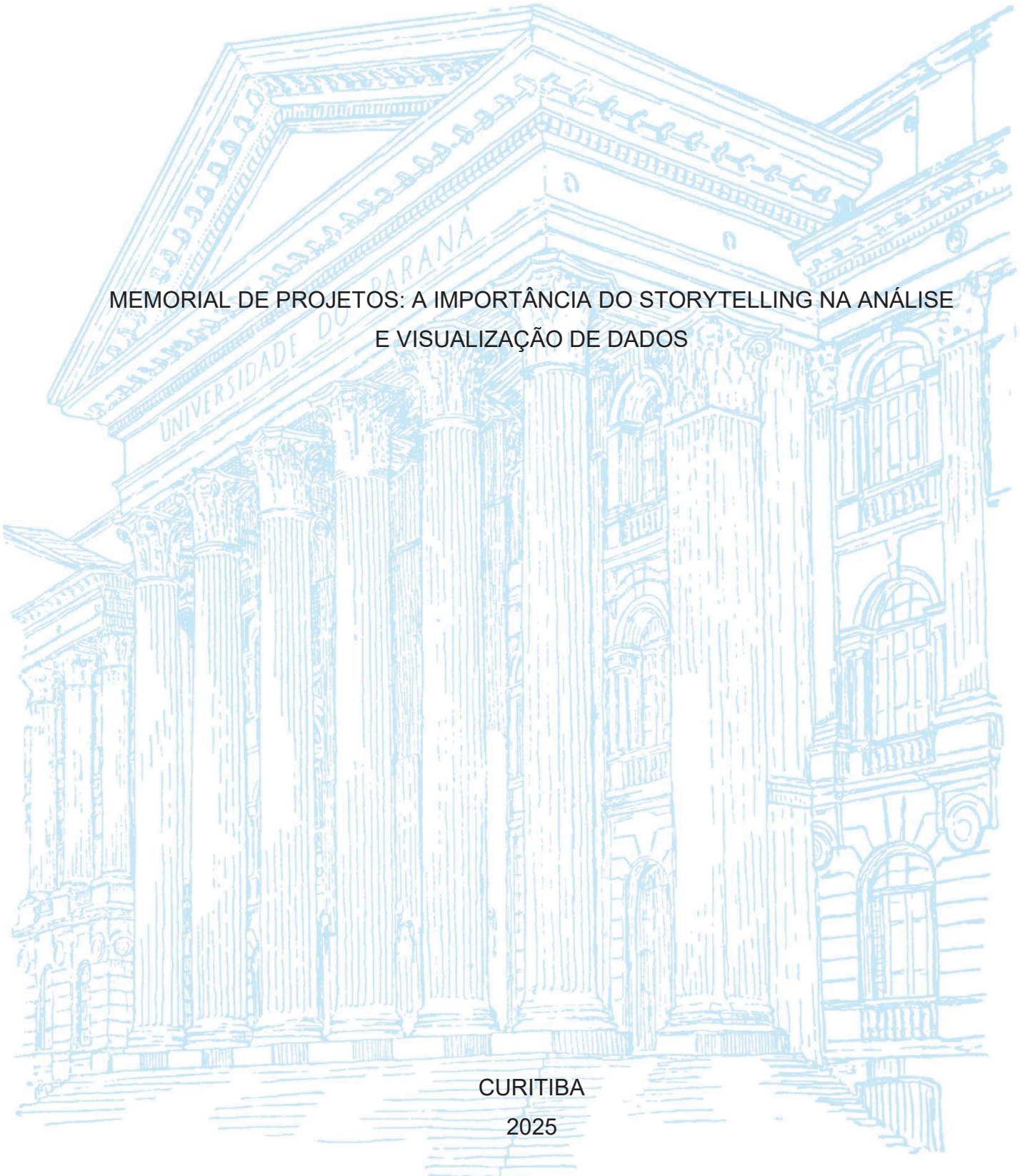
UNIVERSIDADE FEDERAL DO PARANÁ

LÍVIA MARIA DA SILVA FERNANDES



MEMORIAL DE PROJETOS: A IMPORTÂNCIA DO STORYTELLING NA ANÁLISE E VISUALIZAÇÃO DE DADOS

CURITIBA

2025

LÍVIA MARIA DA SILVA FERNANDES

MEMORIAL DE PROJETOS: A IMPORTÂNCIA DO STORYTELLING NA ANÁLISE E VISUALIZAÇÃO DE DADOS

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

Orientador: Prof. Dr. Jaime Wojciechowski

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **LÍVIA MARIA DA SILVA FERNANDES**, intitulada: **MEMORIAL DE PROJETOS: A IMPORTÂNCIA DO STORYTELLING NA ANÁLISE E VISUALIZAÇÃO DE DADOS**, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua _____ 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, 23 de Outubro de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Os dados podem ser definidos como uma simplificação do mundo real, de forma que sua visualização funcione como uma abstração da realidade, permitindo compreender relações e padrões que de outra forma permaneceriam ocultos. A conexão entre eles e a representação real é essencial para dar significado à parte visual da análise, sendo responsabilidade do pesquisador assegurar essa correspondência. Diante da crescente disponibilidade e complexidade de dados, a capacidade de apresentá-los de modo acessível e inteligível torna-se uma habilidade essencial no mundo contemporâneo. Com isso, esse trabalho tem como objetivo evidenciar a importância de uma análise de dados bem fundamentada e estruturada, fundamental em um mundo cada vez mais imerso na cultura *data-driven*. Para elaborar um *storytelling* é necessário ter atenção a três conceitos principais: quem é o público ao qual será apresentado, o quê está sendo comunicado e como será realizada a análise. Além disso, é fundamental que o orador demonstre domínio dos dados que está apresentando, articulando sua narrativa de forma clara, objetiva e concisa, de modo a garantir que a mensagem seja realmente compreendida e cause impacto. A evolução da tecnologia permitiu o avanço de pesquisas no tema de visualização de dados, mas também traz desafios, como a grande variedade de dados provenientes do *Big Data.* Com isso, fica clara a necessidade de compreensão dos dados e dos métodos para analisá-los.

Palavras-chave: Narrativa de dados; análise de dados, visualização de dados; orientação por dados; dados

ABSTRACT

Data can be defined as a simplification of the real world, in such a way that their visualization functions as an abstraction of reality, allowing the identification of relationships and patterns that would otherwise remain hidden. The connection between data and their real-world representation is essential to give meaning to the visual aspect of the analysis, and it is the researcher's responsibility to ensure this correspondence. Given the growing availability and complexity of data, the ability to present them in an accessible and intelligible manner has become an essential skill in the contemporary world.In this context, this study aims to highlight the importance of a well-founded and structured data analysis, which is fundamental in a world increasingly immersed in a data-driven culture. To construct effective storytelling, it is necessary to pay attention to three main concepts: who the target audience is, what is being communicated, and how the analysis will be conducted. Furthermore, it is crucial that the presenter demonstrates mastery of the data being presented, articulating their narrative clearly, objectively, and concisely, so as to ensure that the message is truly understood and creates impact. Technological evolution has enabled advances in research on data visualization but also poses challenges, such as the wide variety of data originating from Big Data. Therefore, the need to understand data and the methods used to analyze them becomes evident.

Keywords: Storytelling; data analysis; data visualization; data-driven; data

SUMÁRIO

1 PARECER TÉCNICO

Nathan Yau (2013), em *Data Points*, define os dados como uma simplificação do mundo real, de forma que sua visualização funcione como uma abstração da realidade. Ele defende que essa visualização não atua de forma negativa, mas permite que o foco seja afastado dos dados individuais, uma vez que a forma como um dado específico se relaciona com o conjunto é muito importante em uma análise.

Esse trabalho tem como objetivo evidenciar a importância de uma análise de dados bem fundamentada, com *storytelling* estruturado e apresentação criada corretamente de acordo com o público alvo da análise. Seu tema é fundamental em um ambiente corporativo cada vez mais voltado para dados, e com cultura *data-driven*.

Houve um tempo em que a Ciência da Informação focava suas preocupações em como coletar, organizar, armazenar e recuperar dados, no entanto atualmente, a visualização desses dados é um importante tema, com foco em proporcionar novas interpretações e significados (Rodrigues; Dias, 2017). A utilização das técnicas de visualização de dados é considerada uma estratégia emergente e de inovação e tem recebido cada vez mais atenção em pesquisas (Cairo, 2012; Manovich, 2011; Dur, 2014 citados por Rodrigues; Dias, 2017).

Silva (2021) aborda o avanço da Tecnologia da Informação como possibilitador da cultura visual e representação gráfica de dados. Ele defende que a visualização de dados é em geral compreendida como um "equivalente moderno da comunicação visual" (Silva, 2021, p. 207), com objetivo de transmitir informações com clareza e eficiência através de gráficos e tabelas, auxiliando na análise e raciocínio sobre os dados. O autor também diferencia os conceitos de infografia e visualização de dados. Enquanto a visualização de dados conta com facilidade de manipulação, que pode ser realizada com a instalação de *softwares* (Sato, 2017), a infografia é estática.

A conexão entre os dados e a representação real é fundamental e responsabilidade do pesquisador (Yau, 2013), ela é importante para dar significado à parte visual da análise, e fundamental para uma análise de dados reflexiva. A capacidade de contar histórias com dados é uma habilidade importante no mundo atual, de grande volume de dados. Uma visualização de dados eficaz pode decidir o sucesso ou fracasso na comunicação de um estudo (Knaflic, 2019).

Cole Knaflic (2019) faz uma interessante abordagem do tema, quando ao definir o que de fato deve ser o foco em uma análise de dados, difere análise exploratória e análise explanatória. De acordo com a autora, o passo de analisar de forma exploratória faz com que o analista compreenda os dados e defina quais deles levarão à uma análise que transforme dados em informação. Enquanto a
explanatória é responsável por passar ao público uma história dos dados específicos selecionados anteriormente. Ela defende que uma análise deve começar pelo entendimento de três conceitos: quem, o quê e como.

O conceito de quem é referente ao público da análise. Deve-se entendê-lo e também assimilar como ele o interpreta. É importante ser o mais específico possível ao definir o público, para fazer uma comunicação eficiente.

Quanto ao o quê, deve-se dar relevância ao que está comunicando. Ela orienta que o analista tome uma postura confiante ao apresentar a análise, uma vez que tem conhecimento sobre os dados os quais está apresentando. É importante também definir o mecanismo de comunicação com o público, que delimita como a informação será transmitida ao público e por qual canal.

O conceito de como diz respeito a quais dados estão disponíveis e ajudarão na apresentação da ideia. Além disso, quais são as ferramentas utilizadas para apresentar os dados visualmente.

É crucial ao orador transmitir ao público da análise que possui o domínio dos dados do estudo. Knaflic cita Nancy Duarte (2020) ao falar sobre o conceito de ‘A Grande Ideia’, que corrobora com a necessidade de articular sua história de forma clara e concisa. São três componentes que definem a Grande Ideia: articular o ponto de vista único, transmitir o que é de interesse e formar uma frase completa. Abordando ainda a necessidade de conhecer efetivamente o material do estudo, Knaflic também define o conceito de ‘História de 3 minutos’, em que deve-se pensar formas de sintetizar suas ideias principais, a fim de garantir que o analista seja claro e consiga articular a história que quer contar.

A visualização de dados torna-se cada vez mais um tema importante, com mais atenção em pesquisas e tendo a evolução da tecnologia como um possibilitador desse avanço (Silva, 2021). Ela tem como objetivo transmitir informações de forma clara e eficiente, através de gráficos e tabelas, de forma a auxiliar análises sobre os dados (Silva, 2021). No entanto, é imprescindível que haja conexão entre os dados e a representação real, para que a análise tenha significado (Yau, 2013), uma vez que a

eficiência da visualização de dados é fundamental para o sucesso de um estudo (Knaflic, 2019). É crucial também, ao realizar uma análise, ter um público alvo bem definido, a fim de criar uma comunicação eficiente (Knaflic, 2019). O avanço da tecnologia também traz desafios para o tema, uma vez que com o *Big Data*, existe uma grande velocidade e variedade de dados, aumentando a dificuldade em garantir a qualidade e segurança deles. Diante do exposto, pode-se concluir que esses desafios corroboram com a necessidade de compreensão dos dados e dos metadados e também de quais métodos vão ser utilizados para tratá-los e analisá-los.

REFERÊNCIAS

CAIRO, A. **El arte funcional: una introducción a los gráficos de información y visualización**. 2012.

DUARTE, N. **Ressonância: apresente histórias visuais que encantem o público**. 1. ed. Rio de Janeiro: Alta Books, 2020.

DUR, B. I. U. **Data visualization and infographics in visual communication design education at the age of information**. Journal of arts and humanities, v. 3, n. 5, p. 39-50, 2014.

KNAFLIC, C. N. **Storytelling com Dados: um guia sobre visualização de dados para profissionais de negócio**. Alta Books Editora, 2018.

MANOVICH, L. **Trending: The promises and the challenges of big social data**. Debates in the digital humanities, v. 2, n. 1, p. 460-475, 2011.

RODRIGUES, A. A.; DIAS, G. A. **Estudos sobre visualização de dados científicos no contexto da Data Science e do Big Data**. *Pesquisa Brasileira em Ciência da Informação e Biblioteconomia*, João Pessoa, v. 12, n. 1, p. 219-228, 2017

SATO, S. N. **A infografia na divulgação científica: um estudo de caso da revista Pesquisa FAPESP**. 2017. 155f. Dissertação (Mestrado) -Escola de Comunicação e Artes, Universidade de São Paulo, 2017.

SILVA, F. C. C. **Visualização de dados: passado, presente e futuro**. LIINC em revista. Rio de Janeiro, RJ. Vol. 15, n. 2 (nov. 2019), p. 205-223, 2019.

YAU, N. **Data points: Visualization that means something**. John Wiley & Sons, 2013.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

A – ENUNCIADO

1 ChatGPT

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

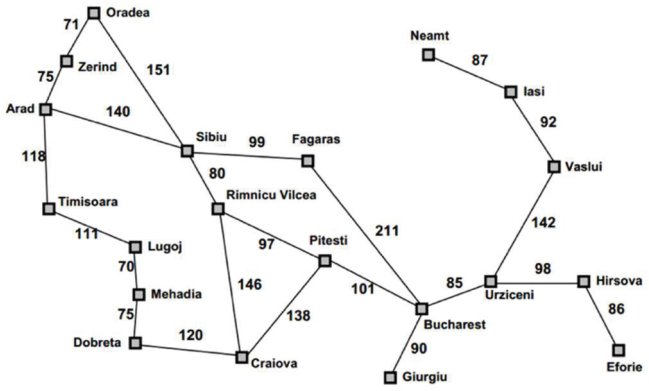
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

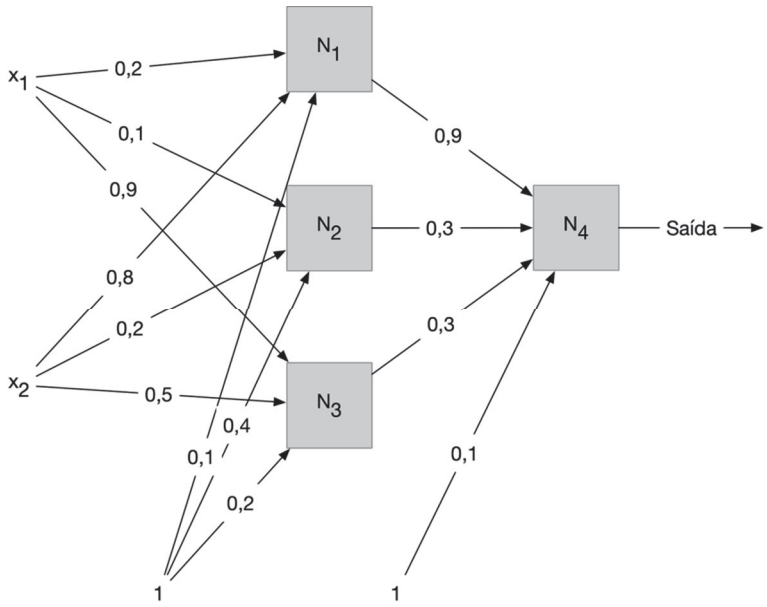
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1a.

Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos que podem realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui habilidades como aprendizado, raciocínio, resolução de problemas, reconhecimento de padrões, compreensão de linguagem natural e tomada de decisões.

Os sistemas de IA são projetados para processar grandes quantidades de dados, identificar padrões e tomar decisões ou realizar ações com base nesses padrões. Eles podem usar uma variedade de técnicas, incluindo aprendizado de máquina, redes neurais, lógica simbólica, algoritmos de otimização e muito mais.

A IA tem aplicações em uma ampla gama de áreas, incluindo reconhecimento de fala, visão computacional, processamento de linguagem natural, diagnóstico médico, automação industrial, finanças, jogos, entretenimento e muito mais. Nos últimos anos, tem havido avanços significativos na IA, impulsionados pelo aumento da capacidade computacional, melhores algoritmos e grandes conjuntos de dados disponíveis para treinamento de modelos de IA.

1b.

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

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

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

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

1c.

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

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

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

comparações de preferências para orientar o modelo em direção ao comportamento desejado.

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

Referências:

GUINNESS, Harris. How does ChatGPT work?. 2023. Disponível em https://zapier.com/blog/how-does-chatgpt-work/

HETLER, Amanda. Definition: ChatGPT. 2023. Disponível em https://www.techtarget.com/whatis/definition/ChatGPT

OPENAI. What is ChatGPT?. 2024. Disponível em https://help.openai.com/en/articles/6783457-what-is-chatgpt

1d.

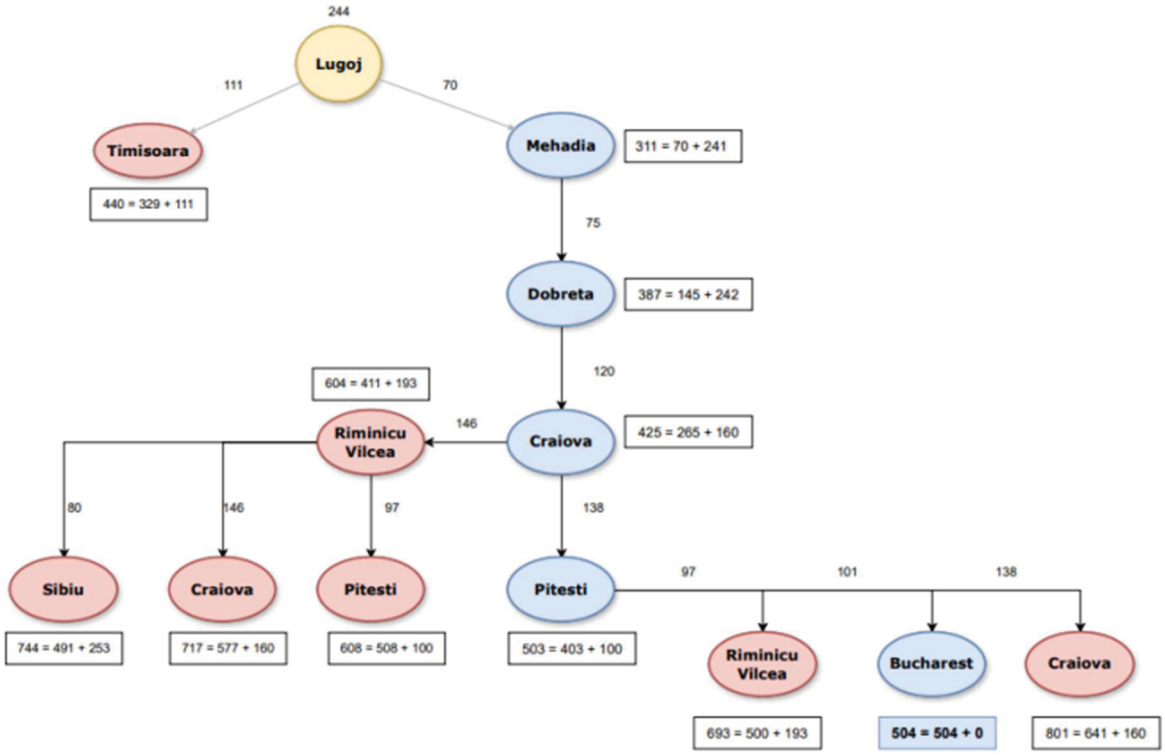
Agir como Humanos: Essa abordagem se enquadra com a forma como o ChatGPT funciona, pois o objetivo não é definir o que é pensamento nem implementar algum processo cognitivo, já que não é necessário verificar respostas corretas, basta que ele consiga imitar o comportamento humano, além de utilizar habilidades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

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

2a.

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

FIGURA 1 - Rota escolhida pelo algoritmo



FONTE: A autora (2025).

3a.

Legenda:
p: Uvas caem
q: Raposa come as uvas
r: Uvas estão maduras

Base de Conhecimento (BC):
R1: $p \rightarrow q$ (Se as uvas caem, então a raposa as come)
R2: $q \rightarrow r$ (Se a raposa as come, então estão maduras)
R3: $\neg r \vee p$ (As uvas estão verdes ou as uvas caem)
R4: $r \rightarrow p$ COND, R3
R5: $q \rightarrow p$ SH, R2, R4
R6: $(q \rightarrow p) \wedge (p \rightarrow q)$ CONJ, R5, R1
R7: $q \leftrightarrow p$ BICOND, R6

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

4.

Dada a entrada x1 = - 3, x2 = 1, dê:

a. $\Sigma xi + wi = -3 * 0.2 + 1 * 0.8 + 1 * 0.1 = 0.3$
u = 0.3
linear(u) = u = 0.3
Saída N1 = 0.3

b. $\Sigma xi + wi = -3 * 0.1 + 1 * 0.2 + 1 * 0.4 = 0.3$
u = 0.3
linear(u) = u = 0.3
Saída N2 = 0.3

c. $\Sigma xi + wi = -3 * 0.9 + 1 * 0.5 + 1 * 0.2 = -2$
u = − 2
linear(u) = u = − 2
Saída N3 = − 2

d. $\Sigma xi + wi = 0.3 * 0.9 + 0.3 * 0.3 + (-2) * 0.3 + 1 * 0.1 = -0.14$
u = − 0. 14
tan(u) = $(e^{u} - e^{-u})/((e^{u} + e^{-u}) = (e^{-0.14} - e^{-(-0.14)})/((e^{-0.14} + e^{-(-0.14)})$ = − 0. 1391
Saída N4 = − 0. 1391

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

1a.

```
import pandas as pd
import matplotlib.pyplot as mpb
import seaborn as sns
#lendo o arquivo com dados carros
pd.read_csv('/content/drive/MyDrive/IAA_UFPR/IntroducaoPyt
hon/dados.csv’)
```

1b.

```
#conferindo a existência de dados faltantes
carros.isna().sum()
```

QUADRO 1 - Quantidade de dados faltantes por variável

Variável	Quantidade
year_of_reference	65245
month_of_reference	65245
fipe_code	65245
authentication	65245
brand	65245
model	65245
fuel	65245
gear	65245
engine_size	65245
year_model	65245
avg_price_brl	65245

FONTE: A Autora (2025).

Observou-se que em 24% das linhas (65245) dos dados originais, todas as variáveis estavam vazias. Como nesses casos nenhuma informação foi preenchida, o mais prudente é a exclusão dessas linhas.

1c.

```
#verificando a existência de dados duplicados
carros2.duplicated().sum()
#vamos olhar quais são as linhas duplicadas
linhas_duplicadas =
carros2[carros2.duplicated(keep=False)]
linhas_duplicadas.head()
```

Foram encontradas 2 linhas duplicadas, cada uma com uma gêmea, e como a base refere-se ao cadastro de modelos de carros, a duplicidade não faz sentido e por isso foi excluída.

1d.

```
#estatisticas descritivas
carros2[colunas_numericas].describe()
carros2[colunas_categoricas].describe()
```

QUADRO 2 - Estatísticas descritivas de dados numéricos

	year_of_reference	engine_size	avg_price_brl
count	202295	202295	202295
mean	2021,56	1,82	52756,77
std	0,57	0,73	51628,91
min	2021	1,00	6647
25%	2021	1,40	22855
50%	2022	1,60	38027
75%	2022	2,00	64064
max	2023	6,20	979358

FONTE: A autora (2025).

QUADRO 3 - Estatísticas descritivas de dados categóricos

	month	fipe_code	authentication	brand	model	fuel	gear
count	202295	202295	202295	202295	202295	202295	202295
unique	12	2091	202295	6	2112	3	2
top	January	003281-6	cfzlctzfwrqp	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual
freq	24260	425	1	44962	425	168684	161883

FONTE: A autora (2025).

1e.

```
carros2['model'].value_counts(normalize=True)
```

FIGURA 2 - Contagem por marca e modelo dos automóveis

FONTE: A autora (2025).

1f.

```
carros2['fuel'].value_counts(normalize=True)
carros2['gear'].value_counts(normalize=True)
```

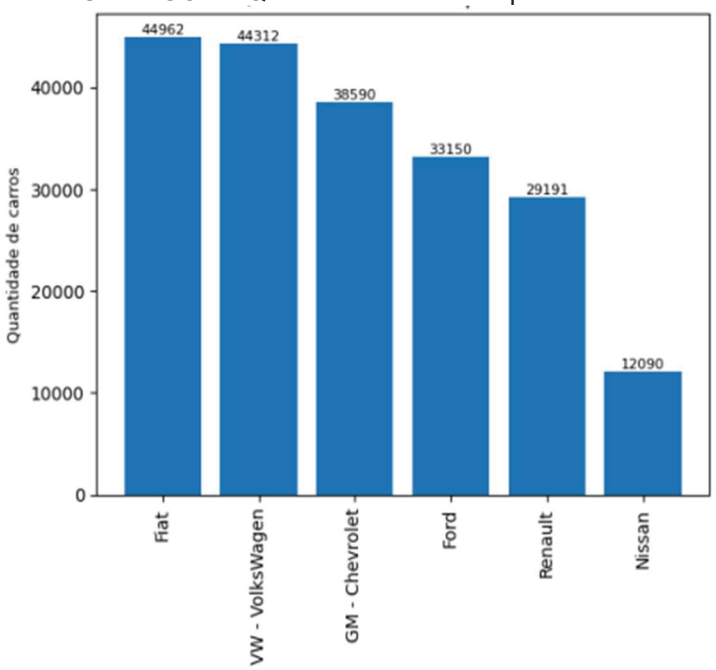
Através da análise exploratória, observou-se que a base de dados analisada é composta por 2112 modelos de carros distintos, com ano de fabricação entre 2000 e 2023. A mediana do preço médio dos carros foi de R$38 mil reais, e o modelo mais barato e mais caro custavam respectivamente, R$6,6 mil e R$979 mil. A marca mais

frequente dos carros cadastrados foi Fiat, 83% dos automóveis são movidos à gasolina e 80% do tipo manual.

2a.

```
qtd_marcas = carros2['brand'].value_counts()
grafico1 = mpb.bar(qtd_marcas.index,qtd_marcas.values)
mpb.xticks(rotation=90) mpb.bar_label(grafico1, size =8)
mpb.title('Quantidade de carros por marca')
mpb.ylabel('Quantidade de carros', size =9)
```

GRÁFICO 1 - Quantidade de carros por marca

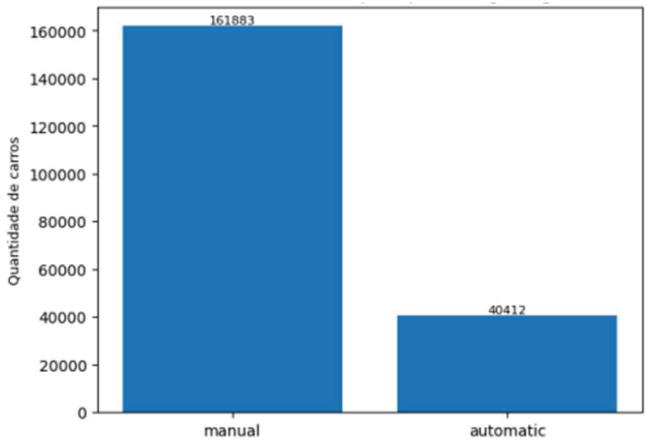


FONTE: A autora (2025).

2b.

```
tipo_engrenagem = carros2['gear'].value_counts()
grafico2 =
mpb.bar(tipo_engrenagem.index,tipo_engrenagem.values)
mpb.bar_label(grafico2, size =8)
mpb.title('Quantidade de carros por tipo de engrenagem')
mpb.ylabel('Quantidade de carros', size =9)
```

GRÁFICO 2 - Quantidade de carros por tipo de embreagem

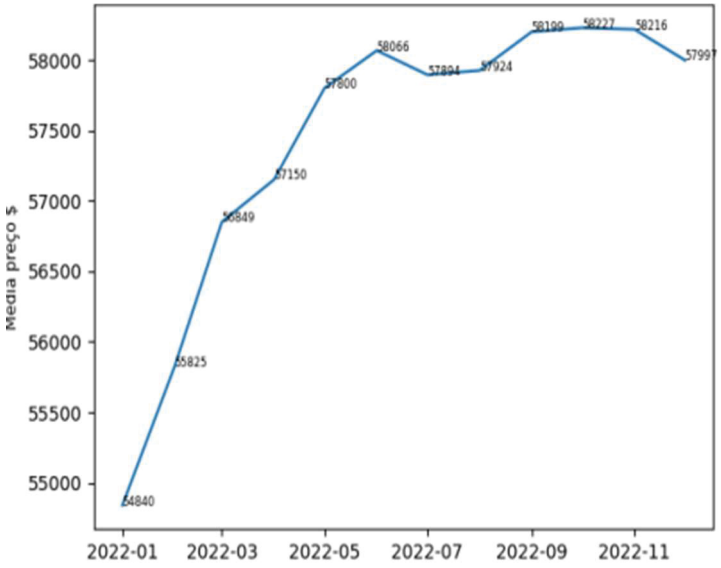


FONTE: A autora (2025).

2c.

```
grafico3 =
mpb.plot(media_preco_mes.index,media_preco_mes.values)
mpb.ylim(0,60000)
for x, y, text in zip(media_preco_mes.index,
media_preco_mes.values, media_preco_mes)
mpb.text(x, y, text,size=6)
mpb.title('Média de preço por mês - ano 2022')
mpb.ylabel('Média preço $', size =9)
```

GRÁFICO 3 - Média de preço por mês (2022)

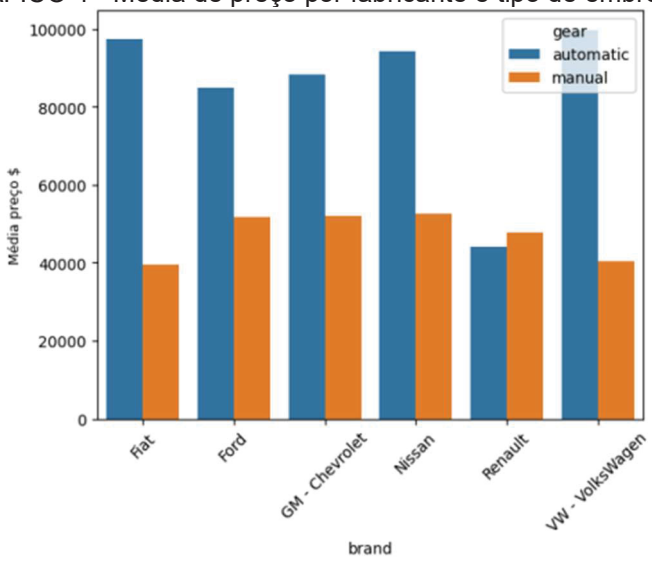


FONTE: A autora (2025).

2d.

```
grafico4 = sns.barplot(x='brand', y='PrecoMedio',
hue='gear', data=media_preco_marca_engrenagem)
mpb.xticks(rotation=45)
```

GRÁFICO 4 - Média de preço por fabricante e tipo de embreagem



FONTE: A autora (2025).

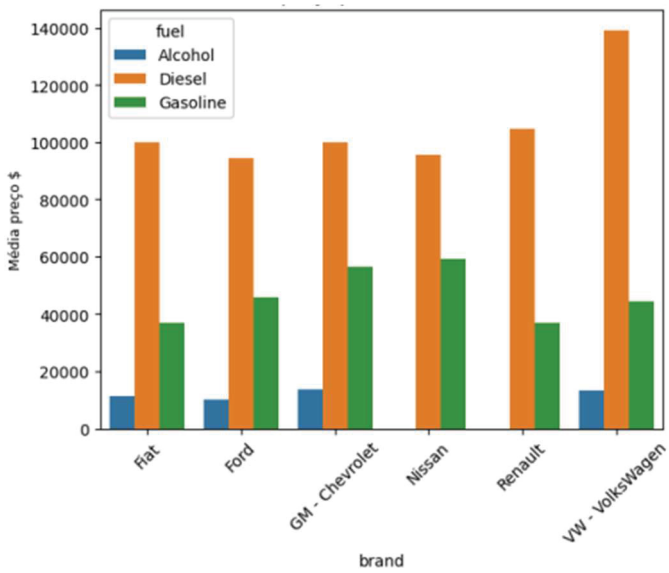
2e.

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

2f.

```
grafico5 =
sns.barplot(x='brand',y='PrecoMedio',hue='fuel',
data=media_preco_marca_combustivel)
mpb.xticks(rotation=45)
mpb.title('Média de preço por marca e combustível')
mpb.ylabel('Média preço $', size =9)
```

GRÁFICO 5 - Média de preço por fabricante e tipo de combustível



FONTE: A autora (2025).

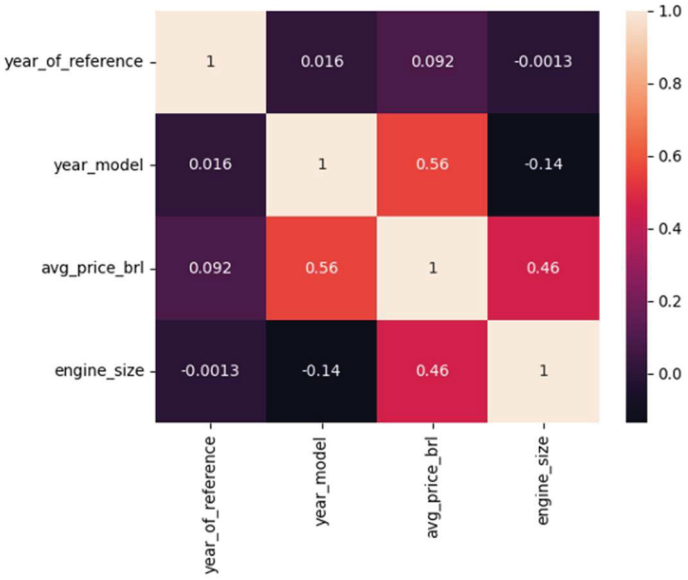
2g.

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

3a.

```
dados_num=carros_modelo[['year_of_reference','year_model'
,'avg_price_brl','engine_size]]
sns.heatmap(dados_num.corr("pearson"), annot = True)
mpb.title("Mapa de Correlação das Variáveis Numéricas",
fontsize = 10)
mpb.show()
```

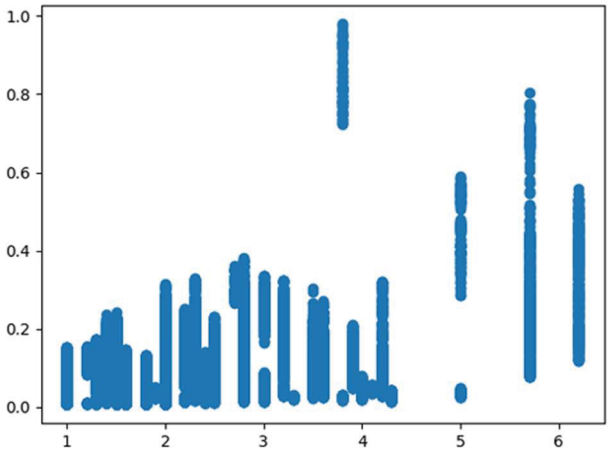
FIGURA 3 - Mapa de Correlação das Variáveis Numéricas



FONTE: A Autora(2025).

```
mpb.scatter(dados_num['engine_size'],dados_num['avg_price
_brl']) mpb.title('Dispersão entre tamanho do motor e
preço médio')
```

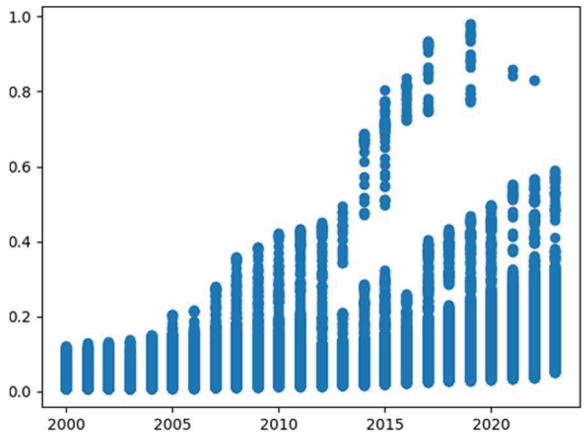
GRÁFICO 6 - Dispersão entre tamanho do motor e preço médio



FONTE: A Autora(2025).

```
mpb.scatter(dados_num['year_model'],dados_num['avg_price_
brl']) mpb.title('Dispersão entre ano do modelo e preço
médio')
```

GRÁFICO 7 - Dispersão entre ano do modelo e preço médio



FONTE: A Autora(2025).

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

3b.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.25, random_state=123)
```

3c.

```
#Random Forest Sem Parâmetros
RF_semParametros=RandomForestRegressor()
RF_semParametros.fit(X_train,Y_train)



#Random Forest Com Parâmetros
RF_comParametros=RandomForestRegressor(min_samples_leaf=3
0, min_samples_split=30, n_estimators=100,
random_state=856
```

```
RF_comParametros.fit(X_train,Y_train)


#XGBost
 XGBoost_semParametro = XGBRegressor()
XGBoost_semParametro.fit(X_train,Y_train)
```

3d.

```
Yhat_RF_semParametros = RF_semParametros.predict(X_test)
Yhat_RF_comParametros = RF_comParametros.predict(X_test)
Yhat_XG = XGBoost_semParametro.predict(X_test)
```

3e.

```
#Random Forest Sem Parâmetros
feature_importances_md1=pd.DataFrame(RF_semParametros.fea
ture_importance_, index = X_train.columns,
columns=['importance’])
feature_importances_md1

#Random Forest Com Parâmetros
feature_importances_md2 =
pd.DataFrame(RF_comParametros.feature_importances_, index
= X_train.columns, columns=['importance’]
feature_importances_md2

#XGBost
feature_importances_md3=pd.DataFrame(XGBoost_semParametro
.feature_importances_, index = X_train.columns,
columns=['importance'])
feature_importances_md3
```

3f.

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

3g.

```
#Random Forest sem parametros
mse_md1 = mean_squared_error(Y_test,
Yhat_RF_semParametros)
mae_md1 = mean_absolute_error(Y_test,
Yhat_RF_semParametros)
r2_score(Y_test, Yhat_RF_semParametros)
```

```
#Random Forest com parametro
mse_md2 = mean_squared_error(Y_test,
Yhat_RF_comParametros)
mae_md2 = mean_absolute_error(Y_test,
Yhat_RF_comParametros)
r2_score(Y_test, Yhat_RF_comParametros)

#XGBost
mse_m3 = mean_squared_error(Y_test, Yhat_XG)
mae_md3 = mean_absolute_error(Y_test,Yhat_XG)
r2_score(Y_test,Yhat_XG)
```

QUADRO 4 - Comparação entre modelos

Modelo	Configuração	MSE	MAE	R²	Variáveis Importantes
RF	Sem parâmetros	53.923.315,28	4.194,11	0,98	engine_size (0,46), year_model (0,40)
RF	min nó folha: 30, min abertura nó: 30, máx árvores: 100, semente bootstrap	134.879.179,34	5.197,45	0,95	engine_size (0,47), year_model (0,42)
XGBoost	Sem parâmetros	69.512.526,98	4.867,86	0,97	engine_size (0,44), fuel (0,20), year_model (0,17)

FONTE: A autora (2025)

Foram ajustados três modelos para comparação: *random forest* sem parâmetros, *random forest* com parâmetros e *XGBost* sem parâmetros. Os três modelos tiveram bom ajuste aos dados, com R2 sempre superior a 0,9. No entanto, a Random Forest sem parâmetros pré definidos, apresentou MSE e MAE menores do que os demais modelos, e teve o melhor valor de R2, 0,98. Com isso, esse modelo,

que teve como variáveis importantes *engine_size* (tamanho do motor) e *year_model* (ano do modelo) foi escolhido como o melhor.

3h.

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

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1.1.

```
mirror <- "cran-r.c3sl.ufpr.br" options(repos = mirror)
# Instale o pacote mlbench se ainda não o tiver instalado
install.packages("mlbench")
# Carregue o pacote
library(mlbench)
data(Satellite)
```

1.2

```
 set.seed(123)
particao <- createDataPartition(Satellite$classes, p = 0.8,
list = FALSE)
```

1.3

```
# 3.1 Treinamento do modelo Random Forest
modelo_rf <- randomForest(classes ~ ., data = dados_treino)
# 3.2 Treinamento do modelo SVM
modelo_svm <- svm(classes ~ ., data = dados_treino)
# 3.3 Treinamento do modelo RNA
modelo_rna <- neuralnet(classes ~ ., data = dados_treino,
hidden = c(5, 2), linear.output =FALSE)
```

1.4

```
library(caret)
# Função para calcular métricas de desempenho
calcular_metricas <- function(matriz_confusao) {
# Precisão (precision)
precisao <- diag(matriz_confusao) /
colSums(matriz_confusao)
# Recall
recall <- diag(matriz_confusao) / rowSums(matriz_confusao)
# F1-score
f1_score <- 2 * (precisao * recall) / (precisao + recall)
# Retornar as métricas
return(data.frame(precisao = precisao, recall = recall,
f1_score = f1_score)) }
# Função para imprimir as métricas
imprimir_metricas <- function(nome_modelo,
matriz_confusao) {
cat("\nModelo:", nome_modelo, "\n")
print(calcular_metricas(matriz_confusao)) }
# Função para plotar a matriz de confusão
plotar_matriz_confusao <- function(nome_modelo,
matriz_confusao) { confusionMatrix(matriz_confusao, main =
nome_modelo) }
```

QUADRO 5 - Métricas modelo Random Forest

	precisao	recall	f1_score
red soil	0,993	0,971	0,982
cotton crop	0,986	0,986	0,986
grey soil	0,967	0,906	0,935
damp grey soil	0,72	0,818	0,766
vegetation stubble	0,851	0,96	0,902
very damp grey soil	0,917	0,899	0,908

FONTE: A Autora(2025).

QUADRO 6 - Métricas modelo SVM

	precisao	recall	f1_score
red soil	0,993	0,959	0,976
cotton crop	0,979	0,978	0,979
grey soil	0,967	0,87	0,916
damp grey soil	0,632	0,712	0,669
vegetation stubble	0,808	0,95	0,874
very damp grey soil	0,87	0,888	0,879

FONTE: A Autora(2025).

QUADRO 7 - Métricas modelo RNA

	precisao	recall	f1_score
red soil	0	NaN	NaN
cotton crop	0	NaN	NaN
grey soil	0	NaN	NaN
damp grey soil	1	0,097	0,177
vegetation stubble	0	NaN	NaN
very damp grey soil	0	NaN	NaN

FONTE: A Autora(2025).

1e.

Na atividade, foram treinados três modelos: *Random Forest*, SVM (*Support Vector Machine*) e RNA (Rede Neural Artificial), utilizando os dados de treinamento. Cada modelo foi treinado para prever as classes com base nas características fornecidas pelo conjunto de dados. Em seguida, avaliamos o desempenho de cada modelo utilizando os dados de teste. Calculamos métricas como precisão, *recall* e *F1-Score*. A precisão representa a proporção de exemplos previstos como positivos que são verdadeiramente positivos, o *recall* é a proporção de exemplos positivos corretamente identificados pelo modelo, e o *F1-Score* é a média harmônica entre

precisão e *recall*, fornecendo uma medida geral do desempenho do modelo. Posteriormente, calculamos os valores de *F1-Score* para cada modelo e identificamos aquele com o maior valor como o melhor modelo para esta tarefa. O modelo SVM obteve o maior *F1-Score*, seguido pelo modelo *Random Forest*. Optamos pelo modelo SVM como o melhor devido ao seu alto F1-Score, indicando um bom equilíbrio entre precisão e *recall*. O SVM obteve um F1-Score superior ao do *Random Forest*, o que confirma sua capacidade na classificação das classes com base nos dados de satélite. Assim, considerando as métricas de desempenho, selecionamos o SVM como o modelo mais adequado para esta tarefa.

2.

2.1

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

2.2

```
dataset <- dataset[, !names(dataset) %in% "NR"]
```

2.3

```
# Setando uma semente de aleatoriedade
 set.seed(123)
 # Criando índices para o treino
 log("Particionando dados em treino e teste")
 indices <- createDataPartition(dataset$VOL, p = 0.8, list
= FALSE)
 # Separando dados em treino e teste
 dados_treino <- dataset[indices, ]
 dados_teste <- dataset[-indices, ]
```

2.4

```
log("Treinando modelo Random Forest")
 rf <- train(VOL ~ ., data = dados_treino, method = "rf")
 log("Treinando modelo SVM")
 svm <- train(VOL ~ ., data = dados_treino, method =
"svmRadial")
 log("Treinando modelo Neural Network")
```

```
 rna <- train(VOL ~ ., data = dados_treino, method =
"neuralnet")
 log("Treinando modelo Alométrico de SPURR")
 alom <- nls(VOL ~ b0 + b1 * (DAP ^ 2) * HT, data =
dados_treino, start = list(b0 = 0.5, b1
```

2.5

```
log("Realizando predições")
 predicoes_rf <- predict(rf, dados_teste)
 predicoes_svm <- predict(svm, dados_teste)
 predicoes_rna <- predict(rna, dados_teste)
 predicoes_alom <- predict(alom, dados_teste)
```

2.6

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

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


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


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

# Função para retornar as metricas de avaliação
calcular_metricas <- function(observacoes, predicoes,
nome_modelo) {
  r2 <- calcular_coef_r2(observacoes, predicoes)
  syx <- calcular_erro_syx(observacoes, predicoes)
  syx_percent <- calcular_erro_syx_percent(observacoes,
predicoes)
  score <- calcular_score(r2, syx_percent / 100)
```

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

2.7

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

- Coeficiente de determinação (R2)
- Erro padrão da estimativa (Syx)
- Porcentagem do erro padrão da estimativa (Syx%)

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

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

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

QUADRO 8 - Comparação entre modelos

Modelo	R^2	S_{xy}	S_{xy} %	Score
RNA	0,88679	0,13545	10,06305	0,89308
Alométrico	0,86944	0,14546	10,80670	0,88069
Random Forest	0,84867	0,15660	11,63489	0,86616
SVM	0,79008	0,18444	13,70321	0,82652

FONTE: A autora (2025).

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

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

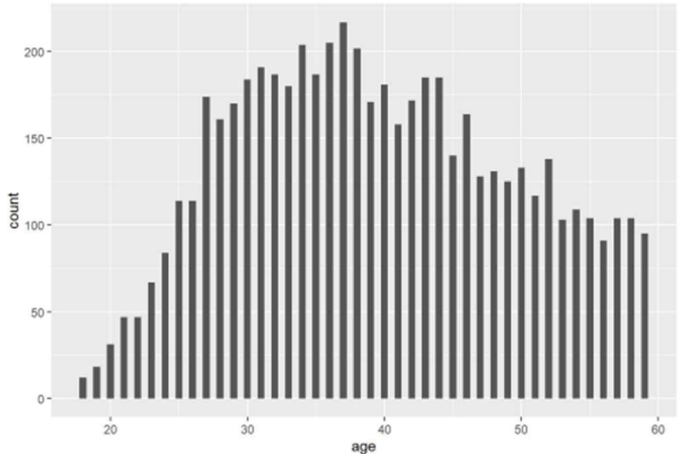
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

1.

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

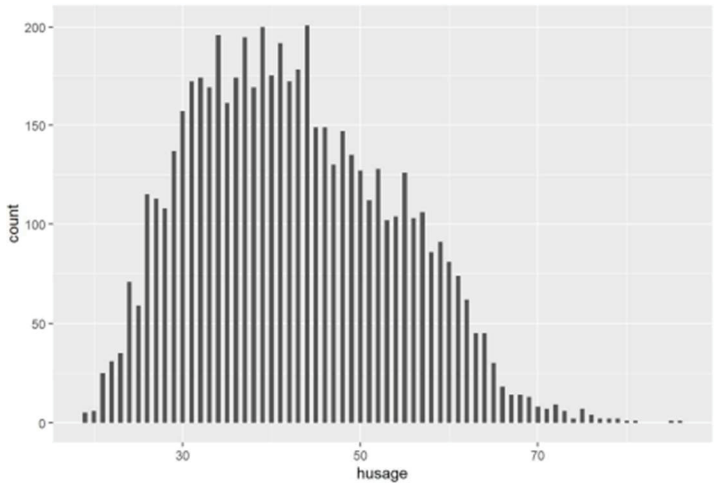
GRÁFICO 8 - Histograma de idade das esposas



FONTE: A autora (2025).

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

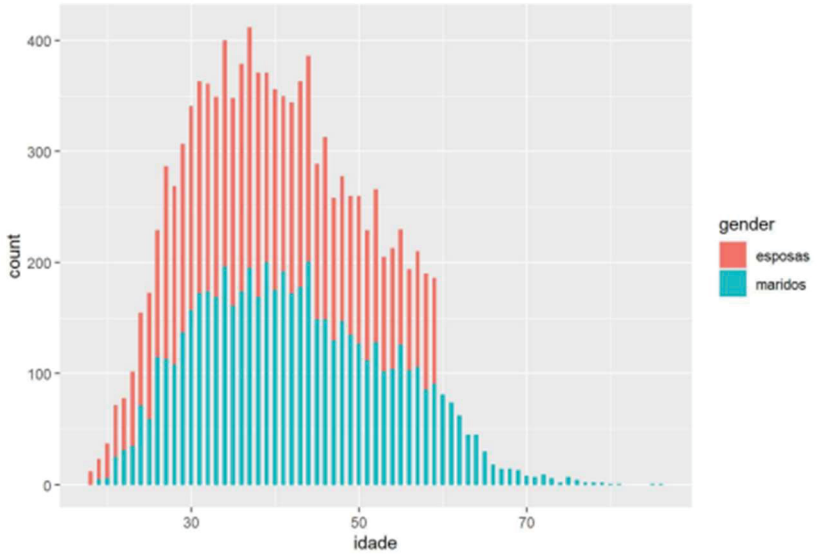
GRÁFICO 9 - Histograma de idade dos maridos



FONTE: A autora (2025).

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

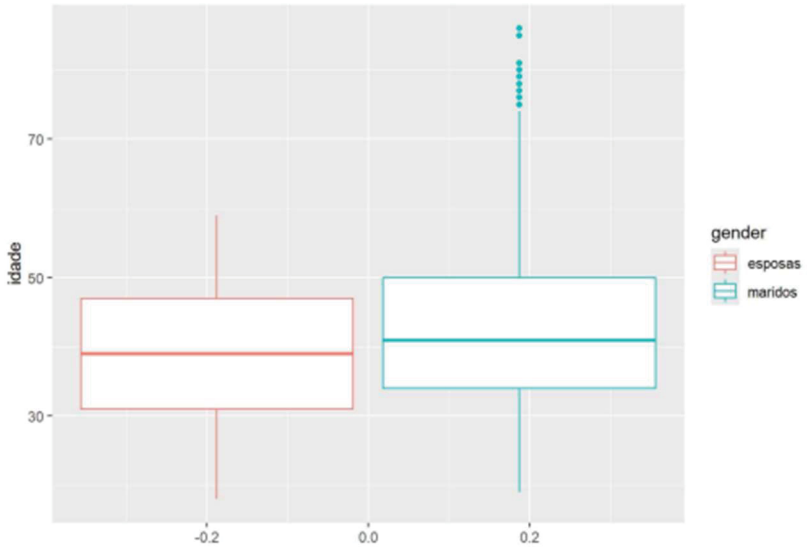
GRÁFICO 10 - Histograma de idades dos maridos e das esposas



FONTE: A autora (2025).

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

GRÁFICO 11 - Boxplot de idade dos maridos e das esposas



FONTE: A autora (2025).

Observamos que a distribuição das idades das esposas está concentrada entre 30 e 50 anos, com idade máxima de 59 anos. Já para as idades dos maridos, observamos nos gráficos, que essa medida atinge valores mais altos, mesmo que tenha concentração também em torno de 30 e 50 anos e que homens e mulheres

tenham mediana de idade similar, observamos que existem maridos com idade superior a 70 anos, que aparecem como outliers no gráfico.

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

QUADRO 9 - Tabela de frequência de idade das esposas

Age	Frequencia	Relativa	Acumulada
18	12	0,0021	0,0021
19	18	0,0032	0,0053
20	31	0,0055	0,0108
21	47	0,0083	0,0192
22	47	0,0083	0,0275
23	67	0,0119	0,0394
24	84	0,0149	0,0543
25	114	0,0202	0,0745
26	114	0,0202	0,0948
27	174	0,0309	0,1257
28	161	0,0286	0,1542
29	170	0,0302	0,1844
30	184	0,0327	0,2171
31	191	0,0339	0,251
32	187	0,0332	0,2842
33	180	0,0319	0,3161
34	204	0,0362	0,3523
35	187	0,0332	0,3855
36	205	0,0364	0,4219
37	217	0,0385	0,4604
38	202	0,0359	0,4963
39	171	0,0304	0,5266
40	181	0,0321	0,5588
41	158	0,028	0,5868
42	172	0,0305	0,6173
43	185	0,0328	0,6502

44	185	0,0328	0,683
45	140	0,0248	0,7078
46	164	0,0291	0,737
47	128	0,0227	0,7597
48	131	0,0233	0,7829
49	125	0,0222	0,8051
50	133	0,0236	0,8287
51	117	0,0208	0,8495
52	138	0,0245	0,874
53	103	0,0183	0,8923
54	109	0,0193	0,9116
55	104	0,0185	0,9301
56	91	0,0162	0,9462
57	104	0,0185	0,9647
58	104	0,0185	0,9831
59	95	0,0169	1

FONTE: A autora (2025).

QUADRO 10 - Tabela de frequência de idade dos maridos

Husage	Frequencia	Relativa	Acumulada
19	5	0,0009	0,0009
20	6	0,0011	0,002
21	25	0,0044	0,0064
22	31	0,0055	0,0119
23	35	0,0062	0,0181
24	71	0,0126	0,0307
25	59	0,0105	0,0412
26	115	0,0204	0,0616
27	113	0,0201	0,0816

28	108	0,0192	0,1008
29	137	0,0243	0,1251
30	157	0,0279	0,153
31	172	0,0305	0,1835
32	174	0,0309	0,2144
33	169	0,03	0,2444
34	196	0,0348	0,2792
35	161	0,0286	0,3078
36	174	0,0309	0,3387
37	195	0,0346	0,3733
38	169	0,03	0,4033
39	200	0,0355	0,4388
40	175	0,0311	0,4698
41	192	0,0341	0,5039
42	172	0,0305	0,5344
43	178	0,0316	0,566
44	201	0,0357	0,6017
45	149	0,0264	0,6282
46	149	0,0264	0,6546
47	130	0,0231	0,6777
48	147	0,0261	0,7038
49	135	0,024	0,7277

50	127	0,0225	0,7503
51	112	0,0199	0,7701
52	128	0,0227	0,7929
53	102	0,0181	0,811
54	104	0,0185	0,8294
55	126	0,0224	0,8518
56	103	0,0183	0,8701
57	106	0,0188	0,8889
58	86	0,0153	0,9042
59	91	0,0162	0,9203
60	81	0,0144	0,9347
61	74	0,0131	0,9478
62	62	0,011	0,9588
63	45	0,008	0,9668
64	45	0,008	0,9748
65	30	0,0053	0,9801
66	18	0,0032	0,9833
67	14	0,0025	0,9858
68	14	0,0025	0,9883
69	13	0,0023	0,9906
70	8	0,0014	0,992
71	7	0,0012	0,9933

72	9	0,0016	0,9949
73	6	0,0011	0,9959
74	2	0,0004	0,9963
75	7	0,0012	0,9975
76	4	0,0007	0,9982
77	2	0,0004	0,9986
78	2	0,0004	0,9989
79	2	0,0004	0,9993
80	1	0,0002	0,9995
81	1	0,0002	0,9996
85	1	0,0002	0,9998
86	1	0,0002	1

FONTE: A autora (2025).

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

2.

```
summary(dados$age)
moda <- sort(table(dados$age), decreasing = T)[1]
moda
summary(dados$husage)
modah <- sort(table(dados$husage), decreasing = T)[1]
modah
var(dados$age)
sd(dados$age)
```

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

QUADRO 11 - Tabela de medidas de dispersão e posição - Idades de maridos e esposas

	Esposas	Maridos
Média	39,4	42,4
Mediana	39	41
Moda	37	44
Variância	99,75	126,07
Desvio Padrão	9,99	11,23
Coef. de Variação	25,33	26,45

FONTE: A autora (2025).

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

3.

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

QUADRO 12 - Tabela de Teste de Normalidade - Teste de Kolmogorov

Parâmetro/Variável	Age	Husage
D	0,058909	0,059662
p-valor	< 2.2e-16	< 2.2e-16

FONTE: A autora (2025).

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

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

QUADRO 13 - Tabela de Teste de Comparação de Medianas - Mann Whitney

Parâmetro	
W	13619912
p-valor	< 2.2e-16
I.C inf	-3
I.C sup	-2

FONTE: A autora (2025).

O p-valor do teste realizado foi inferior a 5%, com isso, rejeitamos a hipótese nula de que a mediana das idades de homens e mulheres são iguais.

O intervalo de confiança da diferença entre as medianas está entre 2 e 3, com mediana da diferença igual a 3.

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

```
library(carData)
library(car)
library(RcmdrMisc)
library(zoo)
library(lmtest)
library(nortest)
library(lmtest)
library(sandwich)
library(caret)
library(glmnet)
```

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

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

# Padronização das variáveis numéricas
cols = c('husage', 'husearns', 'huseduc', 'hushrs',
'earns', 'age', 'educ', 'exper', 'lwage')

# Padronizando a base de treinamento e teste
pre_proc_val <- preProcess(dados_treino[,cols], method =
c("center", "scale"))

dados_treino[,cols] = predict(pre_proc_val,
dados_treino[,cols])
dados_teste[,cols] = predict(pre_proc_val,
dados_teste[,cols])

# Análise do Sumário das variáveis
summary(dados_treino)

summary(dados_teste)


# Crear un objeto con las variables que se usarán en el
modelo
cols_reg <- c('husage', 'husearns', 'huseduc', 'hushrs',
'earns', 'age',
              'educ', 'exper', 'lwage', 'husunion',
'husblack', 'hushisp',
              'kidge6', 'black', 'hispanic', 'union',
'kidlt6')
#cols22 <- setdiff(names(data_salarios), 'lwage')

# Generar variables dummies
dummies <- dummyVars(lwage ~ husage + husearns + huseduc +
hushrs + earns + age + educ + exper +
         husunion + husblack + hushisp + kidge6 + black +
hispanic + union + kidlt6,
                     data = data_salarios)

# Transformar los datos de entrenamiento y prueba usando
las variables dummies
```

```
train_dummies <- predict(dummies, newdata =
dados_treino[,cols_reg])

test_dummies <- predict(dummies, newdata =
dados_teste[,cols_reg])

print(dim(train_dummies)); print(dim(test_dummies))

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

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


# Vamos guardar a matriz de dados de teste das variáveis
# explicativas para o modelo em um objeto chamado
# "x_test"
x_test = as.matrix(test_dummies)

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


# Métricas de avaliação para os futuros modelos

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

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

# Modelo RIDGE
# Cálculo do valor ótimo de lambda

lambdas <- 10^seq(2, -3, by = -.1)
```

```
# Calculando o lambda por validação cruzada:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0, lambda =
lambdas)
# Vamos ver qual o lambda ótimo
best_lambda_ridge <- ridge_lamb$lambda.min
print('O valor ótimo de lambda foi: ')

best_lambda_ridge

#Estimando o modelo ridge
ridge-reg = glnnet(x, y_train, nlambda = 25, aplha = 0,
family = 'gaussian', lambda = best_lambda_ridge)
ridge_reg[["beta"]]

#predição dados treino
prediction_train <- predict (ridge_reg, s =
best_lambda_ridge, newx = x)
metricas_risge_treino <- eval_results(y_train,
predictions_train, dados_treino)
metricas_ridge_treino

#predição e avaliação nos dados de teste
predicition_test <- predict(ridge_reg, s =
best_lambda_ridge, newx = x_test)
metricas_ridge_test <- eval_results(y_test,
predictions_test, dados_teste)
metricas_ridge_test

#Modelo LASSO
lasso_lamb <- cv.glmnet(x,y_train, alpha = 1, lambda =
lambdas, nfolds = 10, standardize = TRUE)
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso


#estimando o modelo
lasso_model <- glmnet(x,y_train,alpha =1, lambda =
best_lambda_lasso, standardize = TRUE)
lasso_model[["beta]]

predictions_train_lasso <- predict(lasso_model, s =
best_lambda_lasso, newx = x)
metricas_lasso_treino <- eval_results(y_train,
predictions_train_lasso, dados_treino)
metricas_lasso_treino

#predição dos dados teste - lasso
predictions_test_lasso <- predict(lasso_model, s =
best_lambda_lasso, newx = x_test)
metricas_lasso_teste <- eval_results(y_test,
predictions_test_lasso, dados_teste)
```

```
metricas_lasso_teste

#Modelo ELASTICNET
train_cont <- trainControl(method = "repeatedcv", number =
10, repeats = 5, search = "random", verboseIter = FALSE)
elastic_reg <- train(lwage ~husage + husearns + huseduc +
hushrs + earns + age + educ + exper + husunion + husblck +
kidge6 + black + hispanic + union + kidlt6 + data =
dados_treino, method = "glmnet", tuneLength = 10, trControl
= train_cont)
elastic_reg$bestTune

#predição dados de treino (elasticnet)
predictions_train_elastic <- predict(elastic_reg, x)
metricas_elastic_treino      <-      eval_results(y_train,
predictions_train_elastic, dados_treino)
metricas_elastic_treino

#predição dados de teste (elasticnet)
predictions_test_elastic <- predict(elastic_reg, x_test)
metricas_elastic_teste       <-       eval_results(y_test,
predictions_test_elastic, dados_teste)
 metricas_elastic_teste

metricas_unificadas                                          <-
rbind.data.frame(metricas_ridge_treino,
metricas_lasso_treino,
metricas_elastic_treino,metricas_ridge_teste,
metricas_lasso_teste, metricas_elastic_teste)
row.names(metricas_unificadas)  <-  c('RIDGE  -  TREINO',
'LASSO - TREINO', 'ELASTICNET - TREINO', 'RIDGE - TESTE',
'LASSO - TESTE', 'ELASTICNET - TESTE' )
metricas_unificadas
```

QUADRO 14 - Métricas de qualidade dos modelos

	Treino		Teste	
	RMSE	R2	RMSE	R2
Ridge	0,5595395	0,6867634	0,5375854	0,7007985
Lasso	0,5596622	0,6866261	0,5340347	0,7047378
Elasticnet	0,5594508	0,6868627	0,5341944	0,7045612

FONTE: A autora (2025).

QUADRO 15 - Valores preditos para caso proposto e intervalos de confiança por modelo

	I.C Inf	Estimado	I.C Sup
Ridge	12,687332	12,977667	13,274646
Lasso	12,949224	13,245552	13,548661
Elasticnet	8,610227	8,807263	9,008807

FONTE: A autora (2025).

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

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

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

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1.

```
import re
 # Cria um bag of words e um stopwords por lingua no
pre_padroes
 def calcula_frequencia_no_dataset(elem_list, dataset):
    for elem in elem_list:
        if elem not in dataset:
            dataset[elem] =1
        else:
            dataset[elem] +=1
    return dataset
 def bagOfWords(pre_padroes):
    bow_por_lingua = {}
    for texto, lingua in pre_padroes:
        pattern_regex = re.compile('[^\w+]', re.UNICODE) #
Regex para identificar caracteres que NÃ
        texto = re.sub(pattern_regex,' ',texto) # Substitui
todos os caracteres que não são alfanum
        texto = texto.lower()
        bow = re.findall(r'\b\w+\b', texto) # Cria lista de
palavras
        # Cria dataset para lingua se ele não existir
        if lingua not in bow_por_lingua:
            bow_por_lingua[lingua] = {}
        bow_por_lingua[lingua]                              =
calcula_frequencia_no_dataset(bow, bow_por_lingua[lingua])
    return bow_por_lingua
 def stopWords(lista_de_linguas):
    stopWords_por_lingua = {}
    for lingua in lista_de_linguas:
        sorted_dict                                         =
sorted(bow_por_lingua[lingua].items(), key=lambda x: x[1],
reverse=True)
        stopWords_por_lingua[lingua]                        =
dict(sorted_dict[:5])
    return stopWords_por_lingua
 lista_de_linguas = set(item[1] for item in pre_padroes)
 bow_por_lingua = bagOfWords(pre_padroes)
 stopWords_por_lingua = stopWords(lista_de_linguas)
 print('Bag of Words')
 for l in lista_de_linguas:
    print(l, list(bow_por_lingua[l].keys())[:20])
    # print(l, bow_por_lingua[l])
 print('\nStop Words')
 for l in lista_de_linguas:
    print(l, list(stopWords_por_lingua[l].keys()))
    # print(l, stopWords_por_lingua[l])
```

```
 # a entrada é o vetor pre_padroes e a saída desse passo
deverá ser "padrões"
 import re
 import numpy as np
 import unicodedata
 if SEED is not None:
 np.random.seed(SEED)
 def tamanhoMedioFrases(texto):
 palavras = re.split("\s",texto)
 palvras = palavras.remove('') # Remove palavras vazias
 # print(palavras)
 tamanhos = [len(s) for s in palavras if len(s)>0]
 #print(tamanhos)
 soma = 0
 for t in tamanhos:
 soma=soma+t
 return soma / len(tamanhos)
 def calcula_frequencia(elem_list):
 contagem_elementos = {}
    for elem in elem_list:
        if elem not in contagem_elementos:
            contagem_elementos[elem] =1
        else:
            contagem_elementos[elem] +=1
    total = sum(contagem_elementos.values())
    # print(total)
    # print(contagem_elementos)
    frequencia_elementos = {}
    for elem, contagem in contagem_elementos.items():
        frequencia_elementos[elem] = contagem / total
    # print(frequencia_elementos)
    return frequencia_elementos
 def conta_ocorrencia(elem_list):
    contagem_elementos = {}
    for elem in elem_list:
        if elem not in contagem_elementos:
            contagem_elementos[elem] =1
        else:
            contagem_elementos[elem] +=1
    return contagem_elementos
 def frequenciaCaracteres(texto):
    texto = texto.lower()
    lista_caracteres = [c for c in texto]
    frequencia_caracteres                                    =
conta_ocorrencia(lista_caracteres)
    sorted_dict   =   sorted(frequencia_caracteres.items(),
key=lambda x: x[1], reverse=True)
    frequencia_caracteres = dict(sorted_dict[:1]) # o mais
frequente
    return frequencia_caracteres
 def frequenciaBigramas(texto):
```

```
    texto = texto.lower()
    texto = re.sub(r'\s+', '', texto) # Regex para remover
todos os espaçoes do texto
    bigramas = []
    for i in range(len(texto)-1):
        bigramas.append(texto[i] + texto[i+1])
    # print(bigramas)
    frequencia_bigramas = conta_ocorrencia(bigramas)
    sorted_dict   =   sorted(frequencia_bigramas.items(),
key=lambda x: x[1], reverse=True)
    frequencia_bigramas = dict(sorted_dict[:2]) # o mais
frequente
    # print(frequencia_bigramas)
    return frequencia_bigramas
 def frequenciaTrigramas(texto):
    texto = texto.lower()
    texto = re.sub(r'\s+', '', texto) # Regex para remover
todos os espaçoes do texto
    trigramas = []
    for i in range(len(texto)-2):
        trigramas.append(texto[i:i+3])
    # print(trigramas)
    frequencia_trigramas = conta_ocorrencia(trigramas)
   sorted_dict   =   sorted(frequencia_trigramas.items(),
key=lambda x: x[1], reverse=True)
    frequencia_trigramas = dict(sorted_dict[:1]) # o mais
frequente
    # print(frequencia_trigramas)
    return frequencia_trigramas
 def frequenciaAcentuacoes(texto):
    texto = texto.lower()
    lista_caracteres_acentuados = []
    for c in texto:
        if c != unicodedata.normalize('NFKD', c):
            lista_caracteres_acentuados.append(c)
    # print(lista_caracteres_acentuados)
frequencia_caracteres_acentuados                          =
conta_ocorrencia(lista_caracteres_acentuados)
 sorted_dict                                              =
sorted(frequencia_caracteres_acentuados.items(),
key=lambda x: x[1], reverse=True
 frequencia_caracteres_acentuados = dict(sorted_dict[:1])
# o mais frequente
 return frequencia_caracteres_acentuados
 def quantidadeAcentuacoes(texto):
 texto = texto.lower()
 qnt = 0
 for c in texto:
 if c != unicodedata.normalize('NFKD', c):
 qnt += 1
 return qnt
```

```
 # def bagOfWords(texto, lingua):
 #     texto = texto.lower()
 #     palavras = re.findall(r'\b\w+\b', texto)
 #     qnt = 0
 #     for p in palavras:
 #         if p in list(bow_por_lingua[lingua].keys()):
 #             qnt += 1
 #     return {f'bw_{lingua}': qnt}
 # def stopWords(texto,lingua):
 #     texto = texto.lower()
 #     palavras = re.findall(r'\b\w+\b', texto)
 #     qnt = 0
 #     for p in palavras:
 #                                         if    p    in
list(stopWords_por_lingua[lingua].keys()):
 #             qnt += 1
 #     return {f'sw_{lingua}': qnt}
 def extraiCaracteristicas(frase):
 # frase é um vetor [ 'texto', 'lingua' ]
 texto, lingua = frase
 pattern_regex = re.compile('[^\w+]', re.UNICODE) # Regex
para identificar caracteres que NÃO sã
 texto = re.sub(pattern_regex,' ',texto) # Substitui todos
os caracteres que não são alfanuméric
 #print(texto)
 caracteristica1=tamanhoMedioFrases(texto)
 caracteristica2=frequenciaCaracteres(texto)
 caracteristica3=frequenciaBigramas(texto)
 caracteristica4=frequenciaTrigramas(texto)
 caracteristica5=frequenciaAcentuacoes(texto)
 caracteristica6=quantidadeAcentuacoes(texto)
 # caracteristicaBagOfWords=bagOfWords(texto,lingua)
 # caracteristicaStopWords=stopWords(texto,lingua)
 # acrescente as suas funcoes no vetor padrao
 padrao = {
 'tamanhoMedioFrases': caracteristica1,
 **caracteristica2,  # O ** é um operador "Spread" de
dicionários. ele retorna todos os itens
**caracteristica3,
 **caracteristica4,
 **caracteristica5,
 'qntAcentuacoes': caracteristica6,
 # **caracteristicaBagOfWords,
 # **caracteristicaStopWords,
 'lingua': frase[1]
 }
 return padrao
 def geraPadroes(frases):
 padroes = []
 for frase in frases:
 padrao = extraiCaracteristicas(frase)
```

```
        padroes.append(padrao)
    return padroes
 # converte o formato [frase classe] em
 # [caracteristica_1, caracteristica_2,... caracteristica
n, classe]
 padroes = geraPadroes(pre_padroes)
 #
 # apenas para visualizacao
 # print(padroes)
 dados = pd.DataFrame(padroes)
 dados.fillna(0, inplace=True) # Substitui o que esta com
NaN para 0
 dados.drop('  ', axis=1, inplace=True) # Remove algum
espaço que tenha ficado
 # print(dict(dados.iloc[0]))
 print(dados.shape)
 dados
from sklearn.model_selection import train_test_split
 if SEED is not None: # Reseta o seed para evitar que de
algum valor diferente durante os testes
    np.random.seed(SEED)
 #from sklearn.metrics import confusion_matrix
 # vet = np.array(padroes)
 classes = np.array(dados['lingua']) #vet[:,-1]          #
classes = [p[-1] for p in padroes]
 # print(len(classes), classes)
 padroes_sem_classe   =      np.array(dados.drop('lingua',
axis=1)) #vet[:,0:-1]
 #print(padroes_sem_classe)
 X_train,       X_test,       y_train,       y_test       =
train_test_split(padroes_sem_classe,              classes,
test_size=0.25, st
 print(X_train.shape,     X_test.shape,     y_train.shape,
y_test.shape)
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))
```

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

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

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

2.

```
import requests, zipfile, io
from io import BytesIO
import numpy as np
import pandas as pd
from scipy.io import arff
import os
r =
requests.get('https://archive.ics.uci.edu/static/public/5
45/rice+cammeo+and+osmancik.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
zip_file_contents = z.namelist()
if 'Rice_Cammeo_Osmancik.arff' in zip_file_contents:
# File found, proceed with extraction
z.extract('Rice_Cammeo_Osmancik.arff')
```

```
 os.chdir(os.getcwd())     #  Change  to  the  directory
containing the extracted file
 dadosfp = arff.loadarff('Rice_Cammeo_Osmancik.arff')
 dados = pd.DataFrame(dadosfp[0])
 print(dados.head())
 print(dados.describe())
from sklearn.preprocessing import StandardScaler
 # Salva o dataframe para testar depois
 Y_orig = np.array(dados['Class'])
 X_orig =  np.array(dados.drop('Class', axis=1))
 scaler = StandardScaler()
 dados_normalizados = scaler.fit_transform(dados)
 dados          =          pd.DataFrame(dados_normalizados,
columns=dados.columns)
 dados['Class'] = Y_orig # Evita de normalizar o class
 # Exibir os dados normalizados
 dados.describe().loc[['mean',  'std']].round(6)  #  Para
facilitar na hora de debugar. O mean deve ser
from sklearn.utils import resample
 dados_cammeo = dados[dados['Class'] == 0]
 dados_osmancik = dados[dados['Class'] == 1]
 # Subamostragem da classe mais prevalente
 dados_osmancik_subamostrados = resample(dados_osmancik,
                                         replace=False,   #
Evita amostras duplicadas

n_samples=len(dados_cammeo),  # Numero de amostras da class
                                         random_state=42)
 # Concatenação dos dados
 dados_subamostrados                                     =
pd.concat([dados_osmancik_subamostrados, dados_cammeo])
 dados = dados_subamostrados # Aplica prevalencia
 dados_subamostrados.groupby('Class').count()
 from sklearn.model_selection import train_test_split
 import numpy as np
 Y = np.array(dados['Class'])
 X =  np.array(dados.drop('Class', axis=1)) #vet[:,0:-1]
 # com os dados originais
 X_oring_train,  X_orig_test,  y_orig_train,  y_orig_test  =
train_test_split(X_orig,
                      Y_orig,              test_size=0.25,
stratify=Y_orig,random_state=10)
 # com os dados tratados
 X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.25,

stratify=Y,random_state=10)
from sklearn import svm
 from sklearn.metrics import confusion_matrix
 from sklearn.metrics import classification_report
 treinador = svm.SVC()  #algoritmo escolhido
```

```
 modelo_orig = treinador.fit(X_oring_train, y_orig_train)
 # predição com os mesmos dados usados para treinar
 y_orig_pred = modelo_orig.predict(X_oring_train)
 cm_orig_train      =      confusion_matrix(y_orig_train,
y_orig_pred)
 print('Matriz de confusão - com os dados ORIGINAIS usados
no TREINAMENTO')
 print(cm_orig_train)
 print(classification_report(y_orig_train, y_orig_pred))
 # predição com os mesmos dados usados para testar
 print('Matriz de confusão - com os dados ORIGINAIS usados
para TESTES')
 y2_orig_pred = modelo_orig.predict(X_orig_test)
 cm_orig_test       =       confusion_matrix(y_orig_test,
y2_orig_pred)
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)
 # predição com os mesmos dados usados para treinar
 y_pred = modelo.predict(X_train)
 cm_train = confusion_matrix(y_train, y_pred)
 print('Matriz de confusão - com os dados TRATADOS usados
no TREINAMENTO')
 print(cm_train)
 print(classification_report(y_train, y_pred))
 # predição com os mesmos dados usados para testar
 print('Matriz de confusão - com os dados ORIGINAIS usados
para TESTES')
 y2_pred = modelo.predict(X_test)
 cm_test = confusion_matrix(y_test, y2_pred)
 print(cm_test)
 print(classification_report(y_test, y2_pred))
```

Selecionamos a base de dados "Rice (Cammeo and Osmancik)" da UCI *Machine Learning* por que ela era indicada para problemas de classificação e tinha grandes chances de poder ser aprimorada utilizando as técnicas de pré-processamento vistas nas aulas. Na parte da correção de prevalência, utilizamos a técnica da subamostragem para remover os dados extra da classe mais prevalente e manter as classes equilibradas. Por mais que a subamostragem reduza um pouco a acurácia, temos um modelo mais generalista e equilibrado nas predições.

Utilizamos um modelo SVM para fazer o treinamento e testes na base de dados. A acurácia do modelo de SVM aplicado sobre os dados originais foi de 88%.

Com os tratamentos realizados, observou-se melhora de 6% nessa medida de qualidade, chegando à uma acurácia de 93%. Com isso chegamos ao objetivo deste trabalho que era obter de 2% a 5% de acréscimo de acurácia a uma base de dados desbalanceada.

APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

```
# 1.a veículos (classificação) - Random Forest Hold Out
install.packages("e1071")
install.packages("caret")
library("caret")
setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-
aprendizado-maquina/bases-de-dados/06-
veículos")
data <- read.csv("6-veiculos.csv")
View(data)
data$a <- NULL
any(is.na(data))
# FALSE
preproc_center_scale <- preProcess(data, method =
c("center", "scale"))
normalized_data <- predict(preproc_center_scale, data)
# Dados normalizados com média centralizada em 0
View(normalized_data)
set.seed(202493)
ind <- createDataPartition(normalized_data$tipo, p = 0.8,
list = F)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]
# --- Hold out ---
set.seed(202493)
rf <- train(tipo ~ ., data = normalized_data, method = "rf")
rf
# mtry = 2
predict.rf <- predict(rf, test)
confusionMatrix(predict.rf, as.factor(test$tipo))
# Accuracy: 1
# --- Novos casos (usando Hold out) ----
new_data <- read.csv("6-veiculos-novos-dados.csv")
View(new_data)
new_data$a <- NULL
```

```
any(is.na(new_data))
# FALSE
preproc_center_scale <- preProcess(new_data, method =
c("center", "scale"))
normalized_new_data <- predict(preproc_center_scale,
new_data)
# Dados normalizados com média centralizada em 0
View(normalized_new_data)
predict.rf_new_data <- predict(rf, normalized_new_data)
# van bus opel
# Levels: bus opel saab van
new_data$tipo <- NULL
result <- cbind(new_data, predict.rf_new_data)
names(result)[names(result) == "predict.rf_new_data"] <-
"tipo"
View(result)
# Visualização do DF com os novos dados e a predição
```

QUADRO 16 - Comparação de modelos - base de veículos

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	bus: 43, 0, 0, 0 opel: 0, 42, 0, 0 saab: 0, 0, 43, 0 van: 0, 0, 0, 39
RNA – Hold-out	size=5 decay=0.1	85% (0,8503)	bus: 41, 0, 1, 2 opel: 0, 30, 7, 1 saab: 1, 9, 35, 0 van: 1, 3, 0, 36
SVM – Hold-out	C=1 Sigma=0.07189928	85% (0,8502)	bus: 43, 0, 0, 1 opel: 0, 26, 7, 0 saab: 0, 12, 35, 0 van: 0, 4, 1, 38
SVM – CV	C=100 Sigma=0.015	84% (0,8443)	bus: 40, 0, 0, 1 opel: 1, 34, 12, 2 saab: 0, 8, 31, 0 van: 2, 0, 0, 36

RNA – CV	size=11 decay=0.4	81% (0,8084)	bus: 39, 0, 1, 1 opel: 0, 27, 9, 1 saab: 1, 11, 32, 0 van: 3, 4, 1, 37
RF – CV	mtry=10	74% (0,7365)	bus: 42, 0, 1, 0 opel: 0, 18, 13, 2 saab: 0, 21, 27, 1 van: 1 ,3, 2, 36
KNN	k=1	68% (0,6766)	bus: 39, 1, 1, 3 opel: 1, 17, 16, 1 saab: 1, 20, 22, 0 van: 2, 4, 4, 35

FONTE: A Autora(2025).

QUADRO 17 - Novos casos - base de veículos

Comp	**Circ**	**Dcir**	**RadRa**	**PrAxisRa**	**MaxLRa**	**ScatRa**	**Elong**	**PrAxisRect**	**MaxLRRect**	**SeVarMaks**	**SeVarmAxis**	**RaGy**	**SkewMaxis**	**Skewmaks**	**KurtMaAxis**	**Kurtmaks**	**Hourl**	**Tipo**
75	55	89	105	133	36	117	60	24	140	130	806	204	70	3	28	192	190	van
89	42	56	241	124	7	200	34	28	148	200	450	261	92	1	29	184	208	bus
115	37	107	315	55	48	251	28	20	125	320	210	139	100	9	37	181	201	opel

FONTE: A Autora(2025).

```
# 1.b diabetes (classificação) - Random Forest – Hold Out
install.packages("e1071")
install.packages("caret")
library("caret")
setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-
aprendizado-maquina/bases-de-dados/10-
diabetes")
data <- read.csv("10-diabetes.csv")
View(data)
data$num <- NULL
any(is.na(data))
# FALSE
preproc_center_scale          <-          preProcess(data,
method=c("center", "scale"))
normalized_data <- predict(preproc_center_scale, data)
# Dados normalizados com média centralizada em 0
View(normalized_data)
set.seed(202493)
ind <- createDataPartition(normalized_data$diabetes, p =
0.8, list = FALSE)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]
# --- Hold out ---
set.seed(202493)
rf <- train(diabetes ~ ., data = normalized_data, method =
"rf")
rf
# mtry = 2
predict.rf <- predict(rf, test)
confusionMatrix(predict.rf, as.factor(test$diabetes))
# Accuracy: 1
# --- Novos casos (usando Hold out) ----
new_data <- read.csv("10-diabetes-novos-dados.csv")
View(new_data)
new_data$num <- NULL
any(is.na(new_data))
# FALSE
preproc_center_scale  <-  preProcess(new_data,  method  =
c("center", "scale"))
normalized_new_data    <-    predict(preproc_center_scale,
new_data)
# Dados normalizados com média centralizada em 0
View(normalized_new_data)
predict.rf_new_data <- predict(rf, normalized_new_data)
predict.rf_new_data
# pos neg neg
# Levels: neg pos
new_data$diabetes <- NULL
result <- cbind(new_data, predict.rf_new_data)
names(result)[names(result)  ==  "predict.rf_new_data"]  <-
"diabetes"
```

```
View(result)
# Visualização do DF com os novos dados e a predição
```

QUADRO 18 - Comparação de modelos - base de diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	neg: 100, 0 pos: 0, 53
SVM – Hold-out	C=0,25 Sigma=0,1258432	78% (0,7843)	neg: 93, 26 pos: 7, 27
RNA – Hold-out	size=1 decay=0,1	78% (0,7778)	neg: 90, 24 pos: 10, 29
RF – CV	mtry=2	77% (0,7712)	neg: 89, 24 pos: 11, 29
SVM – CV	C=2 Sigma=0,015	76% (0,7582)	neg: 92, 29 pos: 8, 24
RNA – CV	size=3 decay=0,1	76% (0,7581)	neg: 88, 25 pos: 12, 28
KNN	k=9	73% (0,7255)	neg: 84, 26 pos: 16, 27

FONTE: A Autora(2025).

QUADRO 19 - Novos casos - base de diabetes

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

FONTE: A Autora(2025).

```
# 2.a admissão (regressão) - Random Forest – Hold Out
install.packages("e1071")
install.packages("kernlab")
install.packages("caret")
install.packages("mice")
library("caret")
library(Metrics)
library(stats)
library(mice)
setwd("/Users/cassi/dev/_estudos/pos-iaa/IAA008-aprendizado-
maquina/bases-de-dados/09-
admissão")
data <- read.csv("9-admissao.csv")
View(data)
data$num <- NULL
any(is.na(data))
# FALSE
target_data <- data[["ChanceOfAdmit"]]
predictors <- data[, colnames(data) != "ChanceOfAdmit"]
preproc_center_scale <- preProcess(predictors,
method=c("center", "scale"))
normalized_predictors <- predict(preproc_center_scale,
predictors)
normalized_data <- cbind(normalized_predictors, target_data)
names(normalized_data)[names(normalized_data) == "target_data"]
<- "ChanceOfAdmit"
View(normalized_data)
set.seed(202493)
ind <- createDataPartition(normalized_data$ChanceOfAdmit, p =
0.8, list = FALSE)
train <- normalized_data[ind,]
test <- normalized_data[-ind,]
# --- Hold out ---
set.seed(202493)
rf_ho <- train(ChanceOfAdmit ~ ., data = normalized_data, method
= "rf")
rf_ho
# mtry = 2
predict.rf_ho <- predict(rf_ho, test)
r2 <- function(predicted, observed) {
return (1 - (sum((predicted - observed) ^ 2) / sum((observed -
mean(observed)) ^ 2)))
}
syx <- function(predicted, observed) {
n <- length(observed)
syx <- sqrt(sum((observed - predicted)^2) / (n - 2))
return(syx)
}
rmse(test$ChanceOfAdmit, predict.rf_ho)
# 0.0333386
r2(predict.rf_ho, test$ChanceOfAdmit)
```

```
# 0.9458273
syx(predict.rf_ho, test$ChanceOfAdmit)
# 0.03368409
cor(test$ChanceOfAdmit, predict.rf_ho) # Pearson (library stats)
# 0.9746234
mae(test$ChanceOfAdmit, predict.rf_ho)
# 0.02295854
# --- Novos casos (usando Hold out) ----
new_data <- read.csv("9-admissao-novos-dados.csv")
View(new_data)
new_data$num <- NULL
any(is.na(new_data))
# FALSE
new_target_data <- new_data[["ChanceOfAdmit"]]
new_predictors    <-    new_data[,    colnames(new_data)    !=
"ChanceOfAdmit"]
preproc_center_scale       <-       preProcess(new_predictors,
method=c("center", "scale"))
normalized_new_predictors   <-   predict(preproc_center_scale,
new_predictors)
normalized_new_data     <-     cbind(normalized_new_predictors,
new_target_data)
names(normalized_new_data)[names(normalized_new_data)        ==
"new_target_data"] <-
"ChanceOfAdmit"
# Dados normalizados com média centralizada em 0
View(normalized_new_data)
predict.rf_ho_new_data <- predict(rf_ho, normalized_new_data)
predict.rf_ho_new_data
# 1 2 3
# 0.6088426 0.7209769 0.7601318
new_data$ChanceOfAdmit <- NULL
result <- cbind(new_data, predict.rf_ho_new_data)
names(result)[names(result)  ==  "predict.rf_ho_new_data"]  <-
"ChanceOfAdmit"
View(result)
# Visualização do DF com os novos dados e a predição
# --- Geração do Gráfico de Resíduos com RF Hold Out e Dados de
teste ---
test_residuals  <-  ((test$ChanceOfAdmit  -  predict.rf_ho)  /
test$ChanceOfAdmit) * 100
plot(
predict.rf_ho,
test_residuals,
col = "blue",
pch = 20,
main = "Resíduos (%) - RF Hold Out (Dados teste)",
xlab = "ChanceOfAdmit (estimado)",
ylab = "Resíduo (%)",
ylim=c(-100, 100)
)
```

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

QUADRO 20 - Comparação de modelos - base de admissão

Técnica	Parâmetro	R²	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0,9458	0,03368	0,97462	0,03334	0,02296
RNA – Hold-out	size=41 decay=0,1	0,8341	0,05895	0,91351	0,058348	0,044075
RNA – CV	size=16 decay=0,1	0,8218	0,06110	0,90938	0,060474	0,048163
SVM – CV	C=50 Sigma=0,015	0,8209	0,61251	0,90913	0,60622	0,04388
RF – CV	mtry=2	0,8046	0,06398	0,89732	0,06332	0,04513
SVM – Hold-out	C=0,5 Sigma=0,1769097	0,8026	0,0643	0,89797	0,063643	0,045819
KNN	k=9	0,7883	0,06659	0,89068	0,065908	0,04751

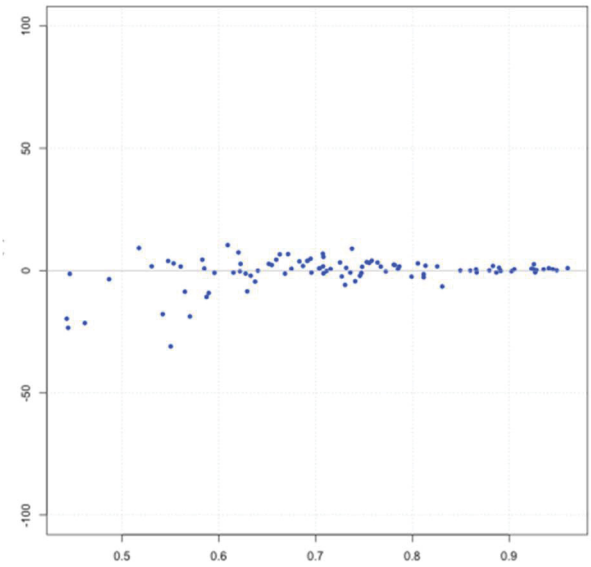
FONTE: A Autora(2025).

QUADRO 21 - Novos casos - base de admissão

num	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	ChanceOfAdmit
1	299	114	3	4	2	8,4	1	0,6088426
2	318	103	2	5	3	8,8	0	0,7209769
3	327	98	5	1	3	8,9	1	0,7601318

FONTE: A Autora(2025).

GRÁFICO 12 - Resíduos (%)



FONTE: A Autora(2025).

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

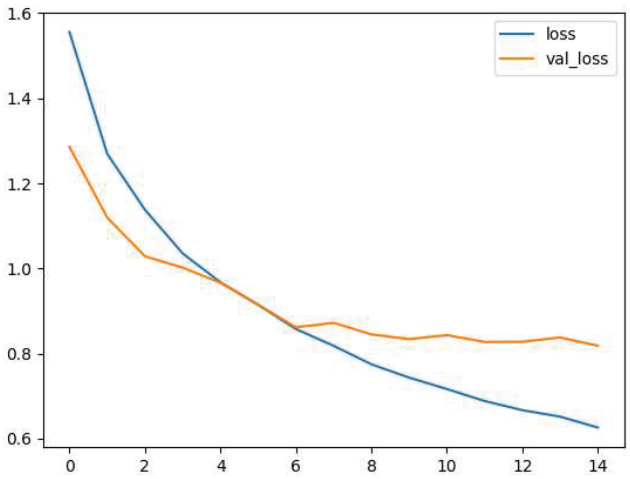
B – RESOLUÇÃO

1.

```
K = len(set(y_train))
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)
x = Flatten()(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation = 'relu')(x)
x = Dropout(0.2)(x)
x = Dense(K, activation = 'softmax')(x)
model = Model(i, x)
model.compile(optimizer = 'adam', loss = 
'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

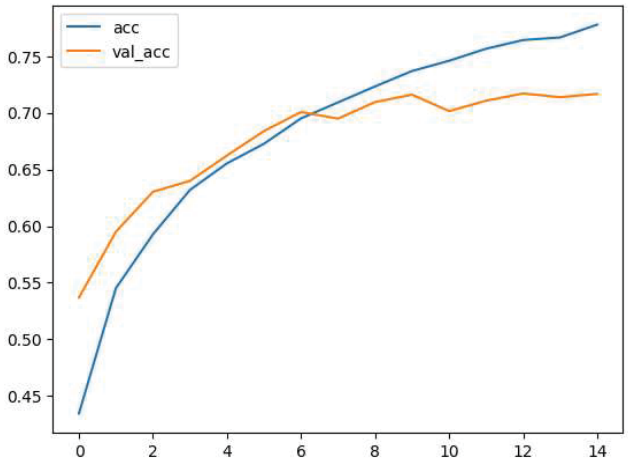
```
r = model.fit(x_train, y_train, validation_data = (x_test,
y_test), epochs = 15)
plt.plot(r.history['loss'], label = 'loss')
plt.plot(r.history['val_loss'], label = 'val_loss')
plt.legend()
plt.show()

plt.plot(r.history['accuracy'], label = 'acc')
plt.plot(r.history['val_accuracy'], label = 'val_acc')
plt.legend()
plt.show()
y_pred = model.predict(x_test).argmax(axis = 1)
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat = cm, figsize = (12,8),
show_normed= True)
```

GRÁFICO 13 - Perda por época modelo CNN



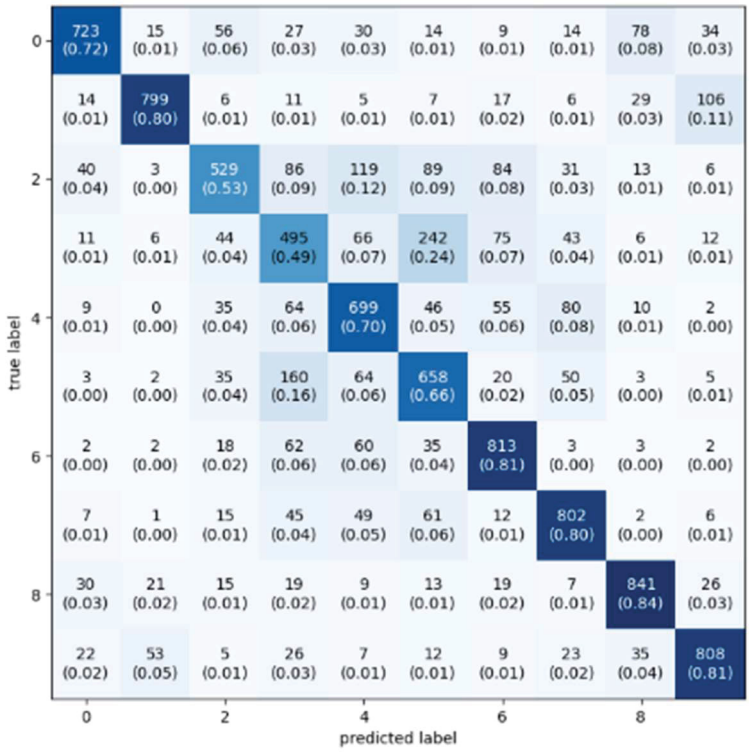
FONTE: A Autora (2025).

GRÁFICO 14 - Acurácia por época modelo CNN



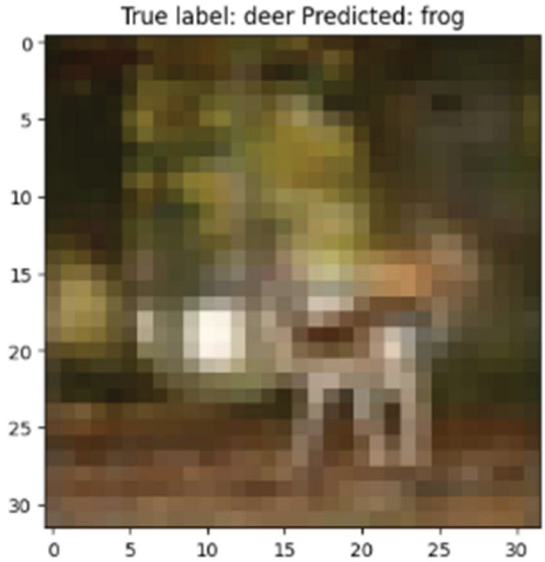
FONTE: A Autora (2025).

FIGURA 4 - Matriz de confusão modelo CNN



FONTE: A Autora(2025).

FIGURA 5 - Teste do modelo gerado - CNN



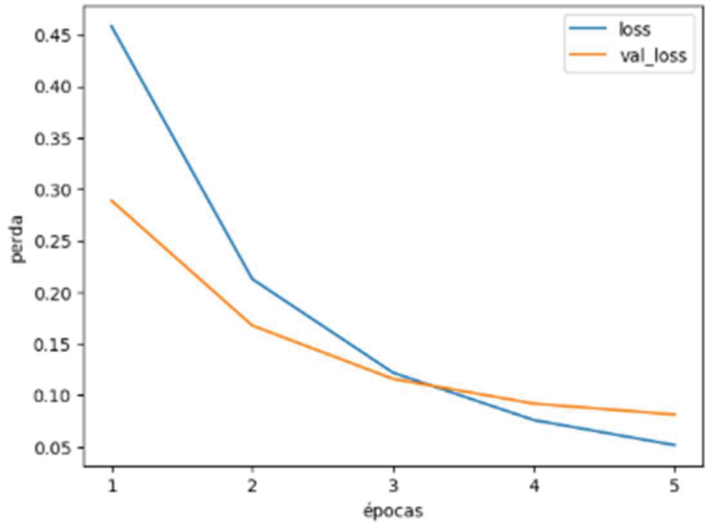
FONTE: A Autora(2025).

2.

```
!wget http://www.razer.net.br/datasets/spam.csv
df = pd.read_csv('spam.csv', encoding='ISO-8859-1')
df.head()
df = df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],
axis=1)
df.columns = ['labels', 'data']
df["b_labels"] = df["labels"].map({"ham": 0, "spam": 1})
y = df['b_labels'].values

x_train,       x_test,       y_train,       y_test       =
train_test_split(df['data'],       y,       test_size=0.33,
random_state=42)
num_words = 20000
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(x_train)
sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print('%s tokens' % V)

data_train = pad_sequences(sequences_train)
T = data_train.shape[1]
data_test = pad_sequences(sequences_test, maxlen=T)

print('data_train.shape = ',data_train.shape)
print('data_test.shape = ',data_test.shape)
D = 20
M = 5
i = Input(shape=(T,))
x = Embedding(V + 1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation='sigmoid')(x)
model = Model(i, x)
model.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))
plt.plot(r.history['loss'],label = 'loss')
plt.plot(r.history['val_loss'],label = 'val_loss')
plt.xlabel('épocas')
plt.ylabel('perda')
plt.xticks(np.arange(0,  epochs,  step  =  1),  labels  =
range(1, epochs +1))
plt.legend()
plt.show()
```

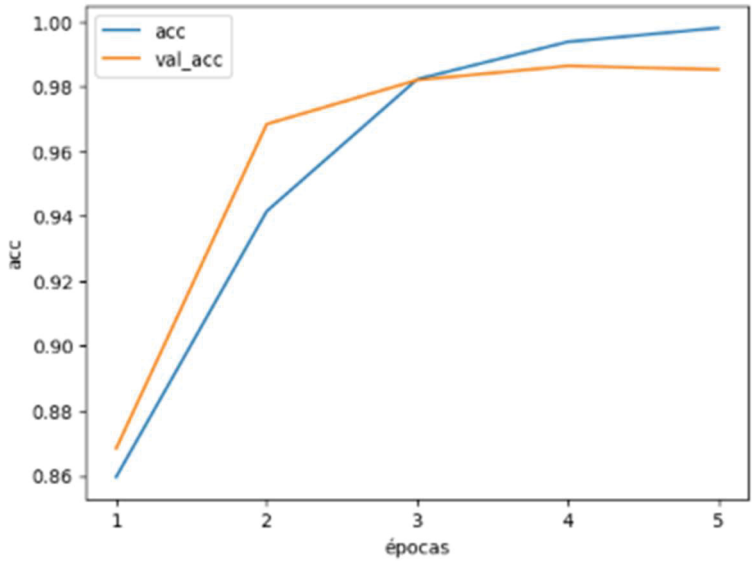
```
plt.plot(r.history['accuracy'],label = 'acc')
plt.plot(r.history['val_accuracy'],label = 'val_acc')
plt.xlabel('épocas')
plt.ylabel('acc')
plt.xticks(np.arange(0, epochs, step = 1), labels =
range(1, epochs +1))
plt.legend()
plt.show()
```

GRÁFICO 15 - Perda por época



FONTE: A Autora(2025).

GRÁFICO 16 - Acurácia por época



FONTE: A Autora(2025).

3.

```
!pip install imageio
!pip install git+https://github.com/tensorflow/docs
(train_imagens, train_labels), (_,_) =
tf.keras.datasets.mnist.load_data()
train_images =
train_imagens.reshape(train_imagens.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5
#normalizando

BUFFER_SIZE = 60000
BATCH_SIZE = 256

train_dataset =
tf.data.Dataset.from_tensor_slices(train_images).shuffle(B
UFFER_SIZE).batch(BATCH_SIZE)

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)
  model.add(layers.Conv2DTranspose(128, (5, 5), strides=
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

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

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

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

discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print(decision)
cross_entropy =
tf.keras.losses.BinaryCrossentropy(from_logits=True)

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


def generator_loss(fake_output):
  return cross_entropy(tf.ones_like(fake_output),
fake_output)

generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint =
tf.train.Checkpoint(generator_optimizer=generator_optimize
r,

discriminator_optimizer=discriminator_optimizer,
                                  generator=generator,

discriminator=discriminator)
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

seed = tf.random.normal([num_examples_to_generate,
noise_dim])
#treinamento
@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_gener
ator, generator.trainable_variables))

discriminator_optimizer.apply_gradients(zip(gradients_of_d
iscriminator, discriminator.trainable_variables))
def train(dataset, epochs):
  for epoch in range(epochs):
    start = time.time()
    for image_batch in dataset:
      train_step(image_batch)
    display.clear_output(wait=True)
    generate_and_save_images(generator, epoch+1, seed)

    if (epoch+1)%15 == 0:
      checkpoint.save(file_prefix = checkpoint_prefix)

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

  display.clear_output(wait=True)
  generate_and_save_images(generator, epochs, seed)

def generate_and_save_images(model,epoch,test_input):
  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))
train(train_dataset, EPOCHS)
```

```
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_d
ir))
def display_image(epoch_no):
  return
PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no
))

display_image(EPOCHS)
anim_file = 'dcgan.gif'

with imageio.get_writer(anim_file,model="I") as writer:
  filenames = glob.glob('image*.png')
  filenames = sorted(filenames)
  for filename in filenames:
    image = imageio.imread(filename)
    writer.append_data(image)
    image = imageio.imread(filename)
    writer.append_data(image)

import tensorflow_docs.vis.embed as embed
embed.embed_file(anim_file)
```

4.

```
examples, metadata = tfds.load(
    'ted_hrlr_translate/pt_to_en',
    with_info=True,
    as_supervised=True
)
train_examples, val_examples = examples['train'],
examples['validation']

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

  print()

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

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/m
odels/{model_name}.zip",
    cache_dir='.',
    cache_subdir='',
```

```
    extract=True
)

tokenizers = tf.saved_model.load(model_name)

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

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

BUFFER_SIZE = 20000
BATCH_SIZE = 64

def make_batches(ds):
  return (
      ds
      .cache()
      .shuffle(BUFFER_SIZE)
      .batch(BATCH_SIZE)
      .map(tokenize_pairs,
num_parallel_calls=tf.data.AUTOTUNE)
      .prefetch(tf.data.AUTOTUNE)
  )

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

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

def positional_encoding(position, d_model):
  angle_rads = get_angles(
      np.arange(position)[:, np.newaxis],
      np.arange(d_model)[np.newaxis, :],
      d_model
  )
  angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
  angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])

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

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

```
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))

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

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

def create_look_ahead_mask(size):
  mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -
1, 0)
  return mask

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

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

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

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

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

    assert d_model % self.num_heads == 0

    self.depth = d_model // self.num_heads

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

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

  def split_heads(self, x, batch_size):
    x = tf.reshape(x, (batch_size, -1, self.num_heads,
self.depth))
    return tf.transpose(x, perm=[0, 2, 1, 3])

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

    q = self.wq(q)
    k = self.wk(k)
    v = self.wv(v)

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

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

    output = self.dense(concat_attention)

    return output, attention_weights

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

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

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

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

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

```
  def call(self, x, training, mask):
    attn_output, _ = self.mha(x, x, x, mask)
    attn_output = self.dropout1(attn_output, training =
training)
    out1 = self.layernorm1(x + attn_output)

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

    return out2

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

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

    self.ffn = point_wise_feed_forward_network(d_model,
dff)

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

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

  def call(self, x, enc_output, training, look_ahead_mask,
padding_mask):
    attn1, attn_weights_block1 = self.mha1(x, x, x,
look_ahead_mask)
    attn1 = self.dropout1(attn1, training = training)
    out1 = self.layernorm1(attn1 + x)

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

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

```
    return out3, attn_weights_block1, attn_weights_block2

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

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

  def call(self, x, training, mask):
    seq_len = tf.shape(x)[1]
    x = self.embedding(x)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x = self.enc_layers[i](x, training, mask)
    return x
class Decoder(tf.keras.layers.Layer):
  def __init__(
      self,
      num_layers,
      d_model,
      num_heads,
      dff,
      target_vocab_size,
      maximum_position_encoding,
      rate = 0.1):
    super().__init__()
    self.d_model = d_model
    self.num_layers = num_layers
```

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

  def call(self, x, enc_output, training, look_ahead_mask,
padding_mask):
    seq_len = tf.shape(x)[1]
    attention_weights = {}

    x = self.embedding(x)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]

    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x, block1, block2 = self.dec_layers[i](
          x,
          enc_output,
          training,
          look_ahead_mask,
          padding_mask
      )
      attention_weights[f'decoder_layer{i+1}_block1'] =
block1
      attention_weights[f'decoder_layer{i+1}_block2'] =
block2

    return x, attention_weights

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

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

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

        return final_output, attention_weights

    def create_masks(self, inp, tar):
        enc_padding_mask = create_padding_mask(inp)
        dec_padding_mask = create_padding_mask(inp)
        look_ahead_mask =
create_look_ahead_mask(tf.shape(tar)[1])
        dec_target_padding_mask = create_padding_mask(tar)
        look_ahead_mask = tf.maximum(dec_target_padding_mask,
look_ahead_mask)

        return enc_padding_mask, look_ahead_mask,
dec_padding_mask

num_layers = 4
```

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

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

  def __call__(self, step):
    step = tf.cast(step, tf.float32)
    arg1 = tf.math.rsqrt(step)
    arg2 = step * (self.warmup_steps ** -1.5)

    return tf.math.rsqrt(self.d_model) *
tf.math.minimum(arg1, arg2)

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

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

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

  return tf.reduce_sum(loss_) / tf.reduce_sum(mask)

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

```
  return tf.reduce_sum(accuracies) / tf.reduce_sum(mask)

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

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

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 ckpt_manager.latest_checkpoint:
  ckpt.restore(ckpt_manager.latest_checkpoint)
  print('Latest checkpoint restored!')

EPOCHS = 25
train_step_signature = [
    tf.TensorSpec(shape = (None, None), dtype = tf.int64),
    tf.TensorSpec(shape = (None, None), dtype=tf.int64)
]

@tf.function(input_signature = train_step_signature)
def train_step(inp, tar):
  tar_inp = tar[:, :-1]
  tar_real = tar[:, 1:]

  with tf.GradientTape() as tape:
    predictions, _ = transformer([inp, tar_inp], training
= True)
    loss = loss_function(tar_real, predictions)
  gradients = tape.gradient(loss,
transformer.trainable_variables)
  optimizer.apply_gradients(zip(gradients,
transformer.trainable_variables))
  train_loss(loss)
  train_accuracy(accuracy_function(tar_real, predictions))
```

```
for epoch in range(EPOCHS):
  start = time.time()
  train_loss.reset_state()
  train_accuracy.reset_state()
  epoch_count = epoch + 1

  for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)

    if batch % 50 == 0:
      print(f"Epoch {epoch + 1} Batch {batch} Loss
{train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}")

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

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

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

  def __call__(self, sentence, max_length = 20):
    assert isinstance(sentence, tf.Tensor)
    if len(sentence.shape) == 0:
      sentence = sentence[tf.newaxis]
    sentence =
self.tokenizers.pt.tokenize(sentence).to_tensor()
    encoder_input = sentence

    start_end = self.tokenizers.en.tokenize([''])[0]
    start = start_end[0][tf.newaxis]
    end = start_end[1][tf.newaxis]

    output_array = tf.TensorArray(dtype = tf.int64, size =
0, dynamic_size = True)
    output_array = output_array.write(0, start)

    for i in tf.range(max_length):

      output = tf.transpose(output_array.stack())
```

```
      predictions, _ = self.transformer([encoder_input,
output], training=False)
      predictions = predictions[:, -1:, :]
      predicted_id = tf.argmax(predictions, axis = -1)
      output_array = output_array.write(i + 1,
predicted_id[0])

      if predicted_id == end:
        break

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

    return text, tokens, attention_weights

translator = Translator(tokenizers, transformer)

sentence = "vamos testar o tradutor."

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

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

APÊNDICE I - BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

RESUMO

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

Palavras-chave: *Big Data. NoSQL. NewSQL*. MongoDB. Modelagem de Dados.

2.1 ABSTRACT

The growing generation and complexity of data drive the need for new approaches to Big Data management. This paper explores the implementation of an e-commerce application using NoSQL and NewSQL technologies to optimize the storage and processing of large data volumes. The case study presents the use of MongoDB for semi-structured data and logs, Cassandra for transaction data, and CockroachDB for structured data and ACID transactions. The analysis highlights the characteristics of each technology, the necessary modeling, and effectiveness in

different data scenarios. The combination of these tools enables a robust, scalable, and efficient solution, tailored to the specific needs of the e-commerce application.

Keywords: Big Data. NoSQL. NewSQL. MongoDB. Data Modeling.

1 INTRODUÇÃO

Com o crescimento exponencial dos dados gerados por empresas e usuários, as soluções tradicionais de banco de dados relacional se tornaram insuficientes para atender às demandas de escalabilidade, desempenho e flexibilidade. Este documento explora a implementação de uma aplicação de *Big Data*, focando em como ferramentas *NoSQL e NewSQL* podem ser usadas para gerenciar grandes volumes de dados. O estudo de caso apresentado envolve uma plataforma de e-commerce que utiliza essas tecnologias para melhorar sua eficiência e experiência do usuário.

2 CARACTERIZAÇÃO DOS DADOS

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

- Dados de Transações: Informações sobre compras, pagamentos e devoluções.

Dados de Usuários: Perfis de clientes, histórico de navegação e preferências.
Dados de Produtos: Detalhes dos produtos, categorias e avaliações.
Dados de Logs: Registros de atividades dos usuários e do sistema.

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

3 FERRAMENTAS UTILIZADAS

1. NoSQL

1.1. MongoDB

- Modelo de Dados: Documentos JSON.

- Características: Alta escalabilidade e flexibilidade. Ideal para dados semiestruturados e não-estruturados, como logs de atividades e perfis de usuários.
- Modelagem: Os dados de usuários e produtos são armazenados em coleções de documentos. Isso permite consultas rápidas e escalabilidade horizontal.

1.2. Cassandra

- Modelo de Dados: Colunas.
- Características: Alta disponibilidade e desempenho para grandes volumes de dados. Adequado para dados de transações e *logs*, onde a escrita e leitura rápida são essenciais.
- Modelagem: Os dados de transações são modelados como linhas em uma tabela de colunas, permitindo consultas rápidas e eficientes.

2. NewSQL

2.1. CockroachDB

- Modelo de Dados: Relacional com suporte a SQL.
- Características: Combina a escalabilidade horizontal dos bancos *NoSQL* com a consistência e a robustez dos bancos de dados relacionais.
- Modelagem: Os dados de produtos e transações são armazenados em tabelas relacionais, garantindo consistência e integridade referencial, enquanto suportam grandes volumes e alta concorrência.

4 MODELAGEM DE DADOS

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

1. *NoSQL* (MongoDB e Cassandra):

- Modelagem de Documentos (MongoDB): Os dados são armazenados em documentos JSON, permitindo a inclusão de campos aninhados e flexíveis, o que é ideal para perfis de usuários e logs de atividades.
- Modelagem de Colunas (Cassandra): As tabelas são desenhadas para suportar grandes volumes de dados com alta taxa de escrita, como as transações de e-

commerce.

2. *NewSQL* (CockroachDB):

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

2.2 5 CONSIDERAÇÕES FINAIS

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

2.3 REFERÊNCIAS

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

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

1a.

```
#lendo bases de treino
path = '/content/drive/MyDrive/Pós IAA -
UFPR/IAA_VISAO_COMPUTACIONAL/base_treino/'

# Criar um dicionário para armazenar imagens por paciente e
suas classes
dados_pacientes = {}

# Percorrer cada diretório (classe)
for classe in os.listdir(path):
    classe_path = os.path.join(path, classe)

    if os.path.isdir(classe_path):  # Verificar se é um
diretório
        for img in os.listdir(classe_path):
            if img.endswith('.png'):
                paciente = img.split('_')[0]  # XX é o número
do paciente

                if paciente not in dados_pacientes:
                    dados_pacientes[paciente] = {"classe":
classe, "imagens": []}

                dados_pacientes[paciente]["imagens"].append(os
.path.join(classe_path, img))

print("Total de pacientes carregados:", len(dados_pacientes))
```

1b.

```
# Criar listas de pacientes e suas respectivas classes
pacientes = list(dados_pacientes.keys())
classes = [dados_pacientes[p]["classe"] for p in pacientes]  #
Pegamos a classe de cada paciente
```

```
# Dividir os pacientes garantindo que todas as classes
apareçam nos dois conjuntos
pacientes_treino, pacientes_validacao = train_test_split(
    pacientes, test_size=0.2, stratify=classes,
random_state=42
)

# Criar os conjuntos de treino e validação
dados_treino = {p: dados_pacientes[p] for p in
pacientes_treino}
dados_validacao = {p: dados_pacientes[p] for p in
pacientes_validacao}

print("Pacientes no treino:", len(dados_treino))
print("Pacientes na validação:", len(dados_validacao))
```

1c.

```
#extração de características com lbp

# Parâmetros do LBP
P = 8  # Número de vizinhos
R = 1  # Raio

def extrair_lbp(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    lbp = local_binary_pattern(img, P, R, method="uniform")

    # Criar histograma
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0,
P+3), range=(0, P+2))
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-6)  # Normalização

    return hist

# Criar CSV de características LBP
dados_lbp = []

for paciente, info in dados_treino.items():
    for img_path in info["imagens"]:
        hist_lbp = extrair_lbp(img_path)
        classe = info["classe"]
        dados_lbp.append([paciente] + hist_lbp.tolist() +
[classe])

# Salvar CSV no Google Drive
df_lbp = pd.DataFrame(dados_lbp)
```

```
df_lbp.to_csv('/content/drive/MyDrive/LBP_features.csv',
index=False)

print("Extração de LBP concluída e salva em CSV.")


#extração de características com vgg16

# Carregar a VGG16 sem a última camada
base_model = VGG16(weights="imagenet", include_top=False,
pooling="avg")
model = Model(inputs=base_model.input,
outputs=base_model.output)

def extrair_vgg16(image_path):
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)

    features = model.predict(img_array)
    return features.flatten()

# Criar CSV de características VGG16
dados_vgg16 = []

for paciente, info in dados_treino.items():
    for img_path in info["imagens"]:
        features_vgg16 = extrair_vgg16(img_path)
        classe = info["classe"]
        dados_vgg16.append([paciente] +
features_vgg16.tolist() + [classe])

# Salvar CSV no Google Drive
df_vgg16 = pd.DataFrame(dados_vgg16)
df_vgg16.to_csv('/content/drive/MyDrive/VGG16_features.csv',
index=False)

print("Extração de VGG16 concluída e salva em CSV.")
```

1d.

```
#treinamento dos modelos

# Carregar as features extraídas do CSV
df_lbp =
pd.read_csv('/content/drive/MyDrive/LBP_features.csv')
df_vgg16 =
pd.read_csv('/content/drive/MyDrive/VGG16_features.csv')
```

```
# Separar features e labels
X_lbp = df_lbp.iloc[:, 1:-1].values  # Pega todas as colunas
exceto paciente e classe
y_lbp = df_lbp.iloc[:, -1].values  # Última coluna é a classe

X_vgg16 = df_vgg16.iloc[:, 1:-1].values
y_vgg16 = df_vgg16.iloc[:, -1].values

# Codificar as labels
encoder = LabelEncoder()
y_lbp = encoder.fit_transform(y_lbp)
y_vgg16 = encoder.transform(y_vgg16)

# Dividir entre treino e validação (80/20)
X_treino_lbp, X_validacao_lbp, y_treino_lbp, y_validacao_lbp =
train_test_split(
    X_lbp, y_lbp, test_size=0.2, random_state=42,
stratify=y_lbp
)

X_treino_vgg16, X_validacao_vgg16, y_treino_vgg16,
y_validacao_vgg16 = train_test_split(
    X_vgg16, y_vgg16, test_size=0.2, random_state=42,
stratify=y_vgg16
)

# Função para treinar os modelos
def treinar_modelos(X_train, y_train):
    rf = RandomForestClassifier(n_estimators=100,
random_state=42)
    svm = SVC(kernel="linear", probability=True,
random_state=42)
    rna = MLPClassifier(hidden_layer_sizes=(100,),
max_iter=500, random_state=42)

    rf.fit(X_train, y_train)
    svm.fit(X_train, y_train)
    rna.fit(X_train, y_train)

    return rf, svm, rna

# Treinar modelos
rf_lbp, svm_lbp, rna_lbp = treinar_modelos(X_treino_lbp,
y_treino_lbp)
rf_vgg16, svm_vgg16, rna_vgg16 =
treinar_modelos(X_treino_vgg16, y_treino_vgg16)

print("Treinamento concluído.")


#avaliação de métricas dos modelos
```

```
def avaliar_modelo(modelo, X_test, y_test, nome_modelo):
    y_pred = modelo.predict(X_test)
    matriz_confusao = confusion_matrix(y_test, y_pred)
    relatorio = classification_report(y_test, y_pred)

    print(f" ◇ Modelo: {nome_modelo}")
    print("Matriz de Confusão:\n", matriz_confusao)
    print("Relatório de Classificação:\n", relatorio)
    print("-" * 50)

# Avaliar os modelos treinados
avaliar_modelo(rf_lbp, X_validacao_lbp, y_validacao_lbp,
"Random Forest (LBP)")
avaliar_modelo(svm_lbp, X_validacao_lbp, y_validacao_lbp, "SVM
(LBP)")
avaliar_modelo(rna_lbp, X_validacao_lbp, y_validacao_lbp, "RNA
(LBP)")

avaliar_modelo(rf_vgg16, X_validacao_vgg16, y_validacao_vgg16,
"Random Forest (VGG16)")
avaliar_modelo(svm_vgg16, X_validacao_vgg16,
y_validacao_vgg16, "SVM (VGG16)")
avaliar_modelo(rna_vgg16, X_validacao_vgg16,
y_validacao_vgg16, "RNA (VGG16)")


# Comparar o desempenho
modelos = ["RF (LBP)", "SVM (LBP)", "RNA (LBP)", "RF (VGG16)",
"SVM (VGG16)", "RNA (VGG16)"]
acuracias = [
    rf_lbp.score(X_validacao_lbp, y_validacao_lbp),
    svm_lbp.score(X_validacao_lbp, y_validacao_lbp),
    rna_lbp.score(X_validacao_lbp, y_validacao_lbp),
    rf_vgg16.score(X_validacao_vgg16, y_validacao_vgg16),
    svm_vgg16.score(X_validacao_vgg16, y_validacao_vgg16),
    rna_vgg16.score(X_validacao_vgg16, y_validacao_vgg16)
]

for modelo, acc in zip(modelos, acuracias):
    print(f"{modelo}: {acc:.4f}")
```

RF (LBP): 0,8421
SVM (LBP): 0,4000
RNA (LBP): 0,5158
RF (VGG16): 0,9579
SVM (VGG16): 0,9789
RNA (VGG16): 0,9895

1e.

```
import os

# Caminho para a base de teste
path_teste = '/content/drive/MyDrive/Pós IAA -
UFPR/IAA_VISAO_COMPUTACIONAL/base_teste/'

# Listas para armazenar caminhos das imagens e seus
respectivos labels
imagens_teste = []
labels_teste = []


for class_dir in os.listdir(path_teste):
    class_path = os.path.join(path_teste, class_dir)

    if os.path.isdir(class_path):  # Verifica se é um
diretório
        for img_name in os.listdir(class_path):
            if img_name.endswith('.png'):
                img_path = os.path.join(class_path, img_name)

                # Adiciona a imagem e seu label correspondente
                imagens_teste.append(img_path)
                labels_teste.append(class_dir)

print("Total de imagens de teste carregadas:",
len(imagens_teste))
print("Total de labels carregadas:", len(labels_teste))
print("Exemplo de mapeamento imagem-label:",
list(zip(imagens_teste[:5], labels_teste[:5])))


import os
import cv2
import numpy as np
import pandas as pd
from skimage.feature import local_binary_pattern

# Parâmetros do LBP
P = 8  # Número de vizinhos
R = 1  # Raio

def extrair_lbp(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    lbp = local_binary_pattern(img, P, R, method="uniform")

    # Criar histograma
    hist, _ = np.histogram(lbp.ravel(), bins=np.arange(0, P +
3), range=(0, P + 2))
```

```
    hist = hist.astype("float")
    hist /= (hist.sum() + 1e-6)  # Normalização

    return hist

# Lista para armazenar os dados LBP do teste
dados_lbp_teste = []

# Loop pelas imagens e labels carregados anteriormente
for img_path, classe in zip(imagens_teste, labels_teste):
    hist_lbp = extrair_lbp(img_path)
    nome_img = os.path.basename(img_path)
    dados_lbp_teste.append([nome_img] + hist_lbp.tolist() +
[classe])

# Criar DataFrame e salvar em CSV
df_lbp_teste = pd.DataFrame(dados_lbp_teste)

# Cria nomes de colunas (ex: 'LBP_0', 'LBP_1', ...)
colunas = ['imagem'] + [f'LBP_{i}' for i in
range(len(df_lbp_teste.columns) - 2)] + ['classe']
df_lbp_teste.columns = colunas

# Salvar no Google Drive
df_lbp_teste.to_csv('/content/drive/MyDrive/LBP_features_teste
.csv', index=False)

print("Extração de LBP concluída e salva em CSV para os dados
de teste.")
print("Total de imagens processadas:", len(df_lbp_teste))


import os
import numpy as np
import pandas as pd
import cv2
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.models import Model
from keras.preprocessing import image

# Carregar a VGG16 sem a última camada (extração de features)
base_model = VGG16(weights="imagenet", include_top=False,
pooling="avg")
model = Model(inputs=base_model.input,
outputs=base_model.output)

def extrair_vgg16(image_path):
    """Extrai o vetor de características da VGG16 para uma
imagem."""
    img = image.load_img(image_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
```

```
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)

    features = model.predict(img_array, verbose=0)
    return features.flatten()

# Lista para armazenar os dados VGG16 do teste
dados_vgg16_teste = []

# Loop pelas imagens e labels carregados anteriormente
for img_path, classe in zip(imagens_teste, labels_teste):
    features_vgg16 = extrair_vgg16(img_path)
    nome_img = os.path.basename(img_path)
    dados_vgg16_teste.append([nome_img] +
features_vgg16.tolist() + [classe])

# Criar DataFrame
df_vgg16_teste = pd.DataFrame(dados_vgg16_teste)

# Nomear as colunas (ex: VGG16_0 ... VGG16_n)
colunas = ['imagem'] + [f'VGG16_{i}' for i in
range(len(df_vgg16_teste.columns) - 2)] + ['classe']
df_vgg16_teste.columns = colunas

# Salvar CSV no Google Drive
df_vgg16_teste.to_csv('/content/drive/MyDrive/VGG16_features_t
este.csv', index=False)

print("Extração de VGG16 concluída e salva em CSV para os
dados de teste.")
print("Total de imagens processadas:", len(df_vgg16_teste))

import os
import numpy as np
import pandas as pd
from sklearn.metrics import classification_report,
confusion_matrix

# Caminho dos arquivos CSV
caminho_saida = "/content/drive/MyDrive/"
csv_teste_lbp = os.path.join(caminho_saida,
"LBP_features_teste.csv")
csv_teste_vgg16 = os.path.join(caminho_saida,
"VGG16_features_teste.csv")

# Carregar os CSVs
df_teste_lbp = pd.read_csv(csv_teste_lbp)
df_teste_vgg16 = pd.read_csv(csv_teste_vgg16)

# Separar labels e features
y_teste = df_teste_lbp["classe"].values  # rótulos
```

```
X_teste_lbp = df_teste_lbp.drop(columns=["imagem",
"classe"]).values
X_teste_vgg16 = df_teste_vgg16.drop(columns=["imagem",
"classe"]).values

# Codificar labels usando o mesmo encoder dos dados de treino
y_teste_encoded = encoder.transform(y_teste)

# ---- Aplicar os modelos treinados nos dados de teste ----

# LBP
y_pred_rf_lbp = rf_lbp.predict(X_teste_lbp)
y_pred_svm_lbp = svm_lbp.predict(X_teste_lbp)
y_pred_rna_lbp = rna_lbp.predict(X_teste_lbp)

# VGG16
y_pred_rf_vgg16 = rf_vgg16.predict(X_teste_vgg16)
y_pred_svm_vgg16 = svm_vgg16.predict(X_teste_vgg16)
y_pred_rna_vgg16 = rna_vgg16.predict(X_teste_vgg16)

# ---- Função para avaliação dos modelos ----
def avaliar_modelo(y_teste, y_pred, nome_modelo):
    print(f"\n ◇  Avaliação do Modelo: {nome_modelo}")
    print("Matriz de Confusão:")
    print(confusion_matrix(y_teste, y_pred))
    print("Relatório de Classificação:")
    print(classification_report(y_teste, y_pred))
    print("-" * 50)

# Avaliar os modelos
avaliar_modelo(y_teste_encoded, y_pred_rf_lbp, "Random Forest
LBP")
avaliar_modelo(y_teste_encoded, y_pred_svm_lbp, "SVM LBP")
avaliar_modelo(y_teste_encoded, y_pred_rna_lbp, "RNA LBP")

avaliar_modelo(y_teste_encoded, y_pred_rf_vgg16, "Random
Forest VGG16")
avaliar_modelo(y_teste_encoded, y_pred_svm_vgg16, "SVM VGG16")
avaliar_modelo(y_teste_encoded, y_pred_rna_vgg16, "RNA VGG16")
```

1h.

Utilizando a métrica de acurácia para comparação, o modelo que apresenta melhor resultado é o RNA VGG16.

2a.

```
import os
import numpy as np
```

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator, load_img, img_to_array
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split

# Função para carregar as imagens e rótulos
def carregar_imagens(dados_pacientes):
    imagens = []
    rotulos = []

    for paciente, dados in dados_pacientes.items():
        for img_path in dados["imagens"]:
            img = load_img(img_path, target_size=(224,
224))  # Redimensionar para 224x224
            img_array = img_to_array(img)  # Converter imagem
para array numpy
            imagens.append(img_array)
            rotulos.append(dados["classe"])  # Armazenar o
rótulo (classe)

    imagens = np.array(imagens)  # Converter a lista de
imagens para um array numpy
    rotulos = np.array(rotulos)  # Converter os rótulos para
um array numpy

    return imagens, rotulos

# Carregar imagens e rótulos para treino e validação
X_treino, y_treino = carregar_imagens(dados_treino)
X_validacao, y_validacao = carregar_imagens(dados_validacao)

# Normalizar as imagens (dividir por 255 para ficar entre 0 e
1)
X_treino = X_treino / 255.0
X_validacao = X_validacao / 255.0

# Codificar as classes com LabelEncoder e converter para one-
hot encoding
label_encoder = LabelEncoder()
y_treino_encoded = label_encoder.fit_transform(y_treino)
y_validacao_encoded = label_encoder.transform(y_validacao)

y_treino_one_hot = to_categorical(y_treino_encoded,
num_classes=4)  # Assumindo 4 classes
y_validacao_one_hot = to_categorical(y_validacao_encoded,
num_classes=4)

print("Formato de X_treino:", X_treino.shape)
print("Formato de X_validacao:", X_validacao.shape)
```

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.applications.vgg16 import
preprocess_input

# Definindo o Data Augmentation para o treino
datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,  #adiciona o pré-
processamento da VGG16
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    # zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# Geradores de treino e validação
train_generator = datagen.flow(X_treino, y_treino_one_hot,
batch_size=32)
val_generator = datagen.flow(X_validacao, y_validacao_one_hot,
batch_size=32)
```

2b.

```
from tensorflow.keras.applications import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense,
GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam

# Carregar o modelo VGG16 com pesos pré-treinados, sem as
camadas superiores
base_model_vgg = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

# Congelar as camadas do modelo pré-treinado
for layer in base_model_vgg.layers:
    layer.trainable = False

# Adicionar novas camadas totalmente conectadas para a
classificação de 4 classes
x = GlobalAveragePooling2D()(base_model_vgg.output)
x = Dense(1024, activation='relu')(x)
x = Dense(4, activation='softmax')(x)  # 4 classes

# Criar o modelo final
model_vgg = Model(inputs=base_model_vgg.input, outputs=x)
```

```
model_vgg.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])

# Treinar o modelo VGG16
history_vgg = model_vgg.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)



from tensorflow.keras.applications import ResNet50

# Carregar o modelo ResNet50 com pesos pré-treinados, sem as
camadas superiores
base_model_resnet = ResNet50(weights='imagenet',
include_top=False, input_shape=(224, 224, 3))

# Congelar as camadas do modelo pré-treinado
for layer in base_model_resnet.layers:
    layer.trainable = False

# Adicionar novas camadas totalmente conectadas para a
classificação de 4 classes
x = GlobalAveragePooling2D()(base_model_resnet.output)
x = Dense(1024, activation='relu')(x)
x = Dense(4, activation='softmax')(x)  # 4 classes

# Criar o modelo final
model_resnet = Model(inputs=base_model_resnet.input,
outputs=x)
model_resnet.compile(optimizer=Adam(),
loss='categorical_crossentropy', metrics=['accuracy'])

# Treinar o modelo ResNet50
history_resnet = model_resnet.fit(
    train_generator,
    epochs=10,
    validation_data=val_generator
)
```

2c.

```
from sklearn.metrics import confusion_matrix,
classification_report

# Fazer previsões no conjunto de validação
y_pred_vgg = model_vgg.predict(X_validacao)
y_pred_resnet = model_resnet.predict(X_validacao)
```

```
# Converter previsões para rótulos (classe com maior
probabilidade)
y_pred_vgg_classes = np.argmax(y_pred_vgg, axis=1)
y_pred_resnet_classes = np.argmax(y_pred_resnet, axis=1)

# Converter os rótulos verdadeiros para valores numéricos
y_true_classes = np.argmax(y_validacao_one_hot, axis=1)


# Matriz de confusão para VGG16
conf_matrix_vgg = confusion_matrix(y_true_classes,
y_pred_vgg_classes)
print("Matriz de Confusão - VGG16:\n", conf_matrix_vgg)

# Matriz de confusão para ResNet50
conf_matrix_resnet = confusion_matrix(y_true_classes,
y_pred_resnet_classes)
print("Matriz de Confusão - ResNet50:\n", conf_matrix_resnet)
```

2d.

```
from sklearn.metrics import recall_score, precision_score,
f1_score

def calcular_metricas(y_true, y_pred, modelo_nome):
    recall = recall_score(y_true, y_pred, average=None)  #
Sensibilidade por classe
    specificity = []  # Lista para armazenar a especificidade
por classe

    conf_matrix = confusion_matrix(y_true, y_pred)
    for i in range(len(conf_matrix)):
        TN = np.sum(conf_matrix) - (conf_matrix[i, :].sum() +
conf_matrix[:, i].sum() - conf_matrix[i, i])
        FP = conf_matrix[:, i].sum() - conf_matrix[i, i]
        FN = conf_matrix[i, :].sum() - conf_matrix[i, i]
        specificity.append(TN / (TN + FP))  # Especificidade
por classe

    f1 = f1_score(y_true, y_pred, average=None)  # F1-Score
por classe

    print(f"\n--- Métricas para {modelo_nome} ---")
    for i in range(len(recall)):
        print(f"Classe {i}: Sensibilidade = {recall[i]:.4f},
Especificidade = {specificity[i]:.4f}, F1-Score =
{f1[i]:.4f}")

    # Média das métricas para avaliar o desempenho geral
```

```
    print(f"Média: Sensibilidade = {np.mean(recall):.4f},
Especificidade = {np.mean(specificity):.4f}, F1-Score =
{np.mean(f1):.4f}")

# Calcular métricas para VGG16
calcular_metricas(y_true_classes, y_pred_vgg_classes, "VGG16")

# Calcular métricas para ResNet50
calcular_metricas(y_true_classes, y_pred_resnet_classes,
"ResNet50")
```

2e.

Como o modelo com VGG16 apresenta especificidade e sensibilidade maior do que o resnet, ele é o melhor modelo a ser escolhido.

APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

7. Controle de Informações: Evitar incluir informações desnecessárias que possam aumentar o comprimento do trabalho sem contribuir significativamente para o conteúdo. Concentre-se em informações relevantes, argumentos sólidos e evidências importantes para apoiar sua análise.
8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.
9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx

B – RESOLUÇÃO

RESUMO

Este estudo investiga as implicações éticas do uso do ChatGPT em aconselhamento psicológico online. Abordamos questões como privacidade dos dados, qualidade do aconselhamento, viés algorítmico e responsabilidade ética. Propomos diretrizes para um uso responsável da IA em contextos sensíveis.

Palavras-chave: ChatGPT. Aconselhamento psicológico. Ética. Inteligência Artificial. Privacidade.

This study examines the ethical implications of using ChatGPT in online psychological counseling. We address issues such as data privacy, counseling quality, algorithmic bias, and ethical responsibility. We propose guidelines for the responsible use of AI in sensitive contexts.

Keywords: ChatGPT. Psychological counseling. Ethics. Artificial intelligence. Privacy.

1 INTRODUÇÃO

Com o avanço da inteligência artificial (IA), assistentes virtuais como o ChatGPT têm sido cada vez mais integrados em diferentes setores, incluindo o campo do aconselhamento psicológico online. Esta aplicação levanta questões profundas sobre ética, especialmente relacionadas à privacidade dos dados dos usuários, qualidade do aconselhamento oferecido, viés algorítmico e responsabilidade ética.

Este estudo de caso explora essas implicações éticas específicas, oferecendo uma análise crítica do uso do ChatGPT em um contexto sensível como o suporte emocional e aconselhamento psicológico online.

2 PRIVACIDADE E CONFIDENCIALIDADE

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

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

3 QUALIDADE DO ACONSELHAMENTO E RESPONSABILIDADE

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

Embora o ChatGPT possa oferecer respostas rápidas e acessíveis, há limitações significativas em sua capacidade de compreender nuances emocionais, contexto individual e dinâmicas interativas que são essenciais para o aconselhamento eficaz. Isso levanta questões sobre a responsabilidade ética das plataformas que oferecem serviços de aconselhamento baseados em IA.

Os desenvolvedores e os provedores de serviços devem estabelecer diretrizes claras para o uso responsável do ChatGPT em contextos terapêuticos, garantindo que o bem-estar dos usuários seja priorizado acima de considerações comerciais.

4 VIÉS ALGORÍTMICO E DISCRIMINAÇÃO

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

No contexto do ChatGPT, é fundamental implementar técnicas avançadas de mitigação de viés algorítmico, como a diversificação dos conjuntos de dados de treinamento, a revisão humana de interações críticas e o monitoramento contínuo das respostas geradas pelo assistente virtual.

Além disso, políticas de inclusão e diversidade devem orientar o desenvolvimento e a implementação de algoritmos para evitar discriminações injustas ou prejudiciais.

5 TOMADA DE DECISÃO ÉTICA

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

É essencial que as plataformas que oferecem aconselhamento baseado em ChatGPT forneçam transparência aos usuários sobre os limites e as capacidades do assistente virtual. Isso inclui educar os usuários sobre a natureza da IA, seus propósitos e as expectativas realistas quanto ao tipo de suporte emocional que pode ser oferecido.

Além disso, é necessário estabelecer procedimentos claros para encaminhar usuários para serviços profissionais de saúde mental sempre que necessário, garantindo uma abordagem integrada e ética ao cuidado psicológico.

6 PROPOSTA E SOLUÇÕES RESPONSÁVEIS

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

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

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

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

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

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

7 CONSIDERAÇÕES FINAIS

Em resumo, o uso do ChatGPT em aconselhamento psicológico online apresenta benefícios potenciais significativos, como a expansão do acesso a serviços de suporte emocional. No entanto, também levanta desafios éticos complexos que exigem uma abordagem cuidadosa e responsável.

Ao enfrentar questões de privacidade dos dados, qualidade do serviço, viés algorítmico e tomada de decisão ética, é possível desenvolver práticas que promovam o uso ético da IA no cuidado psicológico.

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

REFERÊNCIAS

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

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

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

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

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

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

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

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

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

B – RESOLUÇÃO

O estudo tem como objetivo identificar desafios de design de arquitetura de *software*, melhores práticas e decisões para sistemas habilitados para *Machine Learning*. O objetivo é detalhado através dos seguintes tópicos:

• Desafios de *design* de arquitetura de *software* mais comuns para sistemas habilitados para *Machine Learning*

• Melhores práticas na arquitetura de sistemas habilitados para *Machine Learning*

• Principais decisões de design de arquitetura de *software* para sistemas habilitados

Apesar de especialistas pesquisarem sobre as melhores práticas de design para sistemas de ML, faltam estudos que analisem a percepção e a usabilidade dos designs de ML na arquitetura de sistemas e aplicativos de ML pelos profissionais.

Foi utilizado um misto de métodos para extrair os resultados do estudo, através da revisão de literatura e de entrevistas com especialistas e profissionais do tema. Esse método misto foi utilizado para compensar limitações de método único.

A etapa de entrevistas foi realizada através de questionário com quinze perguntas abertas, a fim de que os profissionais aprofundassem seus relatos.

A etapa de revisão de literatura teve um sistema para seleção, partindo de uma busca automática por literatura revisada por pares, com critérios de seleção, também foi feita a complementação de busca utilizando técnicas sugeridas em outros estudos.

Observou-se que a arquitetura de quatro visualizações da Siemens é usada para melhor separação de preocupações, enquanto a arquitetura de microsserviços

ajuda na manutenção e coesão, através da decomposição de sistema, promovendo flexibilidade.

É relatado que o manuseio de dados é desafiador em diversos aspectos, entre eles, gerenciamento, visualização e privacidade. Muitos desses desafios, carecem de melhores práticas de *design*.

A escolha de modelos adequados também é dado como um desafio, e as melhores práticas e decisões de design sugerem deixar que os requisitos e o tipo de domínio conduzam a seleção dos modelos.

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

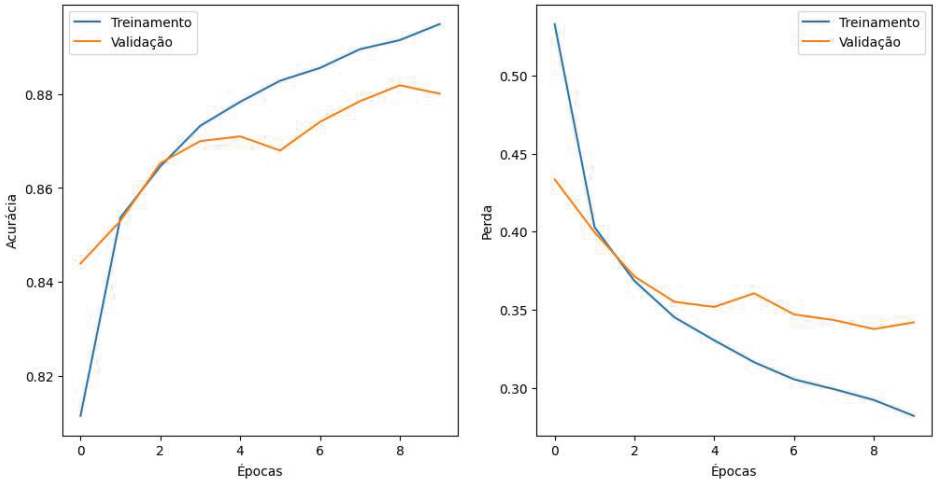
B – RESOLUÇÃO

1.

```
### 1 Classificação (RNA)

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

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

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

# Definir o modelo da rede neural
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),  # Flatten a imagem 28x28 para um vetor 1D
    tf.keras.layers.Dense(128, activation='relu'),    # Camada densa com 128 neurônios e ReLU
    tf.keras.layers.Dropout(0.2),     # Dropout para evitar overfitting
    tf.keras.layers.Dense(10, activation='softmax')  # Camada de saída com 10 classes (uma para cada categoria)
])

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

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

# Treinar o modelo
```

```
history = model.fit(x_train, y_train, epochs=10,
validation_data=(x_test, y_test))

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

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

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

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

plt.show()

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

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

FIGURA 5 - Gráficos de acurácia e perda por épocas



FONTE: A Autora (2025).

Com base nos gráficos de acurácia e função de perda é possível verificar que o treinamento trouxe bons resultados para o modelo, sendo que a acurácia ficou em torno de 88% e a função de perda foi reduzida para um valor em torno de 0,34. Observando o gráfico da função de perda, é possível ver que a queda no valor de perda dos dados de validação começa a reduzir, o que pode significar que, se o treino fosse realizado com mais épocas, possivelmente teríamos um cenário de *overfitting*. Por fim, visto que o modelo, apesar de ter uma acurácia alta, ainda assim pode cometer erros, como é o caso das imagens que foram preditas erradas e estão sendo exibidas na última seção do caderno, no qual tem a classe que foi predita e a classe real da imagem.

2.

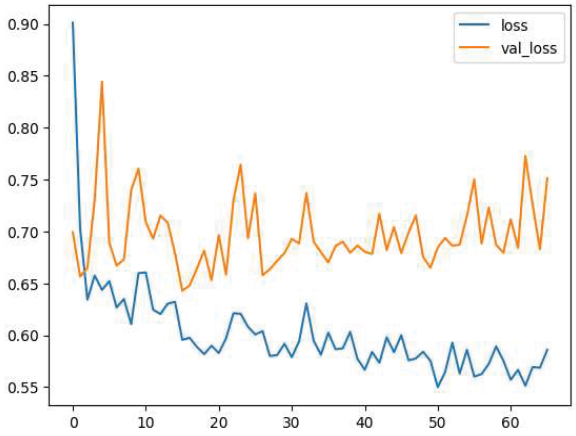
```
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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

#importação dos dados

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

```
data.head()

#mudando nome das colunas
data.columns = [
'acidez_fixa', # fixed acidity
'acidez_volatil', # volatile acidity
'acido_citrico', # citric acid
'acucar_residual', # residual sugar
'cloretos', # chlorides
'dioxido_de_enxofre_livre', # free 130all-b dioxide
'dioxido_de_enxofre_total', # total 130all-b dioxide
'densidade', # density
'pH', # pH
'sulfatos', # 130all-bac
'alcool', # alcohol
'score_qualidade_vinho' # quality
]

data.head()

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

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

#verificando dados faltantes ou infinito

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

#normalização dos dados
from sklearn.preprocessing import StandardScaler

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

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

#separando base de treino e teste
```

```
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y,
test_size=0.3, 131all-b_state=308)


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

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

from keras import backend

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

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

#ajuste do modelo

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

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

#stops para 131all-b

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

#treinamento do modelo
treino_modelo =
modelo_1.fit(x_treino,y_treino,epochs=1000,validation_data=(x_
teste,y_teste),131all-backs = [early_stops])

#avaliação do modelo
plt.plot(modelo_1.history.history['loss'], label='loss')
plt.plot(modelo_1.history.history['val_loss'],
label='val_loss')
plt.legend()
```

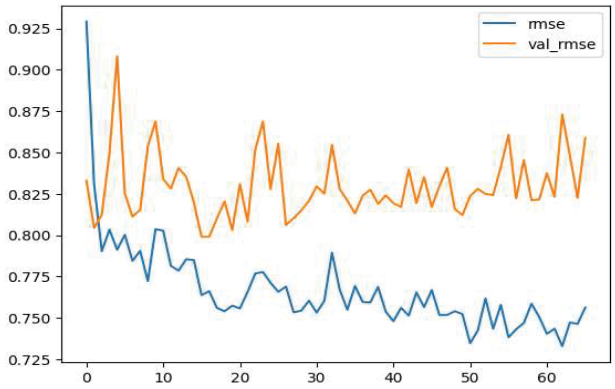
```
#rmse
plt.plot(modelo_1.history.history['rmse'], label='rmse')
plt.plot(modelo_1.history.history['val_rmse'],
label='val_rmse')
plt.legend()
#plotando r2
plt.plot(modelo_1.history.history['r2'], label='r2')
plt.plot(modelo_1.history.history['val_r2'], label='val_r2')
plt.legend()
mse = mean_squared_error(y_teste,y_hat)
rmse = sqrt(mse)
r2 = r2_score(y_teste,y_hat)
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'R2: {r2}')
```

GRÁFICO 17 – Evoluções da Função de Perda



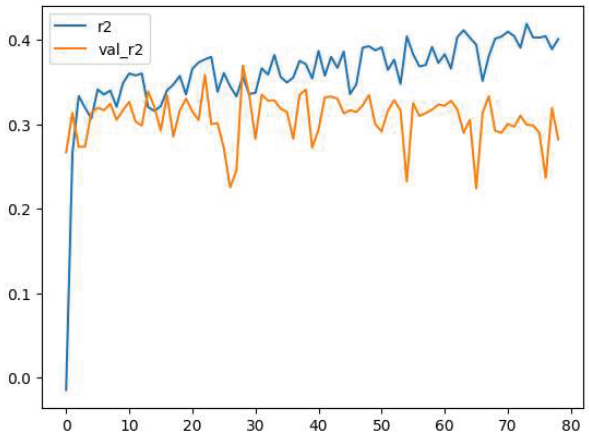
Fonte: A autora (2025).

GRÁFICO 18 - Evolução do RMSE



Fonte: A autora (2025).

GRÁFICO 19 - Evolução do R²



Fonte: A autora (2025).

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

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

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

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

3.

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

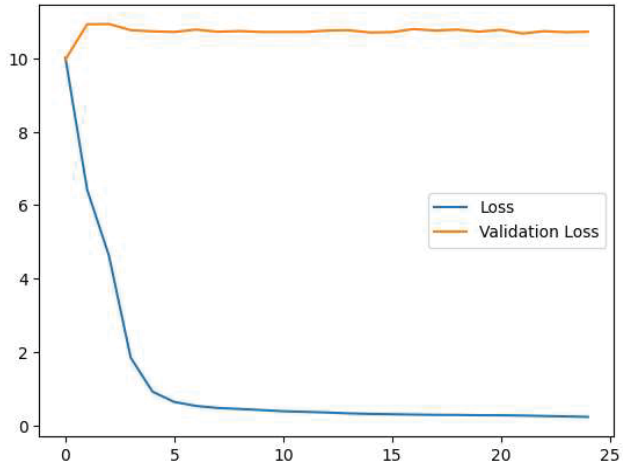
```
from sklearn.utils import shuffle
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

!head '/content/Base_livros.csv'

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

from google.colab import drive
drive.mount('/content/drive')

# 2.2 Visualização básica dos dados
print(df.dtypes)
print('-----')
print('Menor nota: ', df.Notas.min())
print('Maior nota: ', df.Notas.max())
print('-----')
print('Shape: ', df.shape)
print('-----')
print('Distribuição por Nota:')
print(df['Notas'].value_counts().sort_index())

#df.head()

# 2.3 Verificação para ver qual coluna usar em conjunto com o
user id
df['Titulo'].value_counts()

# 2.3 Selecionamos para o treinamento somente os usuarios que
tiveram mais de 20 livros avaliados
mais_avaliados                                              =
df['ID_usuario'].value_counts()[df['ID_usuario'].value_counts(
) > 19].index
mais_avaliados = df[df['ID_usuario'].isin(mais_avaliados)]

df = mais_avaliados
df['ID_usuario'].value_counts()

df.shape

# 2.4 Verificação da distribuição do dataset
plt.figure(figsize=(16, 6))

plt.subplot(1, 2, 1)
score_counts = df['Notas'].value_counts().sort_index()
score_counts.plot()
plt.xlabel('Notas')
plt.ylabel('Frequência')
```

```
plt.xticks(rotation=0)

plt.subplot(1, 2, 2)
score_counts.plot(kind='bar')
plt.xlabel('Notas')
plt.ylabel('Frequência')
plt.xticks(rotation=0)

plt.show()

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

df.Titulo = pd.Categorical(df.Titulo)
df['titulo_cat_codes'] = df.Titulo.cat.codes

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

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

print(df.dtypes)

# 3.2 Conversão de dimensões
# Obter tamanho das listas de Titulos e ID_usuario únicos

N = len(set(df.id_usuario_cat_codes))
M = len(set(df.titulo_cat_codes))

print(f"Número de usuários únicos: {N}")
print(f"Número de livros únicos: {M}")
K = 50

# 4.1 Criação de camadas referentes ao usuario

u = Input(shape=(1, ))
u_emb = Embedding(input_dim=N, output_dim=K)(u)
u_emb = Flatten()(u_emb)

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

# Junção dos conjuntos de camadas

x = Concatenate()([u_emb, i_emb])
x = Dense(256, activation='relu')(x)
x = Dense(1)(x)
```

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

# 5.1 Compilar o modelo
# model.compile(
#     loss='mse',
#     optimizer=SGD(learning_rate=0.08, momentum=0.9)
# )

model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')

# 5.2 Sumário do modelo
model.summary()

from sklearn.model_selection import train_test_split

train, test = train_test_split(df, test_size=0.2,
random_state=42)
print(train.shape)
print(test.shape)

# Verificação das notas de um usuário (id 64)
print(df.loc[df['id_usuario_cat_codes'] == 64, ['Titulo',
'ID_usuario', 'Notas']])

# 7. Treinar o modelo
# Normalização
avg_notas = df.Notas.mean()
train_notas = train.Notas - avg_notas
test_notas = test.Notas - avg_notas

# Treinamento
r = model.fit(
    x=[train.id_usuario_cat_codes,  train.titulo_cat_codes],
    y= train_notas,
    epochs=25,
    batch_size=256,
    verbose=2,
    validation_data=([test.id_usuario_cat_codes,
test.titulo_cat_codes], test_notas)
)
# 8. Plotar a função de perda
plt.plot(r.history['loss'], label='Loss')
plt.plot(r.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
# 9.1 Gerar um array com usuário único
input_usuario = np.repeat(a=64, repeats=M) # 7346 -> 1120
books = np.array(list(set(df.titulo_cat_codes)))
print("input_usuario: ", input_usuario)
print("books: ", books)
print("len input_user: ", len(input_usuario))
```

```
print("len books: ", len(books))
# 9.2 Realiza a predição
preds = model.predict([input_usuario, books])
# 9.3 Tratamento da predição
notas_finais = preds.flatten() + avg_notas
max_idx = np.argmax(notas_finais)
result = df[df.titulo_cat_codes == books[max_idx]]
print(f"Recomendação:       {result.Titulo.values[0]}       por
{result.Autor.values[0]}.  Nota:  {round(notas_finais[max_idx],
1)} ")
```

Gráfico 20 - Evolução da Função de Perda



Fonte: A Autora(2025)

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

O melhor resultado que obtivemos foi utilizando o otimizador Adam e fazendo uma filtragem dos dados para utilizarmos somente os usuários que tivessem feito pelo menos 20 avaliações. Isso melhorou um pouco a acurácia do modelo, mas, devido ao tamanho do *dataset*, não foi possível conseguir uma melhoria significativa.

4.

```
## 1. Importação das bibliotecas
import tensorflow as tf
import numpy as np
import matplotlib as mpl
import IPython.display as display
import PIL.Image

## 2. Importação da imagem
url=
'https://commons.wikimedia.org/wiki/Special:FilePath/Felis_cat
us-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>'))

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

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

# Criação do modelo
```

```
dream_model      =      tf.keras.Model(inputs=base_model.input,
outputs=layers)

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

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

  return  tf.reduce_sum(losses)

## 5. Subida de gradiente (Gradient ascent)
class DeepDream(tf.Module):
  def __init__(self, model):
    self.model = model

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

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

        # Calculo do gradiente da perda em relação aos pixels da
imagem de entrada.
        gradients = tape.gradient(loss, img)
        # Normalizacao dos gradintes
        gradients /= tf.math.reduce_std(gradients) + 1e-8

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

```
      return loss, img

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

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

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

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


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

  return result

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

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

OCTAVE_SCALE = 1.30

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

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

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

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

display.clear_output(wait=True)
img = tf.image.resize(img, base_shape)
img = tf.image.convert_image_dtype(img/255.0, dtype=tf.uint8)
show(img)

end = time.time()
end-start
```

FIGURA 6 - Primeira imagem gerada

FONTE: A Autora(2025)

FIGURA 7 - Última imagem gerada

FONTE: A Autora(2025)

Imagem onírica obtida por Main Loop;

Após o processamento pelo *Main Loop* com camadas Mixed6 e Mixed8, que são partes da rede neural *Inception* (usada no treinamento de visão computacional), padrões visuais abstratos e psicodélicos surgem sobre a imagem. Esses padrões geralmente lembram estruturas orgânicas como olhos, espirais ou texturas semelhantes a folhas e animais.

A técnica funciona "exagerando" características que a rede neural detecta, criando esse efeito de sonho surrealista, como se a máquina estivesse projetando sua própria interpretação da imagem.

Imagem onírica obtida ao levar o modelo até uma oitava;

Após o processamento com a técnica de oitavas, como resultado a imagem original do felino em um cenário de neve foi transformada em uma versão onírica com padrões ainda mais visíveis e elaborados. Nessa versão, há a impressão de múltiplas texturas e formas, como olhos e detalhes geométricos, espalhados de maneira fractal sobre a pelagem do animal e o ambiente ao redor.

Esse efeito mais refinado e detalhado é característico do uso das oitavas, pois ele permite que a rede neural detecte e realce padrões tanto em níveis macro (grandes formas) quanto micro (detalhes finos), gerando uma aparência mais complexa e psicodélica.

Diferenças entre imagens oníricas obtidas com Main Loop e levando o modelo até a oitava.

Main Loop: Foca no processamento direto da imagem em uma única etapa ou em camadas específicas da rede neural (ex. Mixed6, Mixed8). Os padrões visuais oníricos surgem de forma mais sutil e menos detalhada. As formas, como olhos, espirais ou texturas, aparecem mais uniformemente distribuídas pela imagem, mas com menos refinamento em pequenas escalas.

Levando o modelo até a oitava: A imagem é processada em múltiplas resoluções (oitavas), começando em baixa resolução e refinando progressivamente até atingir a imagem completa. O resultado é mais detalhado e complexo, com padrões em múltiplas escalas (macro e micro), criando uma aparência mais fractal e elaborada. Formas como olhos e texturas são mais visíveis, sobrepostas e densamente distribuídas, dando um aspecto mais psicodélico.

Resumo: A técnica de oitavas gera imagens mais detalhadas e refinadas ao realçar padrões em várias escalas, enquanto o *Main Loop* tende a produzir um efeito mais sutil e homogêneo.

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

Escolha um conjunto de dados brutos (ou uma visualização de dados que você acredite que possa ser melhorada) e faça uma visualização desses dados (de acordo com os dados escolhidos e com a ferramenta de sua escolha)

Desenvolva uma narrativa/storytelling para essa visualização de dados considerando os conceitos e informações que foram discutidas nesta disciplina. Não esqueça de deixar claro para seu possível público alvo qual **o objetivo dessa visualização de dados, o que esses dados significam, quais possíveis ações podem ser feitas com base neles.**

Entregue em um PDF:

- O **conjunto de dados brutos** (**ou uma visualização de dados** que você acredite que possa ser **melhorada**);

- Explicação do **contexto e o publico-alvo** da visualização de dados e do storytelling que será desenvolvido;

- A **visualização desses dados** (de acordo com os dados escolhidos e com a ferramenta de sua escolha) **explicando a escolha do tipo de visualização e da ferramenta usada**; (**50 pontos**)

B – RESOLUÇÃO

Os Registros Hospitalares de Câncer são fontes de informações, instalados em hospitais gerais ou oncológicos, públicos, privados, filantrópicos ou universitários, de forma padronizada para todo o país e seguindo padrões internacionais para identificar características pessoais dos pacientes com câncer e de seus tumores. É uma ferramenta utilizada para auxiliar a equipe de saúde e oferecer dados estatísticos sobre o resultado dos tratamentos aplicados.

Através de iniciativas do Instituto Nacional de Câncer, em 1980 as primeiras ações para instalação de um Registro Hospitalar de Câncer foram tomadas, seguindo orientações padronizadas a nível nacional.

Os dados utilizados são dos Registros Hospitalares de Câncer, do ano de 2019, referentes aos pacientes com primeira consulta na unidade hospitalar referente nesse ano.

A fim de dar foco aos tipos de câncer que mais atingem a população brasileira, foram escolhidos para análise os dez tipos de câncer mais frequentes na base utilizada.

O público alvo dessa análise são pesquisadores da área de saúde e gestão pública, a fim de possibilitar melhorias no sistema de saúde.

Foram analisados 220 mil casos de câncer atendidos em unidades hospitalares brasileiras no ano de 2019. Sendo esses, casos de câncer de pele, mama, próstata, colo do útero, brônquio e pulmão, cólon, sistema hematopoético, estômago, reto e tireóide, que foram os 10 tipos mais incidentes na base analisada, de acordo com o CID presente na base de dados. A idade mediana dos pacientes, de forma geral, foi de 64 anos, 55% da base é composta de mulheres e dentre os pacientes com declaração de cor, 49% são brancos e 45% são pardos. Observa-se uma maior concentração de casos nas regiões Sudeste e Sul.

Em conformidade com o que é observado nas estatísticas oficiais, o tipo de câncer mais frequente nos casos analisados, foi o câncer de pele, com 59 mil pacientes. Esse representa 27% dos casos observados a nível nacional, mas são observadas diferenças dessas frequências quando analisados por unidade da federação. Os estados de São Paulo, Paraná, Pernambuco e Rio Grande do Norte apresentam percentuais acima de 30% de atendimento de casos de câncer de pele, enquanto em outros estados, como por exemplo na Bahia, esse percentual é bem inferior aos 27% da representação nacional. Ao analisar o perfil social dos pacientes acometidos com câncer de pele, a mediana de idade era de 70 anos, e 64% dos pacientes com declaração de cor, eram brancos.

Observa-se também que os cânceres de mama e próstata são, após o câncer de pele, os mais frequentes na base analisada (representando 21% e 16% dos casos respectivamente). De acordo com as projeções calculadas pelo INCA para o triênio 2020-2022, os cânceres mais incidentes no período seriam melanoma, mama e próstata. Ao analisar isoladamente os casos de câncer de mama e próstata, observa-se uma diferença etária desses pacientes. Nos dados analisados, o câncer de mama

acomete em sua maioria mulheres mais jovens do que o câncer de próstata em relação aos homens. A mediana de idade dos pacientes com câncer de mama, é de 57 anos, enquanto a mediana de idade dos homens com câncer de próstata na base, é de 70 anos. Em relação à distribuição da cor dos pacientes, dos que tiveram a cor declarada, os casos de câncer de próstata destoam da distribuição geral. Enquanto a nível geral, 50% eram pretos e pardos, nos casos de câncer de próstata esse percentual é de 60%.

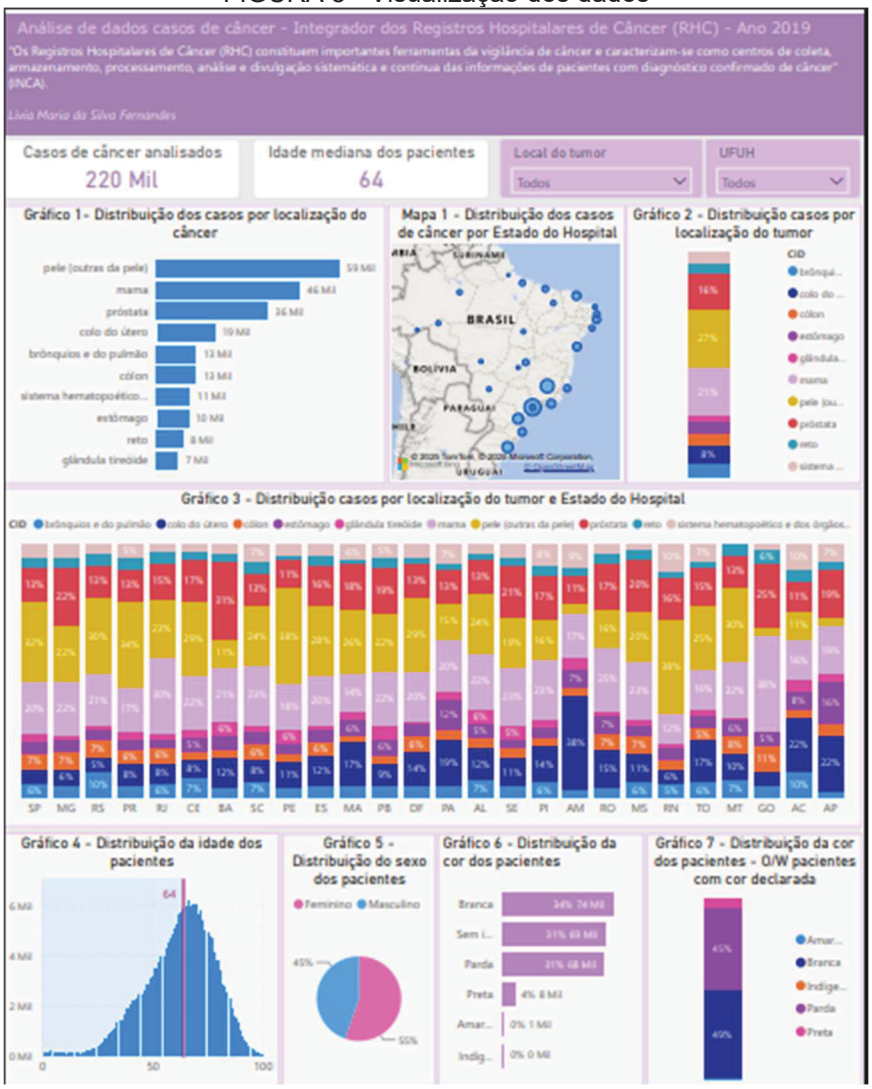
Uma realidade que chama atenção na análise é o percentual exacerbado de casos de câncer de colo de útero na região Norte. Enquanto esse representa apenas 8% dos casos nacionais de câncer, nos estados da região Norte presentes na análise (não há ocorrências do estado de Roraima na base utilizada), eles atingem 22% dos casos de câncer. O câncer de colo de útero está diretamente associado ao HPV (Vírus do Papiloma Humano), e os estados da região são os que apresentam historicamente as mais baixas taxas de cobertura vacinal contra essa doença, de acordo com reportagem da Folha de São Paulo. Dos estados da região Norte, o Amazonas tem destaque ainda maior, por ter 38% dos seus atendimentos para casos de câncer de colo do útero, quase 5 vezes o percentual nacional.

Um fator que pode contribuir com isso, relatado em algumas reportagens e materiais informativos, é a dificuldade de acesso das mulheres à atendimentos de saúde, por ser um estado de grandes proporções e com comunidades muito afastadas. Além da dificuldade de acesso a exames e consultas para prevenção e detecção da doença, de acordo com reportagem do programa Profissão Repórter, o estado do Amazonas tem um único hospital oncológico, em Manaus, para atender os pacientes de todas as cidades do estado. Ainda sobre as altas taxas de câncer de colo do útero no Amazonas, alguns outros fatores, além da dificuldade de acesso à atendimentos de saúde, podem colaborar para esse fenômeno, como a iniciação precoce de atividades sexuais e os hábitos de prevenção à doenças sexualmente transmissíveis. De acordo com a Pense 2019, o Amazonas apresentou o maior percentual de escolares de 13 a 17 anos que já tiveram relação sexual (45,8%), e a “precocidade da iniciação sexual pode estar relacionada com práticas sexuais não seguras e, consequentemente, a exposição aos riscos de contrair infecções sexualmente transmissíveis”.

2.3.1 Referências

- Folha de São Paulo – Câncer de colo de útero é mais frequente no Norte enquanto Sul e Sudeste concentram casos no gástricos
- Divulgação de resultados Pense IBGE 2019 (Agência de Notícias IBGE)
- Globo Repórter
- INCA - Integrador RHC

FIGURA 8 - Visualização dos dados



FONTE: A Autora(2025).

APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

A Solução poderá ser apresentada em: Python (preferencialmente), ou em R, ou em Matlab, ou em C ou em Java.

Considere o seguinte problema de otimização (a escolha do número de 100 cidades foi feita simplesmente para tornar o problema intratável. A solução ótima para este problema não é conhecida).

Suponha que um caixeiro deva partir de sua cidade, visitar clientes em outras 99 cidades diferentes, e então retornar à sua cidade. Dadas as coordenadas das 100 cidades, descubra o percurso de menor distância que passe uma única vez por todas as cidades e retorne à cidade de origem.

Para tornar a coisa mais interessante, as coordenadas das cidades deverão ser sorteadas (aleatórias), considere que cada cidade possui um par de coordenadas (x e y) em um espaço limitado de 100 por 100 pixels.

O relatório deverá conter no mínimo a primeira melhor solução (obtida aleatoriamente na geração da população inicial) e a melhor solução obtida após um número mínimo de 1000 gerações. Gere as imagens em 2d dos pontos (cidades) e do caminho.

Sugestão:

(1) considere o cromossomo formado pelas cidades, onde a cidade de início (escolhida aleatoriamente) deverá estar na posição 0 e 100 e a ordem das cidades visitadas nas posições de 1 a 99 deverão ser definidas pelo algoritmo genético.
(2) A função de avaliação deverá minimizar a distância euclidiana entre as cidades (os pontos).
(3) Utilize no mínimo uma população com 100 indivíduos;
(4) Utilize no mínimo 1% de novos indivíduos obtidos pelo operador de mutação;
(5) Utilize no mínimo de 90% de novos indivíduos obtidos pelo método de cruzamento (crossover-ox);
(6) Preserve sempre a melhor solução de uma geração para outra.

Importante: A solução deverá implementar os operadores de “cruzamento” e “mutação”.

2) Compare a representação de dois modelos vetoriais

Pegue um texto relativamente pequeno, o objetivo será visualizar a representação vetorial, que poderá ser um vetor por palavra ou por sentença. Seja qual for a situação, considere a quantidade de palavras ou sentenças onde tenha no mínimo duas similares e no mínimo 6 textos, que deverão produzir no mínimo 6 vetores. Também limite o número máximo, para que a visualização fique clara e objetiva.

O trabalho consiste em pegar os fragmentos de texto e codificá-las na forma vetorial. Após obter os vetores, imprima-os em figuras (plot) que demonstrem a projeção desses vetores usando a PCA.

O PDF deverá conter o código-fonte e as imagens obtidas.

B – RESOLUÇÃO

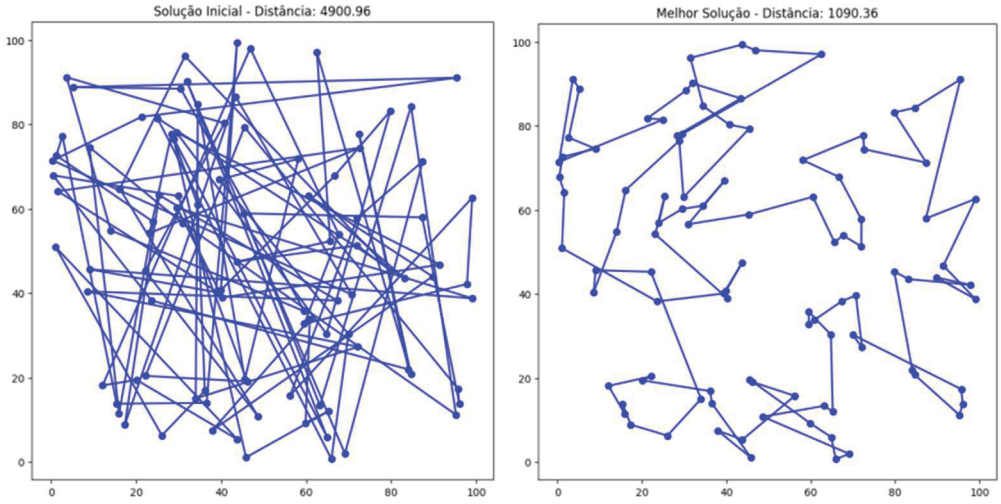
1.

```
import numpy as np
import matplotlib.pyplot as plt
import random
from itertools import permutations
from sklearn.decomposition import PCA
# Definição dos parâmetros do problema
NUM_CIDADES = 100
ESPACO_LIMITE = 100
POPULACAO_SIZE = 100
GERACOES = 1000
MUTACAO_RATE = 0.01
CROSSOVER_RATE = 0.9
# Gerar coordenadas aleatórias para as cidades
cidades = np.random.rand(NUM_CIDADES, 2) * ESPACO_LIMITE
# Função de cálculo da distância euclidiana
def calcular_distancia(percurso):
 distancia = 0
 for i in range(len(percurso) - 1):
 distancia     +=     np.linalg.norm(cidades[percurso[i]]     -
cidades[percurso[i + 1]])
 distancia     +=     np.linalg.norm(cidades[percurso[-1]]     -
cidades[percurso[0]]) # Retorno à cidade inicial
 return distancia

# Inicializar população aleatória
def inicializar_populacao():
 populacao = []
```

```
 for _ in range(POPULACAO_SIZE):
 percurso = list(range(NUM_CIDADES))
 random.shuffle(percurso)
 populacao.append(percurso)
 return populacao
# Função de seleção por torneio
def selecao(populacao):
 candidatos = random.sample(populacao, 5)
 return min(candidatos, key=calcular_distancia)
# Operador de crossover OX (Order Crossover)
def crossover(pai1, pai2):
 tamanho = len(pai1)
 inicio, fim = sorted(random.sample(range(tamanho), 2))
 filho = [-1] * tamanho
 filho[inicio:fim] = pai1[inicio:fim]
 ptr = fim
 for gene in pai2:
 if gene not in filho:
 if ptr >= tamanho:
 ptr = 0
 filho[ptr] = gene
 ptr += 1
 return filho
# Operador de mutação (swap entre duas cidades)
def mutacao(percurso):
 if random.random() < MUTACAO_RATE:
 i, j = random.sample(range(len(percurso)), 2)
 percurso[i], percurso[j] = percurso[j], percurso[i]
 return percurso
# Algoritmo Genético
def algoritmo_genetico():
 populacao = inicializar_populacao()
 melhor_percurso = min(populacao, key=calcular_distancia)
 melhor_distancia = calcular_distancia(melhor_percurso)

 for _ in range(GERACOES):
 nova_populacao = []
 for _ in range(int(POPULACAO_SIZE * CROSSOVER_RATE)):
 pai1, pai2 = selecao(populacao), selecao(populacao)

filho = crossover(pai1, pai2)
 filho = mutacao(filho)
 nova_populacao.append(filho)
 while len(nova_populacao) < POPULACAO_SIZE:
 nova_populacao.append(selecao(populacao))

 populacao = nova_populacao
 melhor_atual = min(populacao, key=calcular_distancia)
 melhor_atual_distancia = calcular_distancia(melhor_atual)

 if melhor_atual_distancia < melhor_distancia:
```

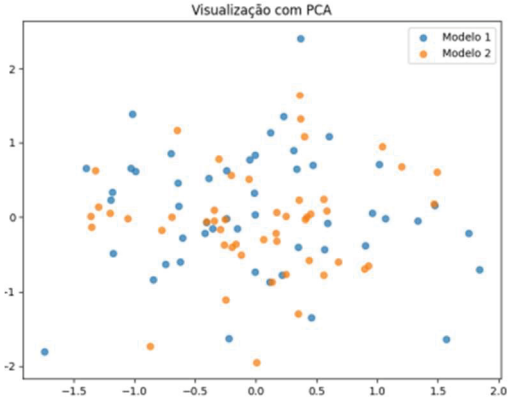
```
 melhor_percurso,     melhor_distancia     =     melhor_atual,
melhor_atual_distancia

 return melhor_percurso, melhor_distancia
# Executar o algoritmo
def plotar_percurso(percurso, titulo):
 plt.figure(figsize=(8, 8))
 caminho = cidades[percurso + [percurso[0]]]
 plt.plot(caminho[:, 0], caminho[:, 1], 'bo-')
 plt.title(titulo)
 plt.show()
# Primeira solução aleatória
populacao = inicializar_populacao()
solucao_inicial = populacao[0]
distancia_inicial = calcular_distancia(solucao_inicial)
plotar_percurso(solucao_inicial, f'Solução Inicial - Distância:
{distancia_inicial:.2f}')
# Melhor solução após evolução
melhor_percurso, melhor_distancia = algoritmo_genetico()
plotar_percurso(melhor_percurso, f'Melhor Solução - Distância:
{melhor_distancia:.2f}')
# Aplicação da PCA em modelos vetoriais de um texto
def aplicar_pca(modelo1, modelo2):
 dados = np.vstack((modelo1, modelo2))
 pca = PCA(n_components=2)
 resultado_pca = pca.fit_transform(dados)

 plt.figure(figsize=(8, 6))
 plt.scatter(resultado_pca[:len(modelo1),                    0],
resultado_pca[:len(modelo1), 1], label='Modelo 1', alpha=0.7)
 plt.scatter(resultado_pca[len(modelo1):,                    0],
resultado_pca[len(modelo1):, 1], label='Modelo 2', alpha=0.7)
 plt.legend()
 plt.title("Visualização com PCA")
 plt.show()

# Exemplo de uso
modelo1 = np.random.rand(50, 300) # Exemplo de embeddings de
palavras
modelo2 = np.random.rand(50, 300)
aplicar_pca(modelo1, modelo2)
```

FIGURA 9 - Solução inicial e Melhor solução - Caixeiro viajante



FONTE: A Autora(2025)

GRÁFICO 21 - Visualização com PCA



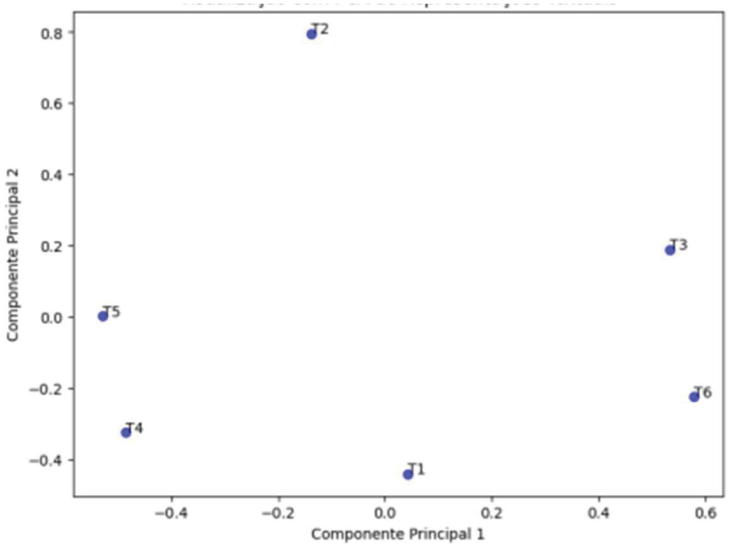
FONTE: A Autora(2025)

2.

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
# Aplicação da PCA em representações vetoriais de textos
textos = [
 "O cachorro correu pelo parque e brincou com a bola.",
 "O gato dormiu no sofá durante a tarde inteira.",
 "As crianças brincaram no parque e correram felizes.",
 "O leão é um animal selvagem que vive na savana.",
 "O cachorro e o gato dormiram juntos na cama.",
 "O parque estava cheio de crianças brincando e correndo."
]
```

```
# Converter textos para vetores usando TF-IDF
vectorizer = TfidfVectorizer()
vetores_texto = vectorizer.fit_transform(textos).toarray()
# Aplicar PCA
pca = PCA(n_components=2)
resultado_pca = pca.fit_transform(vetores_texto)
# Plotar os vetores projetados
plt.figure(figsize=(8, 6))
plt.scatter(resultado_pca[:, 0], resultado_pca[:, 1],
color='blue', alpha=0.7)
for i, txt in enumerate(textos):
 plt.annotate(f'T{i+1}', (resultado_pca[i, 0],
resultado_pca[i, 1]))
plt.title("Visualização com PCA de Representações Textuais")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.show()
```

GRÁFICO 22 - Visualização com PCA de representações textuais



FONTE: A Autora(2025)