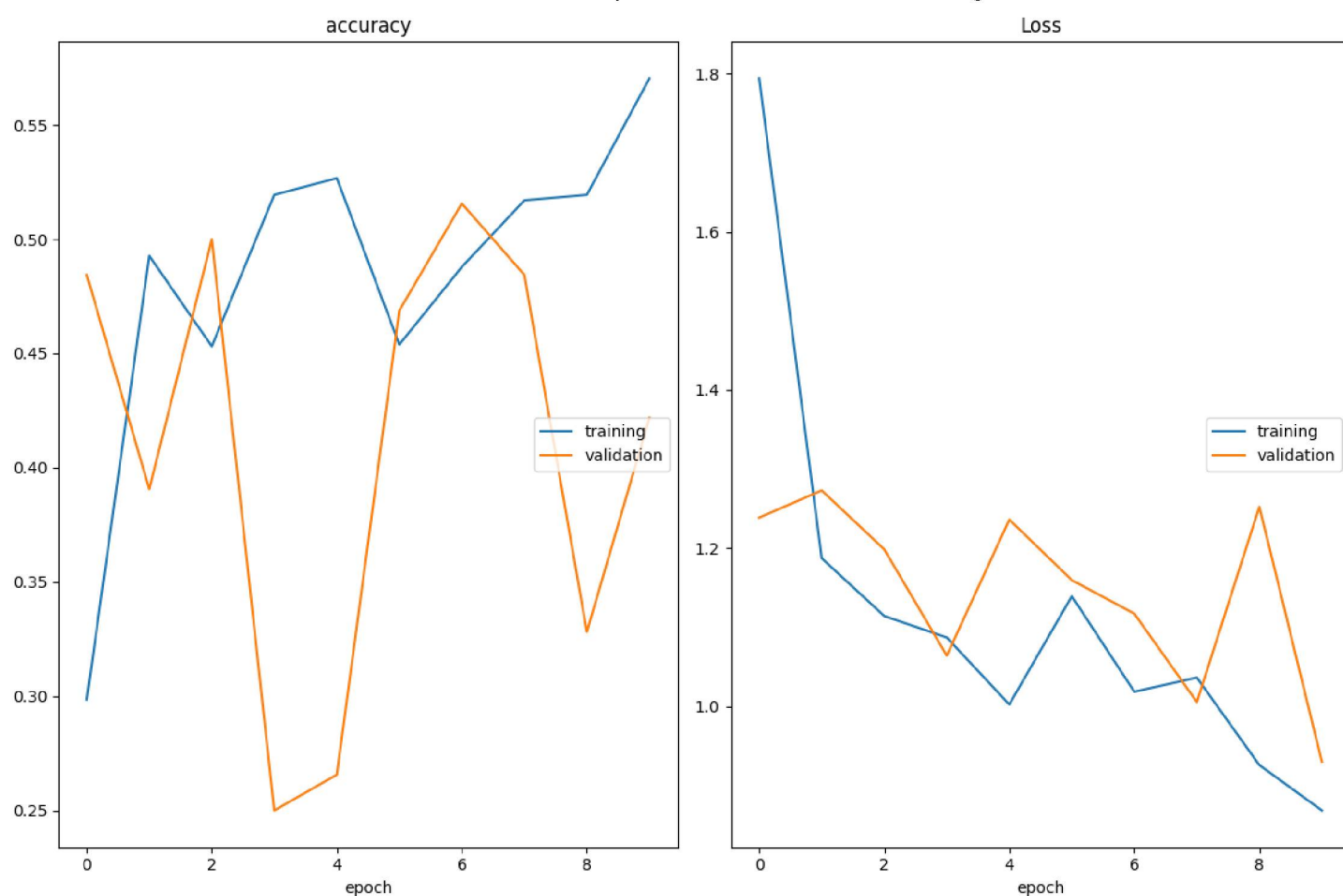
```

```

accuracy
    training    (min: 0.299, max: 0.570, cur: 0.570)
    validation  (min: 0.250, max: 0.516, cur: 0.422)
Loss
    training    (min: 0.869, max: 1.794, cur: 0.869)
    validation  (min: 0.931, max: 1.273, cur: 0.931)

```

FIGURA 33 – Acurácia e perda ResNet50 sem data augmentation



Fonte: O autor (2025).

## 2.2 Com Data Augmentation

```
from keras.optimizers import RMSprop
from keras.callbacks import ModelCheckpoint
from keras.models import Model
from keras.layers import Dense, Flatten
from keras.applications import VGG16, ResNet50
from livelossplot import PlotLossesKeras
from tensorflow.keras.applications.resnet50 import
    preprocess_input

# Definições de parâmetros
BATCH_SIZE = 64 # Tamanho do batch para treinamento
EPOCHS = 10     # Número de épocas para treinamento

# Data Augmentation para o conjunto de treinamento
train_datagen_augmented = ImageDataGenerator(
    rescale=1.0/255.0,
```

```

        rotation_range=90,
        brightness_range=[0.1, 0.7],
        width_shift_range=0.5,
        height_shift_range=0.5,
        horizontal_flip=True,
        vertical_flip=True,
        validation_split=0.2,
        preprocessing_function=preprocess_input
    )

# Carregar dados de treino com Data Augmentation
train_generator_augmented = train_datagen_augmented.
    flow_from_directory(
        'Train_split/',
        target_size=(224, 224),
        batch_size=BATCH_SIZE,
        class_mode='categorical',
    )

# Carregar dados de validação sem Data Augmentation
validation_datagen = ImageDataGenerator(rescale=1.0/255.0,
    preprocessing_function=preprocess_input)
validation_generator = validation_datagen.flow_from_directory(
    'Val_split/',
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
)

# Treinar modelo VGG16
model_vgg_with_augmentation = VGG16(input_shape=(224, 224, 3),
    weights='imagenet', include_top=False) # Use o modelo pré-
    treinado

# Adicionar camadas próprias ao modelo VGG16
x_vgg = Flatten()(model_vgg_with_augmentation.output)
prediction_vgg = Dense(4, activation='softmax')(x_vgg)

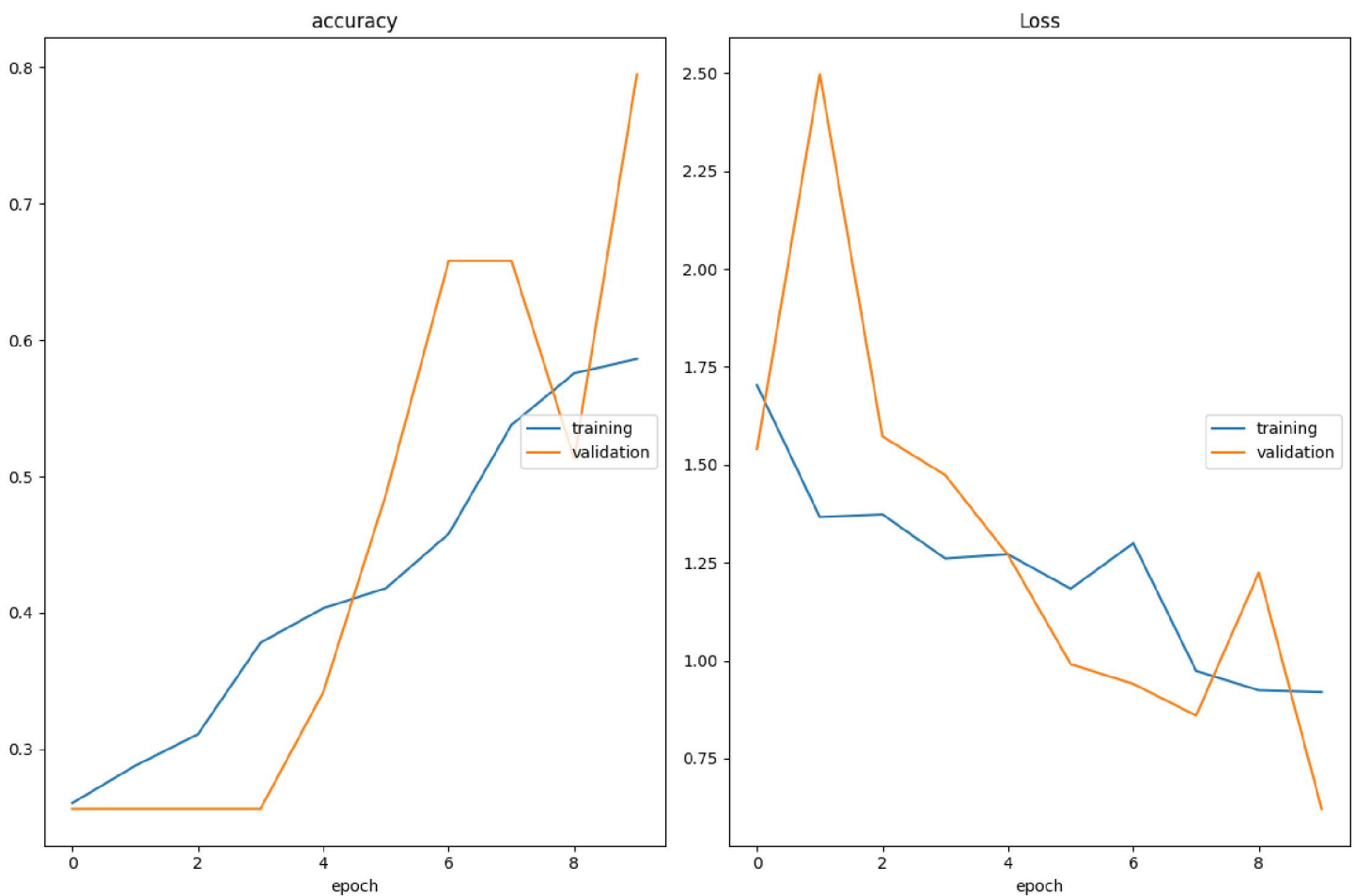
# Criação do Objeto Modelo
model_vgg_with_augmentation = Model(inputs=
    model_vgg_with_augmentation.input, outputs=prediction_vgg)

```

```
# Compilar o modelo
model_vgg_with_augmentation.compile(loss='
    categorical_crossentropy', optimizer=RMSprop(learning_rate
        =0.0001), metrics=['accuracy'])

# Treinamento com Data Augmentation
model_vgg_with_augmentation.fit(
    train_generator_augmented,
    validation_data=validation_generator,
    epochs=EPOCHS,
    callbacks=[PlotLossesKeras()])
)
```

FIGURA 34 – Acurácia e perda VGG16 com data augmentation



Fonte: O autor (2025).

```
accuracy
    training    (min: 0.261, max: 0.586, cur: 0.586)
    validation  (min: 0.256, max: 0.795, cur: 0.795)
Loss
    training    (min: 0.919, max: 1.703, cur: 0.919)
```



```
validation (min: 0.622, max: 2.497, cur: 0.622)
```

```
# Treinar modelo ResNet50
model_resnet_with_augmentation = ResNet50(input_shape=(224,224,3)
    , weights='imagenet', include_top=False)

# Não treinar os pesos existentes
for layer in model_resnet_with_augmentation.layers:
    layer.trainable = False

# Camadas próprias
x_resnet = Flatten()(model_resnet_with_augmentation.output)
prediction_resnet = Dense(4, activation='softmax')(x_resnet)

# Criação do Objeto Modelo
model_resnet_with_augmentation = Model(inputs=
    model_resnet_with_augmentation.input, outputs=
    prediction_resnet)

# Compilar o modelo
model_resnet_with_augmentation.compile(loss='
    categorical_crossentropy', optimizer=RMSprop(learning_rate
    =0.0001), metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor
    resultado
checkpointer = ModelCheckpoint(filepath='img_model_tl.weights.
    best.keras',
                                verbose=1,
                                save_best_only=True)

# Definindo steps_per_epoch e val_steps
steps_per_epoch = train_generator_augmented.samples // BATCH_SIZE
val_steps = validation_generator.samples // BATCH_SIZE

# Treinamento com Data Augmentation para ResNet50
model_resnet_with_augmentation.fit(
    train_generator_augmented, # Usando o gerador de treino com
    data augmentation
    epochs=EPOCHS,
    steps_per_epoch=steps_per_epoch,
    validation_data=validation_generator,
```



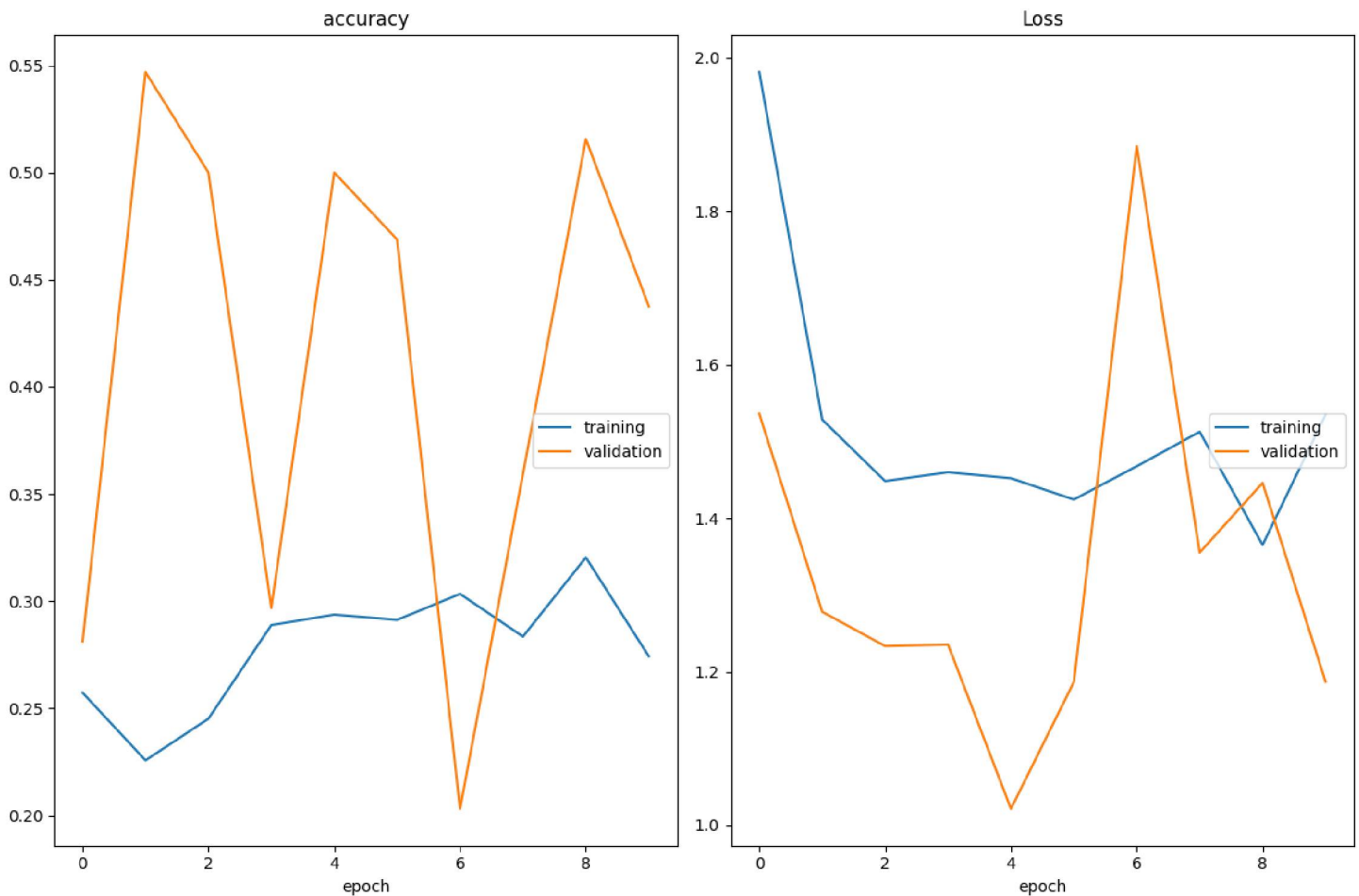
```

validation_steps=val_steps,
callbacks=[checkpointer, PlotLossesKeras()],
verbose=False
)

print("Treinamento concluído para VGG16 e ResNet50, com Data
Augmentation.")

```

FIGURA 35 – Acurácia e perda ResNet50 com data augmentation



Fonte: O autor (2025).

```

accuracy
    training (min: 0.226, max: 0.320, cur: 0.274)
    validation (min: 0.203, max: 0.547, cur: 0.438)
Loss
    training (min: 1.365, max: 1.981, cur: 1.535)
    validation (min: 1.021, max: 1.884, cur: 1.187)

```

3 Aplique os modelos treinados nas imagens da base de Teste

```

test_generator = ImageDataGenerator(preprocessing_function=
    preprocess_input, rescale=1.0/255.0)
testgen = test_generator.flow_from_directory('/content/
    Test_4cl_amostra',
                                           target_size=(224,
224),
                                           batch_size=
BATCH_SIZE,
                                           class_mode=None,
                                           #classes=
class_subset,
                                           shuffle=False,
                                           seed=42)

test_dir = '/content/Test_4cl_amostra'

def carregar_imagens_teste(diretorio):
    imagens = []
    classes = []

    for class_dir in os.listdir(diretorio):
        caminho_classe = os.path.join(diretorio, class_dir)
        if os.path.isdir(caminho_classe):
            for nome_img in os.listdir(caminho_classe):
                caminho_img = os.path.join(caminho_classe,
nome_img)

                # Ler a imagem, redimensionar e normalizar
                img = cv2.imread(caminho_img)
                img = cv2.resize(img, (224, 224))
                img = img / 255.0
                imagens.append(img)
                classes.append(int(class_dir))

    return np.array(imagens), np.array(classes)

#X_test, y_test = carregar_imagens_teste(test_dir)

# Modelos sem Data Augmentation
#y_pred_vgg16_no_aug = model_vgg_no_augmentation.predict(X_test)

```

```

y_pred_vgg16_no_aug = model_vgg_without_augmentation.predict(
    testgen)
y_pred_resnet50_no_aug = model_resnet_without_augmentation.
    predict(testgen)

# Modelos com Data Augmentation
y_pred_vgg16_aug = model_vgg_with_augmentation.predict(testgen)
y_pred_resnet50_aug = model_resnet_with_augmentation.predict(
    testgen)

y_pred_vgg16_no_aug_classes = np.argmax(y_pred_vgg16_no_aug, axis
    =1)
y_pred_resnet50_no_aug_classes = np.argmax(y_pred_resnet50_no_aug
    , axis=1)

y_pred_vgg16_aug_classes = np.argmax(y_pred_vgg16_aug, axis=1)
y_pred_resnet50_aug_classes = np.argmax(y_pred_resnet50_aug, axis
    =1)

```

4 Calcule as métricas de Sensibilidade, Especificidade e F1-Score com base em suas matrizes de confusão

```

# Cálculo das métricas para cada modelo
y_true = testgen.classes

# VGG16 sem Data Augmentation
conf_matrix_vgg16_no_aug = confusion_matrix(y_true,
    y_pred_vgg16_no_aug_classes)
sensibilidade_vgg16_no_aug = recall_score(y_true,
    y_pred_vgg16_no_aug_classes, average='macro')
f1_vgg16_no_aug = f1_score(y_true, y_pred_vgg16_no_aug_classes,
    average='macro')
especificidade_vgg16_no_aug = calcular_especificidade_multiclasse
    (conf_matrix_vgg16_no_aug)

print("Métricas VGG16 sem Data Augmentation:")
print(f"F1-Score: {f1_vgg16_no_aug:.4f}")
print(f"Sensibilidade: {sensibilidade_vgg16_no_aug:.4f}")
print(f"Especificidade: {especificidade_vgg16_no_aug:.4f}\n")

# ResNet50 sem Data Augmentation

```

```

conf_matrix_resnet50_no_aug = confusion_matrix(y_true,
        y_pred_resnet50_no_aug_classes)
sensibilidade_resnet50_no_aug = recall_score(y_true,
        y_pred_resnet50_no_aug_classes, average='macro')
f1_resnet50_no_aug = f1_score(y_true,
        y_pred_resnet50_no_aug_classes, average='macro')
especificidade_resnet50_no_aug =
        calcular_especificidade_multiclasse(
        conf_matrix_resnet50_no_aug)

print("Métricas ResNet50 sem Data Augmentation:")
print(f"F1-Score: {f1_resnet50_no_aug:.4f}")
print(f"Sensibilidade: {sensibilidade_resnet50_no_aug:.4f}")
print(f"Especificidade: {especificidade_resnet50_no_aug:.4f}\n")

# VGG16 com Data Augmentation
conf_matrix_vgg16_aug = confusion_matrix(y_true,
        y_pred_vgg16_aug_classes)
sensibilidade_vgg16_aug = recall_score(y_true,
        y_pred_vgg16_aug_classes, average='macro')
f1_vgg16_aug = f1_score(y_true, y_pred_vgg16_aug_classes, average
        ='macro')
especificidade_vgg16_aug = calcular_especificidade_multiclasse(
        conf_matrix_vgg16_aug)

print("Métricas VGG16 com Data Augmentation:")
print(f"F1-Score: {f1_vgg16_aug:.4f}")
print(f"Sensibilidade: {sensibilidade_vgg16_aug:.4f}")
print(f"Especificidade: {especificidade_vgg16_aug:.4f}\n")

# ResNet50 com Data Augmentation
conf_matrix_resnet50_aug = confusion_matrix(y_true,
        y_pred_resnet50_aug_classes)
sensibilidade_resnet50_aug = recall_score(y_true,
        y_pred_resnet50_aug_classes, average='macro')
f1_resnet50_aug = f1_score(y_true, y_pred_resnet50_aug_classes,
        average='macro')
especificidade_resnet50_aug = calcular_especificidade_multiclasse
        (conf_matrix_resnet50_aug)

print("Métricas ResNet50 com Data Augmentation:")
print(f"F1-Score: {f1_resnet50_aug:.4f}")

```



```
print(f"Sensibilidade: {sensibilidade_resnet50_aug:.4f}")  
print(f"Especificidade: {especificidade_resnet50_aug:.4f}")
```

Métricas VGG16 sem Data Augmentation:

F1-Score: 0.2912

Sensibilidade: 0.3240

Especificidade: 0.7704

Métricas ResNet50 sem Data Augmentation:

F1-Score: 0.4255

Sensibilidade: 0.4898

Especificidade: 0.8316

Métricas VGG16 com Data Augmentation:

F1-Score: 0.8277

Sensibilidade: 0.8444

Especificidade: 0.9486

Métricas ResNet50 com Data Augmentation:

F1-Score: 0.3287

Sensibilidade: 0.5000

Especificidade: 0.8301

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

## RESULTADO

- F1-Score: VGG16 com Data Augmentation
- Sensibilidade: VGG16 com Data Augmentation
- Especificidade: VGG16 com Data Augmentation

VGG16 com Data Augmentation apresentou melhor resultado em todas as métricas.

Após avaliar as performances dos modelos VGG16 e ResNet50 em diferentes configurações, a VGG16 com Data Augmentation se destacou como a melhor opção. Este modelo apresentou as melhores métricas de desempenho, com uma F1-Score superior, sensibilidade e especificidade elevadas.

Em contraste, a ResNet50, embora eficaz, não alcançou resultados tão robustos quanto o VGG16 em situações semelhantes. Assim, a VGG16 com Data Augmen-

tation foi escolhida como o modelo ideal para a tarefa em questão, garantindo uma classificação mais precisa e confiável.



## APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

### A - ENUNCIADO

#### 1 Objetivo

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

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

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

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

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

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

#### 2 Orientações adicionais

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

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

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

- Qual o objetivo do estudo descrito pelo artigo?
- Qual o problema/oportunidade/situação que levou a necessidade de realização deste estudo?

- Qual a metodologia que os autores usaram para obter e analisar as informações do estudo?
- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

## **B - RESOLUÇÃO**

### **Nome do artigo escolhido:**

Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows

### **Resumo do artigo**

O artigo, em sua introdução, enaltece o Aprendizado de Máquina (ML) como promessa de ser o fornecedor de soluções para grandes problemas da atualidade. Porém, sendo uma tecnologia com difícil entendimento e aplicação, há um esforço grande para chegar na precisão desejada. Segue explicando que o Auto-ML é um possível atalho para a utilização dos algoritmos de ML. E, para entender o quanto essas tecnologias estão efetivamente atendendo as necessidades, serão conduzidas entrevistas com dezesseis usuários de Auto-ML a fim de realizar um estudo qualitativo. Em seguida, no capítulo 2, o artigo apresenta 3 diferentes modelos de Auto-ML: sistemas Auto-ML existentes, práticas de trabalho ML centradas no humano e ferramentas ML com humanos no circuito. Cada um deles apresentando suas características e principais soluções existentes no mercado. Sistemas Auto-ML existentes: soluções Auto-ML disponíveis no mercado categorizadas em: software open-source, soluções comerciais hospedadas oferecidas por provedores de nuvem e soluções empresariais oferecidas por empresas dedicadas ao desenvolvimento de plataformas Auto-ML. Práticas de Trabalho ML Centradas no Humano: desenvolvimento de ML deve ser uma atividade colaborativa entre usuários humanos e a máquina, na qual os usuários especificam seu conhecimento de domínio ou comportamento desejável do modelo em alto nível e o sistema executa alguma forma de busca automatizada para gerar um pipeline ou modelo ML apropriado. Ferramentas ML com Humanos no Circuito: busca incorporar conhecimento humano ao desenvolvimento do fluxo de trabalho de ML. Continuando, no capítulo 3, é apresentado o design do estudo. Os 16 participantes recrutados tinham

experiência prévia na aplicação de Auto-ML em capacidades profissionais. Em média, 10 anos de experiência em programação e uma média de 5 anos de experiência com ML. As entrevistas foram realizadas entre outubro de 2019 a março de 2020 e duraram aproximadamente 1 hora. O foco da entrevista foram os 3 componentes abaixo:

- Os participantes foram convidados a descrever seu papel profissional, organização e casos de uso de ML.
- Perguntamos aos participantes sobre sua experiência no desenvolvimento de fluxos de trabalho de ML (sem Auto-ML) e os desafios que enfrentaram.
- Perguntamos aos participantes sobre sua experiência no uso de ferramentas Auto-ML, incluindo os recursos da ferramenta específica usada e como eles integraram o Auto-ML em seus fluxos de trabalho de ML, e suas percepções sobre as ferramentas Auto-ML em geral, incluindo personalização, eficácia, interpretabilidade e transparência.

Todas as entrevistas, com exceção de 1, foram gravadas e transcritas para posterior análise de resultados. Posteriormente, no capítulo 4, o artigo traz os resultados do estudo. Inicialmente definindo os grupos de usuários (inovadores de ML, engenheiros de ML e novatos) e as categorias de uso (aplicação de produção, protótipo e pesquisa). Define também as tarefas de Pré-processamento de Dados, Modelagem e Pós-processamento. Em seguida, mostra, segundo o estudo, os benefícios do Auto-ML:

- Habilita e Empodera Novatos
- Padroniza o fluxo de trabalho ML para melhor reprodutibilidade, manutenção de código e compartilhamento de conhecimento
- Previne resultados subótimos devido a idiosincrasias dos inovadores de ML
- Constrói modelos de forma mais eficaz e eficiente
- Permite prototipagem rápida
- Promove Aprendizado

Após, mostra, segundo o estudo, as deficiências do Auto-ML:

- Falta de suporte abrangente de ponta a ponta
- Pré-processamento de dados limitado
- Suporte limitado para modelos e tipos de dados complexos

- Causa falhas no sistema devido a cargas de trabalho intensivas de computação
- Falta de Personalização
- Falta de Transparência e Interpretabilidade

Na sequência mostra como os profissionais impulsionam o desempenho e a eficiência do Auto-ML realizando operações em meio ao “ML automatizado”. No capítulo 5, denominado Discussões, os autores trazem possíveis melhorias as soluções de Auto-ML:

- Adaptar-se ao nível de proficiência do usuário pretendido
- Língua Franca para ML
- Plataforma Holística
- Computação Sem Servidor
- Interface de Usuário Adaptativa
- Exploração Interativa
- Equilíbrio entre Controle Humano e Automação

Na conclusão, os autores salientam que o estudo lança luz sobre várias formas de parceria ou colaboração entre humanos e ML/IA, dependendo das motivações, necessidades, habilidades e casos de uso dos usuários, e fornece aos desenvolvedores de ferramentas Auto-ML de próxima geração diretrizes de design para como melhor incorporar orientação pragmática e capacitar o engajamento eficaz com as ferramentas Auto-ML.

### **Qual o objetivo do estudo descrito pelo artigo?**

Entender o uso de Auto-ML pelos profissionais e possíveis pontos de melhoria para as soluções presentes no mercado.

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

Apesar se dizer “ML automatizado”, o uso do Auto-ML não cobre todas as etapas e particularidades de seu uso necessitando conhecimento dos profissionais que o utilizam. Desta forma, o estudo busca trazer a luz melhorias esperadas para as soluções de Auto-ML disponíveis no mercado.

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

Entrevistas com usuários e análise qualitativa dos dados.

**Quais os principais resultados obtidos pelo estudo?**

Foi possível avaliar os benefícios e as deficiências das ferramentas. Além de, qualificar 7 pontos de melhorias para as soluções:

- Adaptar-se ao nível de proficiência do usuário pretendido
- Língua Franca para ML
- Plataforma Holística
- Computação Sem Servidor
- Interface de Usuário Adaptativa
- Exploração Interativa
- Equilíbrio entre Controle Humano e Automação

## APÊNDICE 13 – FRAMEWORKS DE IA

### A - ENUNCIADO

#### 1 Classificação (RNA)

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

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

Informações:

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

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

#### 2 Regressão (RNA)

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

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

Informações:



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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
```

```
data = pd.read_csv(url, delimiter=';')
```

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

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

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

### 3 Sistemas de Recomendação

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

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

Informações:

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

#### 4 Deepdream

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

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

Informações:

- **Base de dados:** [https://commons.wikimedia.org/wiki/File:Felis\\_catus-cat\\_on\\_snow.jpg](https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg)
- **Importação da imagem:** Copiar código abaixo.

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

Dica: Para exibir a imagem utilizando `display` (`display.html`) use o link [https://commons.wikimedia.org/wiki/File:Felis\\_catus-cat\\_on\\_snow.jpg](https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg)

## B - RESOLUÇÃO

### 1 Classificação (RNA)

Importação das bibliotecas

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
```

Importação dos dados

```
fashion_mnist = tf.keras.datasets.fashion_mnist
(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
print("x_train.shape: ", x_train.shape)
print("y_train.shape: ", y_train.shape)
print("x_test.shape: ", y_test.shape)
print("y_test.shape: ", y_test.shape)
```

```
x_train.shape: (60000, 28, 28)
y_train.shape: (60000,)
x_test.shape: (10000,)
y_test.shape: (10000,)
```

Pré-processamento

```
x_train, x_test = x_train/255.0, x_test/255.0
```

Criação do modelo

```
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)

model = tf.keras.models.Model(i, x)
```

## Compilação e treinamento do modelo

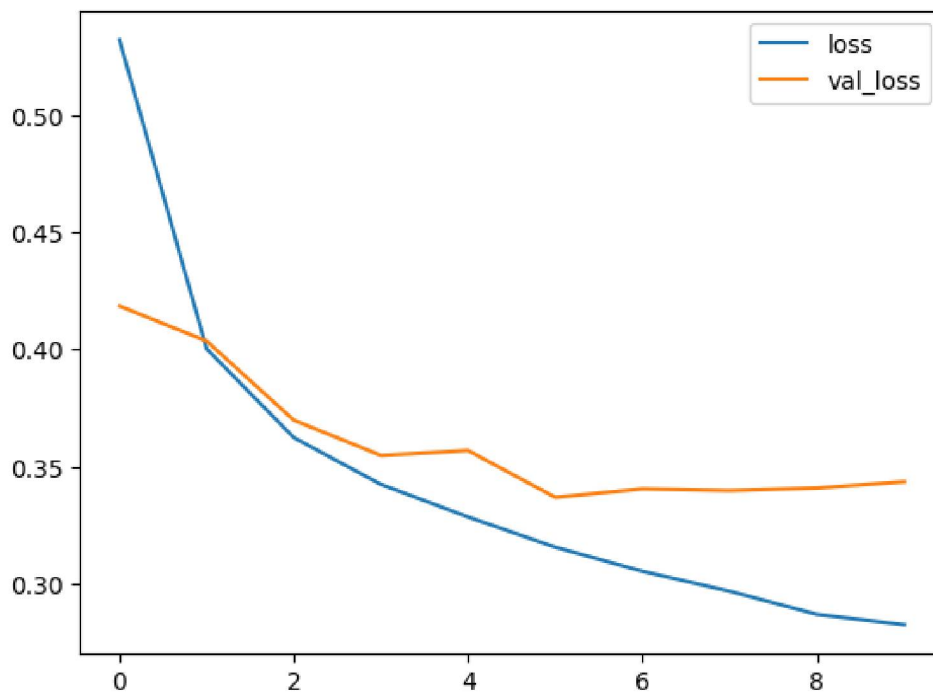
```
model.compile(optimizer='adam',
               loss='sparse_categorical_crossentropy',
               metrics=['accuracy'])

r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)
```

## Avaliação do modelo

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

FIGURA 36 – Função de perda - Fashion MNIST



Fonte: O autor (2025).

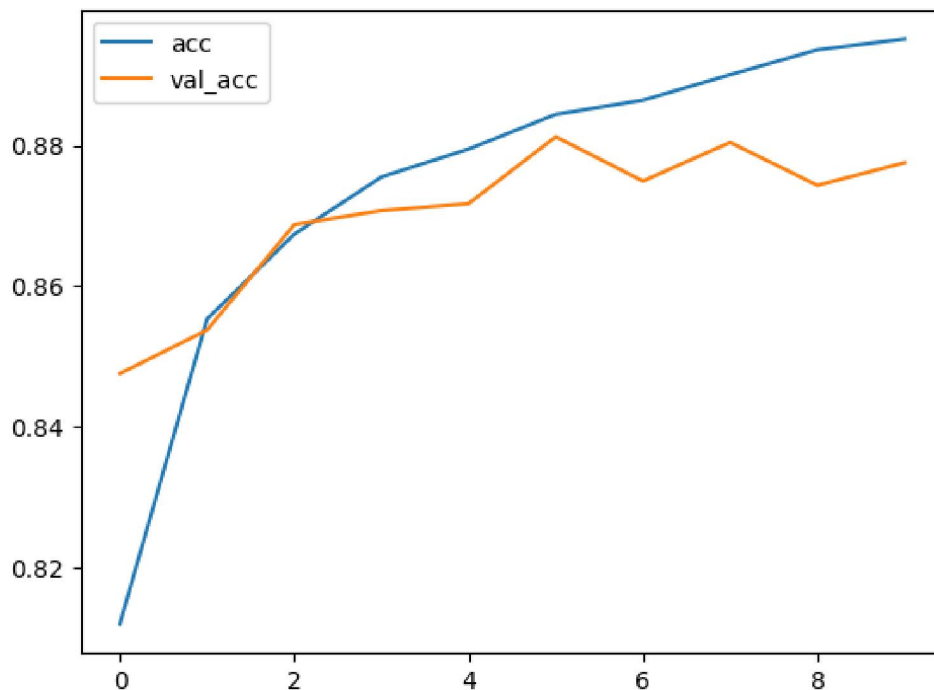
## Breve explicação gráfico de perda

A linha azul 'loss' representa a função de perda no conjunto de treinamento, ou seja, o quão ruins as previsões foram nesse conjunto em relação aos valores verdadeiros. Já a linha laranja 'val\_loss' mostra a perda no conjunto de validação. A

perda diminui gradualmente mostrando que a rede está convergindo para melhores resultados conforme vai sendo treinada.

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

FIGURA 37 – Acurácia - Fashion MNIST



Fonte: O autor (2025).

#### Breve explicação gráfico de acurácia

A linha azul 'acc' representa a acurácia no conjunto de treinamento, ou seja, quantas foram as previsões corretas durante o treinamento. Já a linha laranja 'val\_acc' representa a acurácia no conjunto de validação, mostrando como a rede generaliza para dados que não foram utilizados no treinamento. Ambas as acurácias aumentam gradativamente indicando que a rede está aprendendo e atinge um valor mais de 87% o que é uma boa acurácia.

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

```
accuracy: 0.8782 - loss: 0.3406
[0.3435542583465576, 0.8774999976158142]
```

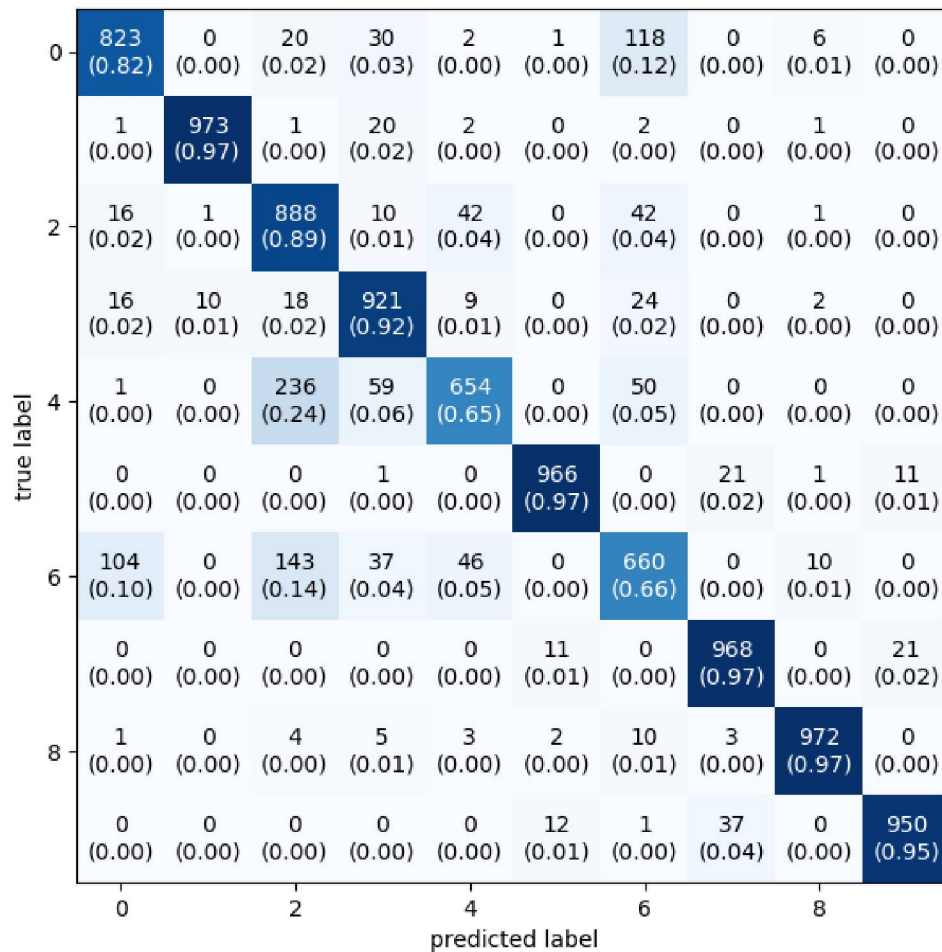


## Predições

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

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

FIGURA 38 – Matriz de confusão - Fashion MNIST



Fonte: O autor (2025).

## Mostrar algumas classificações erradas

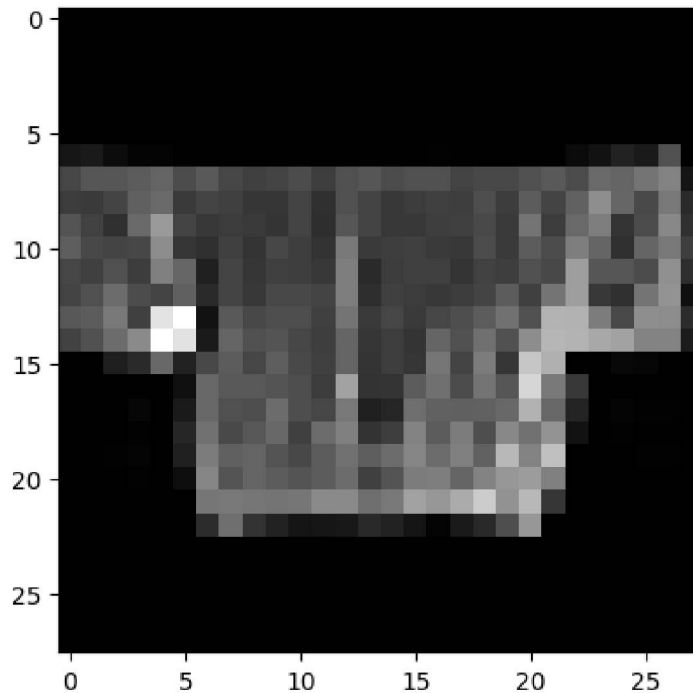
```
misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)

plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
)
```



FIGURA 39 – Classificação errada - Fashion MNIST  
True label: 6 Predicted: 8



Fonte: O autor (2025).

## 2 Regressão (RNA)

Importação das bibliotecas

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

from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from sklearn.utils.class_weight import compute_class_weight
from tensorflow.python.keras import backend
```

Importação dos dados

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

data.head()
```

| ID | Fixed acidity | Volatile acidity | Citric acid | Residual sugar | Chlorides |
|----|---------------|------------------|-------------|----------------|-----------|
| 0  | 7.4           | 0.70             | 0.00        | 1.9            | 0.076     |
| 1  | 7.8           | 0.88             | 0.00        | 2.6            | 0.098     |
| 2  | 7.8           | 0.76             | 0.04        | 2.3            | 0.092     |
| 3  | 11.2          | 0.28             | 0.56        | 1.9            | 0.075     |
| 4  | 7.4           | 0.70             | 0.00        | 1.9            | 0.076     |

| ID | Free SO <sub>2</sub> | Total SO <sub>2</sub> | Density | pH   | Sulphates | Alcohol | Quality |
|----|----------------------|-----------------------|---------|------|-----------|---------|---------|
| 0  | 11.0                 | 34.0                  | 0.9978  | 3.51 | 0.56      | 9.4     | 5       |
| 1  | 25.0                 | 67.0                  | 0.9968  | 3.20 | 0.68      | 9.8     | 5       |
| 2  | 15.0                 | 54.0                  | 0.9970  | 3.26 | 0.65      | 9.8     | 5       |
| 3  | 17.0                 | 60.0                  | 0.9980  | 3.16 | 0.58      | 9.8     | 6       |
| 4  | 11.0                 | 34.0                  | 0.9978  | 3.51 | 0.56      | 9.4     | 5       |

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

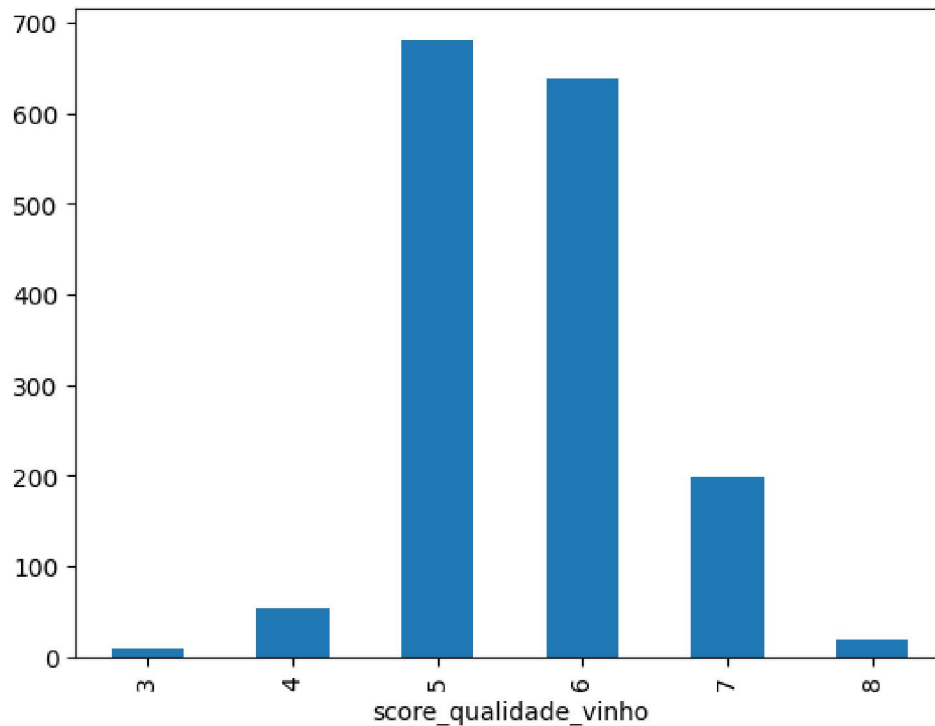
```
data.columns = [
    'acidez_fixa', # fixed acidity
    'acidez_volatil', # volatile acidity
    'acido_citrico', # citric acid
    'acucar_residual', # residual sugar
    'cloretos', # chlorides
    'dioxido_de_enxofre_livre', # free sulfur dioxide
    'dioxido_de_enxofre_total', # total sulfur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

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

```
X = data.iloc[:, :-1] # Todas as colunas exceto a última
Y = data.iloc[:, -1].astype(float) # Apenas a última coluna
```

```
# Verificar distribuição dos dados
data['score_qualidade_vinho'].value_counts().sort_index().plot(
    kind='bar')
plt.show()
```

FIGURA 40 – Distribuição dos dados - Wine quality



Fonte: O autor (2025).

```
# Melhorar distribuição dos dados

#Aplicar SMOTE (oversampling) ou RandomUnderSampler (
    undersampling)
#Tentativa 1
smote = SMOTE(random_state=42)
#Tentativa 2
#undersampler = RandomUnderSampler(random_state=42)

#Tentativa 1
X, Y = smote.fit_resample(X, Y)
#Tentativa 2
#x_train, y_train = undersampler.fit_resample(x_train, y_train)

# Verificar a distribuição das classes após o balanceamento
print(pd.Series(Y).value_counts())
```

```
score_qualidade_vinho
5.0 681
6.0 681
7.0 681
4.0 681
8.0 681
3.0 681
Name: count, dtype: int64
```

### Separação da base em treino e teste (80/20)

```
x_train, x_test, y_train, y_test = train_test_split(X, Y,
                                                    test_size=0.2)
```

### Pré-processamento dos dados

```
# Normalização dos dados de entrada
scaler = StandardScaler()
#scaler = MinMaxScaler()

# Ajustar o scaler aos dados de treinamento e transformar x_train
x_train = scaler.fit_transform(x_train)

# Transformar x_test
x_test = scaler.transform(x_test)
```

### Criação do modelo

```
# 3 camadas
i = tf.keras.layers.Input(shape=(11,))
#x = tf.keras.layers.Dense(50, activation='relu')(i)
x = tf.keras.layers.Dense(64, activation='relu')(i)
d = tf.keras.layers.Dropout(0.2)(x)
x1 = tf.keras.layers.Dense(32, activation='relu')(d)
d1 = tf.keras.layers.Dropout(0.2)(x1)
x2 = tf.keras.layers.Dense(1)(d1)

model = tf.keras.models.Model(i, x2)
```

## Compilação e treinamento do modelo

```
# Criação de funções para as métricas R2 e RMSE serem inseridas
no modelo

def rmse(y_true, y_pred):
    # Converter y_true para float32 para garantir compatibilidade
    y_true = backend.cast(y_true, dtype='float32')

    return backend.sqrt(backend.mean( backend.square(y_pred -
        y_true), axis=-1) )

def r2(y_true, y_pred):
    # Converter y_true para float32 para garantir compatibilidade
    y_true = backend.cast(y_true, dtype='float32')

    media = backend.mean(y_true)
    num    = backend.sum (backend.square(y_true - y_pred))
    den    = backend.sum (backend.square(y_true - media))
    return (1.0 - num/den)

# Compilação
#optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
#optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
optimizer=tf.keras.optimizers.SGD(learning_rate=0.01, momentum
    =0.9)
#optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum
    =0.5)
#optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
               loss="mse",
               metrics=[rmse, r2])

# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                monitor='val_loss',
                patience=50,
                restore_best_weights=True)

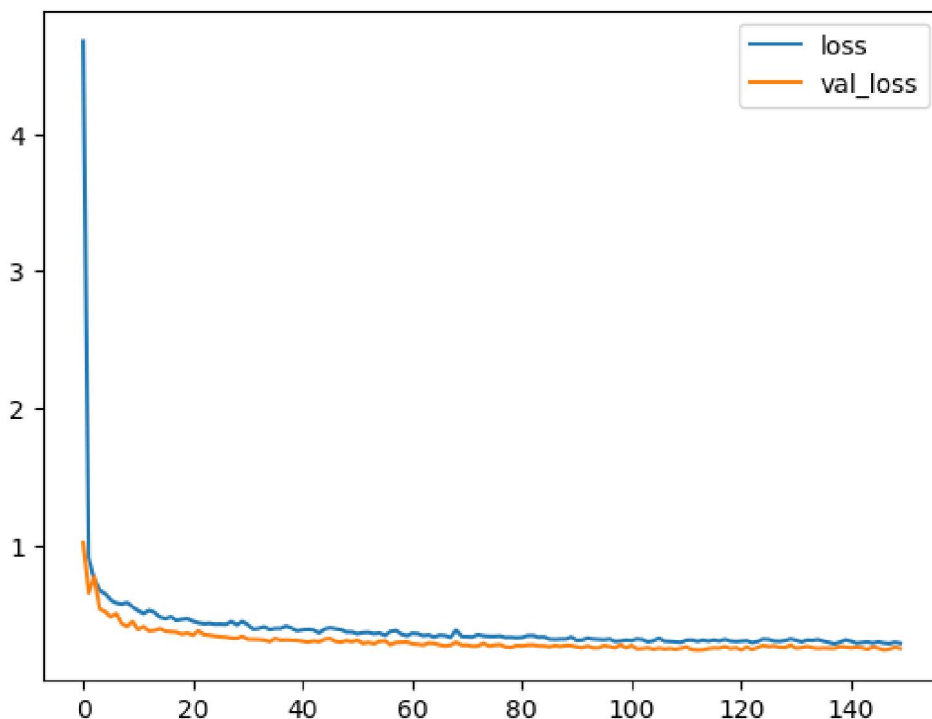
#Treinar o modelo com ou sem pesos de classe
#r = model.fit(x_train, y_train, epochs=100, batch_size=32,
    validation_data=(x_test, y_test), callbacks=[early_stop])
```

```
r = model.fit(x_train, y_train, batch_size=64, epochs=150,
              validation_data=(x_test, y_test), callbacks=[early_stop])
```

### Avaliação do modelo

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

FIGURA 41 – Função de perda - Wine quality



Fonte: O autor (2025).

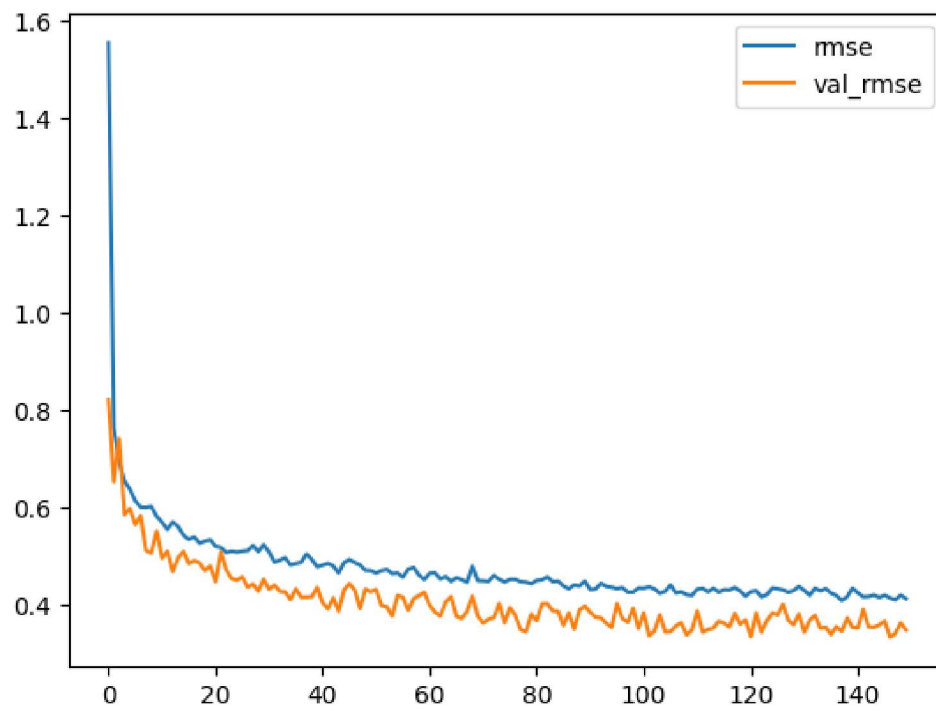
### Breve explicação gráfico loss

A linha azul 'loss' representa a função de perda no conjunto de treinamento, ou seja, o quão ruim as previsões foram nesse conjunto em relação aos valores verdadeiros. Já a linha laranja 'val\_loss' mostra a perda no conjunto de validação. A rede convergiu rapidamente para um baixo valor de perda, e depois se mantém quase constante com uma mínima variação, sugerindo que talvez possa ser diminuído o número de épocas no treinamento.

```
plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()
```



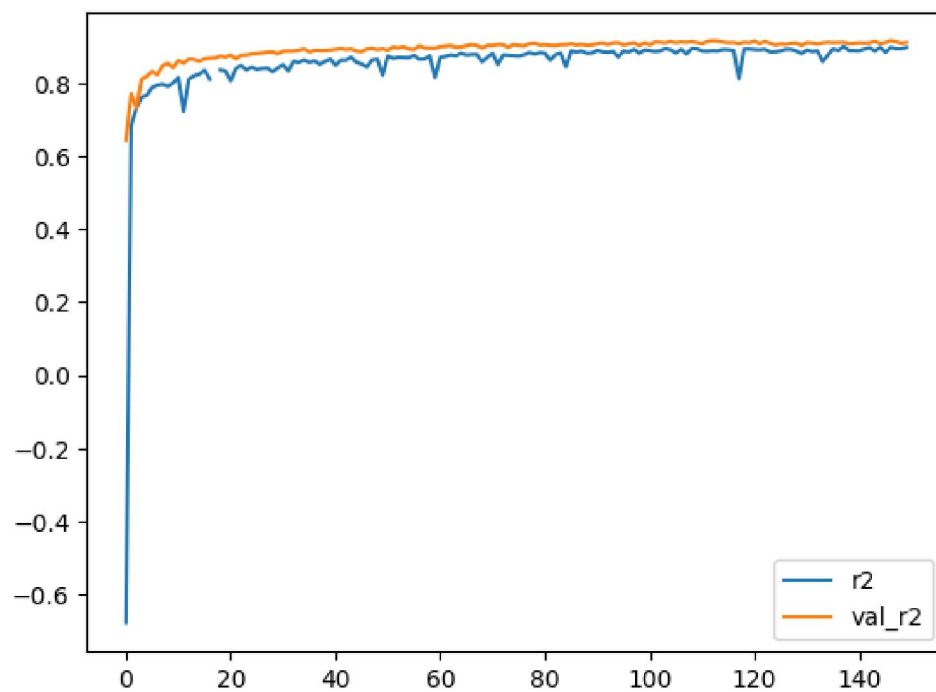
FIGURA 42 – RMSE - Wine quality



Fonte: O autor (2025).

```
plt.plot( r.history["r2"], label="r2" )  
plt.plot( r.history["val_r2"], label="val_r2" )  
plt.legend()
```

FIGURA 43 – R2 - Wine quality



Fonte: O autor (2025).

## Predições

```
# Predição
y_pred = model.predict(x_test).flatten()

print(y_test[:10])
print(y_pred[:10])
```

```
837 7.0
1303 5.0
2092 3.0
3659 8.0
2747 4.0
723 5.0
3230 7.0
1164 5.0
808 5.0
3133 7.0
Name: score_qualidade_vinho, dtype: float64
6.3655353 6.268461 2.9856465 7.419491 4.506182 4.958458 5.962226
4.9503975 5.1298738 6.320078
```

```
# Cálculo das métricas de acurácia: mse, r2 e rmse
mse = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)

# Resultados das métricas de acurácia
print("mse      = ", mse)
print("rmse     = ", rmse)
print("r2       = ", r2)
```

```
mse = 0.23897800376884185
rmse = 0.4888537652190498
r2 = 0.9187425337397466
```

## Breve explicação das métricas de avaliação

- O MSE de  $\approx 0.2389$  sugere que a média dos erros quadráticos não é muito alta, o que implica que o modelo fez previsões razoavelmente boas.

- O RMSE de  $\approx 0.489$  indica que o modelo está errando pouco.
- Um  $R^2$  de  $\approx 0.9187$  significa que aproximadamente 91.87% das qualidades dos vinhos é explicada corretamente pelo modelo. Isso é um excelente desempenho para uma regressão, indicando que a RNA tem uma boa capacidade preditiva.

### 3 Sistemas de Recomendação

#### Importação das bibliotecas

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense, Embedding,
    Flatten, Concatenate, Dropout, Layer
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.utils import shuffle

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

#### Importação dos dados

```
import pandas as pd

rows = []

# Abrindo o arquivo original
with open("/content/Base_livros.csv", "r") as file:
    next(file) # Pula a primeira linha
    for line in file:
        isbn, rest = line.split(",", 1)

        rest_parts = rest.rsplit(",", 5)

        if len(rest_parts) == 6:
            titulo, autor, ano, editora, id_usuario, notas =
rest_parts
            rows.append({
                "ISBN": isbn.strip(),
```

```

        "Titulo": titulo.strip(),
        "Autor": autor.strip(),
        "Ano": ano.strip(),
        "Editora": editora.strip(),
        "ID_usuario": id_usuario.strip(),
        "Notas": notas.strip()
    })

# Limpeza geral
df['ISBN'] = df['ISBN'].str.replace(r'^"|"$', '', regex=True) #
    Remove aspas extras no início e fim
df['Titulo'] = df['Titulo'].str.replace(r'^"|"$', '', regex=
    True) # Remove aspas ao redor do título
df['Notas'] = df['Notas'].str.replace(r';+', '', regex=True) #
    Remove os ';' extras nas notas
df['Notas'] = df['Notas'].str.replace(r'^"|"$', '', regex=True)
    # Remove aspas ao redor

#VERIFICAR VALORES NAN
has_nan = df.isnull().values.any()

```

## Conversão de userId e movieId para categoria

```

# ID_usuario e ISBN não estão no formato certo para usar
# Embeddings > devem ser categóricos

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

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

#DATA AUGMENTATION

def stratified_sampling(df, n_samples=5):
    augmented_samples = []

    # Para cada usuário, pegar os livros com as diferentes notas
    for user in df['new_user_id'].unique():
        user_books = df[df['new_user_id'] == user]
        all_ratings = user_books['Notas'].unique()

```

```

        for rating in all_ratings:
            # Pegar livros com essa nota
            sampled_books = user_books[user_books['Notas'] ==
rating].sample(n=n_samples, replace=True)
            for _, row in sampled_books.iterrows():
                augmented_samples.append([row['new_user_id'], row
['new_book_id'], row['Notas']])

        # Criar DataFrame com as interações aumentadas
        augmented_df = pd.DataFrame(augmented_samples, columns=['
new_user_id', 'new_book_id', 'Notas'])
        return augmented_df

# Aplicar amostragem estratificada
augmented_df = stratified_sampling(df)
# PARA DESATIVAR DATA AUGMENTATION
#augmented_df = df

# Dimensões
N = len(set(augmented_df.new_user_id))
M = len(set(augmented_df.new_book_id))

# dimensão do embedding (tentar outros)
K = 10

```

## Criar o modelo

```

# usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# livro
m = Input(shape=(1,))
m_emb = Embedding(M, K)(m) # saída : num_samples, 1, K
m_emb = Flatten()(m_emb)   # saída : num_samples, K

x = Concatenate()([u_emb, m_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

```

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

## Compilação do modelo

```
model.compile(
    loss="mse",
    #optimizer=SGD(learning_rate=0.08, momentum=0.9)
    optimizer=Adam(learning_rate=0.001)
)
```

## Separação dos dados e pré-processamento

```
from sklearn.preprocessing import StandardScaler

#Padronização das notas, ao descomentar aqui comentar a
#centralização abaixo
scaler = StandardScaler()# Ajuste conforme o nome correto da
#coluna
augmented_df['Notas'] = scaler.fit_transform(augmented_df[['Notas',
#]])

# Tentar converter para numérico, substituindo não numéricos por
#NaN
augmented_df['Notas'] = pd.to_numeric(augmented_df['Notas'],
#errors='coerce')

user_ids, book_ids, ratings = shuffle(augmented_df.new_user_id,
#augmented_df.new_book_id, augmented_df.Notas)

Ntrain = int(0.75 * len(ratings)) # separar os dados 75% x 25%

train_user = user_ids[:Ntrain]
train_book = book_ids[:Ntrain]
train_ratings = ratings[:Ntrain]

test_user = user_ids[Ntrain:]
test_book = book_ids[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
#train_ratings = train_ratings - avg_rating
```



```
#test_ratings = test_ratings - avg_rating

linhas_na = df[df['Notas'].isnull()]
```

## Treinamento do modelo

```
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True
)

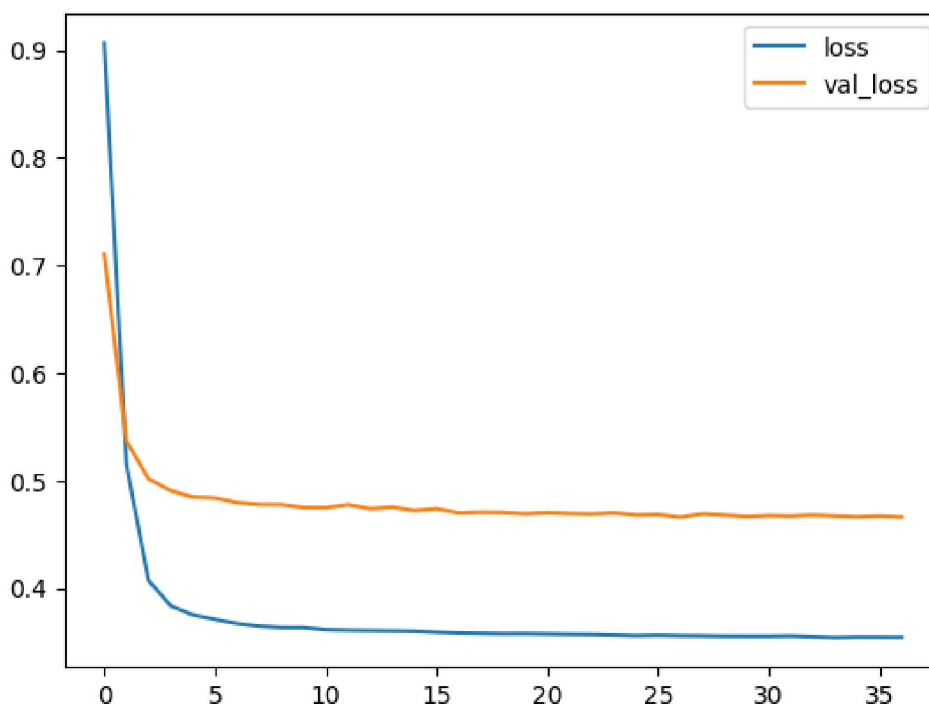
epochs = 50

r = model.fit(
    x=[train_user, train_book],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=(test_user, test_book), test_ratings),
    callbacks=[early_stopping]
)
```

## Plotar a função de perda

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

FIGURA 44 – Função de perda - Base livros



Fonte: O autor (2025).

#### Breve explicação gráfico de perda

A linha azul 'loss' representa a função de perda no conjunto de treinamento, ou seja, o quão ruins as previsões foram nesse conjunto em relação aos valores verdadeiros. Já a linha laranja 'val\_loss' mostra a perda no conjunto de validação. O modelo começa com uma perda alta e, aparentemente, aprende rapidamente no início, o que é indicado pela queda acentuada no início da curva azul. Após um certo ponto, a curva de validação (laranja) começa a se distanciar da curva de treinamento (azul), o que pode ser um sinal de overfitting. Ou seja, o modelo está começando a se ajustar excessivamente aos dados de treinamento, mas não generaliza bem para os dados de validação, resultando em uma perda crescente no conjunto de validação.

#### Recomendações para o usuário 7386

```
# Gerar o array com o usuário único
# repete a quantidade de livros
input_usuario = np.repeat(a=7386, repeats=M)
book = np.array(list(set(book_ids)))

preds = model.predict( [input_usuario, book] )

# descentraliza as previsões
#rat = preds.flatten() + avg_rating
```

```
# Despadroniza as predições
scaler = StandardScaler()
scaler.fit(augmented_df[['Notas']])
mean_rating = scaler.mean_
std_rating = scaler.scale_
rat = preds.flatten() * std_rating + mean_rating

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: ISBN Livro - ", book[idx], " / ", rat[idx] ,
      "*")
```

Recomendação: ISBN Livro - 5853 / 2.1411452293395996 \*

Breve explicação recomendação usuário 7386

O modelo, para o usuário com ID 7386, recomenda o livro com ISBN e a nota associada a ele. No exemplo fornecido, a recomendação foi para o livro com ISBN 5853, e a nota prevista para o livro foi  $\approx 2.141$ .

## 4 Deepdream

Esta prática contém uma implementação mínima do DeepDream, conforme descrito neste post, do blog de Alexander Mordvintsev.

Vamos demonstrar como fazer uma rede neural "sonhar" e aprimorar os padrões surreais que ela vê em uma imagem.

FIGURA 45 – Prática Deepdream



Fonte: O autor (2025).

### Importação das bibliotecas

```
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image
```

### Importação da imagem

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

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



```

        img.thumbnail((max_dim, max_dim))
    return np.array(img)

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

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

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

```

FIGURA 46 – Imagem escolhida - Deepdream



Fonte: O autor (2025).

Preparar o modelo de extração de recursos

Aqui faremos o download um modelo de classificação de imagem pré-treinado. Usaremos o InceptionV3 que é semelhante ao modelo originalmente usado no Deep-

Dream. Observe que qualquer modelo pré-treinado funcionará, embora você precise ajustar os nomes das camadas abaixo, caso deseje alterá-lo.

```
base_model = tf.keras.applications.InceptionV3(include_top=False,
        weights='imagenet')
```

A ideia no DeepDream é escolher uma camada (ou camadas) e maximizar a "perda" de forma que a imagem cada vez mais treine as camadas.

```
# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

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

Cálculo da perda (loss)

A perda é a soma das ativações nas camadas escolhidas.

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

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

    return tf.reduce_sum(losses)
```

Subida de gradiente (Gradient ascent)

Depois de ter calculado a perda para as camadas escolhidas, tudo o que resta é calcular os gradientes em relação à imagem e adicioná-los à imagem original.

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



```

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

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

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

            # Normalizacao dos gradintes
            gradients /= tf.math.reduce_std(gradients) + 1e-8

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

    return loss, img

deepdream = DeepDream(dream_model)

```

### Circuito principal (Main Loop)

```

def run_deep_dream_simple(img, steps=100, step_size=0.01):

    img = tf.keras.applications.inception_v3.preprocess_input(img)
    img = tf.convert_to_tensor(img)
    step_size = tf.convert_to_tensor(step_size)
    steps_remaining = steps

```

```

step = 0
while steps_remaining:
    if steps_remaining>100:
        run_steps = tf.constant(100)
    else:
        run_steps = tf.constant(steps_remaining)
    steps_remaining -= run_steps
    step += run_steps

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

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

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

return result

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

```

FIGURA 47 – Imagem escolhida - Ciclo principal - Deepdream



Fonte: O autor (2025).

Explicação: Imagem gerada durante o ciclo principal do algoritmo DeepDream, que aplica convoluções repetidas vezes para amplificar padrões reconhecidos pela rede neural, em áreas específicas da imagem, gerando texturas e formas psicodélicas, parecidas de um "sonho". Aqui ainda no Main Loop, já é observado um efeito inicial onde partes do gato mostram características "oníricas", mas ainda mantendo a estrutura básica do animal e do fundo da imagem.

Levando o modelo até um oitava

Conseguimos gerar uma imagem. Porém, há alguns problemas com esta primeira tentativa:

1. A saída é ruidosa (isso pode ser resolvido com uma perda `tf.image.total_variation`).
2. A imagem é de baixa resolução.
3. Os padrões parecem estar acontecendo na mesma granularidade.

```
import time
start = time.time()

OCTAVE_SCALE = 1.30

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

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

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

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

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```



FIGURA 48 – Imagem escolhida - Elevado oitava - Deepdream



Fonte: O autor (2025).

Explicação: Aqui onde o modelo é elevado uma oitava a imagem gerada mostra uma intensificação maior dos detalhes oníricos. A imagem do gato começa a perder destaque para o fundo que ganha o destaque com maior preenchimento de padrões e texturas.

Elevar o modelo a uma oitava resolve os problemas de ruído, resolução, e granularidade, através da aplicação da subida de gradiente em uma escala diferente, permitindo que a imagem seja preenchida com detalhes adicionais.

Diferenças entre as imagens do Main Loop e ao elevar à oitava

Nível de detalhe: A imagem após subir uma oitava é mais rica em detalhes, com padrões mais amplificados e complexos em comparação à imagem gerada no Main loop.

Alteração no fundo: No Main Loop, o fundo da imagem ainda mantém maior simplicidade. Após a oitava, ele é preenchido com texturas e formas oníricas adicionais.

Intensidade dos padrões: Na segunda imagem, os padrões criados pela rede neural são mais visíveis e se repetem com maior frequência em todo o corpo do gato.

## APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

### A - ENUNCIADO

Escolha um conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada) e faça uma visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha)

Desenvolva uma narrativa/storytelling para essa visualização de dados considerando os conceitos e informações que foram discutidas nesta disciplina. Não esqueça de deixar claro para seu possível público alvo qual **o objetivo dessa visualização de dados, o que esses dados significam, quais possíveis ações podem ser feitas com base neles.**

**Entregue em um PDF:**

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

### B - RESOLUÇÃO

#### Conjunto de Dados

O conjunto de dados escolhido foi “Crime\_Data\_from\_2020\_to\_Present.csv”, disponível em <https://www.kaggle.com/datasets/samithsachidanandan/crime-data-from-2020-to-present/data>, contendo dados sobre os crimes na cidade de Los Angeles no período entre 2020 e 2024. Para a utilização no relatório, foi criada uma rotina de processamento de dados (“eda-processamento.ipynb”) que filtra somente os crimes de 2023 e substitui valores “nan” para melhor apresentação no relatório. A base preparada (dados\_crimes\_LA\_2023.csv) foi copiada para o Google Drive para melhor utilização com o Looker Studio, devido a integração nativa da ferramenta.

#### Contexto e Público-alvo

O objetivo é trazer um panorama sobre a violência na cidade de Los Angeles para que visitantes e moradores diminuam os riscos de crimes acontecerem. Desta forma, o

relatório deve ser interativo com a finalidade do usuário poder, além de navegar pelas informações, selecionar filtragens específicas.

## **Visualização dos Dados**

Para a visualização dos dados foi escolhida a ferramenta Looker Studio e criado um dashboard com 5 páginas. Esta ferramenta permite a interação entre o usuário e as informações de forma simples e intuitiva, facilitando o objetivo desta visualização. O dashboard está disponível no link: <https://lookerstudio.google.com/reporting/c13ed9a8-61ae-452c-91df-21a9b6d064e8>

## **Descrição do Storytelling**

O dashboard, após a capa, inicia apresentando uma visão baseada nos bairros da cidade, página Crimes por Bairro. Como trata-se de um relatório para visitantes e moradores, responder à pergunta “onde é mais seguro” se torna uma premissa básica. Nesta página, é apresentado os 10 bairros com maior número de crimes e os 10 bairros com menor número de crimes. Além da lista completa de bairros por ordem alfabética e com o totalizador de crimes. Com os gráficos, fica claro afirmar que o bairro com maior número de crimes é o Central e o com menor número de crimes é o Foothill. A próxima página, Tipo do Crime X Status das Investigações, traz a correlação entre os tipos de crimes e qual o status das investigações. Nele, ao clicar em qualquer um dos gráficos, o outro é filtrado. Desta forma, a indicação é clicar no Tipo de Crime e acompanhar os Status das Investigações logo abaixo. Utilizando desta forma, podemos verificar que 95Na página Arma do Crime X Dados das Vítimas, podemos relacionar a arma utilizada nos crimes com dados das vítimas. Da mesma forma que a página anterior, ao selecionar a arma do crime, os dados da vítima são filtrados. Podemos ver, por exemplo, que o crime de ameaça verbal (VERBAL THREAT) tem como vítimas predominantemente mulheres (48,1E, por fim, a página Guia de Destino reúne todas as informações visualizadas no relatório, adicionando uma caixa de seleção suspensa com todos os bairros para que o usuário entenda os crimes no destino escolhido. Como exemplo, ao selecionar Hollywood, é possível verificar que há mais crimes contra mulheres e pessoas brancas, majoritariamente assaltos sem armas.



## APÊNDICE 15 – TÓPICOS EM IA

### A - ENUNCIADO

#### 1 Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

- (1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
- (2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
- (3) Utilize no mínimo uma população com 100 indivíduos;
- (4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
- (5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
- (6) Preserve sempre a melhor solução de uma geração para outra.

**Importante:** A solução deverá implementar os operadores de “cruzamento” e “mutação”.

## 2 Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

## B - RESOLUÇÃO

### 1 Algoritmo Genético

#### Importações

```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations
```

#### Configurações

```
NUM_POPULACAO = 100
GERACOES = 1000
TAXA_MUTACAO = 0.01
TAXA_CRUZAMENTO = 0.9
```

#### Gerar cidades aleatórias

```
cidades = np.random.rand(NUM_POPULACAO, 2) * 100
def populacaoInicial(tamanho):
    resultado = []
    for _ in range(tamanho):
        individuo = list(np.random.permutation(NUM_POPULACAO))
        resultado.append(individuo)
    return resultado
```

#### Desenha solução encontrada

```

def printaSolucao(caminho, titulo):
    # cidadesOrdenadas = np.append(caminho, caminho[0])
    # plt.figure(figsize=(10, 6))
    # plt.scatter(cidades[:, 0], cidades[:, 1], c='red')
    # plt.plot(cidades[cidadesOrdenadas, 0], cidades[cidadesOrdenadas
    #         , 1], 'b-')
    # plt.title(titulo)
    # plt.show()
def printaSolucao(caminho, titulo):
    cidadesOrdenadas = caminho + [caminho[0]]
    plt.figure(figsize=(12, 8))
    plt.scatter([c[0] for c in cidades], [c[1] for c in cidades],
        c='red', s=50, label="Cidades")
    plt.plot([cidades[i % NUM_POPULACAO][0] for i in
    cidadesOrdenadas], [cidades[i % NUM_POPULACAO][1] for i in
    cidadesOrdenadas], 'b-', alpha=0.6, linewidth=1.5)
    for i, (x, y) in enumerate(cidades):
        plt.text(x, y, str(i), fontsize=9, ha='right', color='
black')
    plt.xlim(-10, 110)
    plt.ylim(-10, 110)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.title(titulo)
    plt.legend()
    plt.show()

```

## Calculos distâncias

```

def distanciaEuclidiana(a, b):
    return np.linalg.norm(a - b)
def distanciaEuclidiana(a, b):
    # return np.sqrt((a[0] - b[0])**2 + (a[1] - b[1])**2)
def distanciaTotal(caminho):
    distancia = 0
    for i in range(len(caminho) - 1):
        distancia += distanciaEuclidiana(cidades[caminho[i]],
        cidades[caminho[i+1]])
    distancia += distanciaEuclidiana(cidades[caminho[-1]],
    cidades[caminho[0]])
    return distancia

```

## implementa cruzamento

```
def cruzamento(pai, mae):
    inicio, fim = sorted(random.sample(range(NUM_POPULACAO), 2))
    filho = [-1] * NUM_POPULACAO
    filho[inicio:fim] = pai[inicio:fim]
    ptr = fim
    for gene in mae:
        if gene not in filho:
            while filho[ptr % NUM_POPULACAO] != -1:
                ptr += 1
            filho[ptr % NUM_POPULACAO] = gene
    return np.array(filho)
```

### Implementa mutação

```
def mutacao(individuo):
    if random.random() < TAXA_MUTACAO:
        i, j = random.sample(range(NUM_POPULACAO), 2)
        individuo[i], individuo[j] = individuo[j], individuo[i]
    return individuo
```

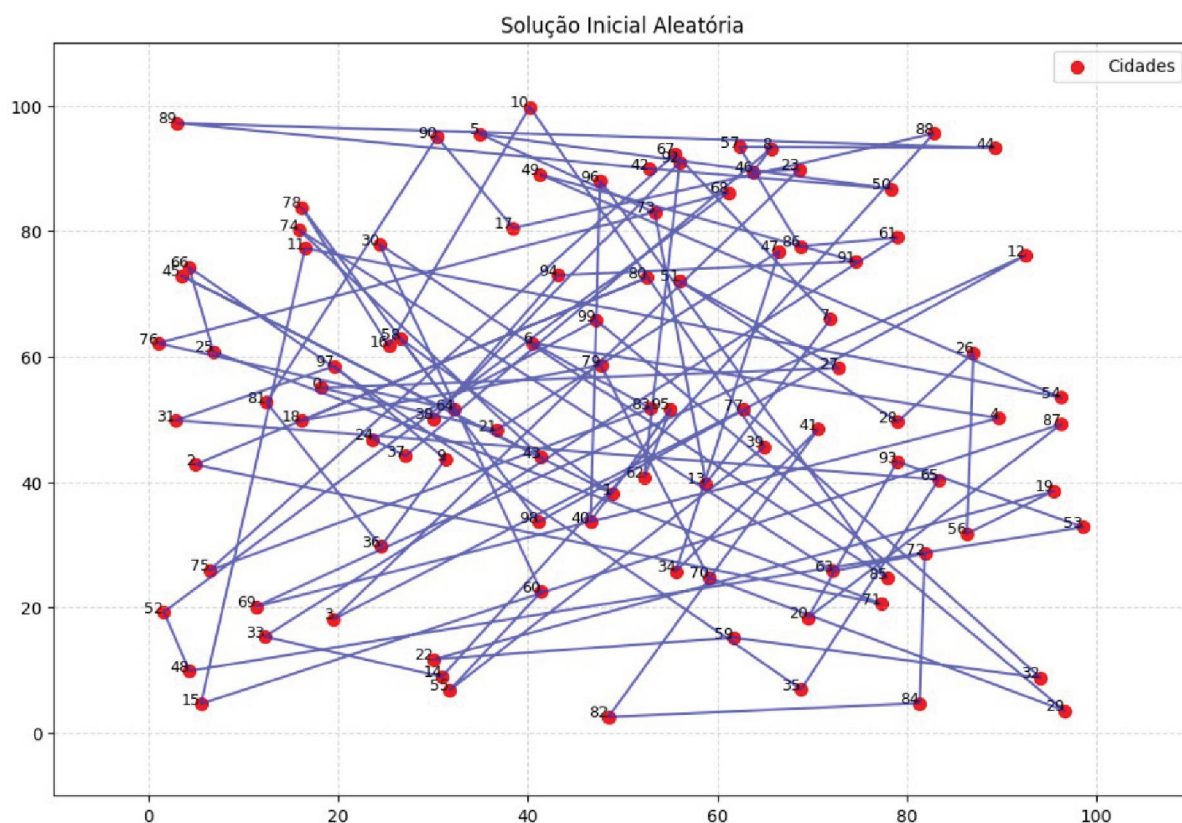
### Implementa seleção

```
def selecao(populacao, fitness):
    total = sum(fitness)
    probabilidades = [a / total for a in fitness]
    idx1, idx2 = random.choices(range(len(populacao)), weights=
    probabilidades, k=2)
    return populacao[idx1], populacao[idx2]
```

### Inicialização

```
populacao = populacaoInicial(NUM_POPULACAO)
melhorSolucao = min(populacao, key=distanciaTotal)
melhorDistancia = distanciaTotal(melhorSolucao)
printaSolucao(melhorSolucao, "Solução Inicial Aleatória")
```

FIGURA 49 – Solução Inicial Aleatória



Fonte: O autor (2025).

### Algoritmo Genético

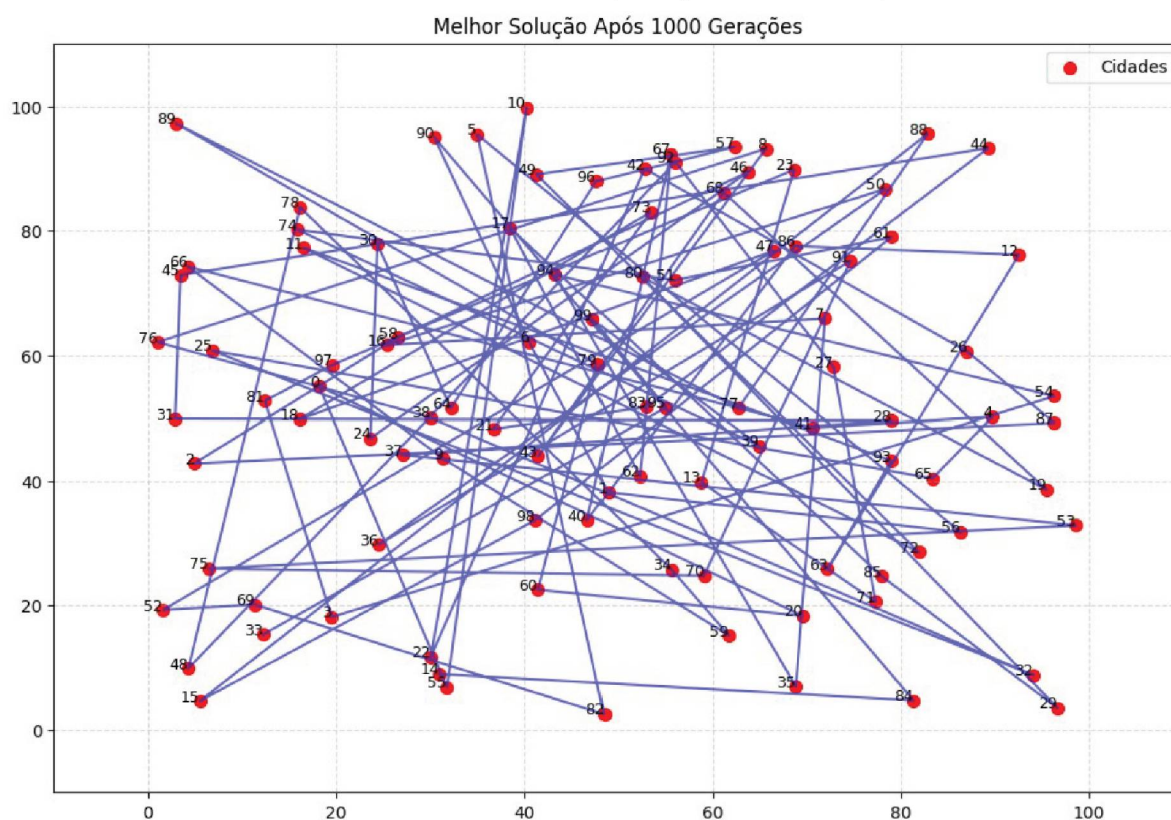
```
for geracao in range(GERACOES):
    fitness = np.array([1 / distanciaTotal(ind) for ind in
    populacao])
    novaPopulacao = [melhorSolucao]
    while len(novaPopulacao) < NUM_POPULACAO:
        if random.random() < TAXA_CRUZAMENTO:
            p1, p2 = selecao(populacao, fitness)
            filho = cruzamento(p1, p2)
        else:
            filho = random.choice(populacao)
            novaPopulacao.append(mutacao(filho))
    populacao = novaPopulacao
    melhorAtual = min(populacao, key=distanciaTotal)
    distanciaAtual = distanciaTotal(melhorAtual)
    if distanciaAtual < melhorDistancia:
        melhorSolucao, melhorDistancia = melhorAtual,
        distanciaAtual
```

Exibe a melhor solução encontrada



```
printaSolucao(melhorSolucao, "Melhor Solução Após 1000 Gerações")
```

FIGURA 50 – Melhor Solução Após 1000 Gerações



Fonte: O autor (2025).

## 2 Compare a representação de dois modelos vetoriais

### Importações

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
```

### Trechos de texto

```
textos = [
    "Um mago nunca se atrasa, nem se adianta. Ele chega  
exatamente quando pretende chegar, pois o tempo dos magos não  
é o mesmo dos homens. Quando ele chega, é o momento certo.",
    "Mesmo a menor das pessoas pode mudar o curso do futuro. O  
que importa não é o tamanho ou a força, mas a coragem que se  
tem de seguir um caminho que poucos ousariam.",
```



"Tudo o que temos de decidir é o que fazer com o tempo que nos é dado. O que está em nossas mãos, ao fim das contas, é como escolhemos usar os momentos que temos para afetar o que está à nossa volta.",

"A jornada não termina aqui. A morte é apenas outro caminho, que todos temos que trilhar. Cada um de nós, por mais que deseje evitar, um dia terá que enfrentar o fim, mas o que fazemos enquanto vivemos é o que realmente importa.",

"O mundo não está em seus livros e mapas. Ele está lá fora, nas montanhas, nas florestas e nas planícies. Não se limita ao que você pode ver com os olhos, mas ao que você está disposto a explorar e descobrir.",

"Os dias passam como folhas ao vento, mas nossas escolhas permanecem. O que fizemos e as decisões que tomamos, seja na alegria ou na dor, ecoam por todo o nosso caminho e deixam uma marca no que está por vir."

]

## Vetorização TF-IDF

```
vetorizador = TfidfVectorizer()
X = vetorizador.fit_transform(textos).toarray()
```

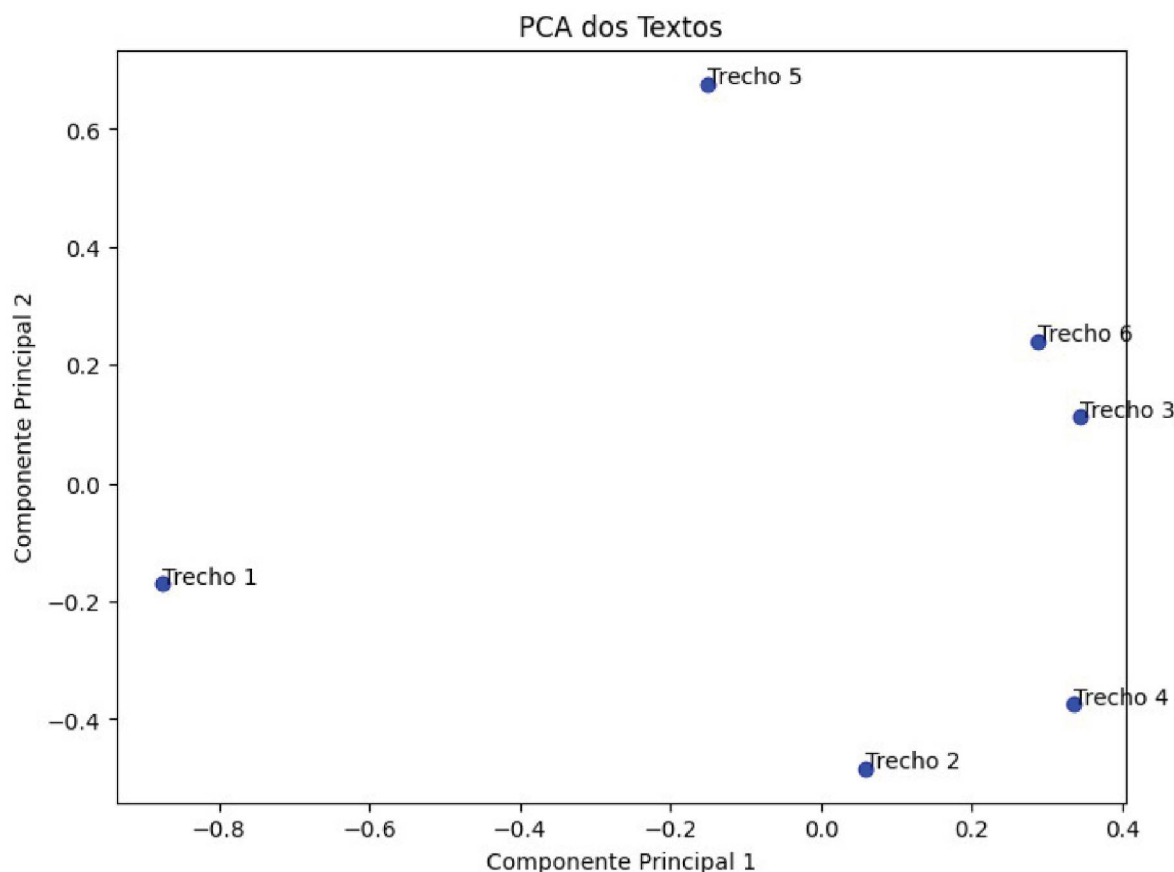
## Redução de dimensionalidade para visualização

```
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)
```

## Plotagem dos vetores

```
plt.figure(figsize=(8, 6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], color='blue')
for i, txt in enumerate(textos):
    plt.annotate(f"Trecho {i+1}", (X_pca[i, 0], X_pca[i, 1]))
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.title("PCA dos Textos")
plt.show()
```

FIGURA 51 – PCA dos Textos



Fonte: O autor (2025).

No gráfico, os trechos 3 ("Tudo o que temos de decidir é o que fazer com o tempo que nos é dado...") e 6 ("Os dias passam como folhas ao vento, mas nossas escolhas permanecem...") ficaram bem próximos, o que faz sentido visto que tratam da mesma ideia sobre a importância das escolhas.

Já os trechos 2 ("Mesmo a menor das pessoas pode mudar o curso do futuro...") e 4 ("A jornada não termina aqui. A morte é apenas outro caminho, que todos temos que trilhar...") estão ligeiramente próximos, e isso também faz sentido. Da para entender que ambos os trechos tem um toque de esperança.

Essas similaridades revelam como a representação vetorial consegue captar certos significados em trechos de textos.