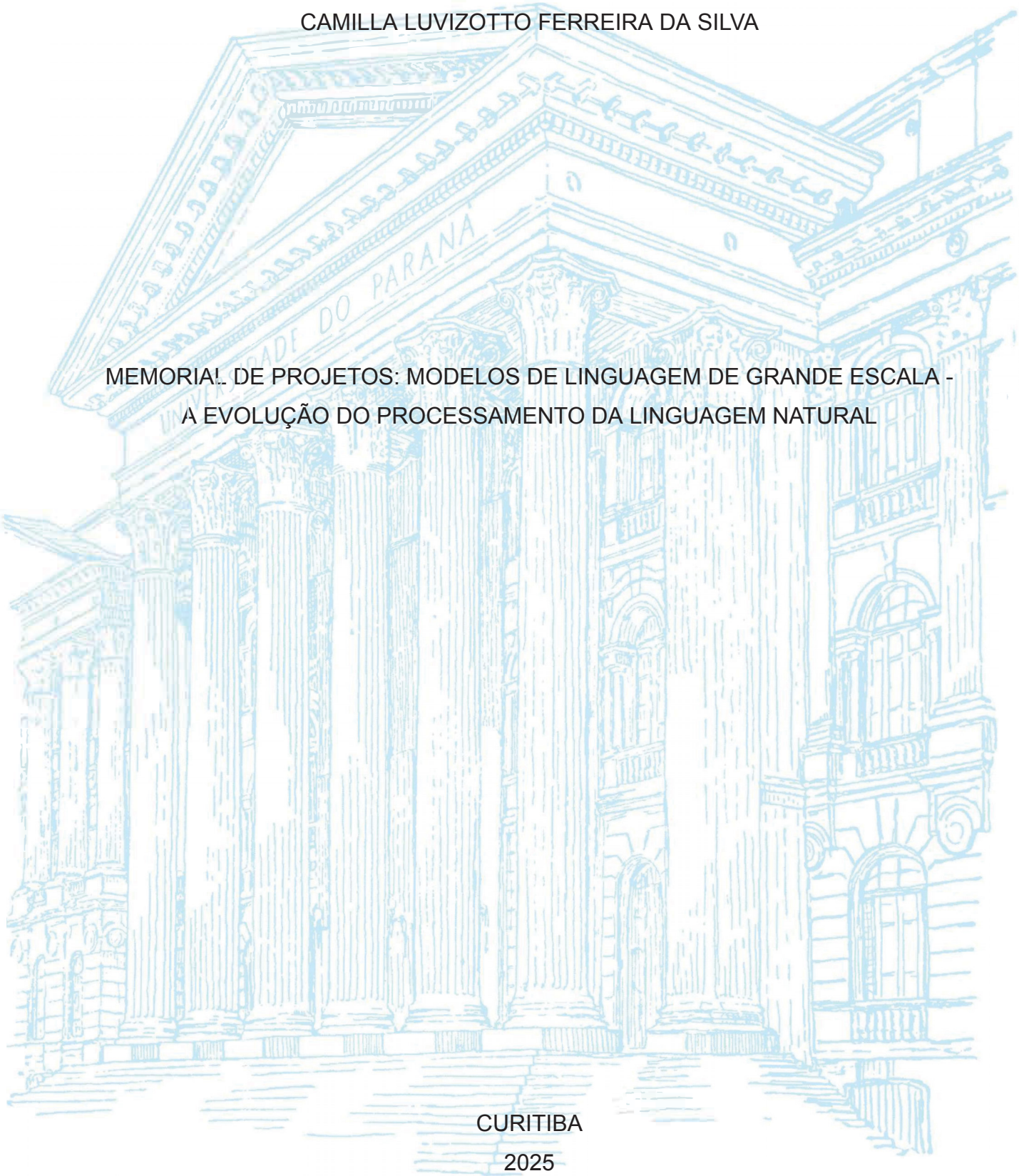
UNIVERSIDADE FEDERAL DO PARANÁ

CAMILLA LUVIZOTTO FERREIRA DA SILVA



MEMORIAL DE PROJETOS: MODELOS DE LINGUAGEM DE GRANDE ESCALA - A EVOLUÇÃO DO PROCESSAMENTO DA LINGUAGEM NATURAL

CURITIBA

2025

CAMILLA LUVIZOTTO FERREIRA DA SILVA

MEMORIAL DE PROJETOS: MODELOS DE LINGUAGEM DE GRANDE ESCALA- A EVOLUÇÃO DO PROCESSAMENTO DA LINGUAGEM NATURAL

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

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 **CAMILLA LUVIZOTTO FERREIRA DA SILVA**, intitulada: **MEMORIAL DE PROJETOS: MODELOS DE LINGUAGEM DE GRANDE ESCALA - A EVOLUÇÃO DO PROCESSAMENTO DA LINGUAGEM NATURAL**, que após terem inquirido a aluna e realizada a avaliação do trabalho, são de parecer pela sua **aprovação** no rito de defesa.

A outorga do título de especialista está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

Curitiba, 29 de Agosto de 2025.

JAIME WOJCIECHOWSKI
Presidente da Banca Examinadora

RAFAELA MANTOVANI FONTANA
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Os Modelos de Linguagem de Grande Escala (LLMs) consolidaram-se como elementos centrais na popularização da Inteligência Artificial (IA), especialmente a partir de 2022, quando ganharam visibilidade com aplicações como o ChatGPT. Esses modelos são fruto de um processo evolutivo que remonta às primeiras discussões sobre a possibilidade de máquinas demonstrarem inteligência, e avançaram significativamente graças ao aumento do poder computacional, da disponibilidade de dados em larga escala e do aperfeiçoamento de arquiteturas de rede. Os LLMs distinguem-se pela capacidade de processar, compreender e gerar linguagem natural de forma coerente, o que os tornou ferramentas versáteis em diversos contextos, desde assistência à programação e busca de informações até aplicações multimodais que combinam texto, imagem e outros tipos de dados. Apesar de seu impacto, sua adoção apresenta desafios expressivos. O treinamento de modelos com bilhões de parâmetros exige elevado consumo de energia e infraestrutura robusta, levantando preocupações ambientais e sociais, além de ampliar desigualdades de acesso, dado que apenas grandes corporações dispõem dos recursos necessários. Também se destacam os desafios éticos e regulatórios, que envolvem riscos de vieses algorítmicos, disseminação de desinformação, produção de conteúdos nocivos e potenciais usos maliciosos, o que reforça a necessidade de governança, transparência e regulamentação, como demonstram iniciativas em curso em diversos países. Em síntese, os LLMs representam um marco disruptivo na evolução da IA, mas sua consolidação depende de um equilíbrio entre inovação tecnológica, sustentabilidade, ética e regulação responsável.

Palavras-chave: Inteligência Artificial; Processamento de Linguagem Natural; Modelos de Linguagem de Grande Escala;

ABSTRACT

Large Language Models (LLMs) have become central to recent advances in Artificial Intelligence (AI), gaining widespread visibility since 2022 through applications such as ChatGPT. These models are the result of a long evolutionary process, benefiting from increased computational power, large-scale data availability, and refined network architectures. LLMs excel at processing, understanding, and generating coherent natural language, enabling versatile applications ranging from programming assistance and information retrieval to multimodal tasks integrating text, images, and other data types. However, their adoption poses significant challenges. Training models with billions of parameters demands substantial energy and infrastructure, raising environmental, social, and access equity concerns. Ethical and regulatory issues, including algorithmic bias, misinformation, harmful content, and potential malicious use, highlight the need for transparency, governance, and regulation. In sum, LLMs represent a disruptive milestone in AI development, but their long-term impact depends on balancing technological innovation with sustainability, ethics, and responsible regulation.

Keywords: Artificial Intelligence; Natural Language Processing; Large Language Models.

SUMÁRIO

1 PARECER TÉCNICO..... 7
REFERÊNCIAS..... 10
APÊNDICE A – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL..... 11
APÊNDICE B – LINGUAGEM DE PROGRAMAÇÃO APLICADA..... 17
APÊNDICE C – LINGUAGEM R..... 34
APÊNDICE D – ESTATÍSTICA APLICADA I..... 41
APÊNDICE E – ESTATÍSTICA APLICADA II..... 51
APÊNDICE F – ARQUITETURA DE DADOS..... 62
APÊNDICE G – APRENDIZADO DE MÁQUINA..... 75
APÊNDICE H – DEEP LEARNING..... 96
APÊNDICE I – BIG DATA..... 124
APÊNDICE J – VISÃO COMPUTACIONAL..... 127
APÊNDICE K – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA..... 136
APÊNDICE L – GESTÃO DE PROJETOS DE IA..... 142
APÊNDICE M – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL..... 145
APÊNDICE N – VISUALIZAÇÃO DE DADOS E STORYTELLING..... 166
APÊNDICE O – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL..... 175

1 PARECER TÉCNICO

Nos últimos anos, os Modelos de Linguagem de Grande Escala (LLMs, do inglês *Large Language Models*) consolidaram-se como protagonistas no campo da Inteligência Artificial (IA). A popularização dessas tecnologias intensificou-se a partir de 2022, com o aumento de aplicações voltadas ao público geral, como o *ChatGPT*, que evidenciaram avanços consideráveis na capacidade de processar e gerar linguagem natural (Minaee et al., 2025).

O desenvolvimento dos LLMs é resultado de uma evolução histórica mais ampla. As origens dessa área de conhecimento remontam à década de 1950, com as primeiras tentativas de formalizar a noção de inteligência de máquina. Um marco simbólico é o Teste de Turing, proposto por Alan Turing, que pretendia avaliar se uma máquina poderia demonstrar comportamento inteligente equivalente ao de um ser humano. Esse teste é tido como um dos motivadores centrais da busca por sistemas capazes de exibir comportamento inteligente (Russel; Norvig, 2021; Zhao et al., 2023).

No campo específico do Processamento de Linguagem Natural (PLN), os primeiros experimentos surgiram nos anos 1960, com sistemas que simulavam interações humanas. O exemplo mais conhecido foi o ELIZA, proposto por Joseph Weizenbaum, que utilizava regras de substituição em um contexto restrito (Russell; Norvig, 2021; Zhao et al., 2023). Embora iniciativas como essa e tantas outras tenham despertado grande interesse, as limitações técnicas acabaram se destacando, o que levou a períodos de estagnação conhecidos como invernos da IA (*AI winters*), nos quais as promessas e expectativas não atingidas resultaram em frustração e diminuição significativa de investimentos na área de IA (Russell; Norvig, 2021).

Com o tempo, entretanto, a área voltou a ganhar destaque. No campo de PLN, inicialmente, com a incorporação de métodos estatísticos e, posteriormente, com a ascensão dos modelos neurais. Abordagens como Word2Vec e BERT introduziram o pré-treinamento em grandes volumes de dados textuais, estabelecendo a base para arquiteturas mais sofisticadas (Minaee et al., 2025).

De modo geral, modelos de linguagem (LMs, do inglês *language models*) são sistemas computacionais treinados para prever a probabilidade de ocorrência de palavras em uma sequência, o que lhes confere a capacidade de compreender e

gerar texto de forma autônoma. Os LLMs expandiram essa lógica por meio do aumento massivo de parâmetros e do uso de técnicas como *in-context learning* e ajuste fino com *feedback* humano, possibilitando respostas mais coerentes e contextualmente relevantes (Chang, 2024). A arquitetura que possibilitou a transição para os LLMs foi a *Transformer* (Vaswani et al., 2017), cujo mecanismo de autoatenção e maior paralelização viabilizaram treinamento eficiente de modelos em escala até então inéditas (Zhao et al., 2023).

O impacto dos LLMs já ultrapassa o domínio da linguagem natural. Atualmente, eles são empregados em assistentes de programação, mecanismos de busca, ferramentas de produtividade e sistemas multimodais capazes de integrar diferentes modalidades, como texto e imagem (Gioia et al., 2025).

Apesar desses avanços, persistem desafios significativos. O treinamento de modelos com bilhões de parâmetros demanda elevado consumo energético. Estima-se que o modelo *GPT-3* tenha requerido centenas de megawatt-hora (MWh) (The Verge, 2024; Wired, 2024). Esse fator levanta preocupações ambientais e amplia desigualdades, uma vez que apenas grandes corporações dispõem da infraestrutura necessária.

Outro desafio crítico é a manutenção da atualidade do conhecimento. Como os modelos de linguagem são treinados em grandes volumes de dados estáticos, existe uma defasagem natural entre o momento do treinamento e a evolução contínua da informação no mundo real. Para enfrentar esse problema, técnicas como a *retrieval-augmented generation* (RAG) têm se mostrado eficazes: nesse método, o modelo recorre a bases externas indexadas e especializadas, ampliando a precisão e a atualidade das respostas sem necessidade de alterar seus parâmetros internos. Importa ressaltar, contudo, que a RAG não garante conhecimento em tempo real, apenas reduz a defasagem temporal. Isso difere de abordagens de fine-tuning contínuo, que efetivamente modificam o modelo. Complementarmente, integrações com APIs estruturadas vêm sendo exploradas para garantir maior confiabilidade (Minaee et al., 2025).

As implicações éticas e sociais também merecem atenção. O viés algorítmico, a disseminação de desinformação, a geração de conteúdo nocivo e o risco de uso malicioso figuram entre as preocupações centrais (Zhao et al., 2023). Nesse cenário, cresce a demanda por mecanismos de governança robustos,

políticas de transparência e regulamentações específicas, como evidenciam iniciativas na União Europeia (*AI Act*), no Brasil e em outras jurisdições.

Quanto às perspectivas futuras, observa-se a tendência de consolidação de modelos multimodais generalistas, capazes de processar texto, imagem, áudio e vídeo em uma mesma estrutura. Essa convergência amplia o escopo das aplicações, mas permanece em aberto o debate sobre se os LLMs, por si só, constituem um caminho viável em direção à chamada Inteligência Artificial Geral (Chang, 2024; Minaee et al., 2025).

Em suma, os LLMs constituem um dos acontecimentos mais relevantes da evolução recente da inteligência artificial. Sua evolução reflete a convergência de progressos técnicos, disponibilidade de dados e poder computacional. Contudo, sua adoção em larga escala traz consigo dilemas éticos, ambientais e regulatórios que exigem resposta coordenada da comunidade científica, órgãos reguladores e população em geral. Assim, o futuro dos LLMs dependerá não apenas da inovação tecnológica, mas também da capacidade coletiva de orientar seu desenvolvimento de forma responsável, inclusiva e sustentável.

REFERÊNCIAS

CHANG, Yupeng; WANG, Xu; WU, Yuan; YANG, Linyi; A Survey on Evaluation of Large Language Models. ACM Transactions on Intelligent Systems and Technology, v. 15, n. 3, p. 1-45, 2024. DOI: 10.1145/3641289. Disponível em: https://dl.acm.org/doi/10.1145/3641289. Acesso em 17 ago. 2025.

GIOIA, Vitor Laguerra Netto; MARCONDES, Cesar Augusto Cavalheiro. Estudo de desempenho de modelos de linguagem de grande escala em maratonas de programação. In: PROJETOS DE PESQUISA EM DATA SCIENCE – Estudos de caso do curso CEDS IBAPE-PR. Editora Científica Digital, 2025. Cap. 17, p. 354-372. DOI: 10.37885/241218370.

MINAEE, Shervin; MIKOLOV, Tomas; NIKZAD, Narjes; et al. Large Language Models: A survey. arXiv, 2025. Disponível em: https://arxiv.org/abs/2402.06196. Acesso em 02 ago. 2025

RUSSELL, Stuart.; NORVIG, Peter. Artificial Intelligence: A modern approach. 4. ed. Pearson, 2021.

THE VERGE. The environmental cost of AI: GPT-3 training consumed massive energy. The Verge, 2024. Disponível em: https://www.theverge.com. Acesso em: 10 ago. 2025.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Łukasz; POLOSUKHIN, Illia. Attention is all you need. arXiv, 2017. Disponível em: https://arxiv.org/abs/1706.03762. Acesso em 27 jul. 2025.

WIRED. The Generative AI Race Has a Dirty Secret. Wired, 2024. Disponível em: https://www.wired.com. Acesso em: 10 ago. 2025.

ZHAO, Wayne; ZHOU, Kun; LI, Junyi; et al. A Survey of Large Language Models. arXiv, 2023. Disponível em: https://arxiv.org/abs/2303.18223. Acesso em 02 ago. 2025.

APÊNDICE A – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

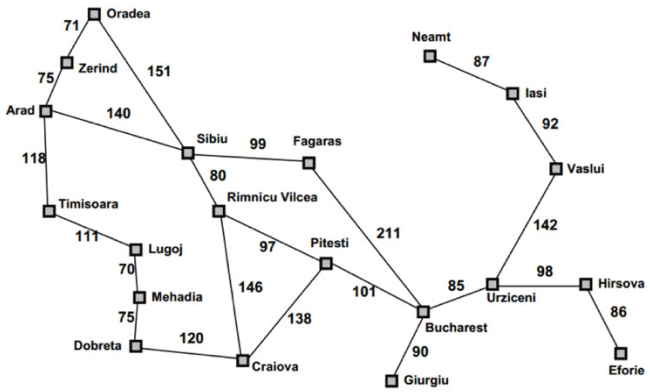
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

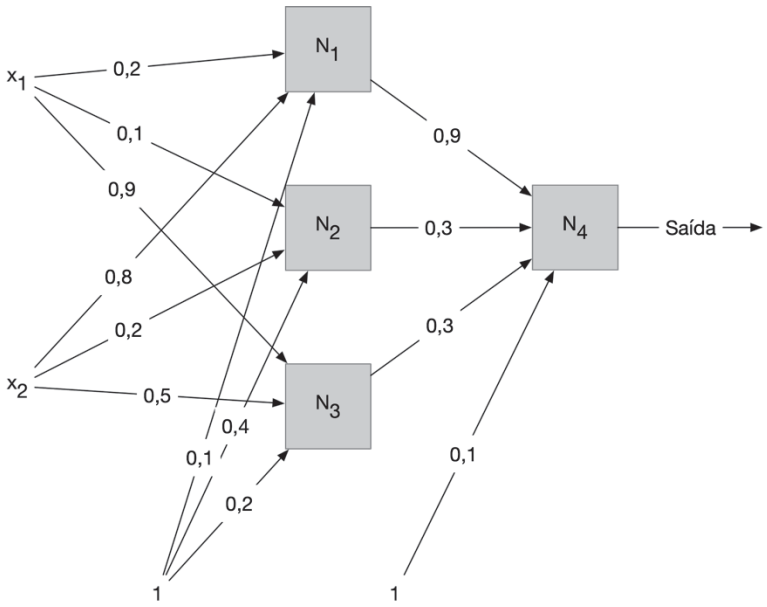
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1.a) Inteligência Artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e máquinas capazes de realizar tarefas que normalmente requerem inteligência humana. Essas tarefas podem incluir raciocínio, aprendizado, reconhecimento de padrões, resolução de problemas, compreensão de linguagem natural, entre outras.

Os sistemas de IA são projetados para simular certos aspectos do comportamento humano, como a capacidade de aprender com experiências passadas, tomar decisões baseadas em dados e adaptar-se a novas situações. Eles podem ser aplicados em uma ampla variedade de áreas, incluindo medicina, finanças, transporte, manufatura, entretenimento, entre outros.

Existem diferentes abordagens e técnicas dentro do campo da IA, incluindo aprendizado de máquina, redes neurais artificiais, lógica simbólica, algoritmos evolutivos, entre outras. O objetivo final da IA é criar sistemas capazes de realizar tarefas de forma autônoma, eficiente e com desempenho comparável ou superior ao humano em determinadas áreas.

1.b) A resposta, acima, dada pelo ChatGPT, pode ser enquadrada nas quatro abordagens, conforme análise a seguir:

Agir como Humano: a resposta descreve a capacidade de sistemas de IA em realizar tarefas que requerem inteligência humana, como raciocínio, reconhecimento de padrões, interpretação de informações e compreensão da linguagem natural. Isso se assemelha com a abordagem de agir como humano, pois imita o comportamento de humanos diante de certas atividades como, por exemplo, ser capaz de interpretar e engajar-se em uma conversa. A representação do conhecimento se dá por meio de processamento de linguagem natural.

Pensar como Humano: essa abordagem busca a implementação do processo de pensamento, ou seja, codificar para o computador a forma de funcionamento do cérebro humano. Esse aspecto pode ser identificado na resposta do ChatGPT quando mencionada a simulação de aspectos do comportamento humano, como a capacidade de aprender com experiências passadas, adaptar-se a novas situações e interpretar informações.

Agir Racionalmente: a resposta destaca a capacidade de sistemas de IA em realizar tarefas que requerem inteligência humana, de tomar decisões e de realizar tarefas de forma eficaz, incluindo adaptação a novas situações. Desta forma, a resposta associa-se ao Agir Racionalmente, visto que esta abordagem é a mais ampla das quatro abordagens. Envolve a implementação de sistemas/agentes capazes de responder a situações e buscar tomar as melhores ações possíveis para atingir objetivo definido.

Pensar Racionalmente: essa abordagem busca modelar o processo de raciocínio para que os sistemas de IA possam operar de acordo com princípios de raciocínio lógico e dedutivo. Na resposta fornecida pelo ChatGPT podemos associar o trecho “sistemas capazes de realizar tarefas” com o Pensar Racionalmente, visto que para realizar uma tarefa é necessário algum uso de raciocínio lógico e dedutivo.

1.c) O ChatGPT pode ser definido como um Modelo de Linguagem de Grande Porte (LLM), projetado para processamento de linguagem natural. Esse tipo de modelo é exposto a quantidade massivas de dados para, assim, aprender os padrões estatísticos da linguagem.

Os LLM são construídos com base na arquitetura de Transformer, uma estrutura de rede neural proposta por Vaswani et al. no trabalho "Attention is All You Need" em 2017. Essa arquitetura se constitui de vários neurônios interconectados (unidades de atenção), o que permite que este modelo processe informações em paralelo, capturando relacionamentos de longo alcance entre palavras em uma sentença. Isso permite que o ChatGPT gere respostas coerentes e contextuais.

Além disso, o ChatGPT é refinado por meio do processo de fine-tuning, onde é ajustado para tarefas específicas e também recebe feedback humano para melhorar seu desempenho. O processo de Aprendizado por Reforço com Feedback Humano (RLHF), onde as interações humanas são usadas para orientar o comportamento do modelo em direção aos resultados desejados, ajuda a garantir que o ChatGPT possa fornecer respostas úteis e evitar a geração de conteúdo problemático. Portanto, tal processo pode ser entendido como uma salvaguarda aplicada ao modelo.

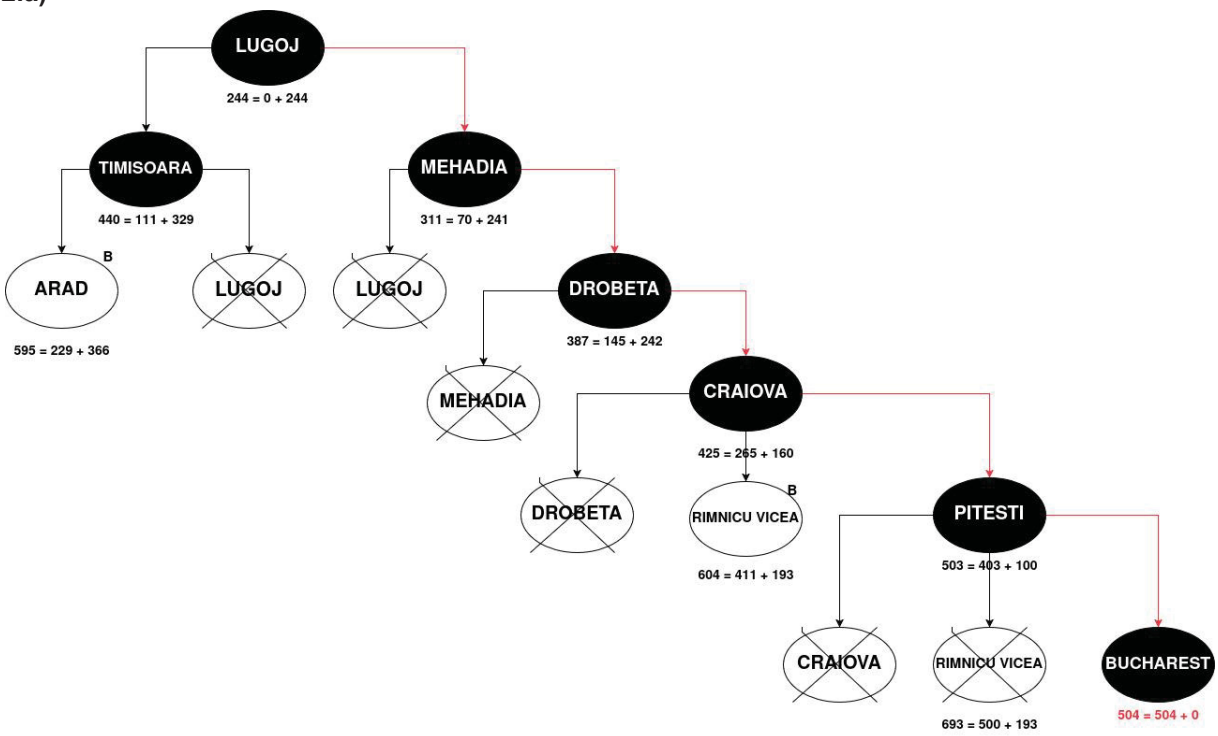
Em resumo, o ChatGPT é uma combinação de técnicas de aprendizado de máquina e processamento de linguagem natural, projetado para entender e responder a perguntas de maneira semelhante a um ser humano.

Referências:

https://help.openai.com/en/articles/6783457-what-is-chatgpt
https://platform.openai.com/docs/introduction
https://www.consultingclub.com.br/post/intelig%C3%AAncias-artificiais-e-o-chat-gpt-o-futuro-j%C3%A1-come%C3%A7ou
https://towardsdatascience.com/how-chatgpt-works-the-models-behind-the-bot-1ce5fca96286
https://www.datacamp.com/blog/a-chat-with-chatgpt-on-the-method-behind-the-bot
https://www.engenhariahibrida.com.br/post/a-tecnologia-por-tras-do-chat-gpt#:~:text=O%20Chat%20GPT%2C%20alimentado%20por,artificial%20por%20meio%20de%20texto
https://www.dio.me/articles/conheca-a-tecnologia-por-tras-do-chatgpt-o-que-e-e-como-usar-a-ferramenta-na-programacao
https://www.youtube.com/watch?v=VcAAXzCKX_g
https://www.youtube.com/watch?v=bSvTVREwSNw
Lee, Peter, et al. A Revolução da Inteligência Artificial na Medicina: GPT-4 e Além. Disponível em: Minha Biblioteca, Grupo A, 2024.

1.d) O modo de funcionamento do ChatGPT enquadra-se na abordagem “Agir como humanos”, devido a compreensão da linguagem natural e a capacidade de responder perguntas, conversar e até mesmo gerar novos conteúdos textuais, utilizando-se da linguagem natural para apresentar seus resultados imitando o comportamento humano. O ChatGPT não pensa como um humano, suas respostas são baseadas em conhecimento obtido através de seus métodos de aprendizagem.

2.a)



3.a)

R1: p → q
R2: q → r
R3: ¬r ∨ p

R4: r → p (Equivalência Implicação em R3)

R5: q → p (Silogismo Hipotético em R2 e R4)
R6: (q → p) ∧ (p → q) (Conjunção entre R5 e R1)
R7: q ↔ p (Equivalência Bi condicional em R6)

4.a)
u(N1) = (0,2 * -3) + (0,8 * 1) + (0,1 * 1)
u(N1) = -0,6 + 0,8 + 0,1
u(N1) = 0,3
f(u) = 0,3

4.b)
u(N2) = (0,1 * -3) + (0,2 * 1) + (0,4 * 1)
u(N2) = -0,3 + 0,2 + 0,4
u(N2) = 0,3
f(u) = 0,3

4.c)
u(N3) = (0,9 * -3) + (0,5 * 1) + (0,2 * 1)
u(N3) = -2,7 + 0,5 + 0,2
u(N3) = -2
f(u) = -2

4.d)
u(N4) = (0,9 * 0,3) + (0,3 * 0,3) + (0,3 * -2) + (0,1 * 1)
u(N4) = 0,27 + 0,09 – 0,6 + 0,1
u(N4) = -0,14
f(u) = tanh(u) = -0,139092448

APÊNDICE B – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

1.a)

```
# QUESTÃO 1.a) Importar a base de dados CSV - precos_carros_brasil.csv
tbDados = pd.read_csv("precos_carros_brasil.csv")

# Mostra o cabeçalho e pequena sequencia de dados
tbDados
```

	year_of_reference	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	January	004001-0	cfzlctzfwrcp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2002.0	9162.0
1	2021.0	January	004001-0	cdqwxwpw3y2p	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2001.0	8832.0
2	2021.0	January	004001-0	cb1t3xwwj1xp	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Gasoline	manual	1	2000.0	8388.0
3	2021.0	January	004001-0	cb9gct6j65r0	GM - Chevrolet	Corsa Wind 1.0 MPFI / EFI 2p	Alcohol	manual	1	2000.0	8453.0
4	2021.0	January	004003-7	g15wg0gbz1fx	GM - Chevrolet	Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI	Gasoline	manual	1,6	2001.0	12525.0
...	...	...	...	...	...	...	...	...	...	...	...
267537	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
267538	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
267539	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
267540	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
267541	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

1.b)

```
# QUESTÃO 1.b) Verificar se faltam valores e tratativa para resolução do problema
# *** HÁ ITENS FALTANTES EM TODAS AS 11 COLUNAS ***
tbDados.isna().any()
```

```
year_of_reference     True
month_of_reference    True
fipe_code             True
authentication        True
brand                 True
model                 True
fuel                  True
gear                  True
engine_size           True
year_model            True
avg_price_brl         True
dtype: bool
```

```
# Contagem de itens faltantes
tbDados.isna().sum()
```

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

```
# EXISTEM ITENS FALTANTES - Opção por apagar itens faltantes
tbDados.dropna(inplace=True)

# Verifica novamente se existem itens faltantes
# *** NÃO HÁ MAIS ITENS FALTANTES ***
tbDados.isna().any()
```

```
year_of_reference     False
month_of_reference    False
fipe_code             False
authentication        False
brand                 False
model                 False
fuel                  False
gear                  False
engine_size           False
year_model            False
avg_price_brl         False
dtype: bool
```

```
# Realiza nova contagem dos itens faltantes
tbDados.isna().sum()
```

```
year_of_reference     0
month_of_reference    0
fipe_code             0
authentication        0
brand                 0
model                 0
fuel                  0
gear                  0
engine_size           0
year_model            0
avg_price_brl         0
dtype: int64
```

1.c)

```
# QUESTÃO 1.c) Verificar se há itens duplicados
tbDados.duplicated().sum()
```

```
2
```

```
# Exclui itens duplicados
tbDados.drop_duplicates(inplace=True)

# Verifica novamente se há itens duplicados
tbDados.duplicated().sum()
```

```
0
```

1.d)

```
# QUESTÃO 1.d) Dividir a categoria dos dados e impressão do resumo de
informações das variáveis numéricas e categóricas
num_colms = [col for col in tbDados.columns if tbDados[col].dtype !=
'object']
cat_colms = [col for col in tbDados.columns if tbDados[col].dtype ==
'object']

# Impressão das infomações das variáveis numéricas
tbDados[num_colms].describe()
```

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

```
# Impressão das infomações das variáveis categóricas
tbDados[cat_colms].describe()
```

	month_of_reference	fipe_code	authentication	brand	model	fuel	gear	engine_size
count	202295	202295	202295	202295	202295	202295	202295	202295
unique	12	2091	202295	6	2112	3	2	29
top	January	001216-5	7hbnjmj9z5dqw	Fiat	Palio Week. Adv/Adv TRYON 1.8 mpi Flex	Gasoline	manual	1,6
freq	24260	425	1	44962	425	168684	161883	47420

1.e)

```
# QUESTÃO 1.e) Impressão de valores por contagem de modelo (model) e
marca do carro (brand)
#Impressão por modelo
tbDados["model"].value_counts()
```

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

```
# Impressão por marca
tbDados["brand"].value_counts()
```

```
brand
Fiat              44962
VW - VolksWagen   44312
GM - Chevrolet    38590
Ford              33150
Renault           29191
Nissan            12090
Name: count, dtype: int64
```

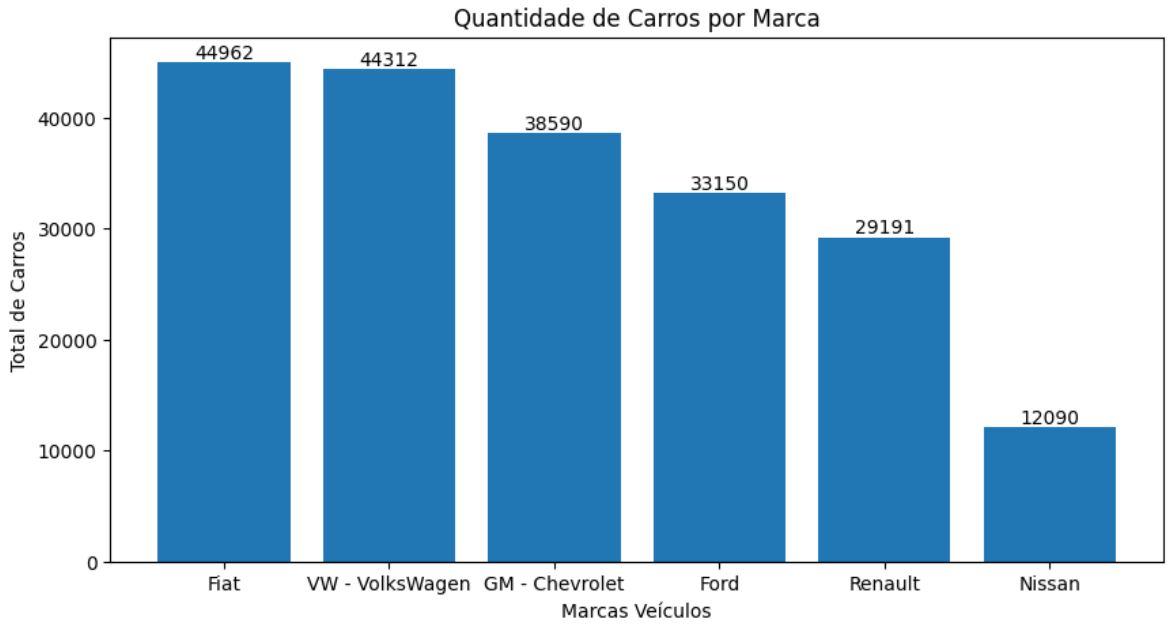
2. a)

```
# QUESTÃO 2.a) Gerar gráfico da distribuição da quantidade de carros por
marca
# Contagem dos valores por marca
valBrand = tbDados["brand"].value_counts()
print(valBrand)
```

```
brand
Fiat              44962
VW - VolksWagen   44312
GM - Chevrolet    38590
Ford              33150
Renault           29191
Nissan            12090
Name: count, dtype: int64
```

```
# Gráfico comparativo
```

```
plt.figure(figsize=(10,5)) # Ajustado tamanho para visualização
simplificada
grfBrand = plt.bar(valBrand.index, valBrand.values) # Variavel "brand" no
eixo X
plt.title('Quantidade de Carros por Marca') # Insere título no gráfico
plt.xlabel('Marcas Veículos')  # Insere título para eixo X
plt.ylabel('Total de Carros')  # Insere título para eixo Y
plt.bar_label(grfBrand, size=10);  # Tamanho da fonte para indicação dos
valores
```



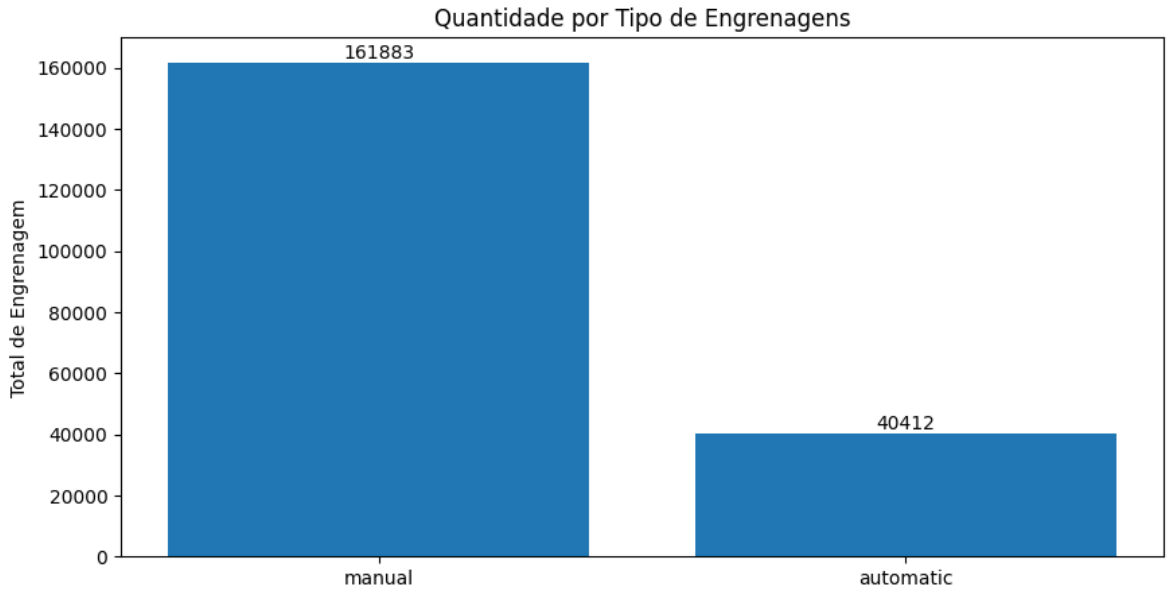
2.b)

```
# QUESTÃO 2.b) Gerar gráfico da quantidade de carros por tipo de
engrenagem
# Contagem dos valores por engrenagem
tpEngr = tbDados["gear"].value_counts()
print(tpEngr)
```

```
gear
manual       161883
automatic     40412
Name: count, dtype: int64
```

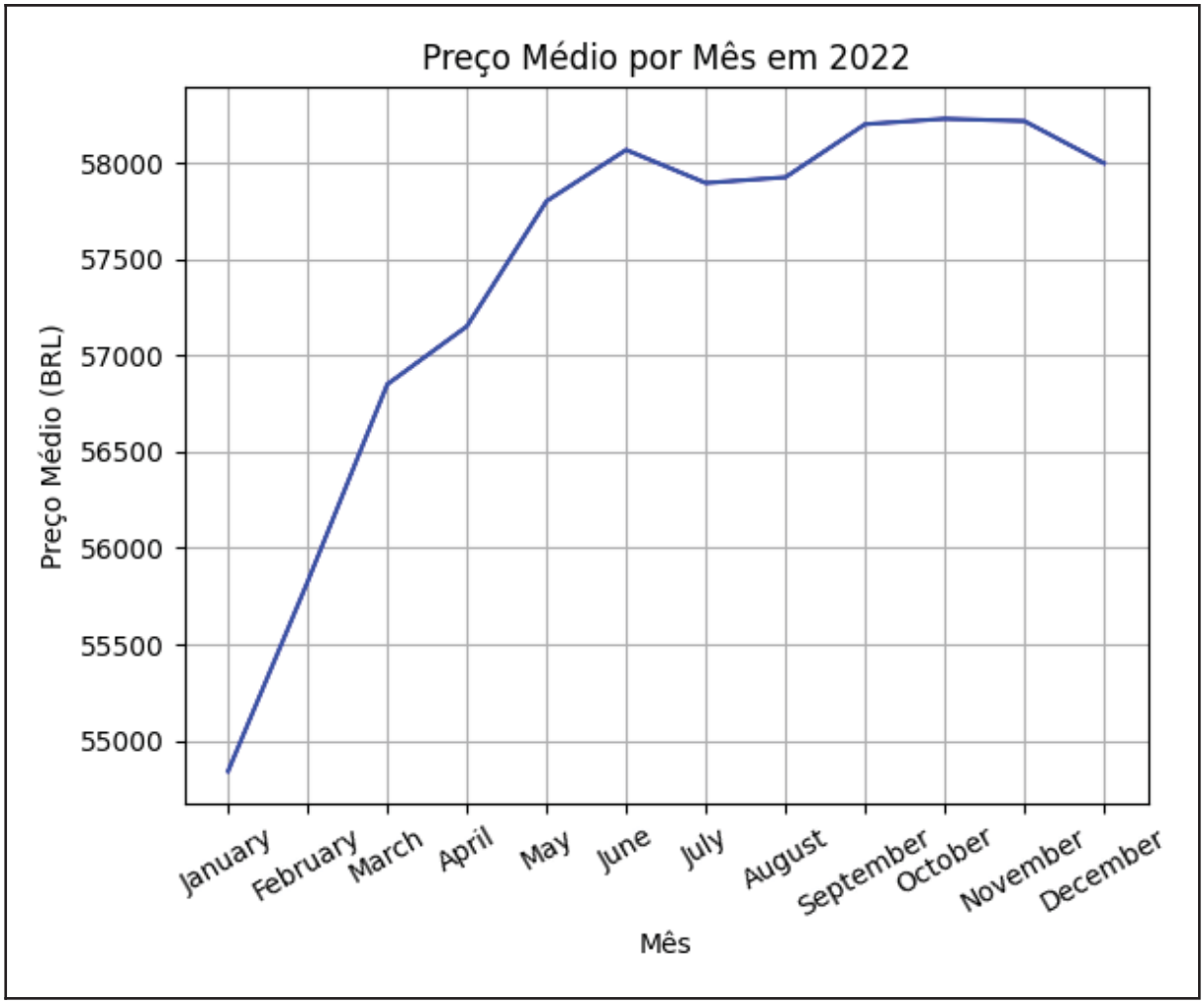
```
# Gráfico comparativo
plt.figure(figsize=(10,5)) # Ajustado tamanho para visualização
simplificada
grfEngr = plt.bar(tpEngr.index, tpEngr.values) # Variavel "gear" no eixo
X
plt.title('Quantidade por Tipo de Engrenagens') # Insere título no
gráfico
plt.ylabel('Total de Engrenagem')  # Insere título para eixo y
plt.bar_label(grfEngr, size=10);  # Tamanho da fonte para indicação dos
```

```
valores
```



2.c)

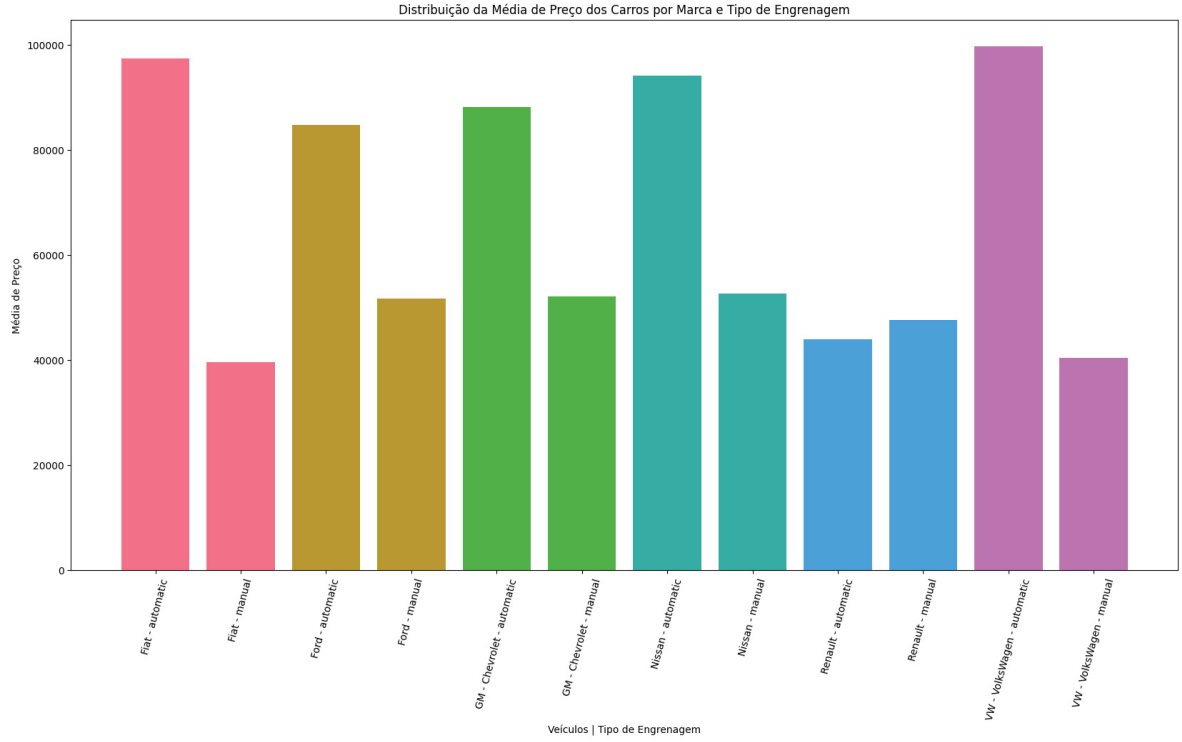
```
# QUESTÃO 2.c) Gerar gráfico da evolução da média de preços dos carros ao
longo do ano de 2022 - Representar os meses no eixo X
# Separando apenas dados de 2022
yearRef = tbDados[tbDados['year_of_reference'] == 2022]

# Criando uma coluna para representar os valorese numericos dos meses
yearRef['month_of_reference_numeric'] = [strptime(mes, '%B').tm_mon for
mes in yearRef['month_of_reference']]
mdMes =
yearRef.groupby(['month_of_reference_numeric','month_of_reference'])['avg
_price_brl'].mean()

# Gerando gráfico de linha
plt.plot(mdMes.index.get_level_values('month_of_reference'), mdMes, "b-")
plt.xlabel("Mês")
plt.ylabel("Preço Médio (BRL)")
plt.title("Preço Médio por Mês em 2022")
plt.xticks(rotation=30)
plt.grid(True)
```



2.d)

```
# QUESTÃO 2.d) Gerar um gráfico da distribuição da média de preços dos
carros por marca e tipo de engrenagem
# Separando apenas dados por preços
refPrice = tbDados.groupby(['brand',
'gear']).avg_price_brl.mean().reset_index()

# Gerando o gráfico
plt.figure(figsize=(20, 10))
Cores = dict(zip(refPrice['brand'].unique(), sns.color_palette("husl",
len(refPrice['brand'].unique()))))

for marca in refPrice['brand'].unique():
    for engrenagem in refPrice['gear'].unique():
        dados = refPrice[(refPrice['brand'] == marca) & (refPrice['gear']
== engrenagem)]
        plt.bar(f'{marca} - {engrenagem}', dados['avg_price_brl'],
label=f'{marca} - {engrenagem}')
        if not dados.empty:
            plt.bar(f'{marca} - {engrenagem}', dados['avg_price_brl'],
```

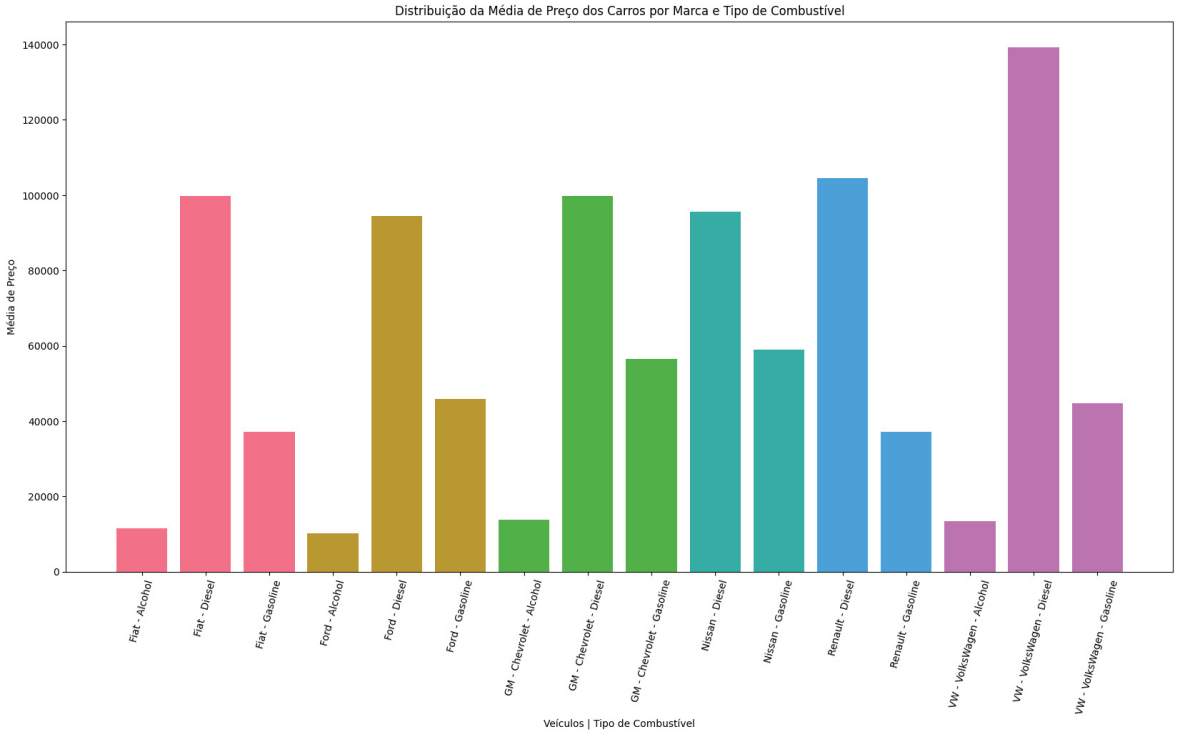
```
color=Cores[marca], label=f'{marca} - {engrenagem}')

plt.xticks(rotation=75)
plt.title('Distribuição da Média de Preço dos Carros por Marca e Tipo de
Engrenagem')
plt.xlabel('Veículos | Tipo de Engrenagem')
plt.ylabel('Média de Preço')
```



2.f)

```
# QUESTÃO 2.f) Gerar um gráfico da distribuição da média de preços dos
carros por marca e tipo de combustível
# Separando apenas dados por preços
refPrice = tbDados.groupby(['brand',
'fuel']).avg_price_brl.mean().reset_index()

# Gerando o gráfico
plt.figure(figsize=(20, 10))
Cores = dict(zip(refPrice['brand'].unique(), sns.color_palette("husl",
len(refPrice['brand'].unique()))))

for marca in refPrice['brand'].unique():
    for combustivel in refPrice['fuel'].unique():
        dados = refPrice[(refPrice['brand'] == marca) & (refPrice['fuel']
== combustivel)]
        for i, row in refPrice.iterrows():
            plt.bar(f'{row["brand"]} - {row["fuel"]}',
row['avg_price_brl'], color=Cores[row['brand']], label=f'{row["brand"]} -
```

```
{row["fuel"]}')

plt.xticks(rotation=75)
plt.title('Distribuição da Média de Preço dos Carros por Marca e Tipo de
Combustível')
plt.xlabel('Veículos | Tipo de Combustível')
plt.ylabel('Média de Preço');
```



3.a)

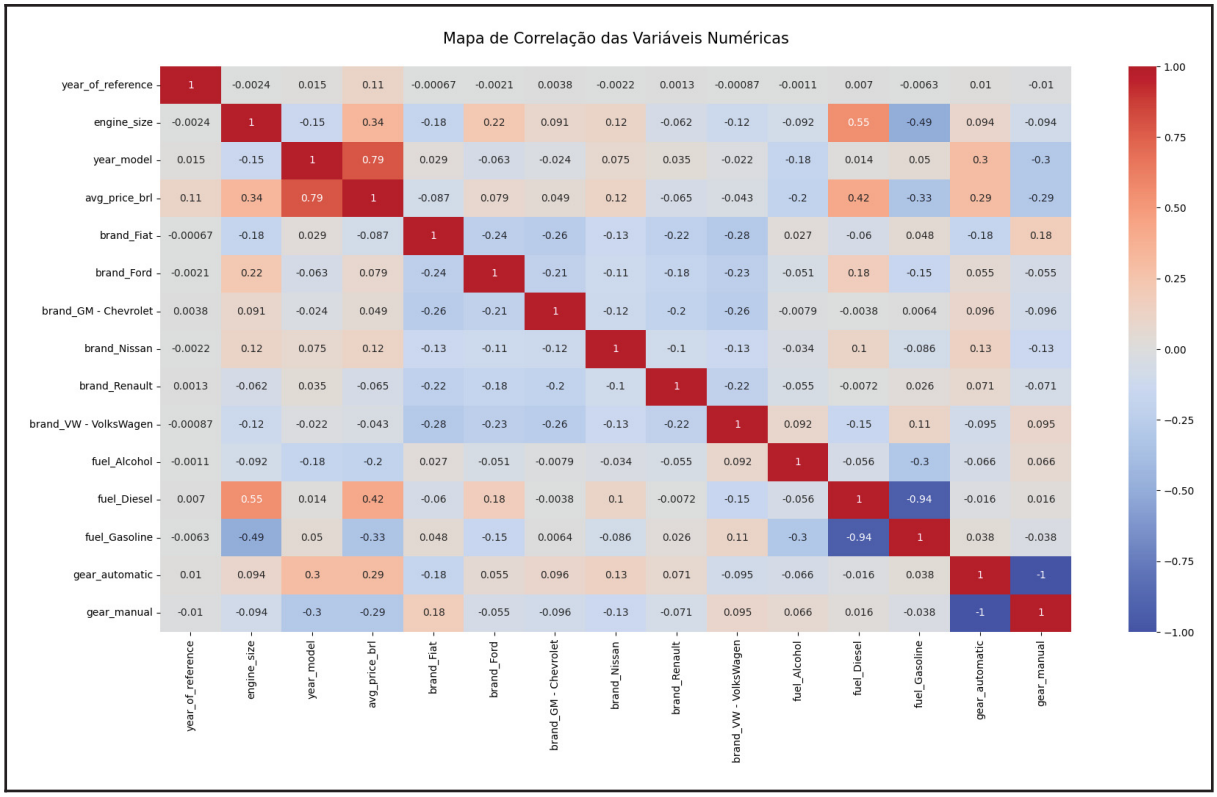
```
# QUESTÃO 3.a) Escolha das variáveis numéricas (modelos de Regressão)
para serem as variáveis independentes do modelo A.
# Elegidas as seguintes variáveis: year_of_reference, brand, fuel, gear,
engine_size, year_model

tbDados_Regr =
tbDados[['year_of_reference','brand','fuel','gear','engine_size','year_mo
del','avg_price_brl']]
tbDados_Regr
```

	year_of_reference	brand	fuel	gear	engine_size	year_model	avg_price_brl
0	2021.0	GM - Chevrolet	Gasoline	manual	1	2002.0	9162.0
1	2021.0	GM - Chevrolet	Gasoline	manual	1	2001.0	8832.0
2	2021.0	GM - Chevrolet	Gasoline	manual	1	2000.0	8388.0
3	2021.0	GM - Chevrolet	Alcohol	manual	1	2000.0	8453.0
4	2021.0	GM - Chevrolet	Gasoline	manual	1,6	2001.0	12525.0
...	...	...	...	...	...	...	...
202292	2023.0	VW - VolksWagen	Gasoline	manual	1,6	2023.0	86038.0
202293	2023.0	VW - VolksWagen	Gasoline	manual	1	2023.0	95997.0
202294	2023.0	VW - VolksWagen	Gasoline	manual	1	2023.0	87828.0
202295	2023.0	VW - VolksWagen	Gasoline	manual	1	2023.0	80845.0
202296	2023.0	VW - VolksWagen	Gasoline	manual	1	2023.0	74458.0

```
# Transformando variáveis categóricas'brand', 'fuel', 'gear' em numéricas
tbDados_Regr = pd.get_dummies(tbDados_Regr,
columns=['brand','fuel','gear'],dtype='int64')

# Tratamento da variável 'engine_size' para indicação numérica
tbDados_Regr['engine_size'] =
LabelEncoder().fit_transform(tbDados_Regr['engine_size'])
tbDados_Regr
```

year_of_reference	engine_size	year_model	avg_price_brl	brand_Fiat	brand_Ford	brand_GM - Chevrolet	brand_Nissan	brand_Renault	brand_VW - VolksWagen	fuel_Alcohol	fuel_Diesel	fuel_Gasoline	gear_automatic	gear_manual
2021.0	0	2002.0	9162.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2001.0	8832.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2000.0	8388.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2000.0	8453.0	0	0	1	0	0	0	1	0	0	0	1
2021.0	5	2001.0	12525.0	0	0	1	0	0	0	0	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2023.0	5	2023.0	86038.0	0	0	0	0	0	1	0	0	1	0	1
2023.0	0	2023.0	95997.0	0	0	0	0	0	1	0	0	1	0	1
2023.0	0	2023.0	87828.0	0	0	0	0	0	1	0	0	1	0	1
2023.0	0	2023.0	80845.0	0	0	0	0	0	1	0	0	1	0	1
2023.0	0	2023.0	74458.0	0	0	0	0	0	1	0	0	1	0	1

```
# Mapa de correlação das variáveis numéricas com variável Target
plt.figure(figsize=(20,10))
sns.heatmap(tbDados_Regr.corr("spearman"), annot = True, cmap="coolwarm")
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```



3.b)

```
# QUESTÃO 3.b) Criar partições contendo 75% de dados para treino e 25% de dados para teste
# Variável X contém as variáveis numéricas de interesse para a análise, excluindo a variável target
varX = tbDados_Regr.drop(['avg_price_brl'],axis = 1)
varX.head()
```

year_of_reference	engine_size	year_model	brand_Fiat	brand_Ford	brand_GM - Chevrolet	brand_Nissan	brand_Renault	brand_VW - VolksWagen	fuel_Alcohol	fuel_Diesel	fuel_Gasoline	gear_automatic	gear_manual
2021.0	0	2002.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2001.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2000.0	0	0	1	0	0	0	0	0	1	0	1
2021.0	0	2000.0	0	0	1	0	0	0	1	0	0	0	1
2021.0	5	2001.0	0	0	1	0	0	0	0	0	1	0	1

```
# Variável Y contém apenas a variável target - 'avg_price_brl'
varY = tbDados_Regr['avg_price_brl']
varY.head()
```

```
0     9162.0
1     8832.0
2     8388.0
3     8453.0
4    12525.0
Name: avg_price_brl, dtype: float64
```

```
# Atribuindo 75% dos dados para treinamento e 25% dos dados para testes
X_train, X_test, Y_train, Y_test = train_test_split(varX, varY, test_size = 0.25, random_state = 42)

# Observando os dados de treinamento
print(X_train.shape)
X_train.head(1)
```

(151721, 14)

	year_of_reference	engine_size	year_model	brand_Fiat	brand_Ford	brand_GM - Chevrolet	brand_Nissan	brand_Renault	brand_VW - VolksWagen	fuel_Alcohol	fuel_Diesel	fuel_Gasoline	gear_automatic	gear_manual
156364	2022.0	10	2020.0	1	0	0	0	0	0	0	1	0	0	1

# Observando a variável target
Y_test.head()

```
180633    42595.0
13130     10989.0
163315     9087.0
121464    26965.0
14044     57102.0
Name: avg_price_brl, dtype: float64
```

3.c)

```
#QUESTÃO 3.c) Treino dos modelos RandomForest e XGBoost para predição dos preços dos carros
# Treino RandomForest
mdlRanForest = RandomForestRegressor()

# Ajuste do modelo conforme variáveis de treinamento
mdlRanForest.fit(X_train, Y_train)
```

RandomForestRegressor
RandomForestRegressor()

```
# Treino XGBoost
mdlXgboost = XGBRegressor()

# Ajuste do modelo conforme variáveis de treinamento
mdlXgboost.fit(X_train, Y_train)
```

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
             num_parallel_tree=None, random_state=None, ...)
```

3.d)

```
# QUESTÃO 3.d) Gravar os valores preditos em variáveis
# Valores preditos em RandomForest com base nos dados de teste
valPredRanForest = mdlRanForest.predict(X_test)
valPredRanForest
```

```
array([ 44889.62859266,  12739.80934143,  15295.85822716, ...,
       117329.56881555,  16274.02237751,  21525.00267423])
```

```
# Valores preditos em XGBoost com base nos dados de teste
valPredXgboost = mdlXgboost.predict(X_test)
valPredXgboost
```

```
array([ 45345.223,  12810.657,  15979.231, ..., 117479.83 ,  15259.941,
        22179.574], dtype=float32)
```

3.e)

```
# QUESTÃO 3.e) Análise da importância das variáveis para estimar cada um
dos modelos
# Análise da importância das variáveis em RandomForest para estimativa do
alvo
mdlRanForest.feature_importances_
feature_importancesRF = pd.DataFrame(mdlRanForest.feature_importances_,
index = X_train.columns,
columns=['importance']).sort_values('importance', ascending = False)
feature_importancesRF
```

	importance
engine_size	0.428020
year_model	0.396104
fuel_Diesel	0.076969
gear_automatic	0.021225
gear_manual	0.018628
year_of_reference	0.013558
fuel_Gasoline	0.012433
brand_Nissan	0.011678
brand_VW - VolksWagen	0.011213
brand_GM - Chevrolet	0.002731
brand_Renault	0.002713
brand_Ford	0.002528
brand_Fiat	0.002182
fuel_Alcohol	0.000018

```
# Análise da importância das variáveis em XGBoost para estimativa do alvo
mdlXgboost.feature_importances_
feature_importancesXG = pd.DataFrame(mdlXgboost.feature_importances_,
index = X_train.columns,
columns=['importance']).sort_values('importance', ascending = False)
feature_importancesXG
```

	importance
fuel_Diesel	0.427805
engine_size	0.237555
year_model	0.167987
gear_automatic	0.049263
brand_Renault	0.026771
brand_Nissan	0.026539
brand_VW - VolksWagen	0.026310
year_of_reference	0.011607
brand_Fiat	0.010896
brand_GM - Chevrolet	0.007659
brand_Ford	0.005939
fuel_Alcohol	0.000907
fuel_Gasoline	0.000762
gear_manual	0.000000

3.g)

```
#QUESTÃO 3.g) Escolha do melhor modelo com base nas métricas de avaliação
MSE, MAE, R2
# Métrica RandomForest
# Resultado análise MSE em RandomForest
valMSErf = mean_squared_error(Y_test, valPredRanForest)
valMSErf
```

```
106634614.2088013
```

```
# Resultado análise MAE em RandomForest
valMAErf = mean_absolute_error(Y_test, valPredRanForest)
valMAErf
```

```
5599.162102900011
```

```
# Resultado análise R2 em RandomForest
valR2rf = r2_score(Y_test, valPredRanForest)
valR2rf
```

```
0.9603773993318255
```

```
# Métricas XGBoost
# Resultado análise MSE em XGBoost
valMSExg = mean_squared_error(Y_test, valPredXgboost)
valMSExg
```

```
107807654.56567411
```

```
# Resultado análise MAE em XGBoost
valMAExg = mean_absolute_error(Y_test, valPredXgboost)
valMAExg
```

```
5668.311479710965
```

```
# Resultado análise R2 em XGBoost
valR2xg = r2_score(Y_test, valPredXgboost)
valR2xg
```

```
0.9599415285784788
```

Questões Discursivas

1.f) Durante a importação e tratamento dos dados observou-se a existência de 65.245 dados faltantes e dois duplicados sendo necessária a aplicação de correções para eliminar estes problemas. Em relação aos dados, observou-se que as marcas Fiat, Volkswagen e Chevrolet são predominantes no conjunto de dados. Foi possível constatar que a maioria dos veículos tem um preço médio superior a 60 mil, são movidos a gasolina, e possuem câmbio manual.

2.e) O gráfico demonstra que as marcas Volkswagem, Fiat, e Nissan, possuem, respectivamente, as médias mais altas de preços para câmbio automático. Já o modelo automático da marca Renault possui média de preço um pouco superior quando comparado aos carros das marcas Volkswagem e Fiat com câmbio manual. Pode-se concluir que os carros automáticos têm preço superior quando comparados a carros manuais.

2.g) Veículos movidos a diesel possuem maior média de preço, seguidos pelos movidos a gasolina e, por fim, pelos movidos a álcool. Adicionalmente, observa-se que as marcas Renault e Nissan não possuem modelos movidos a álcool.

3.f) No modelo de RandomForest as variáveis com maior importância foram o tamanho do motor e o ano do modelo. Essa relevância pode ser observada ao computar com o método feature_importances e também no mapa de correlação de variáveis. Para o XGBoost, as variáveis mais relevantes foram combustível a diesel, tamanho do motor e o ano do modelo.

3.h) Analisando as métricas em ambos os modelos, observa-se que RandomForest e XGBoost tiveram um bom desempenho. No entanto, o melhor modelo foi o do Random Forest que atingiu coeficiente de determinação de 0.96 e menores valores para MSE e MAE.

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

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1.1-1.3)

```
# instalação e carregamento de pacotes necessários
install.packages('e1071')
install.packages('randomForest')
install.packages('kernlab')
install.packages('mlbench')
install.packages('caret')
library('mlbench')
library('caret')

# carregando a base de dados Satellite
set.seed(123)
data("Satellite")

# construindo dataframe apenas com dados de interesse para classificação
database <- data.frame(Satellite[17:20])
database$classes <- Satellite$classes

# criando partições de treino e teste
indices <- createDataPartition(database$classes,p = 0.8,list = FALSE)
treino <- database[indices,]
teste <- database[-indices,]

# treinando modelos

# randomForest
rf <- caret::train(classes~., data=treino, method='rf')
```

```
# svm
svm <- caret::train(classes~.,data=treino, method='svmRadial')

# rna
rna <- caret::train(classes~., data=treino, method='nnet', trace=FALSE)

# fazendo predições com cada modelo

# randomForest
predicoes.rf <- predict(rf,teste)

#svm
predicoes.svm <- predict(svm,teste)

#rna
predicoes.rna <- predict(rna,teste)

# avaliando matrizes de confusão

# randomForest
confusionMatrix(predicoes.rf, teste$classes)

# svm
confusionMatrix(predicoes.svm, teste$classes)

# rna
confusionMatrix(predicoes.rna, teste$classes)
```

1.4) Matriz de Confusão dos Modelos

```
> confusionMatrix(predicoes.rf, teste$classes)
Accuracy : 0.852
95% CI : (0.8314, 0.871)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
> confusionMatrix(predicoes.svm, teste$classes)
Accuracy : 0.8575
95% CI : (0.8371, 0.8762)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
> confusionMatrix(predicoes.rna, teste$classes)
Accuracy : 0.7928
95% CI : (0.7696, 0.8147)
No Information Rate : 0.2383
P-Value [Acc > NIR] : < 2.2e-16
```

1.5) Em problemas de classificação, uma das métricas de referência para avaliar a performance de modelos é a acurácia que indica a proporção de instâncias classificadas corretamente pelo modelo em relação ao total de previsões. Em virtude disso, o modelo escolhido foi o de svm, que obteve 0.8575 de acurácia, a melhor dentre os três modelos.

2.1-2.6)

```
# instalação e carregamento de pacotes necessários
install.packages('e1071')
install.packages('randomForest')
install.packages('kernlab')
install.packages('caret')
library('caret')

# carregar arquivo de volumes
set.seed(123)
data <- read.csv2("http://www.razer.net.br/datasets/Volumes.csv")

# eliminando coluna NR
data$NR <- NULL

# criando partições de treino e teste
indices <- createDataPartition(data$VOL, p=0.8, list=FALSE)
treino <- data[indices,]
teste <- data[-indices,]

# treinando modelos

# randomForest
rf <- caret::train(VOL~.,data=treino,method='rf')

# svm
svm <- caret::train(VOL~., data=treino, method='svmRadial')

# rna
rna <- caret::train(VOL~., data=treino, method='nnet', trace=FALSE)

# alométrico
alom <- nls(VOL ~b0 + b1 * DAP*DAP*HT, data=treino, start=list(b0=0.5,
b1=0.5))

# fazendo as predicoes

#randomForest
predicoes.rf <- predict(rf,teste)

#svm
predicoes.svm <- predict(svm,teste)
```

```
#rna
predicoes.rna <- predict(rna,teste)

#alométrico
predicoes.alom <- predict(alom,teste)

# UDF
# coeficiente de determinação
coef_det <- function(obs, preds){
  sum_pos <- sum((obs - preds) ^ 2)
  sum_neg <- sum ((obs - mean(obs)) ^2)
  result <- 1 - (sum_pos / sum_neg)
  return(result)
}

# erro padrão da estimativa
standard_error <- function(obs, preds){
  size <- length(obs)
  sum_pos <- sum((obs - preds) ^ 2)
  result = sqrt((sum_pos / (size - 2)))
  return(result)
}

percentage_error <- function(obs,preds){
  size <- length(obs)
  sum_pos <- sum((obs - preds) ^ 2)
  partial_result = sqrt((sum_pos / (size - 2)))
  result <- (partial_result/mean(obs)) * 100
}

# métricas por modelo

#randomForest
rf.coef <- coef_det(teste$VOL, predicoes.rf)
rf.error <- standard_error(teste$VOL, predicoes.rf)
rf.percentage_error <- percentage_error(teste$VOL, predicoes.rf)

# svm
svm.coef <- coef_det(teste$VOL, predicoes.svm)
svm.error <- standard_error(teste$VOL, predicoes.svm)
svm.percentage_error <- percentage_error(teste$VOL, predicoes.svm)

# rna
rna.coef <- coef_det(teste$VOL, predicoes.rna)
rna.error <- standard_error(teste$VOL, predicoes.rna)
rna.percentage_error <- percentage_error(teste$VOL, predicoes.rna)

# alométrico
alom.coef <- coef_det(teste$VOL, predicoes.alom)
alom.error <- standard_error(teste$VOL, predicoes.alom)
alom.percentage_error <- percentage_error(teste$VOL, predicoes.alom)
```

2.7)

Após aplicação dos cálculos obteve-se os seguintes resultados:

	RandomForest	SVM	RNA	Alométrico
Coeficiente de determinação R^2	0.8486654	0.7899082	-0.8207433	0.8694429
Erro padrão da estimativa S_{yx}	0.156604	0.1845178	0.5431978	0.1454567
Porcentagem de erro S_{yx}%	11.63489	13.70874	40.35687	10.8067

Levando em consideração o coeficiente de determinação R^2, o melhor modelo foi o Alométrico, visto que é o que mais se aproxima de 1. A estimativa de erro padrão também foi a menor neste modelo.

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.

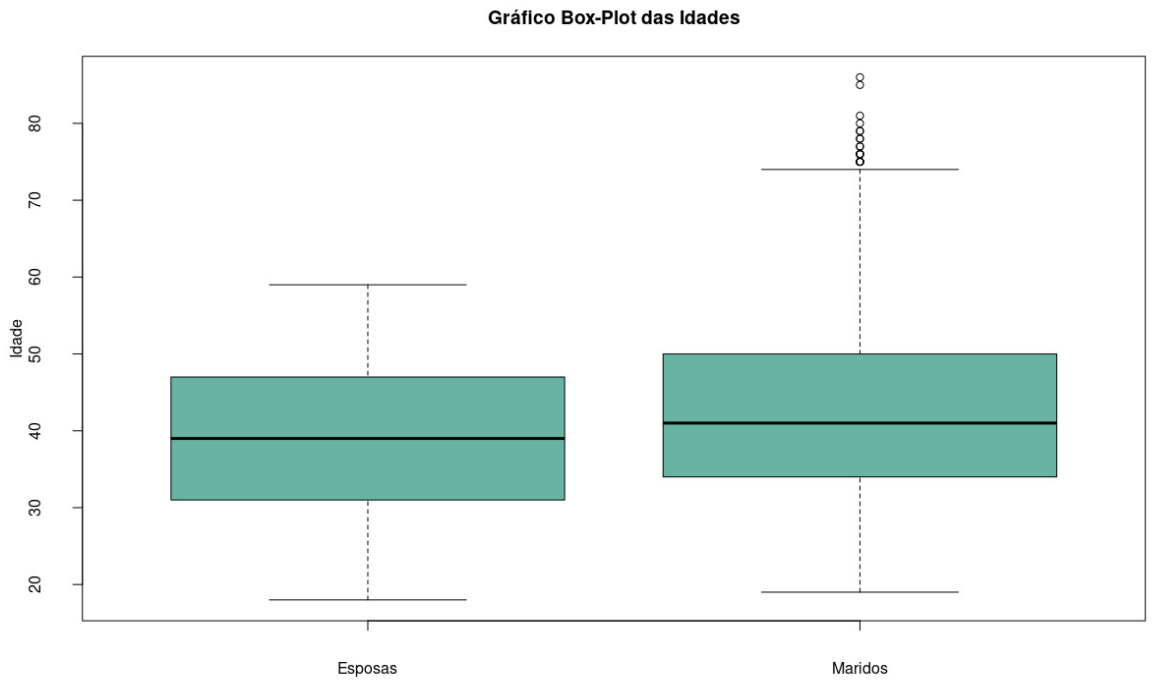
Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

B – RESOLUÇÃO

1.a)

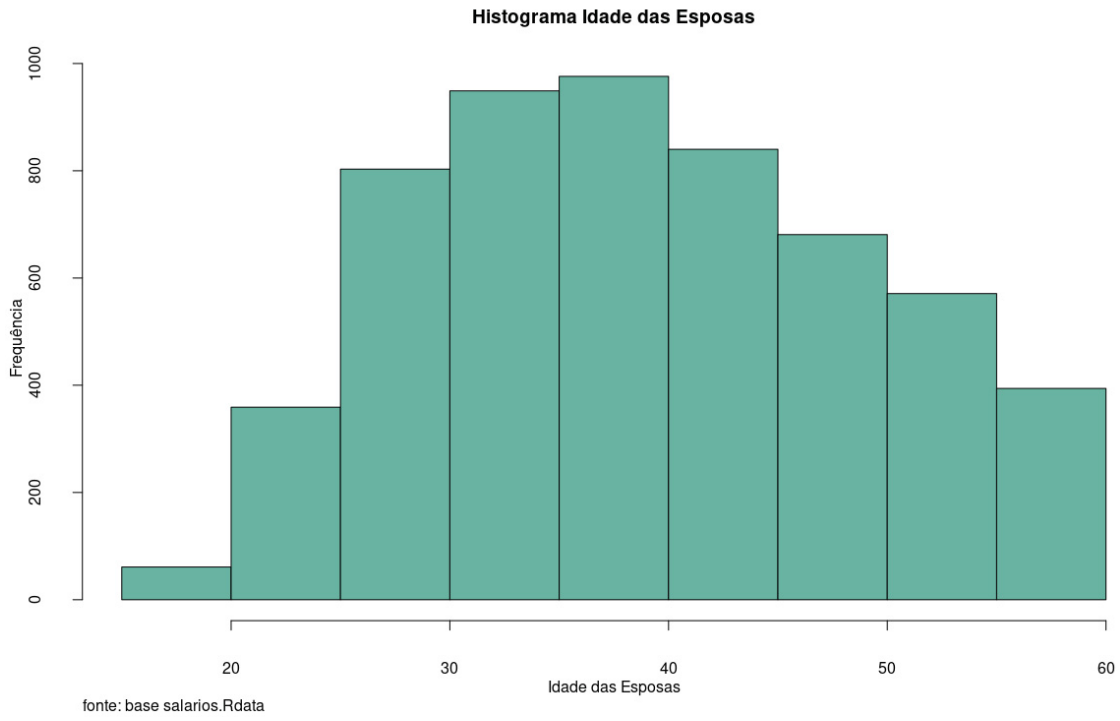
```
# Instalação de pacotes e upload/leitura base de dados:
install.packages("car")
install.packages("dplyr")
library("car")
library(dplyr)
load("salarios.RData")
# Gerar gráfico box-plot com idades das esposas e dos maridos:
ages_list <- list(salarios$age, salarios$husage)
names(ages_list) <- c("Esposas", "Maridos")
par(mfrow=c(1,1),mgp=c(3,2,0))
```

```
boxplot(ages_list, col="#69b3a2", ylab="Idade", main="Gráfico Box-Plot
das Idades")
```

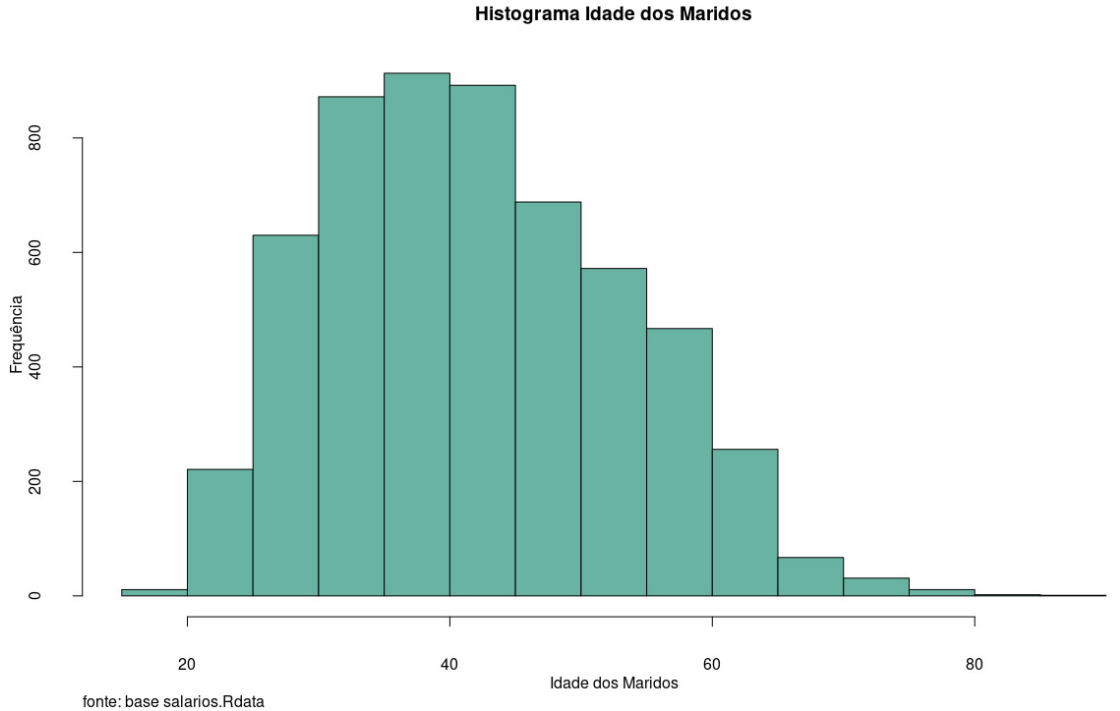


fonte: base salarios.Rdata

```
# Gerar histograma com idades das esposas e dos maridos:
hist(x = salarios$age,xlab = 'Idade das Esposas',ylab = 'Frequência',
col="#69b3a2",
main="Histograma Idade das Esposas")
hist(x=salarios$husage,xlab='Idade dos Maridos',
ylab='Frequência',main="Histograma
Idade dos Maridos",col="#69b3a2")
```



fonte: base salarios.Rdata



fonte: base salarios.Rdata

```
# Complementando com a comparação das informações sumárias:
summary(salarios$age)
Min. 1st Qu. Median Mean 3rd Qu. Max.
18.00 31.00 39.00 39.43 47.00 59.00
summary(salarios$husage)
Min. 1st Qu. Median Mean 3rd Qu. Max.
19.00 34.00 41.00 42.45 50.00 86.00
```

Conclusão:

```
Os maridos possuem idade média superior ao das esposas e tendem a ter mais idade do
que elas. No entanto, a diferença entre as médias e mediana não é expressiva entre ambos.
Algumas observações na amostra analisada indicam outliers nas idades dos maridos que
afetam a média e, em menor grau, poderiam afetar a mediana deste grupo.
```

1.b)

```
# Instalação do pacote necessário:
install.packages("fdth")
library(fdth)
# Gerar tabela de frequência com as idades das esposas:
print(fdt(salarios$age))
```

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

```
# Gerar tabela de frequência com as idades dos maridos:
print(fdt(salarios$husage))
```

Class limits	f	rf	rf(%)	cf	cf(%)
[18.81,23.671)	102	0.02	1.81	102	1.81
[23.671,28.531)	466	0.08	8.27	568	10.08
[28.531,33.392)	809	0.14	14.36	1377	24.44
[33.392,38.253)	895	0.16	15.89	2272	40.33
[38.253,43.114)	917	0.16	16.28	3189	56.60
[43.114,47.974)	629	0.11	11.16	3818	67.77
[47.974,52.835)	649	0.12	11.52	4467	79.29
[52.835,57.696)	541	0.10	9.60	5008	88.89
[57.696,62.556)	394	0.07	6.99	5402	95.88
[62.556,67.417)	152	0.03	2.70	5554	98.58
[67.417,72.278)	51	0.01	0.91	5605	99.49
[72.278,77.139)	21	0.00	0.37	5626	99.86
[77.139,81.999)	6	0.00	0.11	5632	99.96
[81.999,86.86)	2	0.00	0.04	5634	100.00

Conclusão:
Comparando os resultados de ambas tabelas pode-se obter algumas conclusões:

- Observando-se a tabela de frequências das idades das esposas, percebe-se que a frequência maior está entre as idades de 32 a 39 anos. Já a maior frequência dos maridos está entre 28 e 43 anos. Algumas observações referentes aos maridos indicam que pouco mais de 4% deles possuem idade superior a 60 anos, idade esta inferior a máxima idade observada na amostra referente as esposas.

2.a)

```
# Calcular a média da idade das esposas:
```

```
mean(salarios$age)
[1] 39.42758
# Calcular a média da idade dos maridos:
mean(salarios$husage)
[1] 42.45296
# Calcular a mediana da idade das esposas:
median(salarios$age)
[1] 39
# Calcular a mediana da idade das esposas:
median(salarios$husage)
[1] 41
# Calcular as modas das esposas:
table(salarios$age)
subset(table(salarios$age),            table(salarios$age)            ==
max(table(salarios$age)))
37
217
# Calcular as modas dos maridos:
table(salarios$husage)
subset(table(salarios$husage),         table(salarios$husage)         ==
max(table(salarios$husage)))
44
201
```

Tabela resumo:

Item / Grupo	Média	Mediana	Modal	
			Idade	Frequência
Esposas	39.42758	39	37	217
Maridos	42.45296	41	44	201

Conclusão:

- A média das idades dos maridos é 7.67 % maior que a das esposas.
- A mediana das idades dos maridos é 5.13 % maior que a das esposas.
- A moda das idades dos maridos é 18.92 % maior que a das esposas.
- Os resultados indicam que as esposas são, de modo geral, mais jovens que os homens, muito embora a diferença entre ambos os gupos seja pequena;
- Os valores de média e mediana são próximos entre as duas variáveis;

2.b)

```
# Calcular a variância da idade das esposas:
var(salarios$age)
[1] 99.75234
# Calcular a variância da idade dos maridos:
var(salarios$husage)
[1] 126.0717
```

```
# Calcular o desvio padrão das idades das esposas:
sd(salarios$age)
[1] 9.98761
# Calcular o desvio padrão das idades dos maridos:
sd(salarios$husage)
[1] 11.22817
# Calcular o coeficiente de variação da idades das esposas:
(sd(salarios$age)/mean(salarios$age))*100
[1] 25.33153
# Calcular o coeficiente de variação da idades das esposas:
(sd(salarios$husage)/mean(salarios$husage))*100
[1] 26.44849
```

Tabela resumo:

Item de Análise	Esposas	Maridos
Variância	99.75234	126.0717
Desvio padrão	9.98761	11.22817
Coeficiente de variação	25.33153	26.44849

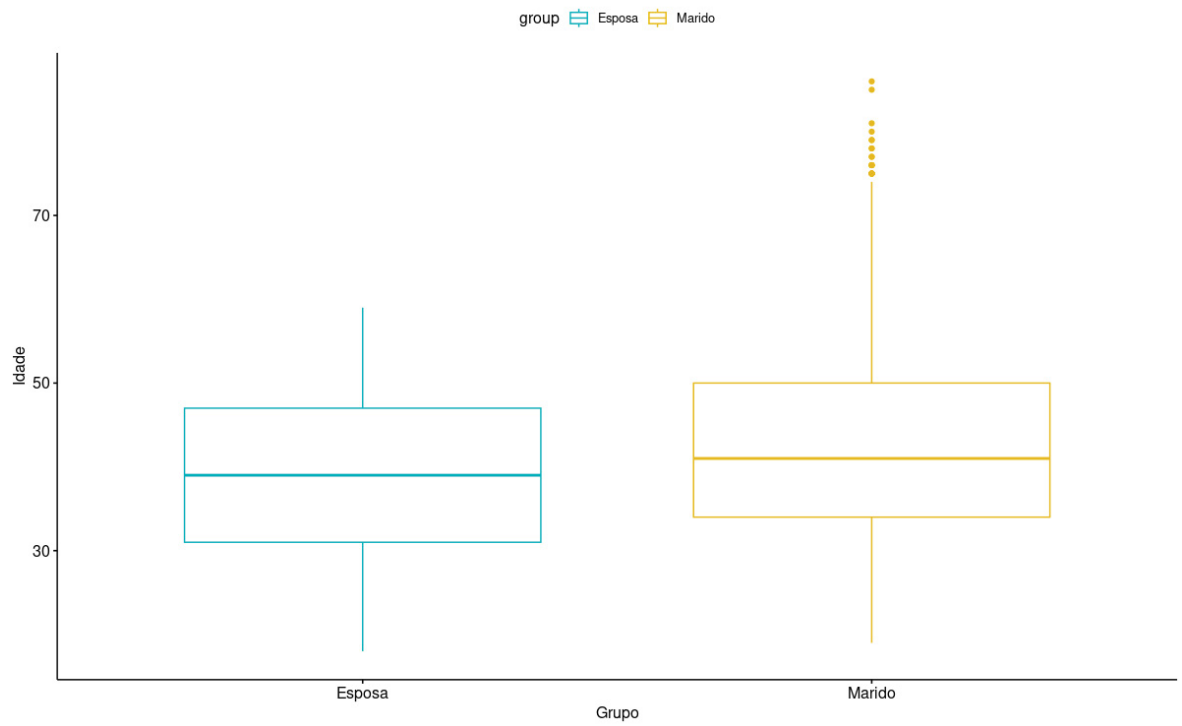
Conclusão:

- A variância das idades dos maridos é 1.26 % maior que a das esposas.
- O desvio padrão das idades dos maridos é 1.12 % maior que a das esposas.
- Os resultados de variância apresentam uma maior distância da idade dos maridos em referência a média padrão deste grupo;
- Comparando os resultados de desvio padrão, as idades dos maridos possuem desvio também maior em relação ao desvio padrão das esposas;
- Com base nos valores de coeficiente de variação e utilizando a "regra de bolso" para análise desse dado, podemos assumir que há uma dispersão média tanto para as idades dos maridos (cv de 26%), quanto das esposas (cv de 25%)

3.a)

```
# Bibliotecas necessárias
library(nortest)
library("ggpubr")
# para evitar notação científica
options(scipen = 999)
# Realizar os testes de Shapiro-Wilk com as seguintes hipóteses:
• H0: Os dados são normalmente distribuídos
• Ha: Os dados não são normalmente distribuídos
with(idades, shapiro.test(idades[grupo == "Marido"]))
```

```
Error in shapiro.test(idades[grupo == "Marido"]) :
sample size must be between 3 and 5000

Conclusão:
A amostra não pode ser testada com o método Shapiro-Wilk, devido o método possuir restrições quanto ao tamanho da amostra; número máximo de até 5000 observações.

# Primeiro vamos fazer o teste de normalidade Kolmogorov-Smirnov para a
# idade das esposas
wife_normality <- lillie.test(salarios$age)
wife_normality
# p-value < 0.05 (valor de 0.00000000000000022), logo não possui
# distribuicao normal
husband_normality <- lillie.test(salarios$husage)
husband_normality
# p-value < 0.05 (valor de 0.00000000000000022), logo não possui
# distribuição normal
# Premissa 3. As duas populações/amostras/grupos possuem
# homogeneidade das variâncias?
# O teste de hipóteses é:
# H0: As variâncias são estatisticamente iguais(homogêneas)
# HA: As variâncias não são estatisticamente iguais(homogêneas)
# criando dataset para comparar idade das esposas da idade dos maridos
idades <- data.frame(group = rep(c("Esposa","Marido"), each =
length(salarios$age)),
idades = c(salarios$age,salarios$husage)
)
# Analisando algumas medidas dos dados
group_by(idades, group) %>%
summarise(
count = n(),
median = median(idades, na.rm = TRUE),
IQR = IQR(idades, na.rm = TRUE)
)
# A tibble: 2 × 4
group count median IQR
<chr> <int> <dbl> <dbl>
1 Esposa 5634 39 16
2 Marido 5634 41 16
# Visualizando boxplot
ggboxplot(idades, x = "group", y = "idades",
color = "group", palette=c("#00AFBB", "#E7B800"),
ylab = "Idade", xlab = "Groups")
```




```
# Usaremos o teste F para testar a homogeneidade nas variâncias.
library("sjPlot")
res.ftest <- var.test(idades ~ group, data = idades)
res.ftest
# Obtendo o valor tabelado da distribuição F
qf(0.95, 5633, 5633)
# temos F=1.04481
# para a outra cauda temos:
1/1.04481
# F = 0.9571118
# Vamos construir o gráfico:
dist_f(f = 1.04481, deg.f1 = 5633, deg.f2 = 5633)
dist_f(f = 0.9571118, deg.f1 = 5633, deg.f2 = 5633)
# O teste de F tem valor crítico entre 0.9571118 e 1.04481 (região de
# não rejeição de H0), os valores acima de 1.04481 e abaixo de
# 0.9571118 estão na região de rejeição de H0.
# O valor da estatística F calculada é de F = 0.79123. Como esse valor
# se encontra na região de rejeição de H0, então rejeitamos a hipótese de
que as variâncias são
# estatisticamente iguais, ou seja, adotamos HA.
# Como não foi verificado normalidade nos dados e nem homogeneidade da
variância, será
adotado um teste não paramétrico
#  e será feito teste para verificar se as medianas das variáveis
age(Idade Esposas) e
# husage (Idade Maridos) são iguais
# Teste Não Paramétrico: Teste U de Mann-Whitney
#Hipóteses
# H0: O idade mediana dos homens é igual estatisticamente a idade
# mediana das mulheres;
```

```
# Ha: O idade mediana dos homens não é estatisticamente igual a
# idade mediana das mulheres
# Executando teste
res <- wilcox.test(idades ~ group, data = idades,
exact = FALSE, conf.int=TRUE)
res
data: idades by group
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
-3.000024 -2.000033
sample estimates:
difference in location
-2.999966
# O p-value do teste eh 0.00000000000000022 que é menor que o nível de
# significância 0,05. Podemos concluir que a idade mediana dos
# homens é estatisticamente diferente da idade mediana das
# mulheres, ou seja, rejeitamos H0, e adotamos HA.
#  A diferença na localização, a diferença nas medianas, é estimada em
-2.999966.
# O intervalo de confiança de 95% para a diferença nas localizações é de
-3.000024 a -
2.000033.
```

APÊNDICE E – ESTATÍSTICA APLICADA II

A – ENUNCIADO

1) 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

1.

```
# Carregando os pacotes necessarios
library(plyr)
library(readr)
library(dplyr)
library(caret)
library(ggplot2)
```

```
library(repr)
library(glmnet)

set.seed(123)

# Carregando base de dados
load('trabalhosalarios.RData')

# armazenando base de dados
dataset <- trabalhosalarios

# diminuindo a variancia de husage e age - as outras possuem zero
dataset['husage'] <- log(dataset['husage'])
dataset['age'] <- log(dataset['age'])

# criando índice para separar em base de teste e de treino
index <- sample(1:nrow(dataset), 0.8*nrow(dataset))

# criando base de treino
train <- dataset[index,]

# criando base de teste
test <- dataset[-index,]

# verificando dimensões das bases de treino e teste
dim(train)
dim(test)

# Vamos padronizar as variaveis
# Vamos criar um objeto com as variaveis para padronizar
# As variaveis binarias nao sao padronizadas
cols = c('husage', 'husearns', 'huseduc', 'hushrs',
      'age', 'educ', 'exper')

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

summary(train)
summary(test)

#############################################################
#                  REGRESSAO RIDGE                          #
#############################################################

# Vamos criar um objeto com as variaveis que usaremos no
# modelo
cols_reg = c('husage','husunion', 'husearns', 'huseduc', 'hushrs',
             'age', 'educ','husblck',
             'hushisp', 'kidge6', 'black', 'hispanic',
```

```
            'union', 'kidlt6','exper','lwage')

# Vamos gerar variaveis dummies para organizar os datasets
# em objetos tipo matriz
# Estamos interessados em estimar o salario-hora da esposa em logaritmo
neperiano (lwage)
dummies <- dummyVars(lwage~husage+husunion+husearns+huseduc+hushrs+
                         age+educ+husblck+hushisp+
                         kidge6+black+hispanic+union+kidlt6+exper,
                   data = dataset[,cols_reg])
train_dummies = predict(dummies, newdata = train[,cols_reg])
test_dummies = predict(dummies, newdata = test[,cols_reg])
print(dim(train_dummies)); print(dim(test_dummies))

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

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

# Vamos guardar a matriz de dados de teste das variaveis
# 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 variavel
# dependente para o modelo em um objeto chamado "y_test"
y_test = test$lwage

# Vamos calcular o valor otimo de lambda;
# alpha = "0", eh para regressao Ridge
# Vamos testar os lambdas de 10^-3 ate 10^2, a cada 0.1
lambdas <- 10^seq(2, -3, by = -.1)
# Calculando o lambda:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
                         lambda = lambdas)
# Vamos ver qual o lambda otimo
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge

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

# Vamos ver o resultado (valores) da estimativa
# (coeficientes)
ridge_reg[["beta"]]
```

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

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

# Predicao e avaliacao nos dados de treinamento:
predictions_train <- predict(ridge_reg,
                             s = best_lambda_ridge,
                             newx = x)

# As metricas da base de treinamento sao:
eval_results(y_train, predictions_train, train)
#  RMSE     Rsquare
#1 0.4361332 0.306454

# Predicao e avaliacao nos dados de teste:
predictions_test <- predict(ridge_reg,
                        s = best_lambda_ridge,
                        newx = x_test)

# As metricas da base de teste sao:
eval_results(y_test, predictions_test, test)
#  RMSE     Rsquare
#1 0.4503715 0.2346912

# Se compararmos as metricas de treinamento e teste
# percebemos que o R² é relativamente baixo em ambas, o que sugere
# que o modelo não está adequado para capturara variabilidade
# dos dados. Em resumo, o modelo não demonstra
# ter um bom poder explicativo.

#############################################################
#                 REGRESSAO LASSO                           #
#############################################################

# Vamos atribuir alpha = 1 para implementar a regressao
# lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,
                        lambda = lambdas,
                        standardize = TRUE, nfolds = 5)
```

```
# Vamos guardar o lambda "otimo" em um objeto chamado
# best_lambda_lasso
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso

# Vamos estimar o modelo Lasso
lasso_model <- glmnet(x, y_train, alpha = 1,
                  lambda = best_lambda_lasso,
                  standardize = TRUE)

# Vamos visualizar os coeficientes estimados
lasso_model[["beta"]]

# Vamos fazer as predicoes na base de treinamento e
# avaliar a regressao Lasso
predictions_train_lasso <- predict(lasso_model,
                                   s = best_lambda_lasso,
                                   newx = x)

# Vamos calcular o R^2 dos valores verdadeiros e
# preditos

# As metricas da base de treinamento sao:
eval_results(y_train, predictions_train_lasso, train)
#  RMSE     Rsquare
#1 0.436358 0.305739

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

# As metricas da base de teste sao:
eval_results(y_test, predictions_test_lasso, test)
#   RMSE   Rsquare
#1  0.4503179 0.2348735

# Novamente, se compararmos as metricas de treinamento e teste
# percebemos que o R² é relativamente baixo em ambas, o que sugere
# que este modelo também não está adequado para capturara variabilidade
# dos dados. Em resumo, o modelo não demonstra
# ter um bom poder explicativo.


#################################################################
#                 REGRESSAO ELASTICNET                          #
#################################################################

# Vamos configurar o treinamento do modelo por
# cross validation, com 10 folders, 5 repeticoes
# e busca aleatoria dos componentes das amostras
# de treinamento, o "verboseIter" eh soh para
```

```
# mostrar o processamento.
train_cont <- trainControl(method = "repeatedcv",
                           number = 10,
                           repeats = 5,
                           search = "random",
                           verboseIter = TRUE)

# Vamos treinar o modelo
elastic_reg <- train(lwage~husage+husunion+husearns+huseduc+hushrs+
                           age+educ+husblck+hushisp+
                           kidge6+black+hispanic+union+kidlt6+exper,
                     data = train,
                     method = "glmnet",
                     tuneLength = 10,
                     trControl = train_cont)

# O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

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

# Vamos fazer as predicoes e avaliar a performance do
# modelo

# Vamos fazer as predicoes no modelo de treinamento:
predictions_train_elasticnet <- predict(elastic_reg, x)

# As metricas de performance na base de treinamento
# sao:
eval_results(y_train, predictions_train_elasticnet, train)
#   RMSE    Rsquare
# 1 0.437097 0.3033854

# Vamos fazer as predicoes na base de teste
predictions_test_elasticnet <- predict(elastic_reg, x_test)

# As metricas de performance na base de teste sao:
eval_results(y_test, predictions_test_elasticnet, test)
#  RMSE     Rsquare
#1 0.4508652 0.23301243

# O modelo com elasticnet também apresentou métricas de desempenho
# relativamente baixas, tanto para treino quanto para teste.
# Em resumo, o modelo não demonstra
# ter um bom poder explicativo.

######################################################################
# Escolha do melhor modelo
# O modelo de Lasso parece ser a melhor escolha, pois apresenta o menor
RMSE
# e o maior R² na base de teste, indicando uma melhor capacidade de
```

```
previsão e
# explicação da variabilidade dos dados de teste em comparação aos outros
modelos.

#No entanto, as diferenças são mínimas, e todos os modelos apresentam
desempenho muito semelhante.

######################################################################
# Preparando valores para as predições

# husage = 40 anos (idade do marido)
husage = (log(40)-pre_proc_val[["mean"]][["husage"]])/
  pre_proc_val[["std"]][["husage"]]

husunion = 0

# husearns = 600 (rendimento do marido em US$)
husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
  pre_proc_val[["std"]][["husearns"]]

# huseduc = 13 (anos de estudo do marido)
huseduc = (13-pre_proc_val[["mean"]][["huseduc"]])/
  pre_proc_val[["std"]][["huseduc"]]

# husblck = 1 (o marido eh preto)
husblck = 1

# hushisp = 0 (o marido nao eh hispanico)
hushisp = 0

# hushrs = 40 (o marido trabalha 40 horas semanais)
hushrs = (40-pre_proc_val[["mean"]][["hushrs"]])/
  pre_proc_val[["std"]][["hushrs"]]

# kidge6 = 0 (nao tem filhos maiores de 6 anos)
kidge6 = 1

# age = 38 anos (idade da esposa)
age = (log(38)-pre_proc_val[["mean"]][["age"]])/
  pre_proc_val[["std"]][["age"]]

# black = 0 (esposa nao eh preta)
black = 0

# educ = 13 (esposa possui 13 anos de estudo)
educ = (13-pre_proc_val[["mean"]][["educ"]])/
  pre_proc_val[["std"]][["educ"]]

# hispanic = 1 (esposa eh hispanica)
hispanic = 1

# union = 0 (o casal nao possui uniao registrada)
```

```
union = 0

# exper = 18 (esposa possui 18 anos de experiência)
exper = (18-pre_proc_val[["mean"]][["exper"]])/
  pre_proc_val[["std"]][["exper"]]

# kidlt6 = 0 (nao possui filhos com menos de 6 anos)
kidlt6 = 1

# Vamos construir uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(husage=husage,
                                husunion=husunion,
                                husearns=husearns,
                                huseduc=huseduc,
                                husblck=husblck,
                                hushisp=hushisp,
                                hushrs=hushrs,
                                kidge6=kidge6,
                                age=age,
                                black=black,
                                educ=educ,
                                hispanic=hispanic,
                                union=union,
                                exper=exper,
                                kidlt6=kidlt6))

#############################################################
#                  PREDIÇÃO RIDGE                           #
#############################################################
# Fazendo a predicao:
predict_our_ridge <- predict(ridge_reg,
                                s = best_lambda_ridge,
                                newx = our_pred)
# O resultado da predicao eh:
predict_our_ridge

# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
lwage_pred_ridge=exp(predict_our_ridge)

# O resultado eh:
lwage_pred_ridge
# Este eh o valor predito do salario por hora (US$),
# segundo as caracteristicas que atribuimos

# O intervalo de confianca para o nosso exemplo eh:
n <- nrow(train) # tamanho da amostra
m <- lwage_pred_ridge # valor medio predito
s <- sd(dataset$lwage) # desvio padrao
dam <- s/sqrt(n) # distribuicao da amostragem da media
CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior
```

```
# Os valores sao:
CIlwr_ridge
CIupr_ridge

# Entao, segundo as caracteristicas que atribuimos o
# salario-hora da esposa é em media US$6.27305 e pode
# variar entre US$6.2505 e US$6.295599

################################################################
#                  PREDIÇÃO LASSO                              #
################################################################

# Fazendo a predição
predict_our_lasso <- predict(lasso_model,
                             s = best_lambda_lasso,
                             newx = our_pred)
# O resultado da predicao eh:
predict_our_lasso

# O resultado eh um valor padronizado, vamos converte-lo
# para o valor nominal, consistente com o dataset original
lwage_pred_lasso = exp(predict_our_lasso)

# O resultado eh:
lwage_pred_lasso

# Vamos criar o intervalo de confianca para o nosso
# exemplo
n <- nrow(train)
m <- lwage_pred_lasso
s <- sd(dataset$lwage) # desvio padrao
dam <- s/sqrt(n)
CIlwr_lasso <- m + (qnorm(0.025))*dam
CIupr_lasso <- m - (qnorm(0.025))*dam

# O intervalo de confianca eh:
CIlwr_lasso
CIupr_lasso
# Entao, o salario medio eh de US$6.253206 e pode variar
# entre US$6.230657 e US$6.275756

################################################################
#                  PREDIÇÃO ELASTICNET                         #
################################################################
# Vamos fazer a predicao com base nos parametros que
# selecionamos
predict_our_elastic <- predict(elastic_reg,our_pred)
predict_our_elastic

# Novamente, o resultado eh padronizado, nos temos que
# reverte-lo para o nivel dos valores originais do
```

```
# dataset, vamos fazer isso:
lwage_pred_elastic=exp(predict_our_elastic)
lwage_pred_elastic
# Entao o salario-hora medio da esposa predito com base
# nas caracteristicas informadas eh US$7.8397

# Vamos criar o intervalo de confianca para o nosso
# exemplo
n <- nrow(train)
m <- lwage_pred_elastic
s <- sd(dataset$lwage) # desvio padrao
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam

# Os valores minimo e maximo sao:
CIlwr_elastic
CIupr_elastic
# Entao, o salario-hora medio da esposa eh de US$7.8397
# e pode variar entre US$7.81715 e US$7.86225
```

2.

Tipo de Regressão	Treino		Teste	
	R^2	RMSE	R^2	RMSE
Ridge	0.306454	0.4361332	0.2346912	0.4503715
Lasso	0.305739	0.436358	0.2348735	0.4503179
ElasticNet	0.3033854	0.437097	0.23301243	0.4508652

Escolha do melhor modelo:

O modelo de Lasso parece ser a melhor escolha, pois apresenta o menor RMSE e o maior R2 na base de teste, indicando uma melhor capacidade de previsão e explicação da variabilidade dos dados de teste em comparação aos outros modelos.

No entanto, as diferenças são mínimas, e todos os modelos apresentam desempenho muito semelhante.

3.

Tipo de Regressão	Resultado	Resultado com antilog	Intervalo Inferior	Intervalo Superior
Ridge	1.836263	6.27305	6.2505	6.295599

Lasso	1.833094	6.253206	6.230657	6.275756
ElasticNet	2.059201	7.8397	7.81715	7.86225

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.

Construção de Características – Identificador Automático de Idiomas

Resultado dos testes após adição de novas características:

Acurácia	
No treino	74,63%
No teste	70,59%

Idioma	Precision	Recall	F1-Score	Support
Inglês (0)	0.60	1.00	0.75	3
Espanhol (1)	0.67	0.40	0.50	5
Português (2)	0.78	0.78	0.78	9
accuracy			0.71	17
macro avg	0.68	0.73	0.68	17
weigthed avg	0.71	0.71	0.69	17

```
import re
from collections import Counter
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.model_selection import RepeatedStratifiedKFold,
cross_val_score

ingles = [
"Hello, how are you?",
"I love to read books.",
"The weather is nice today.",
"Where is the nearest restaurant?",
```

```
"What time is it?",
"I enjoy playing soccer.",
"Can you help me with this?",
"I'm going to the movies tonight.",
"This is a beautiful place.",
"I like listening to music.",
"Do you speak English?",
"What is your favorite color?",
"I'm learning to play the guitar.",
"Have a great day!",
"I need to buy some groceries.",
"Let's go for a walk.",
"How was your weekend?",
"I'm excited for the concert.",
"Could you pass me the salt, please?",
"I have a meeting at 2 PM.",
"I'm planning a vacation.",
"She sings beautifully.",
"The cat is sleeping.",
"I want to learn French.",
"I enjoy going to the beach.",
"Where can I find a taxi?",
"I'm sorry for the inconvenience.",
"I'm studying for my exams.",
"I like to cook dinner at home.",
"Do you have any recommendations for restaurants?",
]

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

```
"Quiero aprender italiano.",
"Disfruto ir a la playa.",
"¿Dónde puedo encontrar un taxi?",
"Lamento las molestias.",
"Estoy estudiando para mis exámenes.",
"Me gusta cocinar la cena en casa.",
"¿Tienes alguna recomendación de restaurantes?",
]

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


import random

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

for frase in espanhol:
```

```
  pre_padroes.append( [frase, 'espanhol'])

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

random.shuffle(pre_padroes)

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

def tamanhoMedioFrases(texto):
      palavras = re.split("\s", texto)
      tamanhos = [len(s) for s in palavras if len(s) > 0]
      return sum(tamanhos) / len(tamanhos)

def tamanho_frase(frase):
      return len(frase.split())

def encontros_pt(frase):
  encontros = ['ss','rr', 'ão']
  if any(char in frase for char in encontros):
      return 1
  else:
      return 0

def art_prep_espanhol(frase):
  artigos_pre_espanhois = ['el', 'la', 'los', 'las', 'un', 'una', 'unos',
'unas', 'lo'
  "bajo","en","hacia", "hasta","según", "sin","vía"
  ]
  palavras = re.split('\s', frase.lower())

  if any(artigo in palavras for artigo in artigos_pre_espanhois):
      return 1
  else:
      return 0


def caracter_espanhol(frase):
  caracteres_espanhois = ['Ñ','ñ', '¡', '¿']
  if any(char in frase for char in caracteres_espanhois):
      return 1
  else:
      return 0

def frequencia_letras(frase):
      letras = re.findall(r'\w', frase.lower())
      contador = Counter(letras)
      total_letras = sum(contador.values())
      frequencia = {f'freq_{letra}': count / total_letras for letra,
count in contador.items()}
```

```
      return frequencia

def ocorrencia_simbolos_especiais(frase):
      especiais = re.findall(r'[ñç]', frase.lower())
      contador = Counter(especiais)
      frequencia = {f'simbolo_{s}': count for s, count in
contador.items()}
      return frequencia

def sufixos_palavras(frase, tamanho=3):
      palavras = frase.split()
      sufixos = [palavra[-tamanho:] for palavra in palavras if
len(palavra) >= tamanho]
      contador = Counter(sufixos)
      frequencia = {f'sufixo_{s}': count for s, count in
contador.items()}
      return frequencia

def encontros_es(frase):
  encontros = ['ll','ch', 'qu']
  if any(char in frase for char in encontros):
      return 1
  else:
      return 0

def encontros_ing(frase):
  encontros = ['ll','mm', 'aa','ee', 'ii', 'oo', 'uu', 'ff']
  if any(char in frase for char in encontros):
      return 1
  else:
      return 0

def extraiCaracteristicas(frase):
      texto = frase[0]
      pattern_regex = re.compile('[^\w+]', re.UNICODE)
      texto = re.sub(pattern_regex, ' ', texto)

      caracteristica1 = tamanhoMedioFrases(texto)
      caracteristica2 = tamanho_frase(texto)
      caracteristica4 = ocorrencia_simbolos_especiais(texto)
      caracteristica6 = sufixos_palavras(texto)
      caracteristica8 = art_prep_espanhol(texto)
      caracteristica9 = encontros_pt(texto)
      caracteristica10 = encontros_es(texto)
      caracteristica11 = encontros_ing(texto)
      caracteristica12 = caracter_espanhol(texto)

      padrao = {
      'tamanhoMedioFrases': caracteristica1,
      'tamanho_frase': caracteristica2,
      **caracteristica4,
      **caracteristica6,
```

```
      'pre-espanhol':caracteristica8,
      'en-pt':caracteristica9,
      'en-es':caracteristica10,
      'en-en':caracteristica11,
      'es':caracteristica12,
      'classe': frase[1]
      }

      return padrao

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

padroes = geraPadroes(pre_padroes)
dados = pd.DataFrame(padroes)

colunas_numericas = dados.select_dtypes(include=[np.number]).columns
imputer = SimpleImputer(strategy='mean')
dados[colunas_numericas] =
imputer.fit_transform(dados[colunas_numericas])

scaler = StandardScaler()
dados[colunas_numericas] = scaler.fit_transform(dados[colunas_numericas])

dados = dados.drop_duplicates()
dados = dados.dropna()

X = dados.drop(columns=['classe'])
y = dados['classe']

class_map = {'inglês': 0, 'espanhol': 1, 'português': 2}
y_encoded = y.map(class_map)

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
pca = PCA(n_components=0.95)
X_pca = pca.fit_transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_pca, y_encoded,
test_size=0.2, random_state=42)
modelo = SVC()
modelo.fit(X_train, y_train)

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
scores = cross_val_score(modelo, X_train, y_train, scoring='accuracy',
cv=cv, n_jobs=-1)

acuracia_treinamento = modelo.score(X_train, y_train)
```

```
print("Acurácia no treino: {:.2f}%".format(acuracia_treinamento * 100))
acuracia_teste = modelo.score(X_test, y_test)
print("Acurácia no teste: {:.2f}%".format(acuracia_teste * 100))

y_pred = modelo.predict(X_test)
print(classification_report(y_test, y_pred))
```

2. Melhore uma base de dados ruim:

Para realização do trabalho foi utilizada a base de referência “UCI-Predict students dropout and academic success” disponível em:

https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success

a) Matriz de confusão obtida após treinar e testar modelo SVM com o mínimo de intervenção:

Acurácia no treino: 50%

Classe	Precision	recall	F1-score	Support
Dropout	0.0	0.0	0.0	1066
Enrolled	0.0	0.0	0.0	595
Graduate	0.50	1.0	0.67	1657
Accuracy			0.50	3318
macro avg	0.17	0.33	0.22	3318
weighted avg	0.25	0.50	0.33	3318

Acurácia no teste: 50%

Classe	Precision	recall	F1-score	Support
Dropout	0.0	0.0	0.0	355
Enrolled	0.0	0.0	0.0	199
Graduate	0.50	1.0	0.67	552

Accuracy			0.50	1106
macro avg	0.17	0.33	0.22	1106
weighted avg	0.25	0.50	0.33	1106

b) Matriz de confusão obtida após treinar e testar modelo SVM com dados tratados:

Acurácia no treino: 77%

Classe	**Precision**	**recall**	**F1-score**	**Support**
Dropout	0.87	0.70	0.78	1065
Enrolled	0.66	0.74	0.70	900
Graduate	0.78	0.85	0.81	1050
Accuracy			0.77	3015
macro avg	0.77	0.76	0.76	3015
weighted avg	0.77	0.77	0.77	3015

Acurácia no teste: 70%

Classe	**Precision**	**recall**	**F1-score**	**Support**
Dropout	0.82	0.63	0.71	356
Enrolled	0.58	0.67	0.62	300
Graduate	0.72	0.80	0.76	350
Accuracy			0.70	1006
macro avg	0.71	0.70	0.70	1006
weighted avg	0.71	0.70	0.70	1006

```
# Instalando pacotes necessários
!pip install pandas
!pip install seaborn
!pip install scikit-learn
!pip install numpy
!pip install matplotlib
!pip install ucimlrepo

# Bibliotecas necessárias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.utils import resample
from sklearn.utils import shuffle
from ucimlrepo import fetch_ucirepo

# Carregando base de dados
predict_students_dropout_and_academic_success = fetch_ucirepo(id=697)

X = predict_students_dropout_and_academic_success.data.features
y = predict_students_dropout_and_academic_success.data.targets

# Salvando as colunas
columns = X.columns
columns

# Verificando informações sobre o dataset
# metadata
print(predict_students_dropout_and_academic_success.metadata)

# variable information
print(predict_students_dropout_and_academic_success.variables)

# Executando o SVM com o mínimo de intervenção
X_original = X.copy()
y_original = y.copy()

X_original_train, X_original_test, y_original_train, y_original_test =
train_test_split(X_original,
                 y_original, test_size=0.25,
```

```
stratify=y_original,random_state=10)

treinador = svm.SVC()

modelo_orig = treinador.fit(X_original_train, y_original_train)

# treinamento
y_original_pred = modelo_orig.predict(X_original_train)
cm_orig_train = confusion_matrix(y_original_train, y_original_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no
TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_original_train, y_original_pred))

# teste
print('Matriz de confusão - com os dados ORIGINAIS usados para
TESTES')
y2_original_pred = modelo_orig.predict(X_original_test)
cm_orig_test = confusion_matrix(y_original_test, y2_original_pred)
print(cm_orig_test)
print(classification_report(y_original_test, y2_original_pred))


## Aplicando tarefas de pré processamento
# Verificando se o dataset contém valores ausentes
X.isna().sum()

# verificando tipos das variáveis
X.dtypes

# Verificando valores das variáveis independentes
X.head()

# Aplicando NORMALIZAÇÃO nas variáveis independentes, que não são
booleanas, para ajustar a escala dos valores
bool_columns = ['International','Displaced','Educational special
needs','Debtor','Tuition fees up to date','Gender','Scholarship
holder','Daytime/evening attendance']

non_bool_columns = X.columns.difference(bool_columns)

scaler = StandardScaler()
X_scaled = X.copy()
X_scaled[non_bool_columns] =
scaler.fit_transform(X_scaled[non_bool_columns])
X_scaled

# Verificando relação das variáveis
plt.figure(figsize=(40,20))
sns.heatmap(X_scaled.corr("spearman"), annot = True, cmap="coolwarm")
plt.title("Mapa de Correlação das Variáveis\n", fontsize = 15)
plt.show()
```

```
# Para tratar correlação e evitar possíveis problemas com
multicolinearidade, será aplicado PCA para SELEÇÃO dos atributos
relavantes para passar ao modelo
pca = PCA(n_components=0.95)

# Aplicar o PCA aos dados
X_pca = pca.fit_transform(X_scaled)

# Criar um DataFrame com os componentes principais
columns = [f"PC{i+1}" for i in range(X_pca.shape[1])]
X_pca_df = pd.DataFrame(data=X_pca, columns=columns)

# Visualizar o DataFrame resultante
print(X_pca_df.head())

# Verificando variável dependente
y

# Verificando distribuição das classes
y.value_counts()

# CORREÇÃO DE PREVALÊNCIA
df = pd.concat([X_pca_df, y], axis=1)
df

# separando num df os dados de classe dropout, visto que irá manter a
quantidade
df_dropout = df[df['Target'] == 'Dropout']

# Separar os exemplos por classe onde será realizado correção de
prevalência
# por replicação ou remoção
df_majority = df[df['Target'] == 'Graduate']
df_minority = df[df['Target'] == 'Enrolled']

# Definir o número desejado de exemplos para cada classe
desired_majority = 1400
desired_minority = 1200

# Realizar oversampling da classe minoritária ("Enrolled")
df_minority_oversampled = resample(df_minority,
                                   replace=True,     # Replicar
exemplos (com reposição)
                                   n_samples=desired_minority,   #
Número desejado de exemplos
                                   random_state=42)  # Seed para
reproduzibilidade

# Realizar undersampling da classe majoritária ("Graduate")
df_majority_undersampled = resample(df_majority,
                                   replace=False,    # Não replicar
```

```
exemplos (sem reposição)
                                        n_samples=desired_majority,   #
Número desejado de exemplos
                                        random_state=42)  # Seed para
reproduzibilidade

# Concatenar os DataFrames resultantes
X_balanced = pd.concat([df_majority_undersampled,
df_minority_oversampled, df_dropout])

# Embaralhar os dados
X_balanced = shuffle(X_balanced, random_state=42)

# Verificando dataframe final
X_balanced

# Separar novamente entre X e y para aplicar no modelo
X_trated = X_balanced.drop(columns='Target', axis=1)
y_treated = X_balanced['Target']

# com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X_trated,
y_treated, test_size=0.25,

stratify=y_treated,random_state=10)

# Treinando e testando o modelo
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 TRATADOS 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))
```

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

Seed utilizado: 202401

1. **Classificação**

Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	C=50 Sigma=0.015	0.8529	Reference Prediction bus opel saab van bus 41 0 2 0 opel 0 32 8 0 saab 0 10 35 0 van 4 0 1 37
SVM – Hold-out	C=100 Sigma=0.015	0.8471	Reference Prediction bus opel saab van bus 42 0 1 1 opel 0 32 10 0 saab 0 10 34 0 van 3 0 1 36
RNA – CV	size=31 decay=0.1	0.8235	Reference Prediction bus opel saab van bus 42 1 0 0 opel 0 28 12 0 saab 0 13 33 0 van 3 0 1 37
RF – Hold-out	mtry=5	0.7294	Reference Prediction bus opel saab van bus 42 1 3 0 opel 0 20 14 0 saab 0 18 25 0 van 3 3 4 37

RF – CV	mtry=7	0.7235	Reference Prediction bus opel saab van bus 42 2 3 0 opel 0 19 14 0 saab 0 18 25 0 van 3 3 4 37
KNN	k=1	0.7059	Reference Prediction bus opel saab van bus 43 1 7 2 opel 0 19 13 0 saab 1 19 24 1 van 1 3 2 34
RNA – Hold-out	size=41 decay=0.7	0.6647	Reference Prediction bus opel saab van bus 41 2 6 0 opel 0 6 7 1 saab 2 34 31 1 van 2 0 2 35

Predição para novos casos: Melhor acurácia foi obtida pelo SVM com Cross Validation.

Script em R:

```
#######################################
# CLASSIFICAÇÃO  Veículo                #
#######################################
install.packages("e1071")
install.packages("caret")
install.packages("Metrics")
install.packages("mice")
install.packages("kernlab")
library(mice)
library("caret")
library(kernlab)

# Veículos

# ler arquivo
#setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/class
ificacao-veiculos")
dados <- read.csv("6 - Veiculos - Dados.csv")
# remover atributo desnecessário
```

```
dados$a <- NULL

# separação entre base de treino e teste que serão utilizadas em todos os
modelos
set.seed(202401)
randomIndexes <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[randomIndexes,]
teste <- dados[-randomIndexes,]

## KNN
# grid com opções de K
set.seed(202401)
tuneGrid <- expand.grid(k = c(1,3,5,7,9))

set.seed(202401)
knn <- train(tipo ~ ., data = treino, method = "knn",tuneGrid=tuneGrid)
knn

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

## RNA
### Treinar o modelo com Hold-out
### size, decay
set.seed(202401)
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
### treino com hold-out
rna <- train(tipo~., data=treino, method="nnet",trace=FALSE,  tuneGrid =
grid)
rna

### Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)
### Matriz de confusão
confusionMatrix(predict.rna, as.factor(teste$tipo))

## Treinar o modelo com cross validation CV
### indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)
### executa a RNA com esse ctrl
rna <- train(
  form = tipo~. ,
  data = treino ,
  method = "nnet" ,
  tuneGrid = grid ,
  trControl = ctrl ,
  maxit = 2000,trace=FALSE)
rna

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

```
## SVM
### C e sigma
set.seed(202401)
tuneGrid_svm = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))

### Treinar SVM com a base de Treino
# com Hold-out
svm <- train(tipo~., data=treino, method="svmRadial",
tuneGrid=tuneGrid_svm)
svm

### Aplicar modelos treinados na base de Teste
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

#### Cross-validation SVM
ctrl_svm <- trainControl(method = "cv", number = 10)
set.seed(202401)
svm_cv <- train(tipo~., data=treino, method="svmRadial",
trControl=ctrl_svm, tuneGrid=tuneGrid_svm)
svm_cv

### Matriz de confusão
predict.svm_cv <- predict(svm_cv, teste)
confusionMatrix(predict.svm_cv, as.factor(teste$tipo))

## Random Forest
#### mtry
set.seed(202401)
tuneGridRf = expand.grid(mtry=c(2, 5, 7, 9))

# treinar com hold out
rf <- train(tipo~., data=treino, method="rf", tuneGrid=tuneGridRf)
rf
### Aplicar modelos treinados na base de Teste
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$tipo))

#### treinar com cross-validation
set.seed(202401)
ctrlRf <- trainControl(method = "cv", number = 10)
rf_cv <- train(tipo~., data=treino, method="rf", tuneGrid=tuneGridRf,
trControl=ctrlRf)
rf_cv
### Aplicar modelos treinados na base de Teste
predict.rf_cv <- predict(rf_cv, teste)
confusionMatrix(predict.rf_cv, as.factor(teste$tipo))

## Resultado -> SVM com Cross Validation obteve melhor acurácia
novos_dados <- read.csv('6-Veiculos- Novos-Dados.csv')
```

```
novos_dados$a <- NULL

predict.novos_svm <- predict(svm_cv, novos_dados)
resultado <- cbind(novos_dados, predict.novos_svm)
View(resultado)
```

Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	0.8117	Reference Prediction neg pos neg 102 21 pos 8 23
SVM – Hold-out	C=2 Sigma=0.01	0.7987	Reference Prediction neg pos neg 102 23 pos 8 21
RF – CV	mtry=7	0.7987	Reference Prediction neg pos neg 99 20 pos 11 24
SVM – CV	C=2 Sigma=0.015	0.7922	Reference Prediction neg pos neg 101 23 pos 9 21
KNN	k=9	0.7468	Reference Prediction neg pos neg 94 23 pos 16 21
RNA – CV	size=21 decay=0.1	0.7143	Reference Prediction neg pos neg 90 24 pos 20 20

RNA – Hold-out	size=31 decay=0.7	0.6429	Reference Prediction neg pos neg 90 35 pos 20 9

Predição para novos casos: O Random Forest com Hold-Out obteve o melhor desempenho, logo, os novos casos serão preditos com Random Forest com Hold-Out.

pregOnt	glucose	pressure	triceps	insulin	mass	pedigree	age	predict.novos_rf_h
5	133	74	35	0	33.6	627	56	pos
1	85	66	29	0	26.6	351	41	neg
9	175	63	0	0	23.3	672	34	pos

Script em R:

```
#######################################
# CLASSIFICAÇÃO  Diabetes              #
#######################################
install.packages("e1071")
install.packages("caret")
install.packages("Metrics")
install.packages("mice")
install.packages("kernlab")
library(mice)
library("caret")
library(kernlab)

set.seed(202401)

# Diabetes
#setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/class
ificacao-diabetes")
dados <- read.csv("10 - Diabetes - Dados.csv")
# remover variável desnecessária
dados$num <- NULL

set.seed(202401)
randomIndexes <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[randomIndexes,]
teste <- dados[-randomIndexes,]

## KNN
# grid com opções de K
tuneGrid_knn <- expand.grid(k = c(1,3,5,7,9))

knn <- train(diabetes ~ ., data = treino, method =
"knn",tuneGrid=tuneGrid_knn)
knn

predict.knn <- predict(knn, teste)
```

```
confusionMatrix(predict.knn, as.factor(teste$diabetes))

## RNA
### Treinar o modelo com Hold-out
### size, decay
set.seed(202401)
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE,
tuneGrid = grid)
rna

### Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)
### Matriz de confusão
confusionMatrix(predict.rna, as.factor(teste$diabetes))

## Treinar o modelo com cross validation CV
### indica o método cv e numero de folders 10
set.seed(202401)
ctrl_rna <- trainControl(method = "cv", number = 10)
### executa a RNA com esse ctrl
rna_cv <- train(
  form = diabetes~. ,
  data = treino ,
  method = "nnet" ,
  tuneGrid = grid ,
  trControl = ctrl_rna ,
  maxit = 2000,trace=FALSE)
rna_cv

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

## SVM
### C e sigma
set.seed(202401)
tuneGrid_svm = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))

### Treinar SVM com a base de Treino
# com Hold-out
set.seed(202401)
svm_h <- train(diabetes~., data=treino, method="svmRadial",
tuneGrid=tuneGrid_svm)
svm_h

### Aplicar modelos treinados na base de Teste
predict.svm_h <- predict(svm_h, teste)
confusionMatrix(predict.svm_h, as.factor(teste$diabetes))

#### Cross-validation SVM
```

```
set.seed(202401)
ctrl_svm <- trainControl(method = "cv", number = 10)
svm_cv <- train(diabetes~., data=treino, method="svmRadial",
trControl=ctrl_svm, tuneGrid=tuneGrid_svm)
svm_cv

### Matriz de confusão
predict.svm_cv <- predict(svm_cv, teste)
confusionMatrix(predict.svm_cv, as.factor(teste$diabetes))

## Random Forest
#### mtry
set.seed(202401)
tuneGrid_rf = expand.grid(mtry=c(2, 5, 7, 9))

# treinar com hold out
rf_h <- train(diabetes~., data=treino, method="rf", tuneGrid=tuneGrid_rf)
rf_h
### Aplicar modelos treinados na base de Teste
predict.rf_h <- predict(rf_h, teste)
confusionMatrix(predict.rf_h, as.factor(teste$diabetes))

#### treinar com cross-validation
set.seed(202401)
ctrl_rf <- trainControl(method = "cv", number = 10)
rf_cv <- train(diabetes~., data=treino, method="rf",
tuneGrid=tuneGrid_rf, trControl=ctrl_rf)
rf_cv
### Aplicar modelos treinados na base de Teste
predict.rf_cv <- predict(rf_cv, teste)
confusionMatrix(predict.rf_cv, as.factor(teste$diabetes))

## Resultado -> RF com Hold out obteve a melhor acurácia
novos_dados <- read.csv('10-Diabetes-Novos-Dados.csv')
novos_dados$num <- NULL
novos_dados$diabetes <- NULL

predict.novos_rf_h <- predict(rf_h, novos_dados)
resultado <- cbind(novos_dados, predict.novos_rf_h)
View(resultado)
```

2. **Regressão**

Admissão

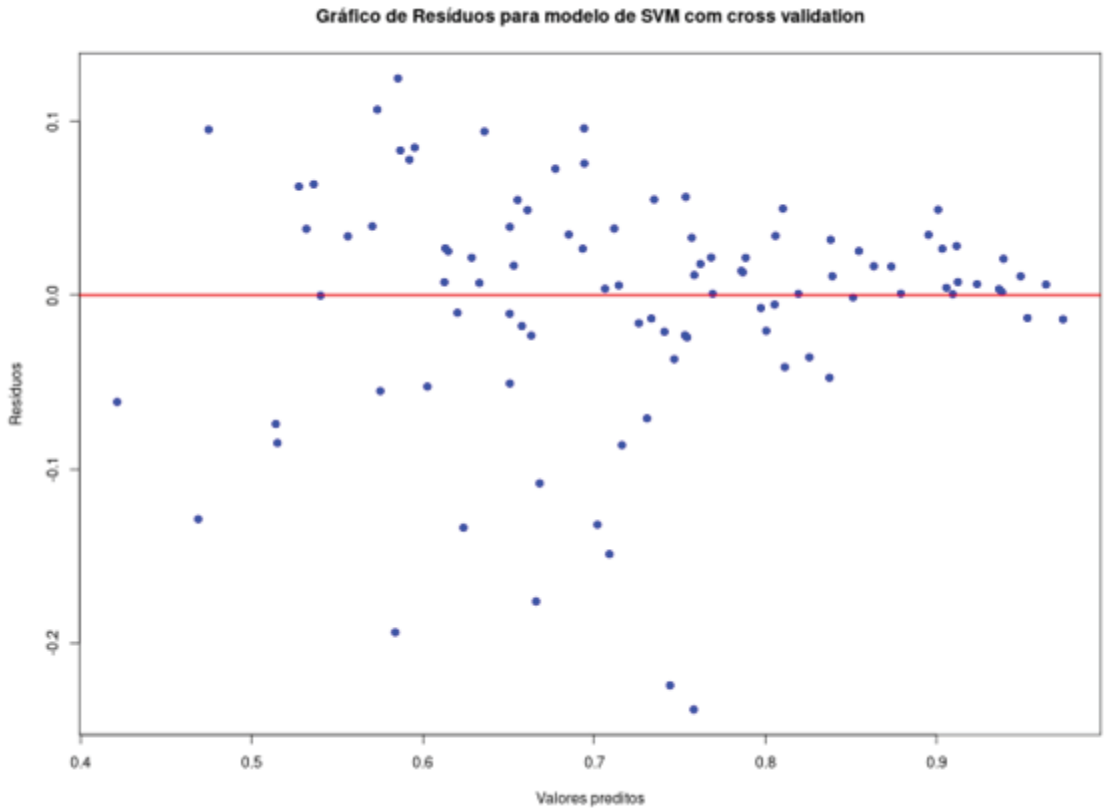
Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	C=10 Sigma=0.01	0.7939	0.0676	0.89133	0.0670	0.0463

SVM – Hold-out	C=2 Sigma=0.01	0.7903	0.0682	0.8894	0.068	0.0470
RNA – CV	size=3 decay=0.1	0.7897	0.0683	0.8927	0.0676	0.0503
RNA – Hold-out	size=3 decay=0.1	0.7873	0.0687	0.8918	0.068	0.0509
RF – Hold-out	mtry=2	0.7646	0.0723	0.8750	0.0716	0.0517
RF – CV	mtry=2	0.7630	0.0725	0.8741	0.0718	0.052
KNN	K=9	0.6717	0.0854	0.8202	0.084	0.0633

Predição para novos casos: O SVM com Cross Validation obteve o melhor R2.

GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.novos_svm_cv
337	118	4	4.5	4.5	9.35	1	0.9137410
323	105	4	4.0	4.5	8.87	1	0.8018283
315	105	3	3.0	3.5	8.00	1	0.6641024

Gráfico dos Resíduos:



FONTE: O autor (2024).

Script em R:

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

#
setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/regres
sao-admissao")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
# remover variável desnecessária
dados$num <- NULL

### Cria arquivos de treino e teste usados em todos os modelos
set.seed(202401)
randomIndexes <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list =
FALSE)
treino <- dados[randomIndexes,]
teste <- dados[-randomIndexes,]

# cria funções necessárias para calculo das métricas

# erro padrão da estimativa - Syx
standard_error <- function(obs, preds){
```

```
  size <- length(obs)
  sum_pos <- sum((obs - preds) ^ 2)
  result = sqrt((sum_pos / (size - 2)))
  return(result)
}

mean_absolute_error <- function(obs, preds){
  size <- length(obs)
  sum_abs <- sum(abs(obs - preds))
  result <- sum_abs / size
  return(result)
}

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

pearson_coefficient <- function(obs, preds) {
  mean_obs <- mean(obs)
  mean_preds <- mean(preds)

  numerator <- sum((obs - mean_obs) * (preds - mean_preds))
  denominator <- sqrt(sum((obs - mean_obs)^2) * sum((preds -
mean_preds)^2))

  result <- numerator / denominator

  return(result)
}

## KNN
### Prepara um grid com os valores de k que serão usados
set.seed(202401)
tuneGrid_knn <- expand.grid(k = c(1,3,5,7,9))
### Executa o Knn com esse grid
set.seed(202401)
knn <- train(ChanceOfAdmit ~ ., data = treino, method = "knn",
             tuneGrid=tuneGrid_knn)
knn

predict.knn <- predict(knn, teste)
### Calcula as métricas do knn
knn_rmse <- rmse(teste$ChanceOfAdmit, predict.knn)
knn_r2 <-r2(predict.knn,teste$ChanceOfAdmit)
knn_syx <- standard_error(teste$ChanceOfAdmit, predict.knn)
knn_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.knn)
knn_pearson <- pearson_coefficient(teste$ChanceOfAdmit, predict.knn)

## RNA
set.seed(202401)
tuneGrid_rna <- expand.grid(size = seq(from = 1, to = 3, by = 1), decay =
```

```
seq(from = 0.1, to = 0.7, by = 0.3))
## treino com hold out
rna_h <- train(ChanceOfAdmit~., data=treino, method="nnet", linout=T,
trace=FALSE, tuneGrid=tuneGrid_rna)
rna_h
predict.rna_h <- predict(rna_h, teste)

### calcular métricas do rna com hold out
rna_h_rmse <- rmse(teste$ChanceOfAdmit, predict.rna_h)
rna_h_r2 <-r2(predict.rna_h,teste$ChanceOfAdmit)
rna_h_syx <- standard_error(teste$ChanceOfAdmit, predict.rna_h)
rna_h_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.rna_h)
rna_h_pearson <- pearson_coefficient(teste$ChanceOfAdmit, predict.rna_h)

## treino com cross validation CV
set.seed(202401)
control_rna <- trainControl(method = "cv", number = 10)
rna_cv <- train(ChanceOfAdmit~.,  data=treino, method="nnet",
trainControl=control_rna, tuneGrid=tuneGrid_rna, linout=T,
                  MaxNWts=10000, maxit=2000, trace=F)
rna_cv
predict.rna_cv <- predict(rna_cv, teste)

### calcular métricas do rna com hold out
rna_cv_rmse <- rmse(teste$ChanceOfAdmit, predict.rna_cv)
rna_cv_r2 <-r2(predict.rna_cv,teste$ChanceOfAdmit)
rna_cv_syx <- standard_error(teste$ChanceOfAdmit, predict.rna_cv)
rna_cv_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.rna_cv)
rna_cv_pearson <- pearson_coefficient(teste$ChanceOfAdmit,
predict.rna_cv)


## SVM
set.seed(202401)
#### Vários C e sigma
tuneGrid_svm = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))
### Treinar SVM com a base de Treino e Hold Out
svm_h <- train(ChanceOfAdmit~., data=treino, method="svmRadial",
tuneGrid=tuneGrid_svm)
svm_h

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

### calcular métricas do svm com hold out
svm_h_rmse <- rmse(teste$ChanceOfAdmit, predict.svm_h)
svm_h_r2 <-r2(predict.svm_h,teste$ChanceOfAdmit)
svm_h_syx <- standard_error(teste$ChanceOfAdmit, predict.svm_h)
svm_h_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.svm_h)
svm_h_pearson <- pearson_coefficient(teste$ChanceOfAdmit, predict.svm_h)
```

```
## treinar com cross validation
set.seed(202401)
ctrl_svm <- trainControl(method = "cv", number = 10)

svm_cv <- train(ChanceOfAdmit~., data=treino, method="svmRadial",
trControl=ctrl_svm, tuneGrid=tuneGrid_svm)
svm_cv
### Aplicar modelos treinados na base de Teste
predict.svm_cv <- predict(svm_cv, teste)

### calcular métricas do svm com hold out
svm_cv_rmse <- rmse(teste$ChanceOfAdmit, predict.svm_cv)
svm_cv_r2 <-r2(predict.svm_cv,teste$ChanceOfAdmit)
svm_cv_syx <- standard_error(teste$ChanceOfAdmit, predict.svm_cv)
svm_cv_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.svm_cv)
svm_cv_pearson <- pearson_coefficient(teste$ChanceOfAdmit,
predict.svm_cv)

# Random Forest
set.seed(202401)
#### Vários mtry
tuneGrid_rf = expand.grid(mtry=c(2, 5, 7, 9))

# treinar com hold out
rf_h <- train(ChanceOfAdmit~., data=treino, method="rf",
tuneGrid=tuneGrid_rf)
rf_h

### Aplicar modelos treinados na base de Teste
predict.rf_h <- predict(rf_h, teste)
### calcular métricas do rf com hold out
rf_h_rmse <- rmse(teste$ChanceOfAdmit, predict.rf_h)
rf_h_r2 <-r2(predict.rf_h,teste$ChanceOfAdmit)
rf_h_syx <- standard_error(teste$ChanceOfAdmit, predict.rf_h)
rf_h_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.rf_h)
rf_h_pearson <- pearson_coefficient(teste$ChanceOfAdmit, predict.rf_h)

# treinar com cross validation
set.seed(202401)
ctrl_rf <- trainControl(method = "cv", number = 10)
rf_cv <- train(ChanceOfAdmit~., data=treino, method="rf",
tuneGrid=tuneGrid_rf, trControl=ctrl_rf)
rf_cv

### Aplicar modelos treinados na base de Teste
predict.rf_cv <- predict(rf_cv, teste)
### calcular métricas do rf com cross validation
rf_cv_rmse <- rmse(teste$ChanceOfAdmit, predict.rf_cv)
rf_cv_r2 <-r2(predict.rf_cv,teste$ChanceOfAdmit)
rf_cv_syx <- standard_error(teste$ChanceOfAdmit, predict.rf_cv)
rf_cv_mae <- mean_absolute_error(teste$ChanceOfAdmit, predict.rf_cv)
rf_cv_pearson <- pearson_coefficient(teste$ChanceOfAdmit, predict.rf_cv)
```

```
## Resultado -> SVM com cross validation obteve as melhores métricas
novos_dados <- read.csv('9-Admissao-Novos-Dados.csv')
novos_dados$num <- NULL

predict.novos_svm_cv <- predict(svm_cv, novos_dados)
resultado <- cbind(novos_dados, predict.novos_svm_cv)
View(resultado)

residuals_svm_cv <- teste$ChanceOfAdmit - predict.svm_cv

# Create the residual plot
plot(predict.svm_cv, residuals_svm_cv,
      main="Gráfico de Resíduos para modelo de SVM com cross validation",
      xlab="Valores preditos",
      ylab="Resíduos",
      pch=19, col="blue")
abline(h=0, col="red", lwd=2)
```

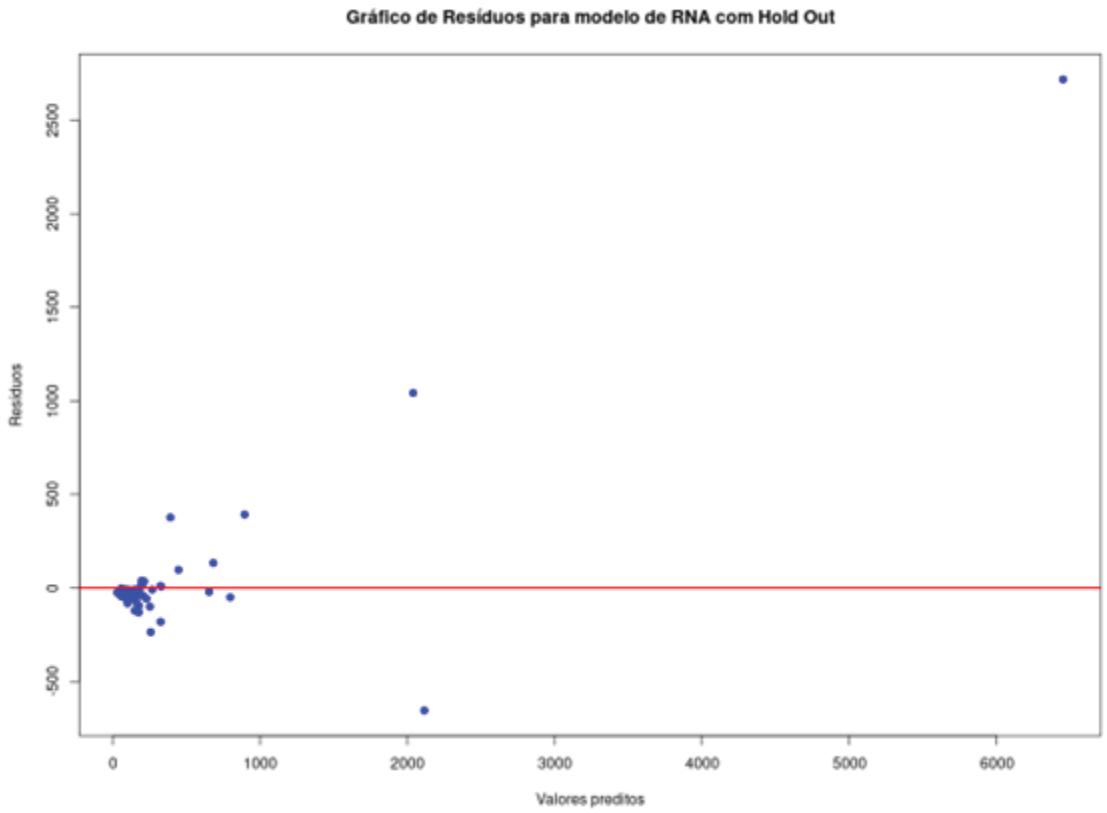
Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RNA – Hold-out	size=3 decay=0.4	0.8967	403.93	0.9860	397.14	133.92
RF – Hold-out	mtry=2	0.8853	425.59	0.9825	418.43	115.32
RF – CV	mtry=2	0.8815	432.63	0.9834	425.36	114.19
SVM – Hold-out	C=100 Sigma=0.01	0.8258	524.55	0.9590	515.73	162.30
SVM – CV	C=100 Sigma=0.01	0.8258	524.55	0.9590	515.73	162.30
RNA – CV	size=2 decay=0.4	0.8065	552.74	0.9524	543.45	142.14
KNN	K=1	0.7776	592.61	0.9420	582.65	156.58

Predição para novos casos: O RNA com Hold-Out obteve o melhor R2.

dap	h	Me	predict.novos_rna_h
6.2	5.0	1.03	109.3212
7.2	5.0	1.04	120.0018
7.9	5.7	1.04	129.8521

Gráfico dos Resíduos:



Script em R:

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

#
setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/regressao-biomassa")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)

### Cria arquivos de treino e teste
set.seed(202401)
randomIndexes <- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
```

```
treino <- dados[randomIndexes,]
teste <- dados[-randomIndexes,]

# cria funções necessárias para calculo das métricas

# erro padrão da estimativa - Syx
standard_error <- function(obs, preds){
  size <- length(obs)
  sum_pos <- sum((obs - preds) ^ 2)
  result = sqrt((sum_pos / (size - 2)))
  return(result)
}

mean_absolute_error <- function(obs, preds){
  size <- length(obs)
  sum_abs <- sum(abs(obs - preds))
  result <- sum_abs / size
  return(result)
}

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

pearson_coefficient <- function(obs, preds) {
  mean_obs <- mean(obs)
  mean_preds <- mean(preds)

  numerator <- sum((obs - mean_obs) * (preds - mean_preds))
  denominator <- sqrt(sum((obs - mean_obs)^2) * sum((preds -
mean_preds)^2))

  result <- numerator / denominator

  return(result)
}

# KNN
### Prepara um grid com os valores de k que serão usados
set.seed(202401)
tuneGrid_knn <- expand.grid(k = c(1,3,5,7,9))
### Executa o Knn com esse grid
set.seed(202401)
knn <- train(biomassa ~ ., data = treino, method = "knn",
             tuneGrid=tuneGrid_knn)
knn

predict.knn <- predict(knn, teste)
### Calcula as métricas do knn
knn_rmse <- rmse(teste$biomassa, predict.knn)
knn_r2 <-r2(predict.knn,teste$biomassa)
```

```
knn_syx <- standard_error(teste$biomassa, predict.knn)
knn_mae <- mean_absolute_error(teste$biomassa, predict.knn)
knn_pearson <- pearson_coefficient(teste$biomassa, predict.knn)

# RNA
set.seed(202401)
tuneGrid_rna <- expand.grid(size = seq(from = 1, to = 3, by = 1), decay =
seq(from = 0.1, to = 0.7, by = 0.3))
## treino com hold out
rna_h <- train(biomassa~., data=treino, method="nnet", linout=T,
trace=FALSE, tuneGrid=tuneGrid_rna)
rna_h
predict.rna_h <- predict(rna_h, teste)

### calcular métricas do rna com hold out
rna_h_rmse <- rmse(teste$biomassa, predict.rna_h)
rna_h_r2 <-r2(predict.rna_h,teste$biomassa)
rna_h_syx <- standard_error(teste$biomassa, predict.rna_h)
rna_h_mae <- mean_absolute_error(teste$biomassa, predict.rna_h)
rna_h_pearson <- pearson_coefficient(teste$biomassa, predict.rna_h)

## treino com cross validation CV
set.seed(202401)
control_rna <- trainControl(method = "cv", number = 10)
rna_cv <- train(biomassa~.,  data=treino, method="nnet",
trainControl=control_rna, tuneGrid=tuneGrid_rna, linout=T,
                  MaxNWts=10000, maxit=2000, trace=F)
rna_cv
predict.rna_cv <- predict(rna_cv, teste)

### calcular métricas do rna com hold out
rna_cv_rmse <- rmse(teste$biomassa, predict.rna_cv)
rna_cv_r2 <-r2(predict.rna_cv,teste$biomassa)
rna_cv_syx <- standard_error(teste$biomassa, predict.rna_cv)
rna_cv_mae <- mean_absolute_error(teste$biomassa, predict.rna_cv)
rna_cv_pearson <- pearson_coefficient(teste$biomassa, predict.rna_cv)


## SVM
set.seed(202401)
#### Vários C e sigma
tuneGrid_svm = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015,
0.2))
### Treinar SVM com a base de Treino e Hold Out
svm_h <- train(biomassa~., data=treino, method="svmRadial",
tuneGrid=tuneGrid_svm)
svm_h

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

### calcular métricas do svm com hold out
```

```
svm_h_rmse <- rmse(teste$biomassa, predict.svm_h)
svm_h_r2 <-r2(predict.svm_h,teste$biomassa)
svm_h_syx <- standard_error(teste$biomassa, predict.svm_h)
svm_h_mae <- mean_absolute_error(teste$biomassa, predict.svm_h)
svm_h_pearson <- pearson_coefficient(teste$biomassa, predict.svm_h)

## treinar com cross validation
set.seed(202401)
ctrl_svm <- trainControl(method = "cv", number = 10)

svm_cv <- train(biomassa~., data=treino, method="svmRadial",
trainControl=ctrl_svm, tuneGrid=tuneGrid_svm)
svm_cv
### Aplicar modelos treinados na base de Teste
predict.svm_cv <- predict(svm_cv, teste)

### calcular métricas do svm com hold out
svm_cv_rmse <- rmse(teste$biomassa, predict.svm_cv)
svm_cv_r2 <-r2(predict.svm_cv,teste$biomassa)
svm_cv_syx <- standard_error(teste$biomassa, predict.svm_cv)
svm_cv_mae <- mean_absolute_error(teste$biomassa, predict.svm_cv)
svm_cv_pearson <- pearson_coefficient(teste$biomassa, predict.svm_cv)

## Random Forest
set.seed(202401)
#### Vários mtry
tuneGrid_rf = expand.grid(mtry=c(2, 5, 7, 9))

# treinar com hold out
rf_h <- train(biomassa~., data=treino, method="rf", tuneGrid=tuneGrid_rf)
rf_h

### Aplicar modelos treinados na base de Teste
predict.rf_h <- predict(rf_h, teste)
### calcular métricas do rf com hold out
rf_h_rmse <- rmse(teste$biomassa, predict.rf_h)
rf_h_r2 <-r2(predict.rf_h,teste$biomassa)
rf_h_syx <- standard_error(teste$biomassa, predict.rf_h)
rf_h_mae <- mean_absolute_error(teste$biomassa, predict.rf_h)
rf_h_pearson <- pearson_coefficient(teste$biomassa, predict.rf_h)

# treinar com cross validation
set.seed(202401)
ctrl_rf <- trainControl(method = "cv", number = 10)
rf_cv <- train(biomassa~., data=treino, method="rf",
tuneGrid=tuneGrid_rf, trControl=ctrl_rf)
rf_cv

### Aplicar modelos treinados na base de Teste
predict.rf_cv <- predict(rf_cv, teste)
### calcular métricas do rf com cross validation
rf_cv_rmse <- rmse(teste$biomassa, predict.rf_cv)
```

```
rf_cv_r2 <-r2(predict.rf_cv,teste$biomassa)
rf_cv_syx <- standard_error(teste$biomassa, predict.rf_cv)
rf_cv_mae <- mean_absolute_error(teste$biomassa, predict.rf_cv)
rf_cv_pearson <- pearson_coefficient(teste$biomassa, predict.rf_cv)

## Resultado -> RNA com Hold Out obteve as melhores métricas
novos_dados <- read.csv('5-Biomassa-Novos-Dados.csv')
novos_dados$biomassa <- NULL

predict.novos_rna_h <- predict(rna_h, novos_dados)
resultado <- cbind(novos_dados, predict.novos_rna_h)
View(resultado)

residuals_rna_h <- teste$biomassa - predict.rna_h

# Create the residual plot
plot(predict.rna_h, residuals_rna_h,
      main="Gráfico de Resíduos para modelo de RNA com Hold Out",
      xlab="Valores preditos",
      ylab="Resíduos",
      pch=19, col="blue")
abline(h=0, col="red", lwd=2)
```

3. Agrupamento

Veículo

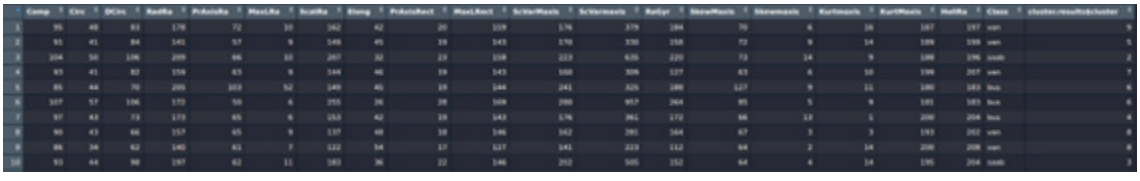
Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usar 10 clusters no experimento.

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

```
K-modes clustering with 10 clusters of sizes 63, 130, 125, 96, 67, 87, 48, 89, 82, 59

Cluster modes:
   Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarnaxis RaGyr SkewMaxis Skewnaxis Kurtnaxis KurtMaxis HollRa Class
1    89   36    53   127       59      6    140    47         18      125       165       290   139        70         1        21       189    184  saab
2   104   51   104   197       60     10    213    31         24      162       223       669   218        72         0        11       188    199  saab
3   100   55   103   197       62     11    219    30         25      172       228       706   214        71         0         7       189    198  opel
4    85   43    68   120       65      7    150    46         19      145       169       341   171        85         4         8       179    182   bus
5    91   38    75   141       61      7    149    45         19      131       177       327   137        74         2        14       185    191  saab
6    85   45    70   130       56      7    151    45         19      146       170       333   186        81         4        11       181    183   bus
7    93   47    85   147       64      8    153    44         19      144       175       354   174        71         0         2       185    197   van
8    86   37    66   138       55      6    137    49         18      127       159       281   145        64         5        14       186    201   van
9    90   46    85   133       56     10    157    43         20      149       173       351   186        75         1        10       183    193   van
10   89   40    66   125       59      7    133    50         18      139       154       246   157        67         7         6       193    183   van
```



Script em R:

```
### Pacotes necessários
install.packages("mlbench")
install.packages("mice")
## para o kmodes
```

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

### Leitura dos dados
#setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/agrup
amento-veiculo")
dados <- read.csv("4 - Veiculos - Dados.csv")
View(dados)
#remover variável desnecessária
dados$a <- NULL

set.seed(202401)
cluster.results <- kmodes(dados, 10, iter.max = 10, weighted = FALSE )
cluster.results$cluster
cluster.results

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

4. Regras de Associação

Musculação

Regras geradas com uma configuração de Suporte e Confiança.

20 com maior confiança

```
> inspect(sort(rules[1:20], by= confidence ))
     lhs                       rhs           support    confidence coverage   lift     count
[1]  {Afundo, Crucifixo}    => {Gemeos}      0.07692308 1          0.07692308 1.529412 2
[2]  {Crucifixo, Gemeos}    => {Afundo}      0.07692308 1          0.07692308 2.888889 2
[3]  {Afundo, Crucifixo}    => {LegPress}    0.07692308 1          0.07692308 1.238095 2
[4]  {Crucifixo, LegPress}  => {Afundo}      0.07692308 1          0.07692308 2.888889 2
[5]  {Crucifixo, Gemeos}    => {LegPress}    0.07692308 1          0.07692308 1.238095 2
[6]  {Crucifixo, LegPress}  => {Gemeos}      0.07692308 1          0.07692308 1.529412 2
[7]  {Adutor, Agachamento}  => {LegPress}    0.11538462 1          0.11538462 1.238095 3
[8]  {Adutor, LegPress}     => {Agachamento} 0.11538462 1          0.11538462 3.250000 3
[9]  {Esteira, Flexor}      => {Extensor}    0.07692308 1          0.07692308 2.000000 2
[10] {Extensor, Flexor}     => {Esteira}     0.07692308 1          0.07692308 2.166667 2
[11] {Esteira, Flexor}      => {Bicicleta}   0.07692308 1          0.07692308 1.857143 2
[12] {Bicicleta, Flexor}    => {Esteira}     0.07692308 1          0.07692308 2.166667 2
[13] {Esteira, Flexor}      => {LegPress}    0.07692308 1          0.07692308 1.238095 2
[14] {Flexor, LegPress}     => {Esteira}     0.07692308 1          0.07692308 2.166667 2
[15] {Extensor, Flexor}     => {Bicicleta}   0.07692308 1          0.07692308 1.857143 2
[16] {Bicicleta, Flexor}    => {Extensor}    0.07692308 1          0.07692308 2.000000 2
[17] {Extensor, Flexor}     => {LegPress}    0.07692308 1          0.07692308 1.238095 2
[18] {Flexor, LegPress}     => {Extensor}    0.07692308 1          0.07692308 2.000000 2
[19] {Bicicleta, Flexor}    => {LegPress}    0.07692308 1          0.07692308 1.238095 2
[20] {Flexor, LegPress}     => {Bicicleta}   0.07692308 1          0.07692308 1.857143 2
```

Script em R:

```
### Instalação dos pacotes necessários
```

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

set.seed(202401)
# ler arquivo
#setwd("~/Documentos/IBM/pós/IAA008-Aprendizado-de-Maquina/trabalho/assoc
iacao-musculacao")
dados <-  read.transactions(file="2 - Musculacao -
Dados.csv",format="basket",sep=";")
summary(dados)
# ver frequencia dos itens
itemFrequencyPlot(dados, topN=10, type="absolute")

# extrair regras
set.seed(202401)
rules <- apriori(dados, parameter = list(supp = 0.001, conf = 0.7,
minlen=3))
summary(rules)

# visualizar as 20 regras com maior confiança
inspect(sort(rules[1:20], by="confidence"))
```

APÊNDICE H – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos *Fake* (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1.

```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Input, Conv2D, Dense, Flatten,
Dropout, AveragePooling2D
from tensorflow.keras.models import Model
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Base
cifar10 = tf.keras.datasets.cifar10
(x_train, y_train), (x_test, y_test) = cifar10.load_data()

# Normalização dos dados
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
y_train, y_test = y_train.flatten(), y_test.flatten()

K = len(set(y_train))

i = Input(shape=x_train[0].shape)
x = Conv2D(32, (3, 3), strides=1, activation="relu")(i)
x = AveragePooling2D(pool_size=(2,2),strides=2, padding="valid")(x)
x = Conv2D(64, (3, 3), strides=1, activation="relu")(x)
x = AveragePooling2D(pool_size=(2,2),strides=2, padding="valid")(x)
x = Conv2D(128, (3, 3), strides=1, activation="relu")(x)
x = AveragePooling2D(pool_size=(2,2),strides=2, padding="valid")(x)

x = Flatten()(x)

x = Dense(512, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

model = Model(i, x)

# Visualizando arquitetura da rede
model.summary()
```

Model: "functional"

Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 32, 32, 3)	0
conv2d (Conv2D)	(None, 30, 30, 32)	896
average_pooling2d (AveragePooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
average_pooling2d_1 (AveragePooling2D)	(None, 6, 6, 64)	0
conv2d_2 (Conv2D)	(None, 4, 4, 128)	73,856
average_pooling2d_2 (AveragePooling2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262,656
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 10)	5,130

Total params: 361,034 (1.38 MB)

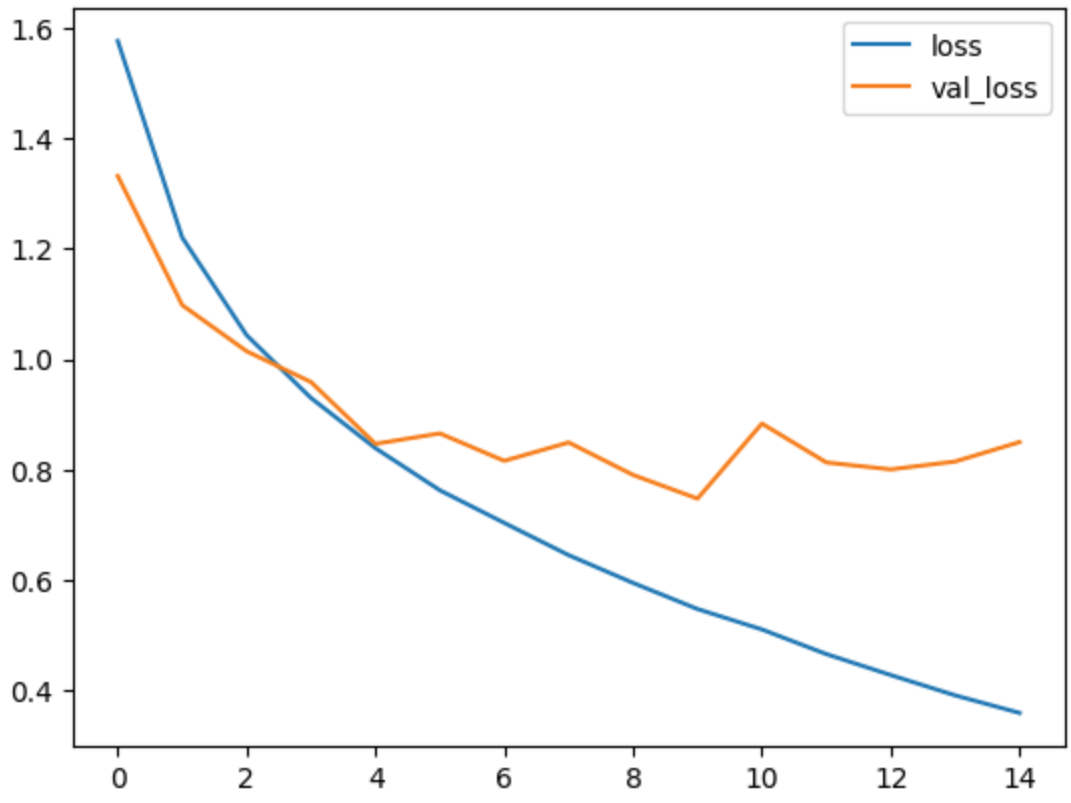
Trainable params: 361,034 (1.38 MB)

Non-trainable params: 0 (0.00 B)

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

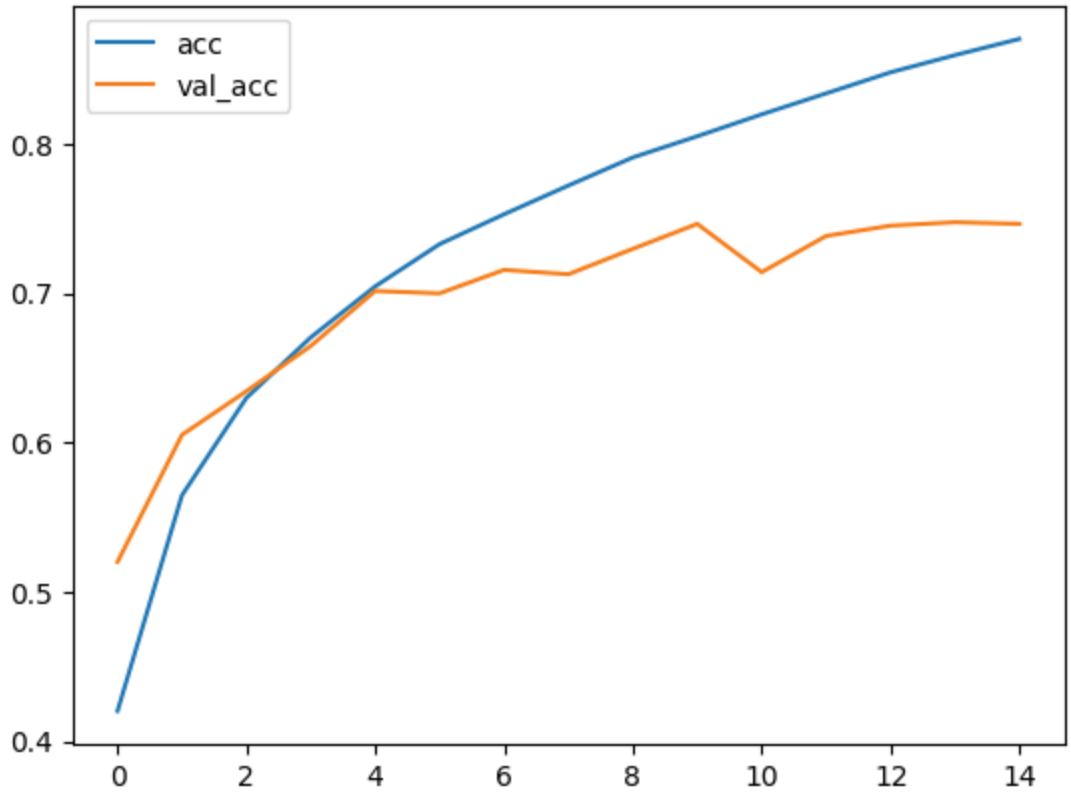
```
Epoch 1/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 18s 8ms/step - accuracy: 0.3411 - loss: 1.7784 - val_accuracy: 0.5201 - val_loss: 1.3315
Epoch 2/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.5413 - loss: 1.2757 - val_accuracy: 0.6051 - val_loss: 1.0980
Epoch 3/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6247 - loss: 1.0578 - val_accuracy: 0.6343 - val_loss: 1.0145
Epoch 4/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.6633 - loss: 0.9502 - val_accuracy: 0.6646 - val_loss: 0.9588
Epoch 5/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6999 - loss: 0.8506 - val_accuracy: 0.7014 - val_loss: 0.8469
Epoch 6/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7329 - loss: 0.7605 - val_accuracy: 0.6998 - val_loss: 0.8659
Epoch 7/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7512 - loss: 0.7069 - val_accuracy: 0.7155 - val_loss: 0.8162
Epoch 8/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7718 - loss: 0.6441 - val_accuracy: 0.7126 - val_loss: 0.8495
Epoch 9/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.7927 - loss: 0.5895 - val_accuracy: 0.7296 - val_loss: 0.7907
Epoch 10/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8098 - loss: 0.5380 - val_accuracy: 0.7463 - val_loss: 0.7479
Epoch 11/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.8277 - loss: 0.4921 - val_accuracy: 0.7140 - val_loss: 0.8837
Epoch 12/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8371 - loss: 0.4607 - val_accuracy: 0.7382 - val_loss: 0.8133
Epoch 13/15
...
Epoch 14/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.8678 - loss: 0.3750 - val_accuracy: 0.7474 - val_loss: 0.8150
Epoch 15/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.8775 - loss: 0.3414 - val_accuracy: 0.7463 - val_loss: 0.8502
```

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

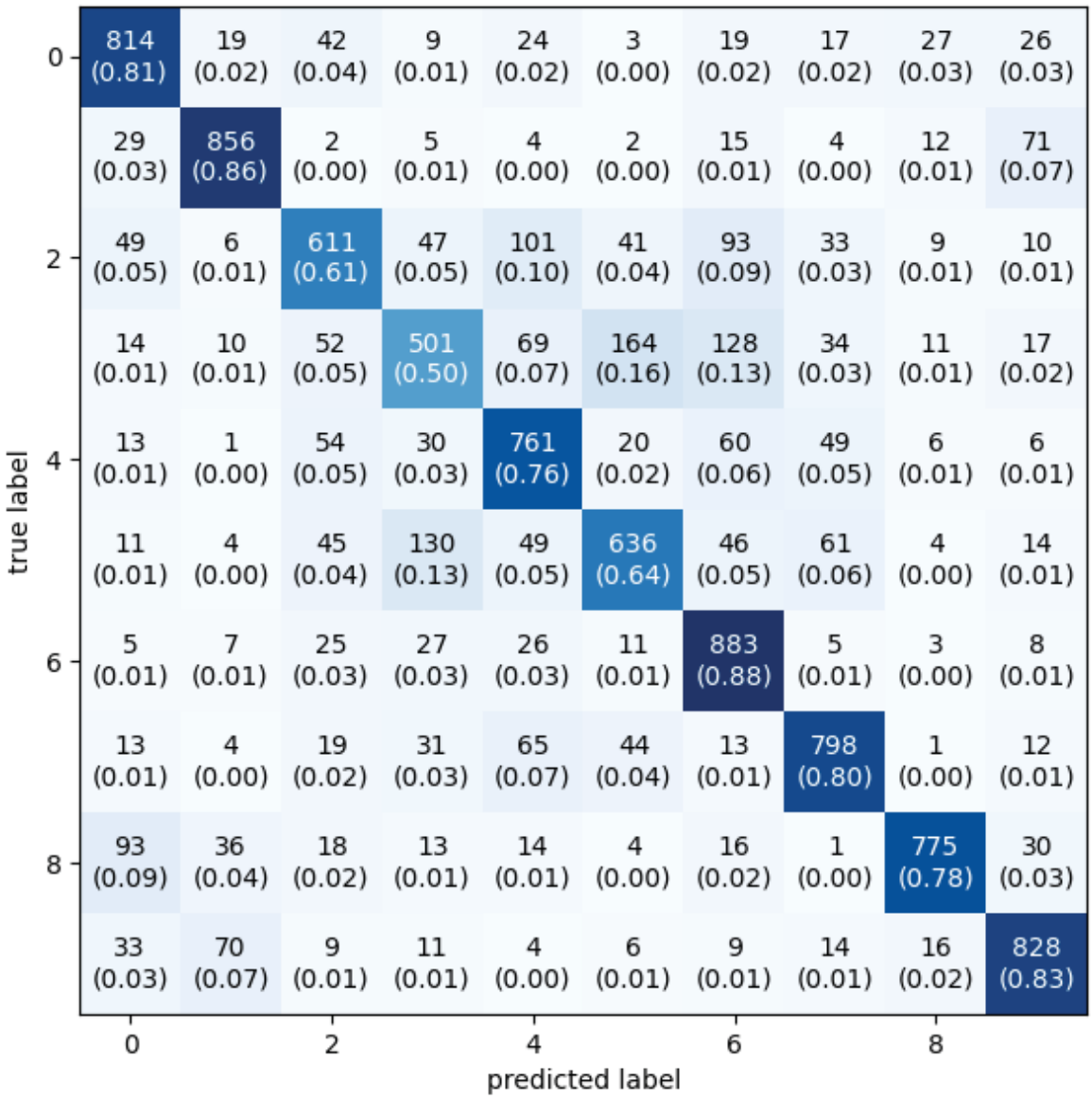



```
# Plotar acurácia, treino e validação
```

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

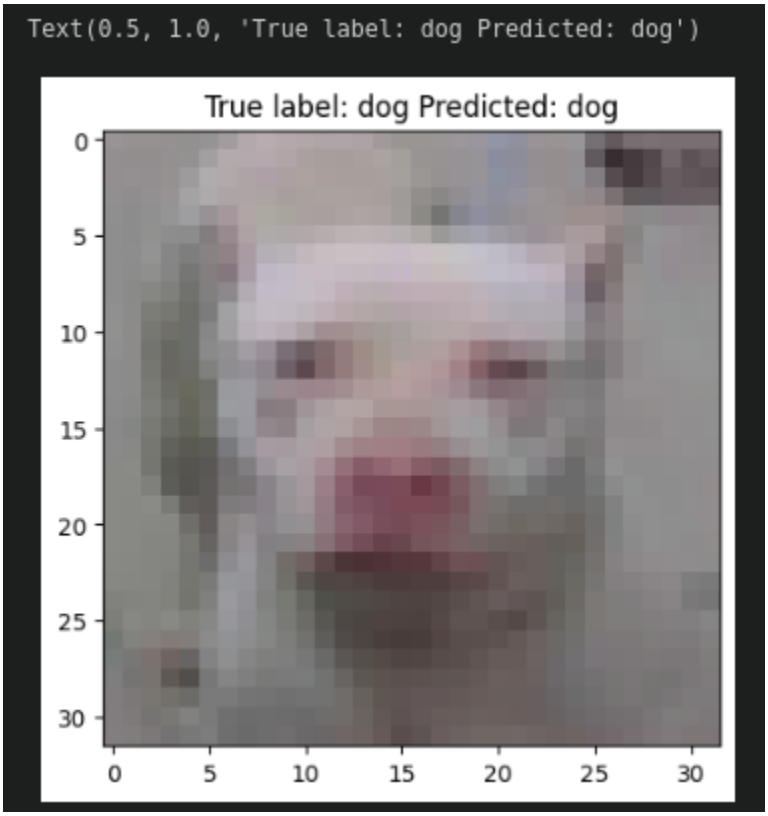



```
# Predições na base de teste
y_pred = model.predict(x_test).argmax(axis=1)

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



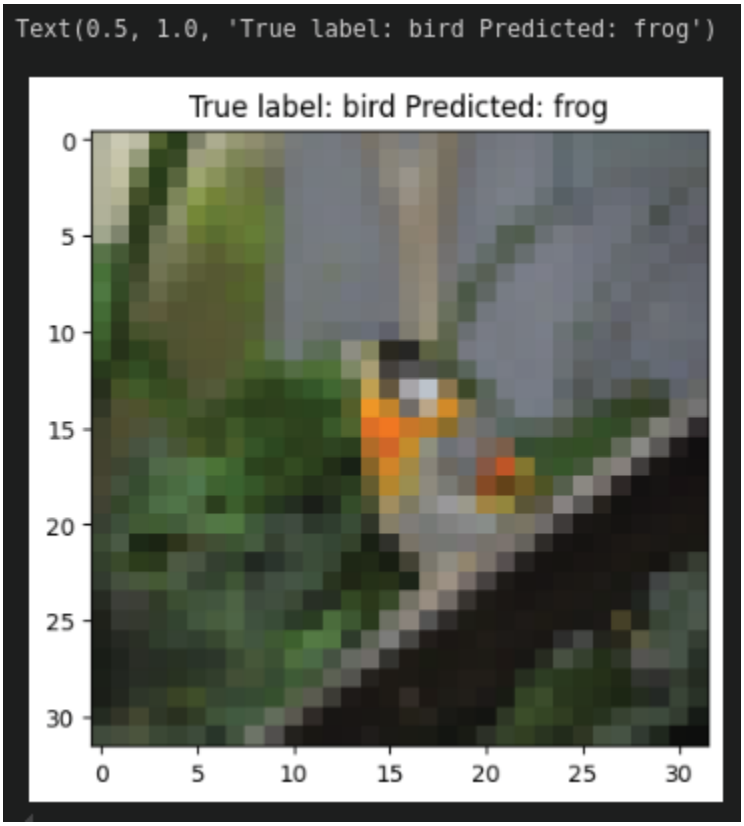

```
# Exemplo de classificação correta
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog",
"frog", "horse", "ship", "truck"]
classified = np.where(y_pred == y_test)[0]
i = np.random.choice(classified)
plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],
labels[y_pred[i]]))
```

```
Text(0.5, 1.0, 'True label: dog Predicted: dog')
```



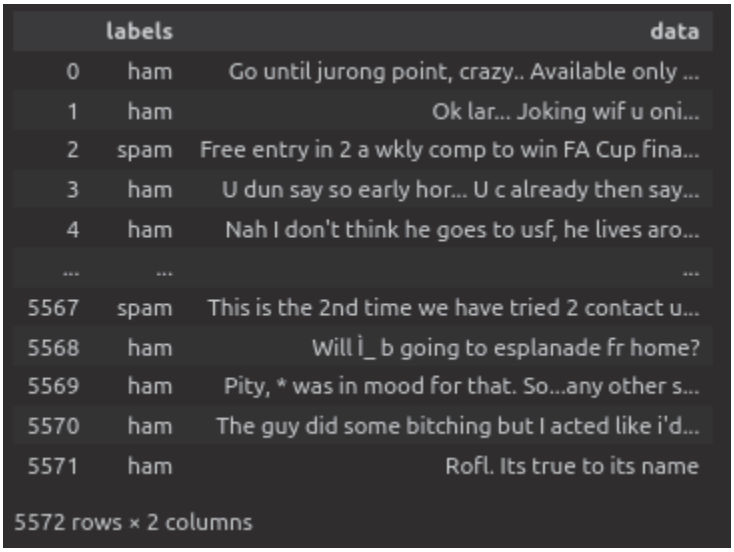

```
# Exemplo de classificação errada
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog",
"frog", "horse", "ship", "truck"]
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)
plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],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

```
Text(0.5, 1.0, 'True label: bird Predicted: frog')
```



2.

```
!pip install tensorflow

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

!wget http://www.razer.net.br/datasets/spam.csv

df = pd.read_csv("spam.csv", encoding="ISO-8859-1", usecols=['v1',
'v2']).rename(columns={'v1': 'labels', 'v2': 'data'})
df
```



	labels	data
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

```
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)

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(V)
7180

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

print("data_train.shape:", data_train.shape)
print("data_test.shape:", data_test.shape)
data_train.shape: (3733, 162)
data_test.shape: (1839, 162)

D = 20
M = 5
i = Input(shape=(T,))
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation = 'sigmoid')(x)

model = Model(i,x)
model.summary()
```

```
Model: "functional"
```

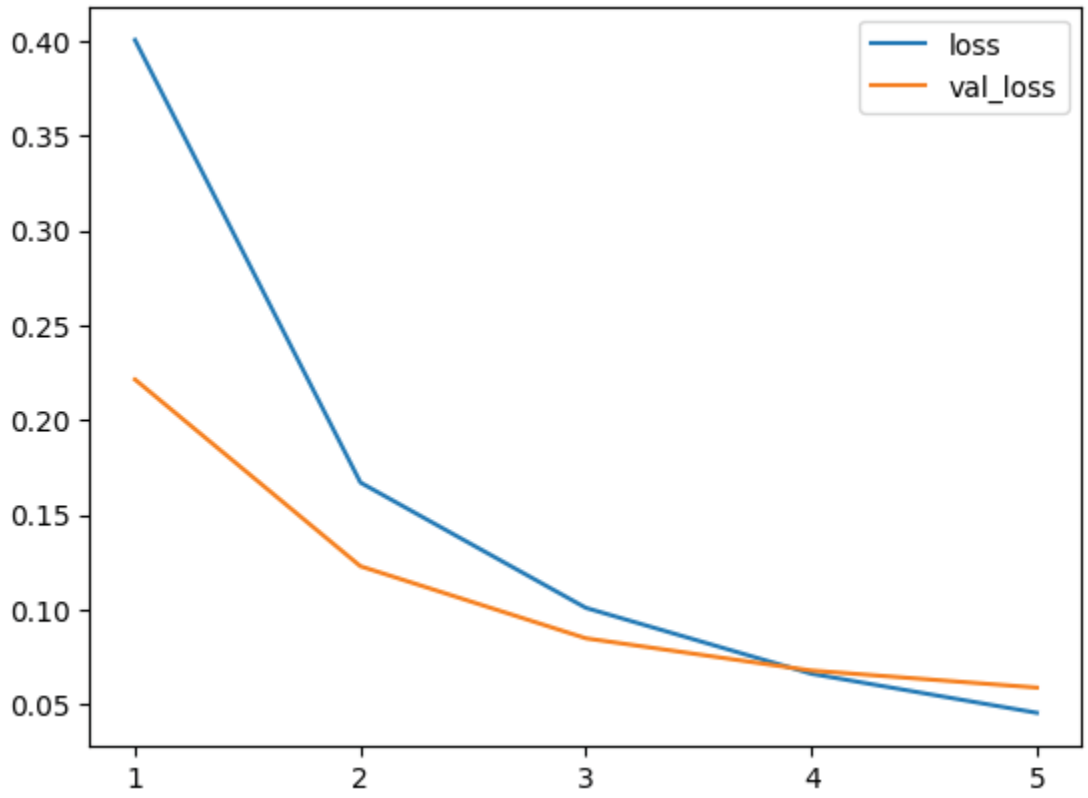
Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 162)	0
embedding_1 (Embedding)	(None, 162, 20)	143,620
lstm_1 (LSTM)	(None, 5)	520
dense_1 (Dense)	(None, 1)	6

```
Total params: 144,146 (563.07 KB)

Trainable params: 144,146 (563.07 KB)

Non-trainable params: 0 (0.00 B)
```

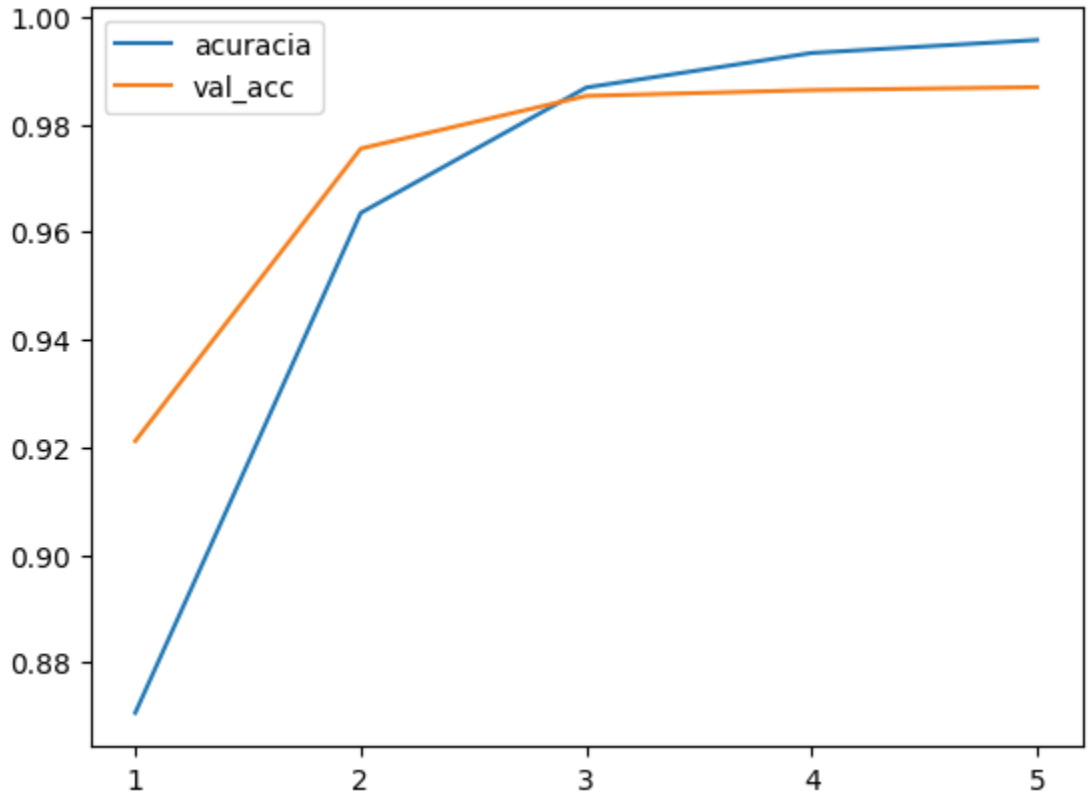
```
model.compile(loss='binary_crossentropy', optimizer = 'adam', metrics =
['accuracy'])
epochs = 5

r = model.fit(data_train, y_train, epochs = epochs, validation_data =
(data_test, y_test))
```

```
Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 11s 72ms/step - accuracy: 0.8505 - loss: 0.5173 - val_accuracy: 0.9212 - val_loss: 0.2214
Epoch 2/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - accuracy: 0.9475 - loss: 0.1988 - val_accuracy: 0.9755 - val_loss: 0.1229
Epoch 3/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 8s 72ms/step - accuracy: 0.9826 - loss: 0.1147 - val_accuracy: 0.9853 - val_loss: 0.0848
Epoch 4/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - accuracy: 0.9935 - loss: 0.0701 - val_accuracy: 0.9864 - val_loss: 0.0678
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 23s 167ms/step - accuracy: 0.9961 - loss: 0.0481 - val_accuracy: 0.9869 - val_loss: 0.0588
```

```
plt.plot(r.history['loss'], label = 'loss')
plt.plot(r.history['val_loss'], label = 'val_loss')
plt.xticks(np.arange(0, epochs, step=1), labels = range(1, epochs+1))
plt.legend()
plt.show()
```




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



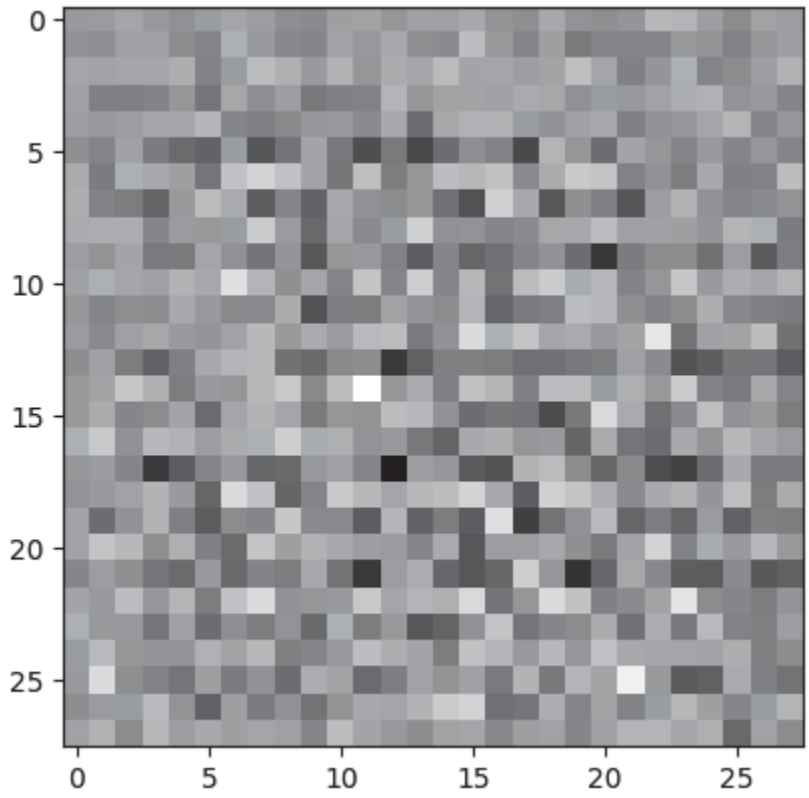

```
texto = "Is your car dirty? Discovery our new product. Free for all.
Click the link."
seq_texto = tokenizer.texts_to_sequences([texto])
data_texto = pad_sequences(seq_texto, maxlen=T)

pred = model.predict(data_texto)
print(pred)
print("Spam" if pred >= 0.5 else 'ok')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 345ms/step
[[0.6511604]]
Spam
```

3.

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

import tensorflow as tf
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
```

```
import time
from tensorflow.keras.losses import BinaryCrossentropy
from tensorflow.keras.optimizers import Adam, SGD, RMSprop

from IPython import display

(train_images, train_labels), (_,_) = tf.keras.datasets.mnist.load_data()

train_images = train_images.reshape(train_images.shape[0], 28, 28,
1).astype('float32')
train_images = (train_images - 127.5) / 127.5

BUFFER_SIZE = 60000
BATCH_SIZE = 256

train_dataset = tf.data.Dataset.from_tensor_slices(train_images)
train_dataset = train_dataset.shuffle(BUFFER_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=(1,1), 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))
  assert model.output_shape == (None, 28, 28, 1)

  return model

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

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



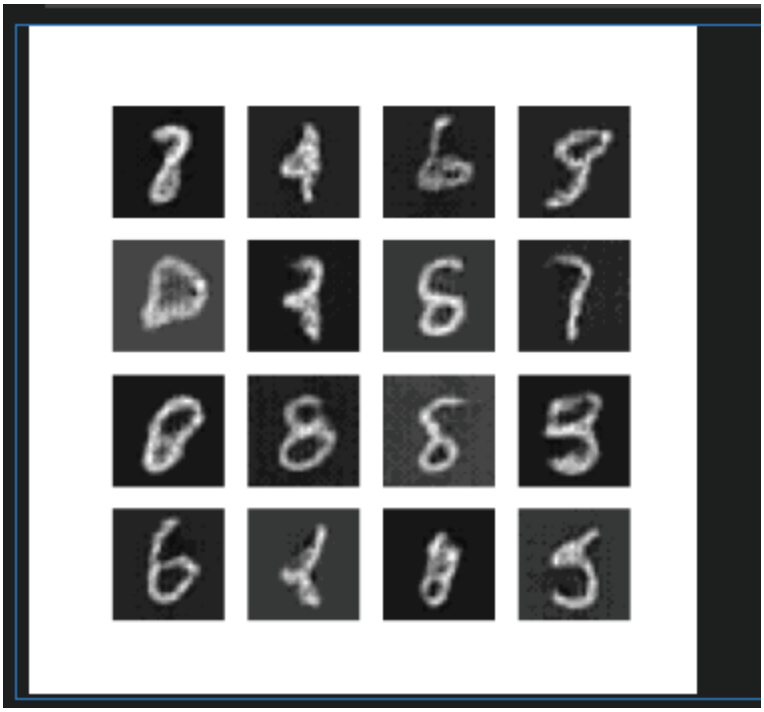

```
def make_discriminator_model():
  model = tf.keras.Sequential()
  model.add(layers.Conv2D(64, (5,5), strides = (2,2), padding = 'same',
input_shape = [28,28,1]))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))

  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)
tf.Tensor([[0.00111132]], shape=(1, 1), dtype=float32)

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_chekcpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, 'ckpt')
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
                                 discriminator_optimizer =
discriminator_optimizer,
                                 generator=generator,
                                 discriminator = discriminator)

EPOCHS = 100
noise_dim = 100
num_example_to_generate = 16

seed = tf.random.normal([num_example_to_generate, noise_dim])

@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_image = generator(noise, training = True)

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

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

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

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

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))
  plt.show()

train(train_dataset, EPOCHS)
```

```
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))

import tensorflow_docs.vis.embed as embed

def display_image(epoch_no):
```

```
      return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

display_image(EPOCHS)

anim_file = 'dcgan.gif'

with imageio.get_writer(anim_file, mode='I') as writer:
      filenames = glob.glob('image_at_epoch_*.png')
      filenames = sorted(filenames)

      for filename in filenames:
      image = imageio.imread(filename)
      writer.append_data(image)

      if filenames:
      image = imageio.imread(filenames[-1])
      writer.append_data(image)

embed.embed_file(anim_file)
```

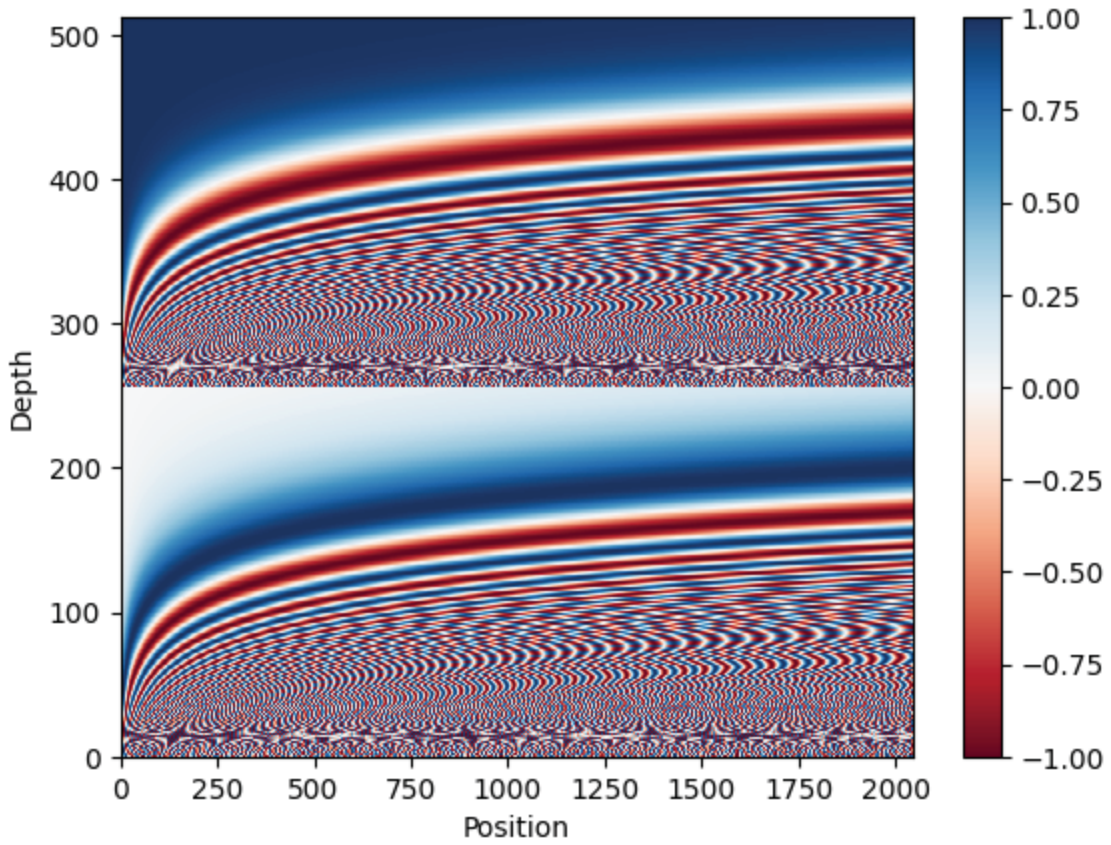


4.

```
# INSTALAÇÃO DE PACOTES
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0

# IMPORTAÇÃO DE BIBLIOTECAS
import logging
import time
```

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

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

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

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

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

# TOKENIZAÇÃO E DESTOKENIZAÇÃO
# Tokenização e Destokenização do texto
model_name = "ted_hrlr_translate_pt_en_converter"
tf.keras.utils.get_file(f"{model_name}.zip",
f"https://storage.googleapis.com/download.tensorflow.org/models/{model_na
me}.zip", cache_dir='.', cache_subdir='', extract=True)

# tokenizers.en tokeniza e detokeniza
tokenizers = tf.saved_model.load(model_name)

# PIPELINE DE ENTRADA
# Definindo função para codificar/tokenizar lotes de texto puro
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

# Pipeline: processa, embaralha, agrupa os dados, prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

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

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

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

def positional_encoding(position, d_model):
  angle_rads = get_angles(np.arange(position)[:,
np.newaxis],np.arange(d_model)[np.newaxis, :], d_model)
  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)

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

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

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




```
# DEFININDO FUNÇÕES PARA MASCARAMENTO POR 0 E 1
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

# DEFININDO FUNÇÃO DE ATENÇÃO
def scaled_dot_product_attention(q, k, v, mask):
  matmul_qk = tf.matmul(q, k, transpose_b=True)
  dk = tf.cast(tf.shape(k)[-1], tf.float32)
  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(MultiHeadAttention, self).__init__()
      self.num_heads = num_heads
```

```
      self.d_model = d_model
      assert d_model % self.num_heads == 0
      self.depth = d_model // self.num_heads
      self.wq = tf.keras.layers.Dense(d_model)
      self.wk = tf.keras.layers.Dense(d_model)
      self.wv = tf.keras.layers.Dense(d_model)
      self.dense = tf.keras.layers.Dense(d_model)

  def split_heads(self, x, batch_size):
      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

# DEFININDO FUNÇÃO PARA REDE FEED-FORWARD
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)])

# DEFININDO CLASSE E FUNÇÕES PARA CAMADA DO CODIFICADOR
class EncoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
      super(EncoderLayer, self).__init__()
      self.mha = MultiHeadAttention(d_model, num_heads)
      self.ffn = point_wise_feed_forward_network(d_model, dff)
      self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
      self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
      self.dropout1 = tf.keras.layers.Dropout(rate)
      self.dropout2 = tf.keras.layers.Dropout(rate)

  def call(self, x, training, mask):
      attn_output, _ = self.mha(x, x, x, mask)
      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

# DEFININDO CLASSE E FUNÇÕES PARA CAMADA DO DECODIFICADOR
class DecoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
      super(DecoderLayer, self).__init__()

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

      self.ffn = point_wise_feed_forward_network(d_model, dff)

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

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

  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
      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

# DEFININDO CLASSE E FUNÇÕES PARA ENCODER
class Encoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads,
dff,input_vocab_size, maximum_position_encoding, rate=0.1):
      super(Encoder, self).__init__()
      self.d_model = d_model
      self.num_layers = num_layers
      self.embedding = tf.keras.layers.Embedding(input_vocab_size,
d_model)
      self.pos_encoding = positional_encoding(maximum_position_encoding,
```

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

  def call(self, x, training, mask):
      seq_len = tf.shape(x)[1]
      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

# DEFININDO CLASSE E FUNÇÕES PARA DECODER
class Decoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff,
target_vocab_size,maximum_position_encoding, rate=0.1):
      super(Decoder, self).__init__()

      self.d_model = d_model
      self.num_layers = num_layers

      self.embedding = tf.keras.layers.Embedding(target_vocab_size,
d_model)
      self.pos_encoding = positional_encoding(maximum_position_encoding,
d_model)

      self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate) for
_ in range(num_layers)]
      self.dropout = tf.keras.layers.Dropout(rate)

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

      x = self.embedding(x)
      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

# DEFININDO CLASSE E FUNÇÕES PARA TRANSFORMER
```

```
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):
      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

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

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

  def __call__(self, step):
      step = tf.cast(step, tf.float32)
      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)

# DEFININDO FUNÇÕES DE PERDA E MÉTRICA DE ACURÁCIA
loss_object =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

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

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

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

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

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

# PROCESSO DE TREINAMENTO
EPOCHS = 20
train_step_signature = [tf.TensorSpec(shape=(None, None),
dtype=tf.int64),tf.TensorSpec(shape=(None, None), dtype=tf.int64), ]
@tf.function(input_signature=train_step_signature)

def train_step(inp, tar):
```

```
  tar_inp = tar[:, :-1]
  tar_real = tar[:, 1:]
  with tf.GradientTape() as tape:
      predictions, _ = transformer([inp, tar_inp], training = True)
      loss = loss_function(tar_real, predictions)
  gradients = tape.gradient(loss, transformer.trainable_variables)
  optimizer.apply_gradients(zip(gradients,
transformer.trainable_variables))
  train_loss(loss)
  train_accuracy(accuracy_function(tar_real, predictions))

# PROCESSO DE TREINAMENTO
for epoch in range(EPOCHS):
  start = time.time()
  train_loss.reset_state()
  train_accuracy.reset_state()

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

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

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

```
Epoch 1 Batch 0 Loss 8.8682 Accuracy 0.0000
Epoch 1 Batch 50 Loss 8.8171 Accuracy 0.0006
Epoch 1 Batch 100 Loss 8.7122 Accuracy 0.0207
Epoch 1 Batch 150 Loss 8.5957 Accuracy 0.0279
Epoch 1 Batch 200 Loss 8.4550 Accuracy 0.0320
Epoch 1 Batch 250 Loss 8.2877 Accuracy 0.0353
Epoch 1 Batch 300 Loss 8.0990 Accuracy 0.0408
Epoch 1 Batch 350 Loss 7.9000 Accuracy 0.0489
Epoch 1 Batch 400 Loss 7.7113 Accuracy 0.0567
Epoch 1 Batch 450 Loss 7.5416 Accuracy 0.0652
Epoch 1 Batch 500 Loss 7.3933 Accuracy 0.0736
Epoch 1 Batch 550 Loss 7.2591 Accuracy 0.0820
Epoch 1 Batch 600 Loss 7.1317 Accuracy 0.0903
Epoch 1 Batch 650 Loss 7.0123 Accuracy 0.0980
Epoch 1 Batch 700 Loss 6.9018 Accuracy 0.1049
Epoch 1 Batch 750 Loss 6.8002 Accuracy 0.1112
Epoch 1 Batch 800 Loss 6.7055 Accuracy 0.1171
Epoch 1 Loss 6.6896 Accuracy 0.1182
Time taken for 1 epoch: 169.43 secs

Epoch 2 Batch 0 Loss 5.2509 Accuracy 0.2119
Epoch 2 Batch 50 Loss 5.1874 Accuracy 0.2176
Epoch 2 Batch 100 Loss 5.1559 Accuracy 0.2216
Epoch 2 Batch 150 Loss 5.1299 Accuracy 0.2236
Epoch 2 Batch 200 Loss 5.1068 Accuracy 0.2253
...
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.4525 Accuracy 0.6797
Time taken for 1 epoch: 97.16 secs
```

```
# DEFININDO FUNÇÕES DO TRADUTOR
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

# EFETUANDO UMA TRADUÇÃO
translator = Translator(tokenizers, transformer)
sentence = [["Eu li sobre triceratops na enciclopédia.", "Ela chegou como
uma chuva de verão.", "Eu li varios livros em minhas férias.",],
            ["O tempo passou e só agora eles se deram conta do que
perderam.","  O onibus estava lotado quando passou por aqui.","Ninguem viu
o que aconteceu"]]

for txt in sentence:
  for frase in txt:
      translated_text, translated_tokens, attention_weights = translator(
tf.constant(frase))
      print(f'{"Prediction":15s}: {translated_text}')
```

```
Prediction     : b'i read about tarizlings in egypt .'
Prediction     : b'she arrived like a rain rainbow .'
Prediction     : b'i read several books on my vacation .'
Prediction     : b"time has just gone back and they ' ve been told when they lost it ."
Prediction     : b'the onebbus was cut when it went on here .'
Prediction     : b'no one saw what happened .'
```

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

Entretenimento Baseado em Dados: Um Estudo de Caso sobre o Sistema de Personalização, Recomendação e Produção da Netflix

A Netflix é uma das maiores e mais influentes empresas de streaming do mundo, contando com milhões de assinantes em mais de 190 países e um catálogo extenso de filmes, séries, documentários e conteúdo original. Mas como uma empresa fundada em 1997 oferecendo um serviço de aluguel de DVDs por correio, evoluiu de maneira significativa ao longo dos anos, tornando-se um dos principais nomes no mercado global de entretenimento? A resposta é simples: atenção especial aos dados de seus clientes.

Desde os anos 2000 a Netflix começou a usar dados dos clientes para aprimorar a experiência de uso de sua plataforma. A abordagem baseada em dados foi desenvolvida para ajudar os clientes a descobrir o que assistir de forma mais rápida e assertiva, tornando a experiência mais personalizada. Por exemplo, se a maioria dos usuários que gostam de doramas também costuma assistir a filmes de drama e comédia romântica, o sistema da Netflix identifica esses padrões e, com base nisso, recomenda conteúdos similares para novos assinantes que começam a assistir doramas.

No entanto, com o crescimento da base de usuários e do volume de dados, a empresa percebeu a necessidade de aprimorar seus modelos de recomendação e escalar sua infraestrutura. Inicialmente, a Netflix usava algoritmos simples para sugerir filmes com base nas avaliações e no histórico de visualização dos usuários. Mas, à medida que a plataforma cresceu, também aumentou a complexidade de seu sistema de recomendações. Para lidar com essa enorme quantidade de dados e melhorar a precisão das sugestões, a empresa adotou ferramentas avançadas, como Hadoop, Apache Spark e AWS, para processar e analisar dados em grande escala.

A primeira parte desse sistema de recomendação começa com a coleta de dados, ou seja, a Netflix coleta dados sobre as interações que seus usuários têm com a plataforma, desde um simples login até interações como pausar e fechar conteúdos e clicar num link recomendado. Todas essas interações, ou eventos, são processados pelo Apache Kafka e enviados para armazenamento na AWS.

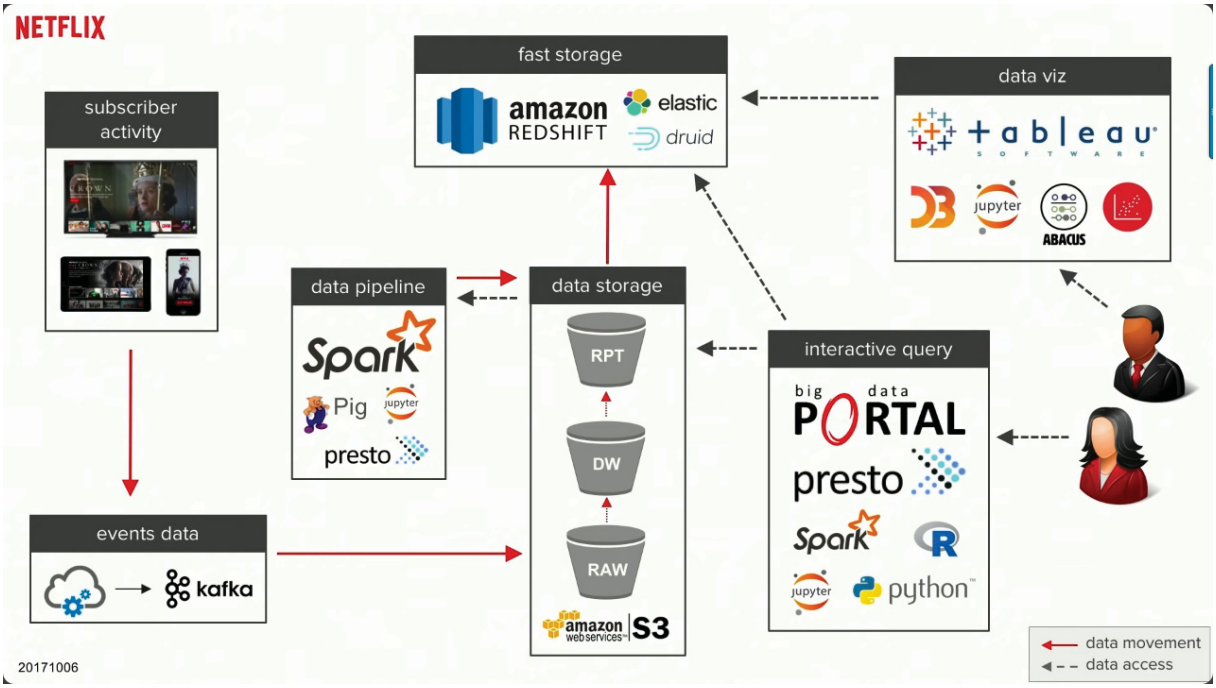
Na AWS, os dados podem ser armazenados de diferentes formas:

- S3: utilizado para armazenar os dados brutos e não estruturados;

- Amazon Redshift: utilizado para armazenar dados estruturados;

Num primeiro momento os dados brutos são apenas armazenados no S3 e, para processar e manipular esses dados, o Netflix utiliza uma gama de tecnologias como Apache Pig, Spark, Jupiter etc. Parte desses dados manipulados são então armazenados em tecnologias que permitem acesso rápido e performático, como o Amazon Redshift.

Possuir dados nesses ambientes é crucial para que tecnologias de visualização de dado, como Tableau, possam ter acesso rápido aos dados. Essas ferramentas permitem que a Netflix oriente suas decisões estratégicas, como direcionamento de orçamento para a aquisição ou produção de novos conteúdos, com base nas tendências de preferências dos usuários e na identificação de nichos de oportunidade a serem explorados.



Para personalizar as recomendações, a primeira etapa foi coletar e armazenar uma vasta quantidade de dados. Além dos dados das interações dos usuários, histórico de visualização e avaliações, a Netflix também coleta informações demográficas, como idade, localização e perfil dos usuários.

Após a coleta, o próximo passo é processar esse grande volume de dados, utilizando diversas ferramentas e tecnologias para otimizar o sistema de recomendação:

- Hadoop: usado para processamento em larga escala, especialmente em análises e processamento batch.
- Apache Kafka: permite a transmissão de dados em tempo real, possibilitando que informações sejam coletadas e processadas à medida que os usuários interagem com a plataforma.
- Apache Flink: utilizado para análises em tempo real, permitindo o processamento contínuo de dados e ajustes rápidos nas recomendações.

A análise de dados é uma parte fundamental do processo, pois, com o processamento em tempo real, os modelos de recomendação são constantemente atualizados. Esse método permite que

os algoritmos de machine learning identifiquem padrões e preferências comuns entre os usuários de forma rápida e precisa. Como resultado, as recomendações tornam-se cada vez mais relevantes, aumentando o engajamento dos usuários. Quanto mais a Netflix acerta nas recomendações, mais tempo os usuários passam assistindo e interagindo com a plataforma, fornecendo ainda mais dados para a empresa.

Atualmente, a Netflix contém um alto volume de dados, contendo 230 milhões de usuários que geram petabytes de dados a cada dia.

A **velocidade** em que os dados precisam ser processados é levado em consideração, pois, os dados são processados em tempo real para fornecer as recomendações imediatas e atualizadas.

A Netflix possui uma **variedade** nos dados, podemos dividi-los em: dados estruturados, que são os dados históricos de visualizações e, também, dados não estruturados como os feedbacks e comentários.

A **veracidade** diz respeito à qualidade e precisão das recomendações, pois, isso depende muito da qualidade dos dados e dos algoritmos de análises garantindo recomendações relevantes.

Podemos concluir que a Netflix faz uso exemplar de big data para se destacar como uma empresa orientada por dados. Ao priorizar a análise e o entendimento dos dados, a empresa cria experiências únicas e relevantes para seus usuários, aumenta a satisfação e o engajamento deles, e impulsiona a retenção na plataforma.

Referências

https://netflixtechblog.com/

https://www.youtube.com/watch?v=nMyuCdqzpZc

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

1. **Extração de Características**

Modelo que apresentou melhor desempenho foi o de RNA treinado com as características extraídas pela vgg16, levando em consideração as métricas de acurácia, com 0.86, mas principalmente o F1-score obtido para cada classe do problema, demonstrando um equilíbrio para identificar corretamente os casos positivos e minimizar falsos positivos e falsos negativos.

```
# código extração de características

## LBP
# Função para calcular o histograma lbp
def get_lbp(img):
      if img is None:
      return None
      # Verifica se a imagem já está em grayscale, se não estiver
converte para grayscale
      if len(img.shape) == 3:
      img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

      # Parâmetros do LBP
      METHOD = 'uniform'
      radius = 3
      n_points = 8 * radius

      # Calcula o histograma do LBP
      lbp_image = local_binary_pattern(img, n_points, radius,
METHOD)
      lbp_hist, _ = np.histogram(lbp_image.flatten(),
bins=np.arange(0, n_points + 3), range=(0, n_points + 2))

      return lbp_hist

# Função para obter valores lbp de todas as imagens
def extract_lbp_features(image_paths):
  lbp_features = []

  for image in image_paths:

      open_image = cv2.imread(image, cv2.IMREAD_GRAYSCALE)
```

```
      lbp_value = get_lbp(open_image)

      lbp_features.append(lbp_value)

  return lbp_features
## VGG16
# Carregando modelo VGG16 sem a camada de classificação
base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(224, 224, 3))

model = Model(inputs=base_model.input, outputs=base_model.output)
# Função para converter as imagens de entrada no formato esperado
pela VGG16
def preprocess_image(image_path):

      img = load_img(image_path, target_size=(224, 224))

      img_array = img_to_array(img)

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

      img_array = preprocess_input(img_array)

      return img_array

# Função para extrair características de uma única imagem com a
VGG16
def get_features_vgg16(image_path):

      img_array = preprocess_image(image_path)

      features = model.predict(img_array)

      features_flattened = features.flatten()

      return features_flattened
# Função para extrair de todas as imagens
def extract_features_vgg16(image_paths):

      features_list = []

      for image_path in image_paths:

      features = get_features_vgg16(image_path)

      features_list.append(features)

      return np.array(features_list)

# Teste dos modelos LBP
```

```
# Random Forest
## acurácia
rf_accuracy_lbp = accuracy_score(y_test, y_test_pred_rf)
print(f'Acurácia {rf_accuracy_lbp}')

## matriz de confusão
rf_cm_lbp = confusion_matrix(y_test, y_test_pred_rf)
print(f"Matriz de Confusão:\n{rf_cm_lbp}")

## sensibilidade, precisão, e F1-score
rf_recall_lbp, rf_precision_lbp, rf_f1_lbp, _ =
precision_recall_fscore_support(y_test, y_test_pred_rf)
print("Sensibilidade por classe:", rf_recall_lbp)
print("Precisão por classe:", rf_precision_lbp)
print("F1-Score por classe:", rf_f1_lbp)
```

```
Acurácia 0.7601078167115903
Matriz de Confusão:
[[86  7  8  0]
 [ 6 30 53  1]
 [ 7  6 77  0]
 [ 0  1  0 89]]
Sensibilidade por classe: [0.86868687 0.68181818 0.55797101 0.98888889]
Precisão por classe: [0.85148515 0.33333333 0.85555556 0.98888889]
F1-Score por classe: [0.86       0.44776119 0.6754386  0.98888889]
```

```
# SVM
## acurácia
svm_accuracy_lbp = accuracy_score(y_test, y_test_pred_svm)
print(f'Acurácia {svm_accuracy_lbp}')

## matriz de confusão
svm_cm_lbp = confusion_matrix(y_test, y_test_pred_svm)
print(f"Matriz de Confusão:\n{svm_cm_lbp}")

## sensibilidade, precisão, e F1-score
svm_recall_lbp, svm_precision_lbp, svm_f1_lbp, _ =
precision_recall_fscore_support(y_test, y_test_pred_svm)
print("Sensibilidade por classe:", svm_recall_lbp)
print("Precisão por classe:", svm_precision_lbp)
print("F1-Score por classe:", svm_f1_lbp)
```

```
Acurácia 0.7358490566037735
Matriz de Confusão:
[[72 26  3  0]
 [ 4 36 50  0]
 [ 2  9 79  0]
 [ 0  2  2 86]]
Sensibilidade por classe: [0.92307692 0.49315068 0.58955224 1.        ]
Precisão por classe: [0.71287129 0.4        0.87777778 0.95555556]
F1-Score por classe: [0.80446927 0.44171779 0.70535714 0.97727273]
```

```
# RNA
## acurácia
```

```
rna_accuracy_lbp = accuracy_score(y_test, y_test_pred_rna)
print(f'Acurácia {rna_accuracy_lbp}')

## matriz de confusão
rna_cm_lbp = confusion_matrix(y_test, y_test_pred_rna)
print(f"Matriz de Confusão:\n{rna_cm_lbp}")

## sensibilidade, precisão, e F1-score
rna_recall_lbp, rna_precision_lbp, rna_f1_lbp, _ =
precision_recall_fscore_support(y_test, y_test_pred_rna)
print("Sensibilidade por classe:", rna_recall_lbp)
print("Precisão por classe:", rna_precision_lbp)
print("F1-Score por classe:", rna_f1_lbp)
```

```
Acurácia 0.7520215633423181
Matriz de Confusão:
[[78 20  3  0]
 [ 4 34 52  0]
 [ 2  6 80  2]
 [ 0  3  0 87]]
Sensibilidade por classe: [0.92857143 0.53968254 0.59259259 0.97752809]
Precisão por classe: [0.77227723 0.37777778 0.88888889 0.96666667]
F1-Score por classe: [0.84324324 0.44444444 0.71111111 0.97206704]
```

```
# Teste dos modelos VGG16

# Random Forest
## acurácia
rf_accuracy_vgg16 = accuracy_score(y_test, y_test_pred_rf_vgg16)
print(f'Acurácia {rf_accuracy_vgg16}')

## matriz de confusão
rf_cm_vgg16 = confusion_matrix(y_test, y_test_pred_rf_vgg16)
print(f"Matriz de Confusão:\n{rf_cm_vgg16}")

## sensibilidade, precisão, e F1-score
rf_recall_vgg16, rf_precision_vgg16, rf_f1_vgg16, _ =
precision_recall_fscore_support(y_test, y_test_pred_rf_vgg16)
print("Sensibilidade por classe:", rf_recall_vgg16)
print("Precisão por classe:", rf_precision_vgg16)
print("F1-Score por classe:", rf_f1_vgg16)
```

```
Acurácia 0.7520215633423181
Matriz de Confusão:
[[83 18  0  0]
 [32 34 23  1]
 [ 1  1 72 16]
 [ 0  0  0 90]]
Sensibilidade por classe: [0.71551724 0.64150943 0.75789474 0.8411215 ]
Precisão por classe: [0.82178218 0.37777778 0.8        1.        ]
F1-Score por classe: [0.76497696 0.47552448 0.77837838 0.91370558]
```

```
# SVM
```

```
## acurácia
svm_accuracy_vgg16 = accuracy_score(y_test, y_test_pred_svm_vgg16)
print(f'Acurácia {svm_accuracy_vgg16}')

## matriz de confusão
svm_cm_vgg16 = confusion_matrix(y_test, y_test_pred_svm_vgg16)
print(f"Matriz de Confusão:\n{svm_cm_vgg16}")

## sensibilidade, precisão, e F1-score
svm_recall_vgg16, svm_precision_vgg16, svm_f1_vgg16, _ =
precision_recall_fscore_support(y_test, y_test_pred_svm_vgg16)
print("Sensibilidade por classe:", svm_recall_vgg16)
print("Precisão por classe:", svm_precision_vgg16)
print("F1-Score por classe:", svm_f1_vgg16)
```

```
Acurácia 0.77088948787062
Matriz de Confusão:
[[88 13  0  0]
 [22 40 24  4]
 [ 0  0 74 16]
 [ 0  0  6 84]]
Sensibilidade por classe: [0.8        0.75471698 0.71153846 0.80769231]
Precisão por classe: [0.87128713 0.44444444 0.82222222 0.93333333]
F1-Score por classe: [0.83412322 0.55944056 0.7628866  0.86597938]
```

```
# RNA
## acurácia
rna_accuracy_vgg16 = accuracy_score(y_test, y_test_pred_rna_vgg16)
print(f'Acurácia {rna_accuracy_vgg16}')

## matriz de confusão
rna_cm_vgg16 = confusion_matrix(y_test, y_test_pred_rna_vgg16)
print(f"Matriz de Confusão:\n{rna_cm_vgg16}")

## sensibilidade, precisão, e F1-score
rna_recall_vgg16, rna_precision_vgg16, rna_f1_vgg16, _ =
precision_recall_fscore_support(y_test, y_test_pred_rna_vgg16)
print("Sensibilidade por classe:", rna_recall_vgg16)
print("Precisão por classe:", rna_precision_vgg16)
print("F1-Score por classe:", rna_f1_vgg16)
```

```
Acurácia 0.8652291105121294
Matriz de Confusão:
[[93  8  0  0]
 [16 68  6  0]
 [ 2  4 70 14]
 [ 0  0  0 90]]
Sensibilidade por classe: [0.83783784 0.85       0.92105263 0.86538462]
Precisão por classe: [0.92079208 0.75555556 0.77777778 1.        ]
F1-Score por classe: [0.87735849 0.8        0.84337349 0.92783505]
```

2. Redes Neurais

Modelo que apresentou melhor desempenho foi o do Resnet50 treinado sem dados provenientes de data augmentation. As métricas consideradas foram as de acurácia, com 0.91, mas principalmente o F1-score obtido para cada classe do problema, demonstrando um equilíbrio para identificar corretamente os casos positivos e minimizar falsos positivos e falsos negativos.

```
# código testando os modelos

## com data augmentation

## VGG16
## acurácia
vgg16_accuracy = accuracy_score(y_test, predicted_vgg16_classes)
print(f'Acurácia {vgg16_accuracy}')

## matriz de confusão
vgg16_cm = confusion_matrix(y_test, predicted_vgg16_classes)
print(f"Matriz de Confusão:\n{vgg16_cm}")

## sensibilidade, precisão, e F1-score
vgg16_recall, vgg16_precision, vgg16_f1, _ =
precision_recall_fscore_support(y_test, predicted_vgg16_classes)
print("Sensibilidade por classe:", vgg16_recall)
print("Precisão por classe:", vgg16_precision)
print("F1-Score por classe:", vgg16_f1)
```

```
Acurácia 0.7169811320754716
Matriz de Confusão:
[[101   0   0   0]
 [ 49   1  40   0]
 [ 11   3  75   1]
 [  0   0   1  89]]
Sensibilidade por classe: [0.62732919 0.25       0.64655172 0.98888889]
Precisão por classe: [1.         0.01111111 0.83333333 0.98888889]
F1-Score por classe: [0.77099237 0.0212766  0.72815534 0.98888889]
```

```
## RESNET50
## acurácia
resnet50_accuracy = accuracy_score(y_test,
predicted_resnet50_classes)
print(f'Acurácia {resnet50_accuracy}')

## matriz de confusão
resnet50_cm = confusion_matrix(y_test, predicted_resnet50_classes)
print(f"Matriz de Confusão:\n{resnet50_cm}")

## sensibilidade, precisão, e F1-score
resnet50_recall, resnet50_precision, resnet50_f1, _ =
precision_recall_fscore_support(y_test, predicted_resnet50_classes)
print("Sensibilidade por classe:", resnet50_recall)
print("Precisão por classe:", resnet50_precision)
```

```
print("F1-Score por classe:", resnet50_f1)
```

```
Acurácia 0.9002695417789758
Matriz de Confusão:
[[101   0   0   0]
 [ 10  77   2   1]
 [  3   9  68  10]
 [  0   0   2  88]]
Sensibilidade por classe: [0.88596491 0.89534884 0.94444444 0.88888889]
Precisão por classe: [1.         0.85555556 0.75555556 0.97777778]
F1-Score por classe: [0.93953488 0.875      0.83950617 0.93121693]
```

```
## sem data augmentation

## VGG16
## acurácia
vgg16_accuracy_without = accuracy_score(y_test,
predicted_vgg16_without_classes)
print(f'Acurácia {vgg16_accuracy_without}')

## matriz de confusão
vgg16_cm_without = confusion_matrix(y_test,
predicted_vgg16_without_classes)
print(f"Matriz de Confusão:\n{vgg16_cm_without}")

## sensibilidade, precisão, e F1-score
vgg16_recall_without, vgg16_precision_without, vgg16_f1_without, _
= precision_recall_fscore_support(y_test,
predicted_vgg16_without_classes)
print("Sensibilidade por classe:", vgg16_recall_without)
print("Precisão por classe:", vgg16_precision_without)
print("F1-Score por classe:", vgg16_f1_without)
```

```
Acurácia 0.7601078167115903
Matriz de Confusão:
[[93  6  2  0]
 [13 13 64  0]
 [ 0  0 87  3]
 [ 0  0  1 89]]
Sensibilidade por classe: [0.87735849 0.68421053 0.56493506 0.9673913 ]
Precisão por classe: [0.92079208 0.14444444 0.96666667 0.98888889]
F1-Score por classe: [0.89855072 0.23853211 0.71311475 0.97802198]
```

```
## RESNET50
## acurácia
resnet50_accuracy_without = accuracy_score(y_test,
predicted_resnet50_without_classes)
print(f'Acurácia {resnet50_accuracy_without}')

## matriz de confusão
resnet50_cm_without = confusion_matrix(y_test,
predicted_resnet50_without_classes)
print(f"Matriz de Confusão:\n{resnet50_cm_without}")
```

```
## sensibilidade, precisão, e F1-score
resnet50_recall_without, resnet50_precision_without,
resnet50_f1_without, _ = precision_recall_fscore_support(y_test,
predicted_resnet50_without_classes)
print("Sensibilidade por classe:", resnet50_recall_without)
print("Precisão por classe:", resnet50_precision_without)
print("F1-Score por classe:", resnet50_f1_without)
```

```
Acurácia 0.9191374663072777
Matriz de Confusão:
[[86 15  0  0]
 [ 1 87  2  0]
 [ 0  4 83  3]
 [ 0  0  5 85]]
Sensibilidade por classe: [0.98850575 0.82075472 0.92222222 0.96590909]
Precisão por classe: [0.85148515 0.96666667 0.92222222 0.94444444]
F1-Score por classe: [0.91489362 0.8877551  0.92222222 0.95505618]
```

APÊNDICE K – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.
9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: hfps://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx

B – RESOLUÇÃO

ESTUDO DE CASO: IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT

RESUMO
A inteligência artificial (IA), desde sua concepção em 1956, tem evoluído rapidamente, especialmente na forma de IA generativa, que cria conteúdos como texto, imagem e áudio baseados em dados de treinamento. O ChatGPT, principal exemplo de IA generativa, exemplifica o potencial e os desafios éticos dessa tecnologia. Treinado com vastos conjuntos de dados da internet, levanta diversas questões éticas, como consentimento dos dados utilizados, preconceito ocasionado por viés algorítmico e facilidade na criação de fake news. Para mitigar esses problemas, propõem-se políticas de transparência e consentimento para o uso de dados, auditorias de viés algorítmico e educação em ética para desenvolvedores. Além disso, medidas de verificação de conteúdo e políticas de alfabetização midiática são necessárias para combater a disseminação de fake news. Em conclusão, apesar dos benefícios significativos do ChatGPT, é essencial abordar suas implicações éticas de maneira cuidadosa.

Palavras-chave: inteligência artificial. inteligência artificial generativa. ética. viés.privacidade.

ABSTRACT
Artificial intelligence (AI), since its conception in 1956, has rapidly evolved, especially in the form of generative AI, which creates content such as text, images, and audio based on training data. ChatGPT, a prominent example of generative AI, exemplifies both the potential and ethical challenges of this technology. Trained on vast datasets from the internet, it raises various ethical questions, including consent for data used, bias introduced by algorithmic bias, and ease of creating fake news. To mitigate these issues, proposed solutions include transparency and consent policies for data usage, algorithmic bias audits, and ethics education for developers. Additionally, content verification measures and media literacy policies are necessary to combat the spread of fake news. In conclusion, despite the significant benefits of ChatGPT, addressing its ethical implications carefully is essential.

Keywords: artificial intelligence. generative artificial intelligence. ethics. bias. Privacy.

1 INTRODUÇÃO

O termo "inteligência artificial" (IA) foi cunhado em 1956, durante uma conferência na Universidade de Dartmouth promovida por John McCarthy, Marvin Minsky, Nathaniel Rochester e Claude Shannon. Desde então, a definição de IA tem sido objeto de debate, mas geralmente se refere à capacidade de máquinas e sistemas computacionais de executar tarefas que normalmente exigiriam a inteligência humana. Alternativamente, IA pode ser entendida como a área de estudo que busca construir agentes inteligentes capazes de realizar a melhor ação possível em uma dada situação[1]. Exemplos de tarefas e agentes inteligentes incluem reconhecimento de voz, reconhecimento de imagens, aprendizado de padrões, diagnóstico médico, veículos autônomos e recomendação de conteúdo.

Recentemente, um subcampo da IA tem ganhado destaque: a IA generativa. A IA generativa representa sistemas capazes de criar novos conteúdos, seja texto, imagem ou áudio, baseados em dados de treinamento. Para alcançar esses resultados, a IA generativa utiliza aprendizado de máquina, em particular redes neurais profundas. O sucesso no uso das redes neurais profundas foi viabilizado pelos avanços na capacidade de processamento e no desenvolvimento de hardware, resultando em processadores mais rápidos e eficientes, como GPUs (unidades de processamento gráfico) e TPUs (unidades de processamento tensorial), que permitem que sistemas de IA realizem cálculos complexos em velocidades sem precedentes.

As grandes companhias de tecnologia, conhecidas como Big Tech, têm investido intensamente em pesquisas para o avanço das inteligências artificiais (IA). As estratégias de coleta de dados e estudo do comportamento humano no meio digital estão cada vez mais sofisticadas, permitindo a sugestão e direcionamento de produtos e serviços com grande eficiência. Os usuários de mídias sociais e da internet em geral tornaram-se produtos valiosos, disputados pelas grandes companhias, gerando uma competição feroz entre as Big Tech e impulsionando a evolução tecnológica de forma exponencial.

No entanto, essa evolução tecnológica supera a capacidade da sociedade tradicional de acompanhá-la com suas leis e regras. Novas tecnologias introduzem vieses complexos, exigindo atenção contínua de governos, órgãos reguladores e sociedade civil para assegurar os direitos e padrões éticos existentes. Entre os desafios estão a proteção da propriedade intelectual, a privacidade dos dados pessoais, os direitos humanos, a criação de notícias falsas (fake news) por algoritmos inteligentes e os vieses decorrentes do mau uso da tecnologia. O uso do ChatGPT, devido à sua popularidade e rápida ascensão[2], exemplifica algumas dessas implicações éticas. Este estudo de caso explora as implicações éticas do uso do ChatGPT, destacando a importância de considerar aspectos como privacidade dos usuários e de seus dados, viés e impacto social.

2 DESENVOLVIMENTO

O ChatGPT, desenvolvido pela OpenAI[3], é o exemplo mais conhecido de IA generativa. Ele pode ser definido como um Modelo de Linguagem de Grande Porte, projetado para o processamento de linguagem natural. Esse tipo de modelo é exposto a quantidades massivas de dados para aprender os padrões estatísticos da linguagem, permitindo que ele "gere" novos textos e sequências

de palavras com base na probabilidade de como um humano usaria as palavras em um dado contexto.

De acordo com estimativas do banco de investimento UBS[2], o ChatGPT conseguiu alcançar mais de 100 milhões de usuários mensais ativos em menos de três meses, marca que o transformou no aplicativo com o mais rápido crescimento na história. Entretanto, essa rápida adoção levanta questões éticas que não podem ser desconsideradas.

Para compreender os padrões linguísticos e, mais recentemente, ser capaz de processar entradas multimodais, ou seja, texto e imagem, o ChatGPT demandou um grande volume de dados. Uma das primeiras questões éticas levantadas diz respeito à proveniência desses dados. Os dados utilizados para treinar modelos como o ChatGPT são, em sua maioria, oriundos de conteúdos públicos da internet. Mas, apesar de estar disponível publicamente, os autores/donos de tais conteúdos não deram consentimento explícito ao ChatGPT. Eles podem até ter concordado com os termos de uso das plataformas em que originalmente postaram conteúdo, mas não imaginaram que, no futuro, seus materiais seriam utilizados para treinar e até mesmo servir de base para IAs generativas. Até que ponto algo que está disponível publicamente e gratuitamente na internet pode ser utilizado, sem consentimento direto dos autores, por empresas privadas no desenvolvimento e melhoria de tecnologias que eventualmente vão gerar lucros para as mesmas?

Esse questionamento se amplia pois, recentemente, a Meta, empresa por traz de aplicativos como Instagram, Facebook e Whatsapp, adicionou uma seção à sua política de privacidade do Instagram para que os usuários possam optar ou não em fornecer suas postagens públicas como fonte de dados para treinamento de IAs generativas da Meta[4]. Ou seja, os usuários de aplicações da Meta vão ter a chance de recusar o uso de seus dados, mas como fica o direito de escolha frente a outros modelos de IA generativa que simplesmente afirmam usar dados compartilhados publicamente na internet? Os usuários saberão algum dia que seus conteúdos foram usados por esses modelos?

Outro caso recente envolvendo IA generativa e suposto uso não autorizado de dados de terceiros é o da Associação Americana da Indústria de Gravação (Recording Industry Assoiation of America - RIAA) que entrou com processos contra duas empresas do ramo de IA generativa para música, a Udio e a Suno. A RIAA alega que estas empresas fizeram uso de uma quantidade massiva de músicas protegidas por direitos autorais para conseguir treinar seus modelos[5]. Caso a RIAA ganhe a causa, pode conseguir abrir precedentes legais relevantes contra casos similares, demonstrando para empresas de IA generativa que elas não podem simplesmente se apropriar de material disponível online para suprir a considerável quantidade de dados necessários para treinar seus modelos.

Outra questão ética envolvendo dados diz respeito à possível introdução de viés algoritmo, visto que os dados são selecionados para o treinamento, ou seja, a imparcialidade do modelo resultante do treinamento depende dos dados apresentados a ele. Caso os dados selecionados sejam preconceituosos ou desiguais entre diferentes grupos, ou ainda, tenha ocorrido processo de rotulação manual tendenciosa, o modelo vai incorporar isso e acabar replicando esses vieses, impactando na desejada neutralidade algorítmica.

A simplicidade no uso de ferramentas de IA generativa, como o ChatGPT, onde o usuário apenas escreve num prompt de comando e rapidamente é respondido com textos ou imagens, em comparação com métodos “antigos” de obter conteúdo generativo, que exigia conhecimento em programação e técnicas de inteligência artificial, ou de editores de fotos, para caso das imagens, levanta outra questão ética: A facilidade, e velocidade, na criação de conteúdos fantasiosos que podem servir como fonte de notícias falsas, as famosas fake news.

A sofisticação dos conteúdos obtidos com ChatGPT dificulta a capacidade de distinguir entre o que é real e o que não é, assim, conteúdos falsos gerados dessa forma podem rapidamente serem distribuídos entre a população, contribuindo para enfraquecimento na confiança pública nas fontes de informação e levando a uma sociedade “informada” por pós verdade.

As implicações éticas em torno do uso do ChatGPT e outras IAs generativas constituem um campo vasto. Este trabalho optou por discutir questões relacionadas à origem dos dados utilizados no treinamento dos modelos de IA, os riscos ideológicos introduzidos durante esse processo e a facilidade na geração de notícias falsas. A seguir, serão apresentadas possíveis soluções para mitigar esses problemas.

Primeiramente, a respeito da privacidade e consentimento dos dados. É imprescindível a implementação de políticas de transparência que informem aos usuários sobre como seus dados poderão ser utilizados para treinamento de modelos de IA. Além disso, assim como a iniciativa da Meta, oferecer uma opção para que o usuário informe que não permite que seus dados sejam usados para este fim.

Com relação ao problema do viés algorítmico, é necessário garantir que os conjuntos de dados utilizados nos treinamentos sejam diversos, garantindo assim representatividade desses dados. Outra medida aplicável para mitigar o viés é a realização de auditorias, buscando identificação e correção de possíveis vieses. Além disso, a capacitação dos desenvolvedores e pesquisadores em boas práticas éticas aplicadas à IA pode contribuir para que os vieses sejam reduzidos ainda na fase de desenvolvimento.

O combate à disseminação de fake news e conteúdos enganosos de modo geral demanda tanto esforços por parte das empresas por trás dos modelos de IA quanto das redes sociais e do governo. As empresas devem implementar ferramentas, ou consumir de empresas terceiras, para verificar e monitorar a qualidade e a veracidade dos conteúdos gerados por seus modelos. Já as plataformas de redes sociais devem sinalizar conteúdos potencialmente falsos ou enganosos, ou no mínimo, mencionar quanto a geração da resposta/conteúdo por um mecanismo de inteligência artificial. Por sua vez, o governo deve promover políticas de alfabetização midiática à população, capacitando as pessoas a discernir entre informações confiáveis e falsas, sem serem influenciadas por crenças pessoais e focando, em especial, na educação infantil, como meio de desenvolver uma geração de pessoas mais bem preparada para uso dos recursos cibernéticos.

Resumindo, o enfrentamento das implicações éticas relacionadas ao ChatGPT exige um esforço conjunto entre desenvolvedores, pesquisadores, empresas, órgãos reguladores e população em geral.

3 CONCLUSÃO

O rápido crescimento das tecnologias de informação e comunicação traz benefícios, mas também desafios para salvaguardar os direitos humanos e a privacidade. A disseminação da IA, acessível a grande parte da população mundial, levanta questões sobre os desafios enfrentados pela sociedade informacional. Compreender e tratar esses desafios éticos e morais é urgente e necessário. Acompanhando a evolução tecnológica e garantindo que regras éticas estejam atualizadas, podemos evitar resultados ineficazes ou obsoletos. O envolvimento de reguladores, desenvolvedores de tecnologia, sociedade e governos é essencial para criar regras que abranjam e arbitrem todos os interesses comuns. Além disso, é necessário adaptar os padrões éticos para a sociedade informacional, considerando o uso das tecnologias desde a infância e a inclusão de gerações anteriores menos familiarizadas com essas inovações.

REFERÊNCIAS

1. RUSSEL, S.; NORVIG, P. Artificial Intelligence: A modern approach. Ed. Prentice-Hall, Egnlewood Cliffs, 1995.

2. AI Business - UBS: ChatGPT May Be the Fastest Growing App of All Time. Disponível em: https://aibusiness.com/nlp/ubs-chatgpt-is-the-fastest-growing-app-of-all-time. Acesso em 16 jun. 2024.

3. OpenAI – What is ChatGPT?. Disponível em: https://help.openai.com/en/articles/6783457-what-is-chatgpt. Acesso em 16 jun. 2024.

4. Aos Fatos - Como impedir que a Meta use seus dados para alimentar modelos de IA. Disponível em: https://www.aosfatos.org/noticias/como-impedir-uso-dados-pessoais-ia-meta-facebook/. Acesso em 21 jun. 2024.

5. Variety – Major Labels Sue AI Music Services Suno and Udio for Copyright Infringement. Disponível em: https://variety.com/2024/music/news/record-labels-sue-ai-music-services-suno-and-udio-copyright-infringement-1236045366/. Acesso em 25 jun. 2024

APÊNDICE L – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido: Whither AutoML? Understanding the Role of Automation in Machine Learning Workflows

Qual o **objetivo** do estudo descrito pelo artigo?	Qual o **problema/ oportunidade/ situação** que levou à necessidade de realização desse estudo?	Qual a **metodologia** que os autores usaram para obter e analisar as informações do estudo?	Quais os **principais resultados** obtidos pelo estudo?
O objetivo do estudo é compreender os papéis das ferramentas de automação em machine learning no dia a dia, analisando como os usuários interagem com elas e em quais momentos do fluxo de machine learning essas ferramentas são empregadas. Assim, busca-se oferecer diretrizes de design para desenvolvedores de ferramentas de automação, contribuindo para a melhoria da experiência dos usuários e promovendo maior confiabilidade no uso dessas ferramentas.	A motivação deste estudo surge da crescente popularidade e facilidade de acesso às ferramentas de automação de machine learning, que permitem que usuários com diferentes níveis de experiência em tecnologia façam uso dessas ferramentas. Devido a essa diversidade de experiências, surgem desafios relacionados à confiança, compreensão e eficácia no uso dessas ferramentas, o que justifica a necessidade de investigar como os usuários interagem com elas e como isso pode guiar o design dessas tecnologias.	A metodologia utilizada foi um estudo qualitativo baseado em entrevistas semi-estruturadas. Dezesseis participantes foram selecionados, com uma variedade de experiências em tecnologia e contextos de uso de ferramentas de automação em projetos reais de machine learning. As entrevistas foram transcritas, e os autores empregaram uma abordagem de codificação/categorização para identificar temas e padrões nas respostas. Essa análise concentrou-se em como os usuários interagiam com as ferramentas de automação de machine learning, suas percepções sobre a automação e as dinâmicas de confiança e controle.	De modo geral, o estudo concluiu que as ferramentas de automação de machine learning são eficazes em tornar o machine learning mais acessível e em simplificar diversas atividades. No entanto, algumas considerações importantes devem ser levadas em conta: a automação completa não é necessária nem desejada pelos usuários. É fundamental que as ferramentas envolvam diferentes etapas do fluxo de trabalho de machine learning, proporcionando uma abordagem mais completa e centralizada, que elimine a necessidade de múltiplas integrações. Além disso, é essencial que as interfaces sejam amigáveis e adaptáveis ao nível de proficiência dos usuários. Por fim, as ferramentas devem permitir uma compreensão clara de suas operações, garantindo, assim, a confiança dos usuários em seu uso.

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.

```
# Importação das bibliotecas
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Importação dos dados
(x_train, y_train), (x_test, y_test) =
tf.keras.datasets.fashion_mnist.load_data()

# Pré processamento dos dados
x_train, x_test = x_train/255.0, x_test/255.0

# Criando modelo
i = tf.keras.layers.Input(shape=(28, 28))
x = tf.keras.layers.Flatten()(i)
x = tf.keras.layers.Dense(128, activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
x = tf.keras.layers.Dense(10, activation="softmax")(x)
```

```
model = tf.keras.models.Model(i, x)

# Compilando modelo
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Treinando o modelo
result = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 10s 4ms/step - accuracy: 0.7670 - loss: 0.6712 - val_accuracy: 0.8414 - val_loss: 0.4391
Epoch 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8498 - loss: 0.4162 - val_accuracy: 0.8605 - val_loss: 0.3869
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.8635 - loss: 0.3718 - val_accuracy: 0.8615 - val_loss: 0.3849
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8727 - loss: 0.3417 - val_accuracy: 0.8711 - val_loss: 0.3553
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8808 - loss: 0.3289 - val_accuracy: 0.8700 - val_loss: 0.3547
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8830 - loss: 0.3155 - val_accuracy: 0.8734 - val_loss: 0.3533
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8850 - loss: 0.3050 - val_accuracy: 0.8785 - val_loss: 0.3354
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8915 - loss: 0.2959 - val_accuracy: 0.8806 - val_loss: 0.3312
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8919 - loss: 0.2865 - val_accuracy: 0.8700 - val_loss: 0.3556
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8927 - loss: 0.2844 - val_accuracy: 0.8825 - val_loss: 0.3324
```

```
# Avaliando o modelo
# Plotar a função de perda
plt.plot(result.history["loss"], label="loss")
plt.plot(result.history["val_loss"], label="val_loss")
plt.legend()
```

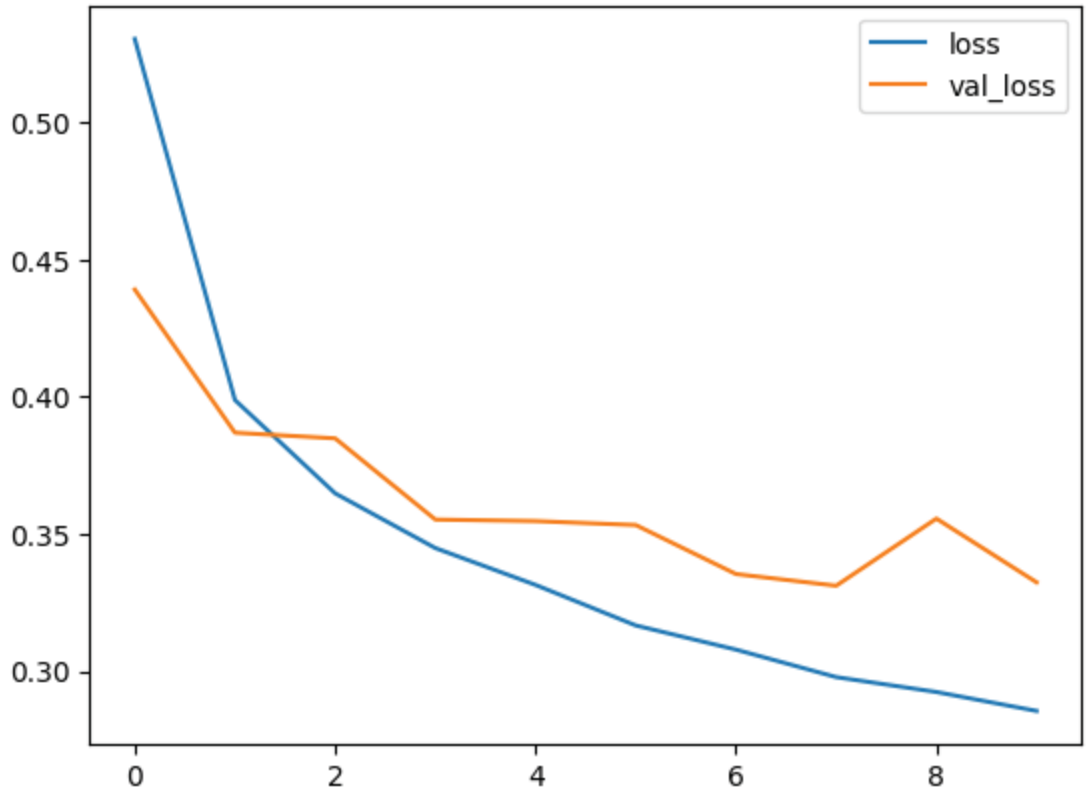


Gráfico de perda
Durante o treinamento do modelo, observamos que, em apenas 10 épocas, houve uma redução significativa na perda do treinamento, que caiu de aproximadamente 0.67 para menos de 0.3. Este resultado é positivo, indicando que o modelo conseguiu aprender a representação dos dados de treinamento de maneira eficiente.

No entanto, a perda de validação não seguiu a mesma tendência de redução esperada, apresentando oscilações ao longo das épocas. Isso pode sugerir a necessidade de mais épocas para observar uma redução mais estável na perda de validação. Alternativamente, se essas oscilações persistirem, pode ser necessário revisar tanto a qualidade e a quantidade dos dados de validação quanto a arquitetura do modelo. Vale destacar que oscilações na perda de validação podem ser indicativas de overfitting, onde o modelo ajusta-se muito bem aos dados de treinamento, mas não generaliza tão bem para dados não vistos.

```
# Plotar a acurácia
plt.plot(result.history["accuracy"], label="acc")
plt.plot(result.history["val_accuracy"], label="val_acc")
plt.legend()
```

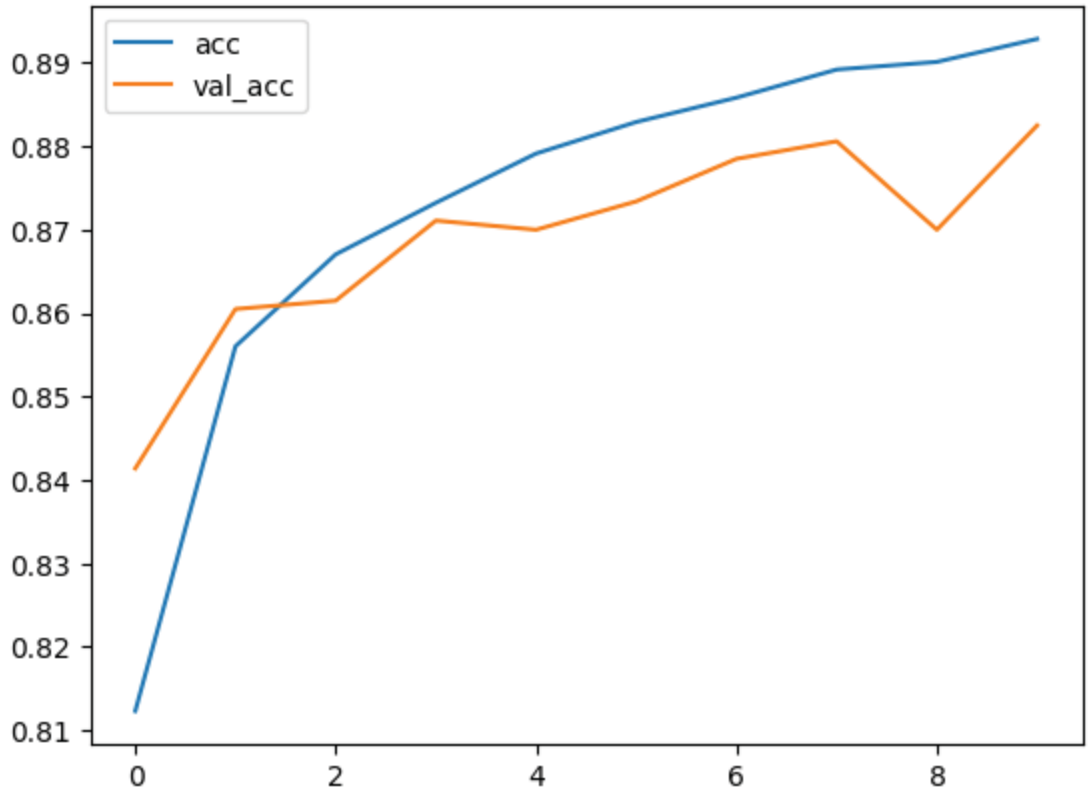
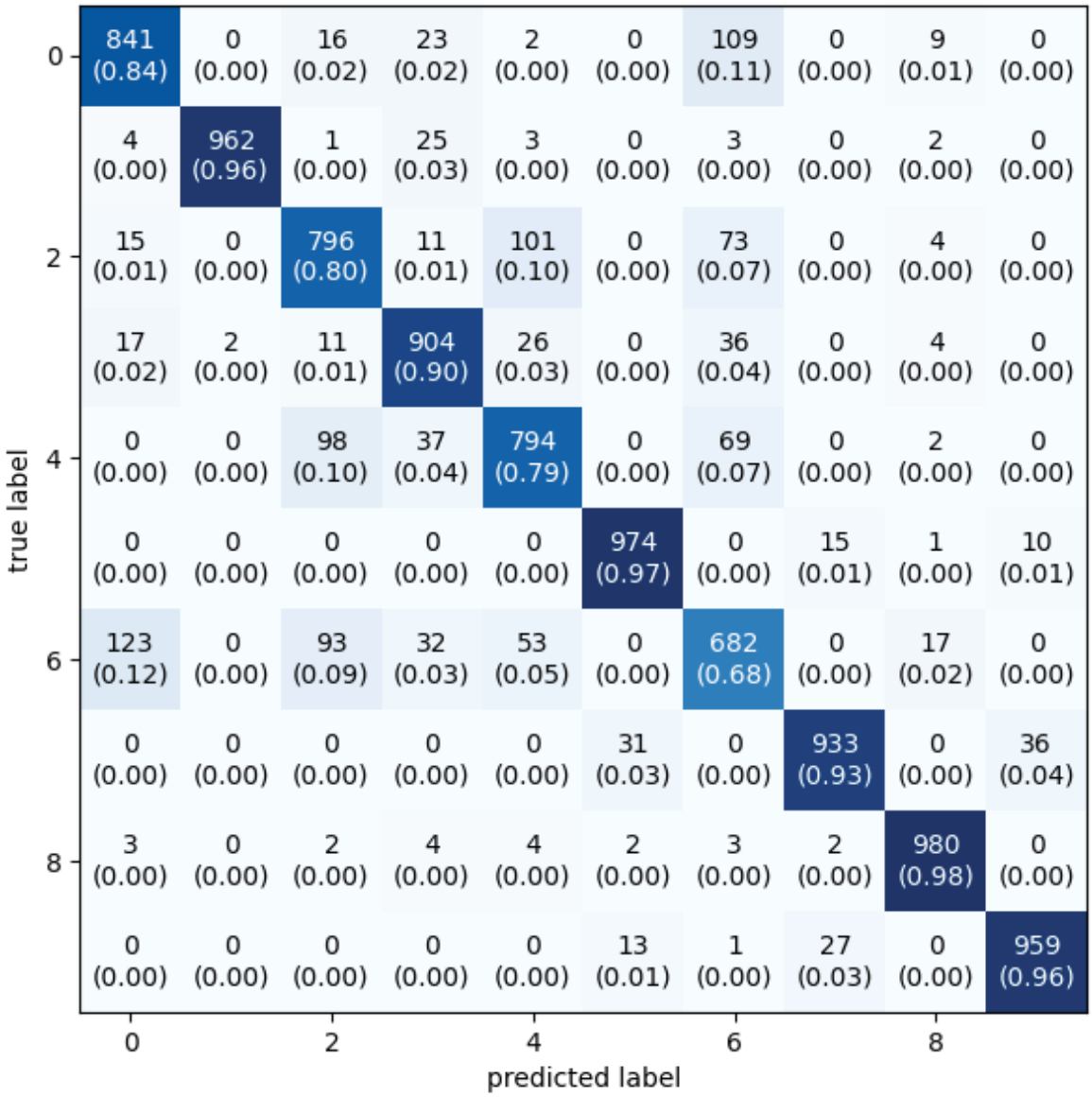


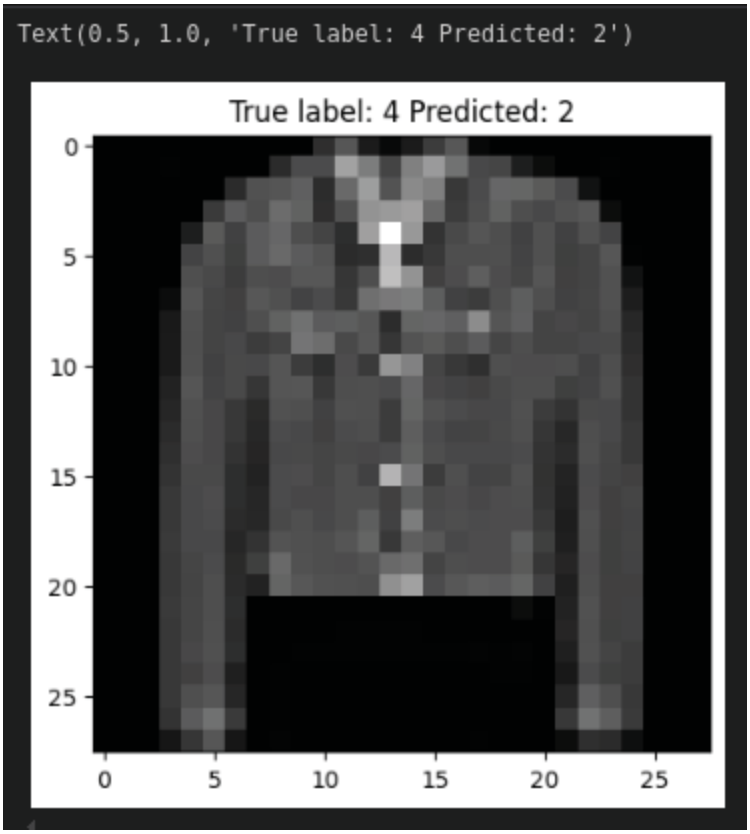
Gráfico de acurácia
Quanto à acurácia, o comportamento observado foi o esperado. A acurácia de treinamento teve um crescimento contínuo, alcançando um valor próximo de 0.9, o que indica que o modelo está aprendendo de forma eficaz. Já a acurácia de validação também cresceu, mas com pequenas flutuações. Embora tenha mantido uma tendência ascendente, essas oscilações podem ser um sinal de que o modelo ainda precisa de mais ajustes para estabilizar a generalização.

```
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8804 -
loss: 0.3317
[0.3324049711227417, 0.8824999928474426]

# Predições
y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)
313/313 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[9 2 1 ... 8 1 5]

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




```
# Visualizando classificações erradas
misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)
plt.imshow(x_test[i].reshape(28, 28), cmap="gray")
plt.title("True label: %s Predicted: %s" % (y_test[i], y_pred[i]))
```

```
Text(0.5, 1.0, 'True label: 4 Predicted: 2')
```



Classificação errada
No exemplo de classificação incorreta, temos uma amostra da categoria 4 que foi erroneamente classificada pelo modelo como categoria 2. Ao analisar a matriz de confusão, observamos que a maior dificuldade do modelo em relação à categoria 4 ocorreu justamente com a categoria 2. Especificamente, o modelo classificou 98 amostras da categoria 4 como pertencentes à categoria 2, resultando na maior taxa de confusão para essa classe.

2.

```
# Importação de bibliotecas
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.preprocessing import StandardScaler

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

```
data.head()
```

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

```
# Separando em variáveis independentes (X) e dependente (y)
X = data.drop("quality", axis=1).values
y = data["quality"].values

# Pré processamento dos dados
scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)

# Dividir os dados em treino e teste (75% treino, 25% teste)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y,random_state=42, test_size=0.25)

# Criando o modelo
i = tf.keras.layers.Input(shape=(11,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

model = tf.keras.models.Model(i, x)

# Compilação
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
loss='mse')

# Early stop para epochs
early_stop = tf.keras.callbacks.EarlyStopping(
                        monitor='val_loss',
                        patience=20,
                        restore_best_weights=True)

# Treinar o modelo
history = model.fit(X_train, y_train, epochs=1500,
validation_split=0.2,callbacks=[early_stop])
```

```
Epoch 1/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 3s 32ms/step - loss: 28.6536 - val_loss: 22.9032
Epoch 2/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 21.1536 - val_loss: 16.2289
Epoch 3/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 14.3322 - val_loss: 10.8584
Epoch 4/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 9.6321 - val_loss: 6.9232
Epoch 5/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 6.1569 - val_loss: 4.5370
Epoch 6/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 4.1051 - val_loss: 3.2617
Epoch 7/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.9594 - val_loss: 2.6936
Epoch 8/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - loss: 2.5560 - val_loss: 2.4147
Epoch 9/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2.2935 - val_loss: 2.2617
Epoch 10/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 2.1241 - val_loss: 2.1590
Epoch 11/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.8515 - val_loss: 2.0708
Epoch 12/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 1.9183 - val_loss: 1.9893
Epoch 13/1500
...
Epoch 191/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.2900 - val_loss: 0.3439
Epoch 192/1500
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2906 - val_loss: 0.3475
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
# Plotar os gráficos de loss
plt.figure(figsize=(12, 6))
plt.plot(history.history['loss'], label='Loss (Treinamento)')
plt.plot(history.history['val_loss'], label='Loss (Validação)')
plt.title('Função de Perda Durante o Treinamento')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

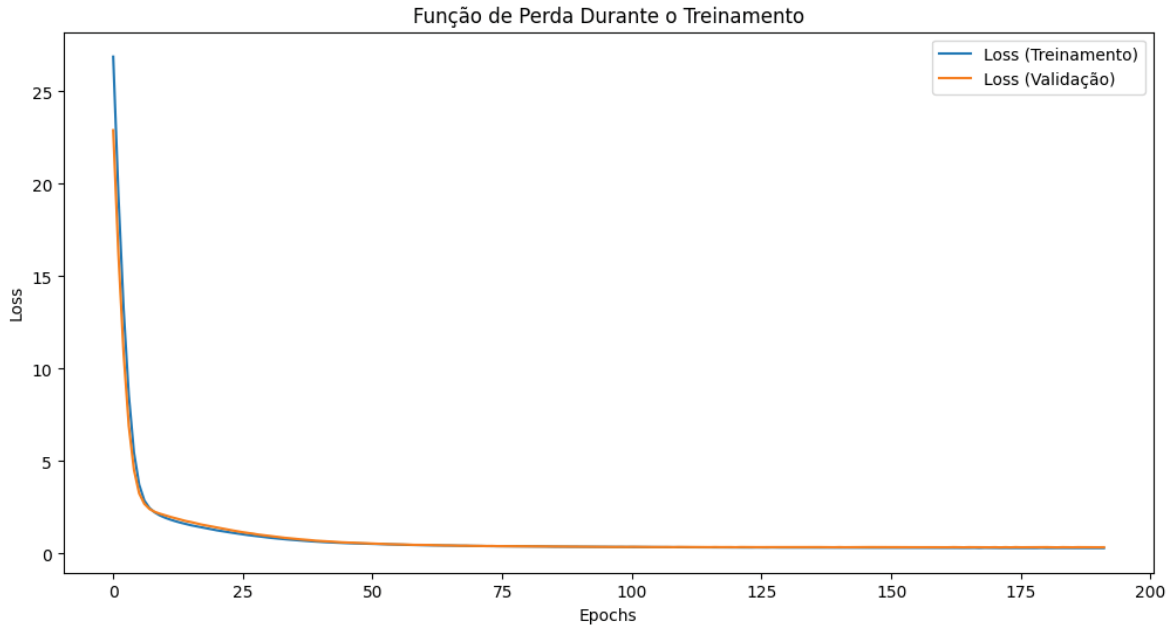


Gráfico de perda

Pelo gráfico de perda, podemos observar que por volta de 40 epochs, o modelo atingiu um valor mínimo de perda, o que implica na não necessidade de continuar treinando o modelo, visto que a perda se tornou estável.

```
# Fazer predições com o conjunto de teste
y_pred = model.predict(X_test).flatten()

# Calcular MAE, MSE e R²
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Plotar as métricas
print(f"MAE: {mae}")
print(f"MSE: {mse}")
print(f"R²: {r2}")
13/13 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
MAE: 0.4805771422386169
MSE: 0.3634197126441913
R²: 0.412506639957428
```

Avaliando as métricas

* MAE (0.48): O erro médio absoluto indica que o modelo prevê valores com uma média de erro moderada. Embora não seja muito alto, ainda há espaço para reduzir a diferença entre as previsões e os valores reais.
* MSE (0.36): Esse valor reforça que o modelo não comete muitos erros grandes, mas ainda apresenta alguma inconsistência que poderia ser melhorada.
* R² (0.41): O modelo explica 41% da variação nas classes, indicando uma correlação modesta com as classes reais. Isso sugere que ele ainda não captura totalmente as variações no padrão dos dados.

Em resumo, o modelo possui um desempenho razoável, mas ajustes finos, como aprimoramento no pré-processamento ou experimentação com diferentes configurações, poderiam melhorar os resultados.

3.

```
# Importação de 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

# Leitura dos arquivos
# Antes de executar, importar para o ambiente o arquivo base
data = pd.read_csv('./Base_livros.csv', sep=',', on_bad_lines='skip')

data.head()
```

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


```
# Limpar caracteres indesejados das colunas
data.columns = data.columns.str.replace(r'[^a-zA-Z0-9\s]', '',
regex=True)
data.head()
```

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

```
# remove caracteres indesejados das notas
data['Notas'] = data['Notas'].str.replace(r'[^a-zA-Z0-9\s]', '',
regex=True)
data.head()
```

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

```
data.dtypes
data['Notas'].unique()

# função para avaliar se é possível converter as notas para int
def castable(value):
```

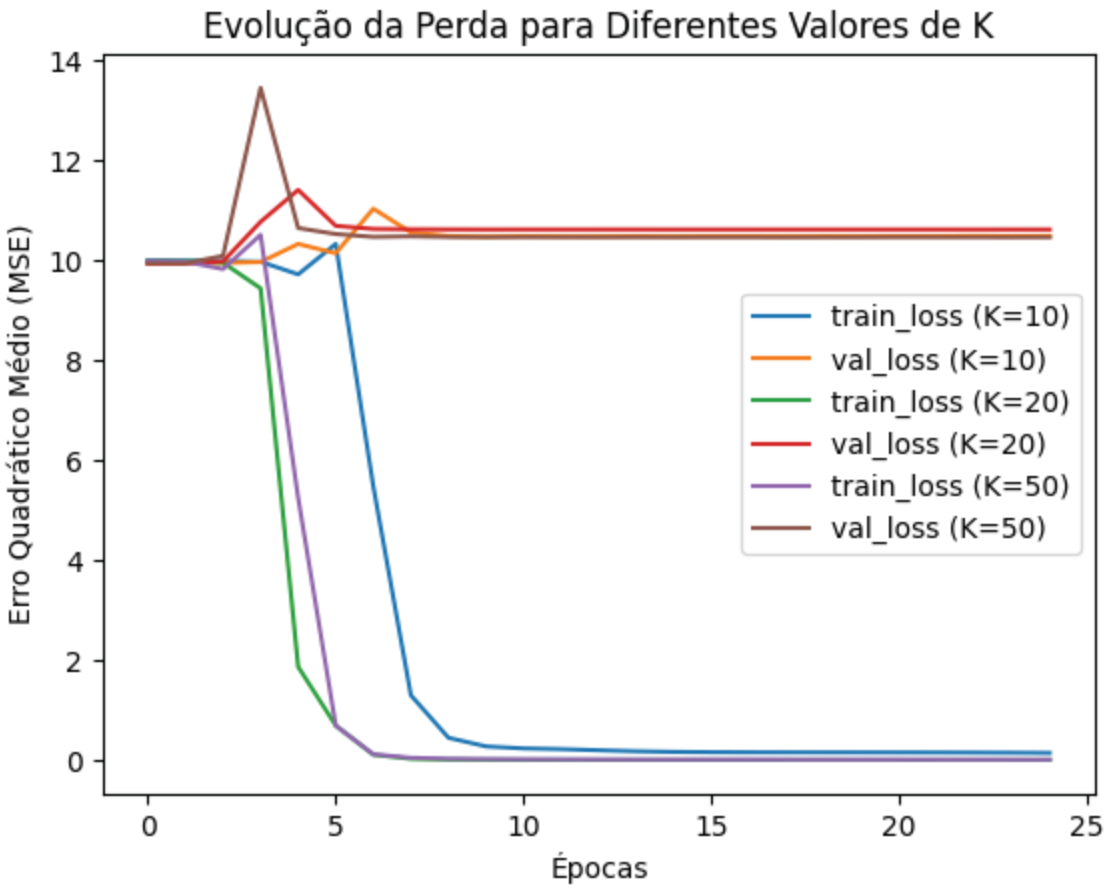
```
 try:
        int(value)
        return True
 except:
        return False

# verifica e remove linhas que não contém notas válidas
data = data[data['Notas'].apply(castable)]
# converte as notas para int
data['Notas'] = data['Notas'].astype(int)

# Converter IDusuario e ISBN em categóricos e criar novos códigos
data.IDusuario = pd.Categorical(data.IDusuario)
data['new_user_id'] = data.IDusuario.cat.codes

data.ISBN = pd.Categorical(data.ISBN)
data['new_isbn'] = data.ISBN.cat.codes

data.head(15)
```

	ISBN	Titulo	Autor	Ano	Editora	IDusuario	Notas	new_user_id	new_isbn
0	2005018	Clara Callan	Richard Bruce Wright	2001	HarperFlamingo Canada	276725	0	6628	22749
1	60973129	Decision in Normandy	Carlo D'Este	1991	HarperPerennial	276726	2	6629	72317
2	374157065	Flu: The Story of the Great Influenza Pandemic...	Gina Bari Kolata	1999	Farrar Straus Giroux	276727	6	6630	40165
3	393045218	The Mummies of Urumchi	E. J. W. Barber	1999	W. W. Norton & Company	276729	1	6631	45721
4	399135782	The Kitchen God's Wife	Amy Tan	1991	Putnam Pub Group	276729	9	6631	47849
5	425176428	What If?: The World's Foremost Military Histor...	Robert Cowley	2000	Berkley Publishing Group	276733	0	6633	50007
6	671870432	PLEADING GUILTY	Scott Turow	1993	Audioworks	276736	3	6634	79250
7	679425608	Under the Black Flag: The Romance and the Real...	David Cordingly	1996	Random House	276737	9	6635	80006
8	074322678X	Where You'll Find Me: And Other Stories	Ann Beattie	2002	Scribner	276744	4	6636	5368
9	771074670	Nights Below Station Street	David Adams Richards	1988	Emblem Editions	276745	9	6637	89328
10	080652121X	Hitler's Secret Bankers: The Myth of Swiss Neu...	Adam Lebor	2000	Citadel Press	276746	7	6638	6005
11	887841740	The Middle Stories	Sheila Heti	2004	House of Anansi Press	276746	3	6638	104024
12	1552041778	Jane Doe	R. J. Kaiser	1999	Mira Books	276746	8	6638	12133
13	1558746218	A Second Chicken Soup for the Woman's Soul (Ch...	Jack Canfield	1998	Health Communications	276746	7	6638	13206
14	1567407781	The Witchfinder (Amos Walker Mystery Series)	Loren D. Estleman	1998	Brilliance Audio - Trade	276746	5	6638	15273

```
# Dimensões
N = len(set(data.new_user_id))
M = len(set(data.new_isbn))
user_ids, isbn, ratings = shuffle(data.new_user_id, data.new_isbn,
data.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_book = isbn[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_book = isbn[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

# Lista de valores de K para testar
embedding_sizes = [10, 20, 50]
```

```
# Dicionário para armazenar os resultados
results = {}

for K in embedding_sizes:
      print(f"\nTreinando modelo com K = {K}")

      # Camada de entrada e embedding para usuários
      u = Input(shape=(1,))
      u_emb = Embedding(N, K)(u)
      u_emb = Flatten()(u_emb)

      # Camada de entrada e embedding para livros
      m = Input(shape=(1,))
      m_emb = Embedding(M, K)(m)
      m_emb = Flatten()(m_emb)

      # Concatenar embeddings e passar pelas camadas densas
      x = Concatenate()([u_emb, m_emb])
      x = Dense(1024, activation="relu")(x)
      x = Dense(1)(x)

      # Criar o modelo
      model = Model(inputs=[u, m], outputs=x)

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

      # Treinamento do modelo
      r = model.fit(
      x=[train_user, train_book],
      y=train_ratings,
      epochs=25,
      batch_size=1024,
      verbose=2,
      validation_data=([test_user, test_book], test_ratings)
      )

      # Salvar histórico de treinamento
      results[K] = {
      "history": r.history,
      "model": model
      }

      # Plotar os gráficos de perda para análise
      plt.plot(r.history["loss"], label=f"train_loss (K={K})")
      plt.plot(r.history["val_loss"], label=f"val_loss (K={K})")

# Mostrar o gráfico consolidado
```

```
plt.title("Evolução da Perda para Diferentes Valores de K")
plt.xlabel("Épocas")
plt.ylabel("Erro Quadrático Médio (MSE)")
plt.legend()
plt.show()
```




```
## Avaliação função de perda
Os resultados obtidos indicam que o modelo apresenta sobreajuste, com a perda de treinamento diminuindo significativamente, mas a perda de validação permanecendo estável em torno de 10,5. Isso sugere que o modelo está ajustando bem os dados de treinamento, mas não consegue generalizar para o conjunto de validação. Além disso, não houve melhorias significativas na perda de validação ao testar diferentes valores para K. Logo, é necessário reavaliar o modelo, modificar sua arquitetura, testar diferentes funções de perda e avaliar a qualidade e representatividade dos dados para melhorar o desempenho.

# Nova recomendação
# Gerar o array com o usuário único
# repete a quantidade de livros
books = np.array(list(set(isbn)))
input_usuario = np.repeat(a=6635, repeats=M)

preds = model.predict( [input_usuario, books] )

# descentraliza as predições
```

```
rat = preds.flatten() + avg_rating

# índice da maior nota
idx = np.argmax(rat)

print("Recomendação: Livro - ", books[idx], " / ", rat[idx] , "*")
3397/3397 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step
Recomendação: Livro -  50343  /  10.029646 *

## Avaliação da recomendação
O modelo recomendou que o usuário com ID 6636 daria nota de
aproximadamente 10 para o livro com ISBN 41471.
```

4.

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

# Importação da imagem
url =
'https://upload.wikimedia.org/wikipedia/commons/b/b6/Felis_catus-cat_on_s
now.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)

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

# Leitura da Imagem
original_img = download(url, max_dim=500)
show(original_img)
display.display(display.HTML('<a
"href=https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg
```

```
"/>'))
```

```
# Preparando modelo de classificação

base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')

# Selecionando as camadas da rede para maximizar a perda
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]
# Criação do modelo dream
dream_model = tf.keras.Model(inputs=base_model.input, outputs=layers)
# Função para calcular a perda
def calc_loss(img, model):
  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)

# DeepDream
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:
      tape.watch(img)
      loss = calc_loss(img, self.model)

      gradients = tape.gradient(loss, img)

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

      img = img + gradients*step_size
      img = tf.clip_by_value(img, -1, 1)

      return loss, img

deepdream = DeepDream(dream_model)

# 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
```

```
# Aplicando Main Loop

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

```
### Explicação resultado Main Loop
A imagem onírica obtida após o Main Loop representa uma visão alucinada e
distorcida da imagem original. Apesar da baixa resolução, podemos
observar que alguns padrões detectados pela rede neural foram ampliados,
contudo esses padrões não parecem ser tão distintos entre si

# Levando o modelo até uma 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
```

Explicação da imagem obtida com oitava

A imagem obtida da aplicação da técnica das oitavas apresenta uma aparência mais complexa e detalhada com relação aos padrões aprendidos pela rede neural. Isso ocorre pois a imagem é passada pela rede em diferentes escalas de tamanho.

Explicação das diferenças entre imagens oníricas obtidas com Main Loop e levando o modelo até a oitava.
A principal diferença entre as imagens está na **complexidade e diversidade** dos padrões resultados conforme a técnica aplicada:
* Com o Main Loop, a imagem onírica é uma visão distorcida da original, onde os padrões detectados são ampliados, mas de maneira uniforme e em baixa resolução, resultando em detalhes pouco distintos e uma aparência menos rica.

* Já com a técnica das oitavas, a imagem passa por diferentes escalas de tamanho, o que cria uma composição mais detalhada e complexa. Assim, os padrões surgem em várias granularidades, desde texturas finas até formas maiores, dando profundidade e diversidade à imagem final.

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

Cobertura Vacinal contra poliomielite no Brasil: Comparação Regional

A poliomielite é uma doença viral contagiosa que pode causar paralisia permanente e, em casos mais graves, levar à morte. Embora tenha sido praticamente erradicada em muitas partes do mundo devido à vacinação, a cobertura vacinal no Brasil tem diminuído nos últimos anos, ficando abaixo da meta estabelecida desde 2016[1]. Essa redução aumenta o risco de reintrodução do vírus, colocando a população, especialmente crianças, em perigo. Este trabalho tem como objetivo analisar e visualizar a cobertura vacinal contra a poliomielite em diferentes regiões do país, destacando áreas críticas que necessitam de maior atenção e estratégias para reverter esse cenário.

[1] BRASIL. Ministério da Saúde. Brasil reverte tendência de queda nas coberturas vacinais e oito imunizantes do calendário infantil registram alta em 2023. Disponível em: https://www.gov.br/saude/pt-br/assuntos/noticias/2023/dezembro/brasil-reverte-tendencia-de-queda-nas-coberturas-vacinais-e-oito-imunizantes-do-calendario-infantil-registram-alta-em-2023. Acesso em: 15/02/2025.

CONJUNTO DE DADOS

Os dados analisados neste estudo foram extraídos do DATASUS[2], abrangendo informações sobre a cobertura vacinal contra a poliomielite no Brasil entre 2016 e 2022. O conjunto de dados apresenta a taxa de vacinação (em porcentagem) por região, permitindo uma avaliação das tendências ao longo dos anos.

Representação gráfica do conjunto de dados utilizado.

	ANO	NORTE	NORDESTE	SUDESTE	CENTRO-OESTE	SUL
0	2016	72.28	81.55	86.31	96.15	87.50
1	2017	75.67	81.92	87.56	84.44	89.82
2	2018	77.06	90.04	92.66	88.59	89.91
3	2019	79.59	82.73	84.54	85.40	89.04
4	2020	65.69	73.11	78.28	80.47	86.50
5	2021	62.29	68.53	71.53	74.22	79.98
6	2022	71.24	78.50	75.14	80.50	83.10

Fonte: Elaboração própria.

CONTEXTO E PÚBLICO ALVO

A queda na cobertura vacinal tem se tornado uma preocupação crescente no Brasil, especialmente em relação à poliomielite, uma doença erradicada no país há 34 anos[3], mas que pode ressurgir caso a imunização continue em declínio. Diante desse cenário, este estudo busca comparar as taxas de vacinação entre as regiões brasileiras, identificando áreas com maior ou menor adesão ao longo do tempo.

O público-alvo são gestores de saúde pública e tomadores de decisão que podem usar essas informações para desenvolver políticas mais eficazes de incentivo à vacinação.

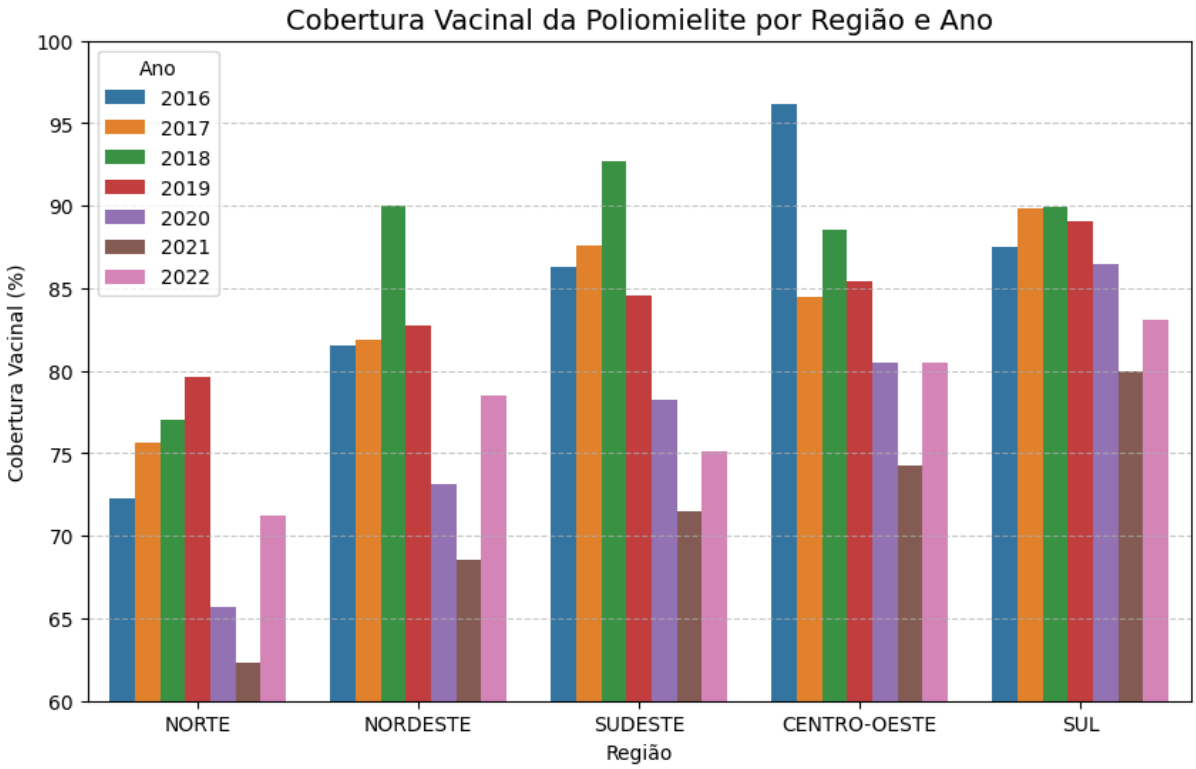
---

[2] BRASIL. Ministério da Saúde. DATASUS. Coberturas vacinais – PNI Brasil. Disponível em: http://tabnet.datasus.gov.br/cgi/dhdat.exe?bd_pni/cpnibr.def. Acesso em: 15/02/2025.

[3] BRASIL. Ministério da Saúde. *Há 34 anos, último caso de poliomielite foi registrado no Brasil.* Disponível em: https://www.gov.br/saude/pt-br/assuntos/noticias/2023/marco/ha-34-anos-ultimo-caso-de-poliomielite-foi-registrado-no-brasil. Acesso em:15/02/2025.

VISUALIZAÇÃO DOS DADOS

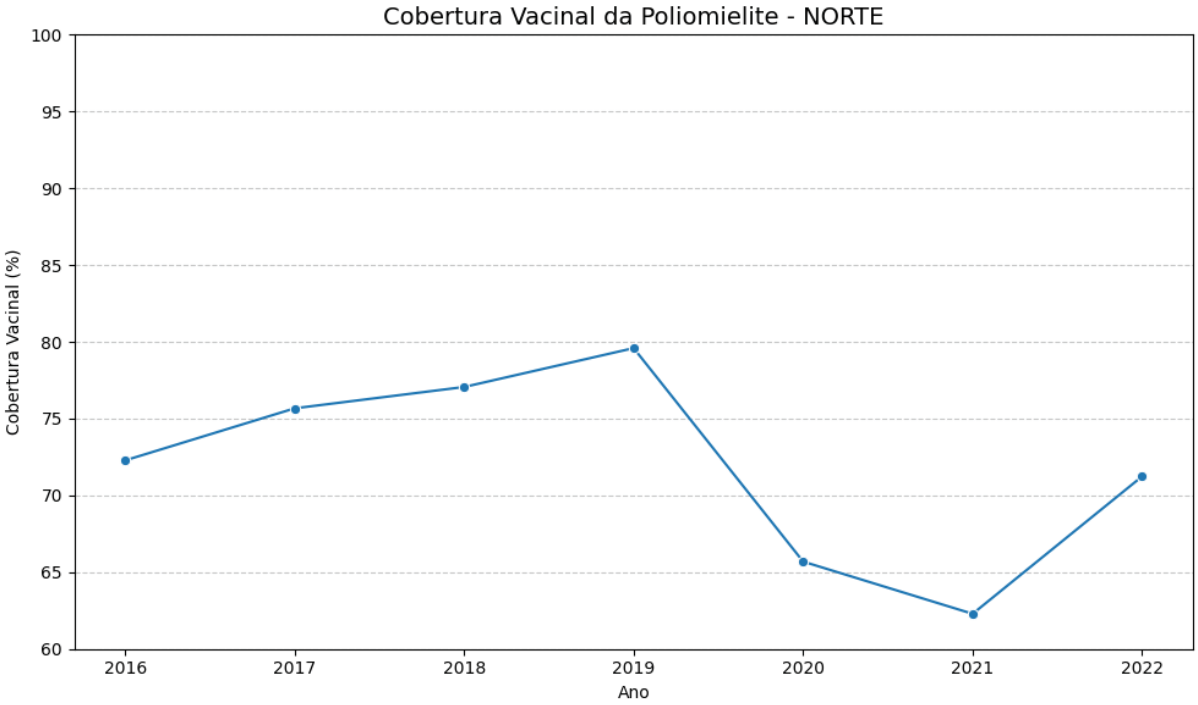
Para elaboração da visualização dos dados, foram utilizadas como ferramentas a linguagem *python* e bibliotecas auxiliares, como *pandas, seaborn e matplotlib*. Foram gerados diferentes tipos de gráficos para facilitar a compreensão da variação da taxa de vacinação ao longo do tempo.



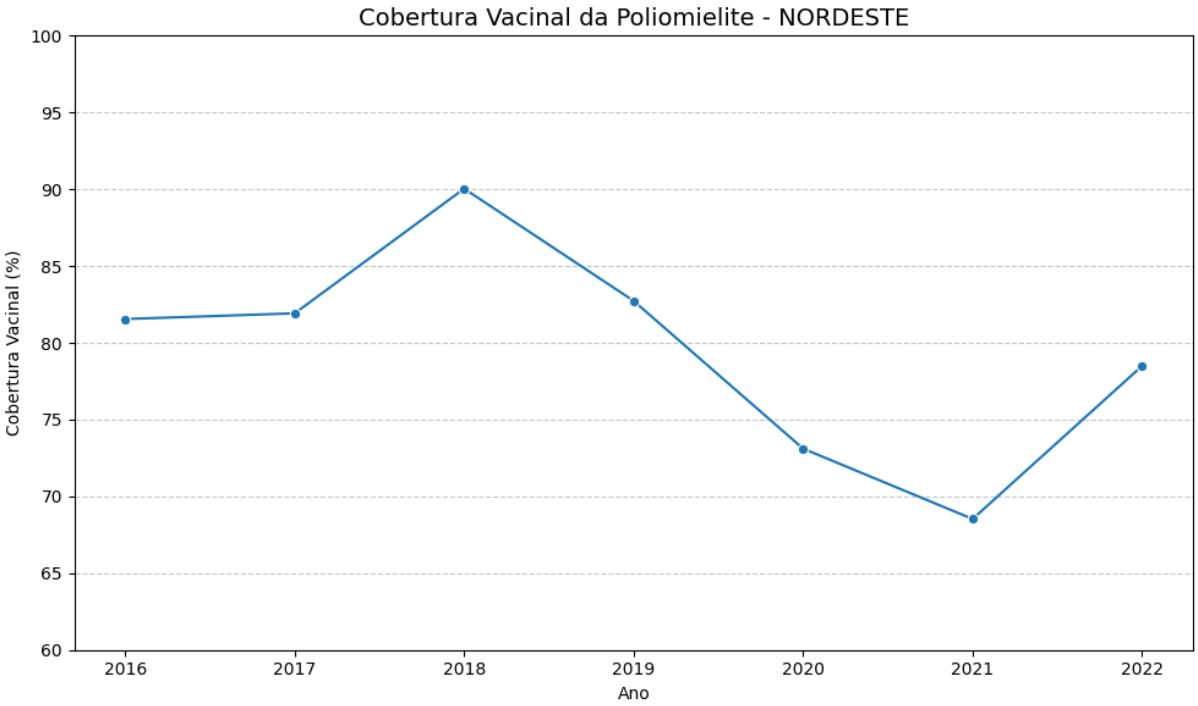
Fonte: Elaboração própria.

O gráfico de barras foi utilizado para ilustrar a taxa de cobertura vacinal por região em um determinado ano. Esse modelo permite uma comparação direta entre as regiões e evidencia diferenças significativas na taxa de imunização, contudo ele possui a limitação de não ser ideal para representar tendências ao longo do tempo.

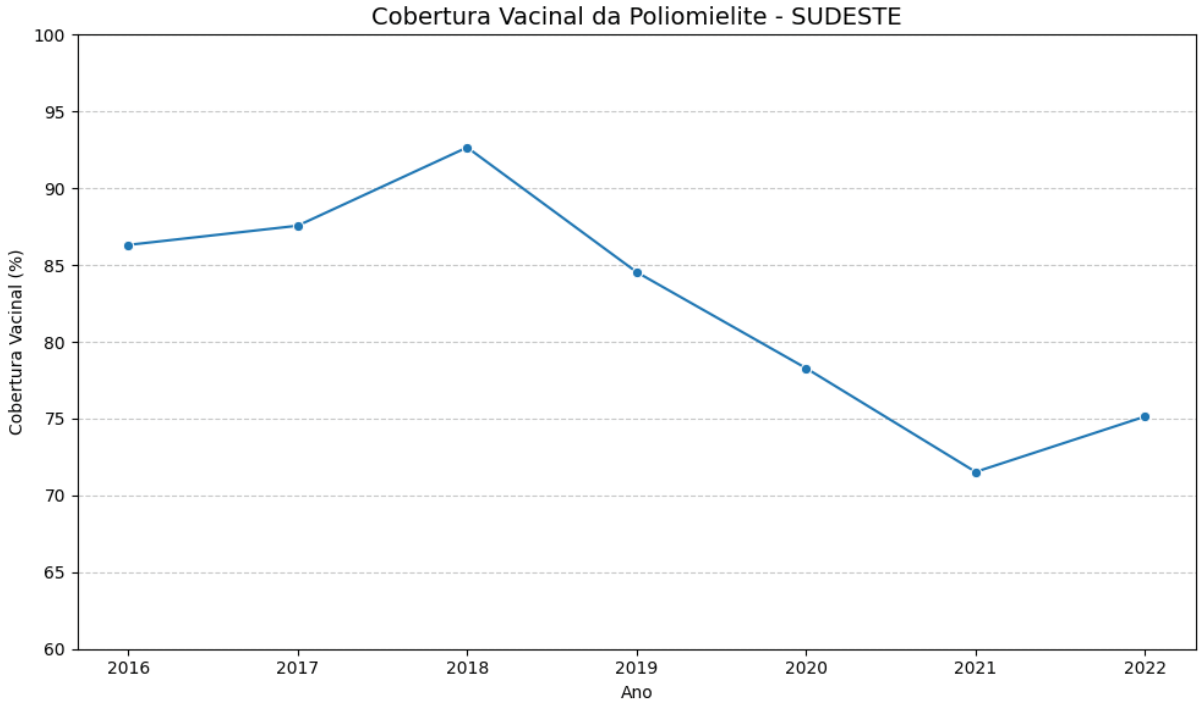
O gráfico de linhas foi escolhido para representar a evolução da taxa de vacinação ao longo dos anos. Esse tipo de visualização é eficaz para identificar padrões e quedas acentuadas. A seguir, serão demonstrados gráficos de linha para cada região.



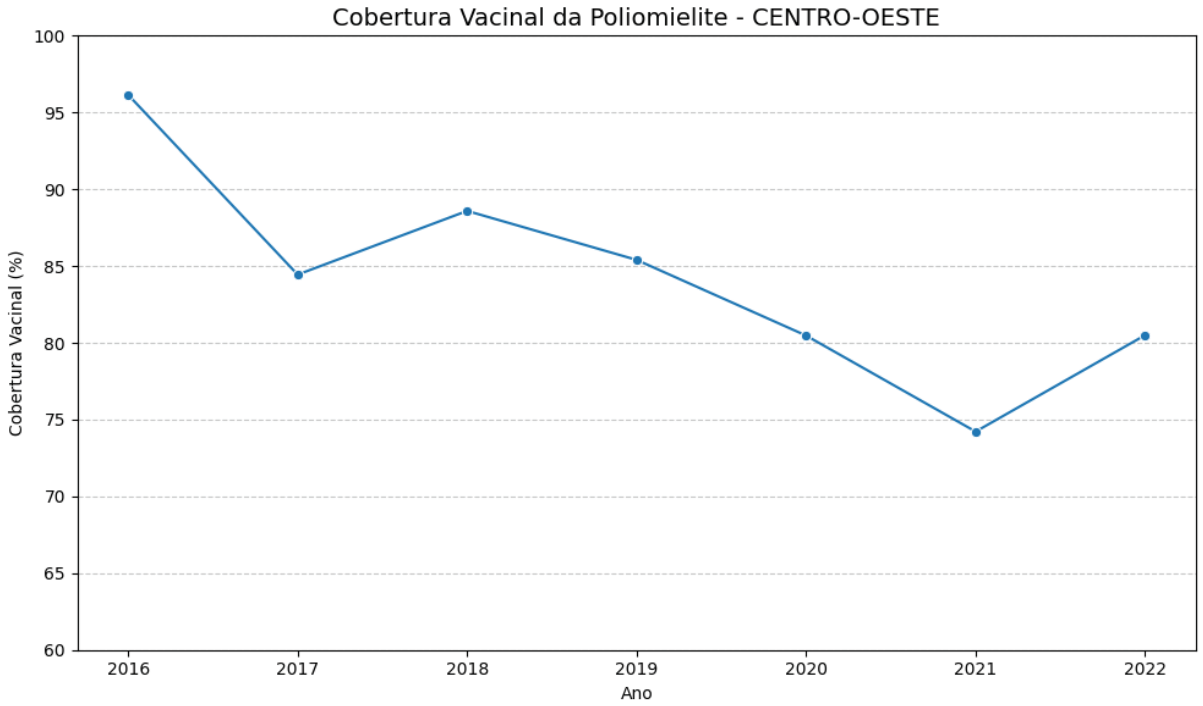
Fonte: Elaboração própria.



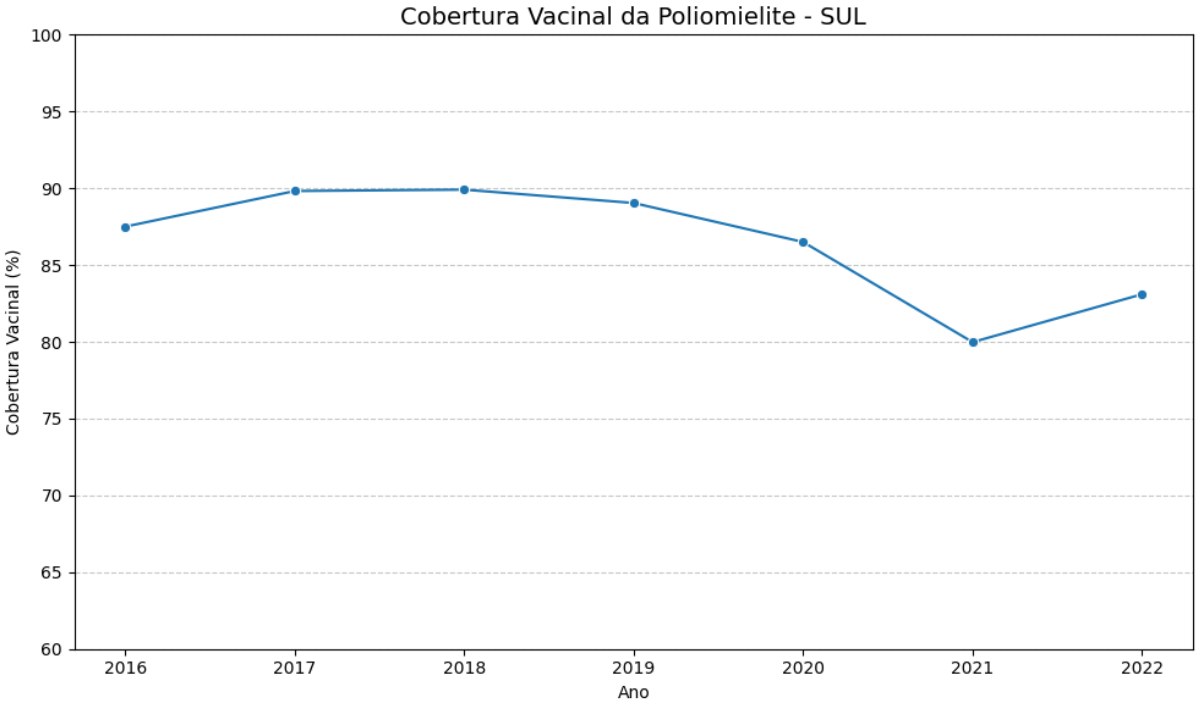
Fonte: Elaboração própria.



Fonte: Elaboração própria.

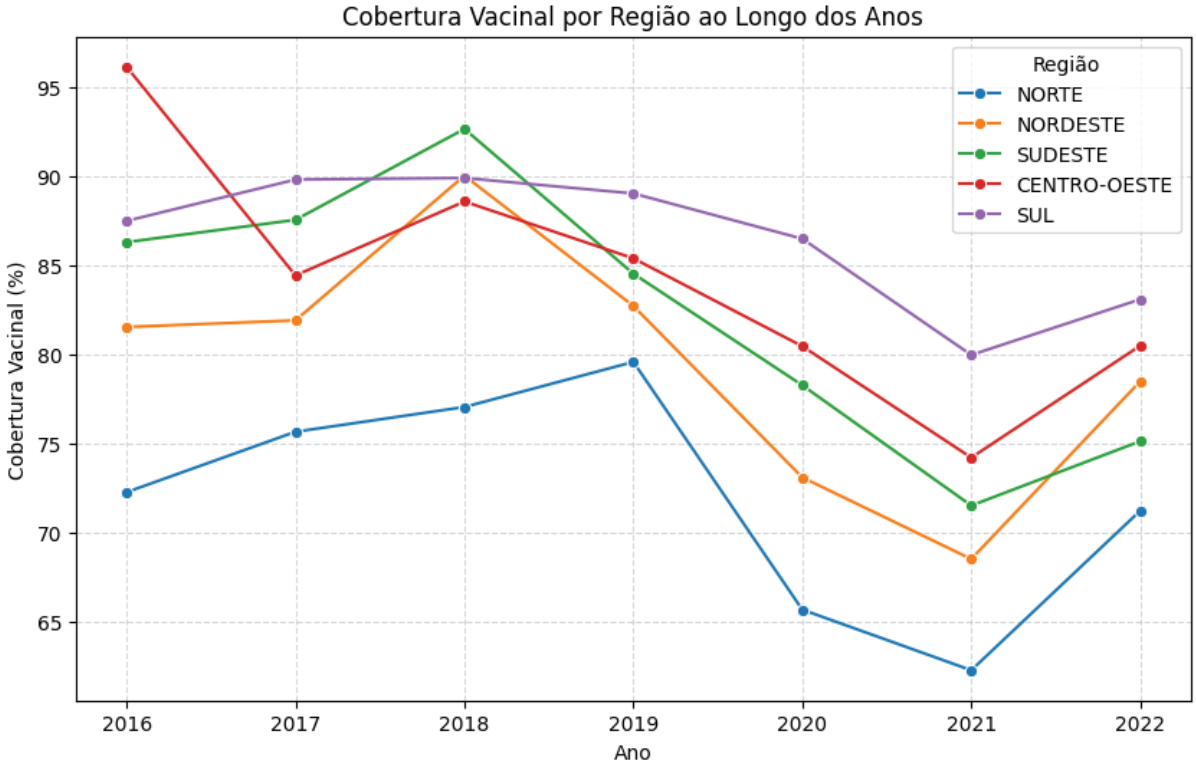


Fonte: Elaboração própria.



Fonte: Elaboração própria.

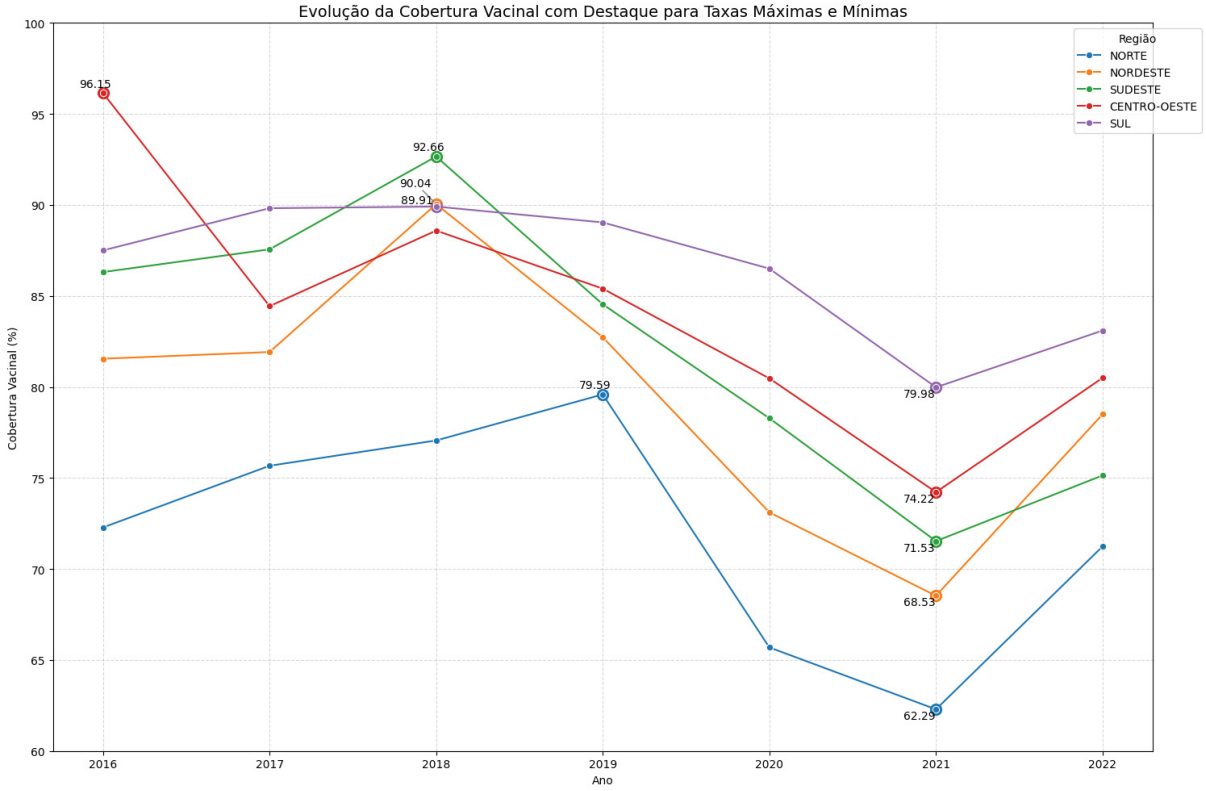
Para facilitar a comparação entre as regiões, foi criado um gráfico de linhas unificado, onde cada região é representada por uma cor diferente. Esse gráfico destaca claramente as diferenças entre as taxas e mostra quais regiões tiveram as quedas mais bruscas.



Fonte: Elaboração própria.

Com base no gráfico apresentado, é possível concluir que todas as regiões apresentaram quedas a partir de 2019, com a região Sul se mantendo mais estável em comparação com as demais, e a região Norte apresentando a menor taxa de vacinação ao longo do período.

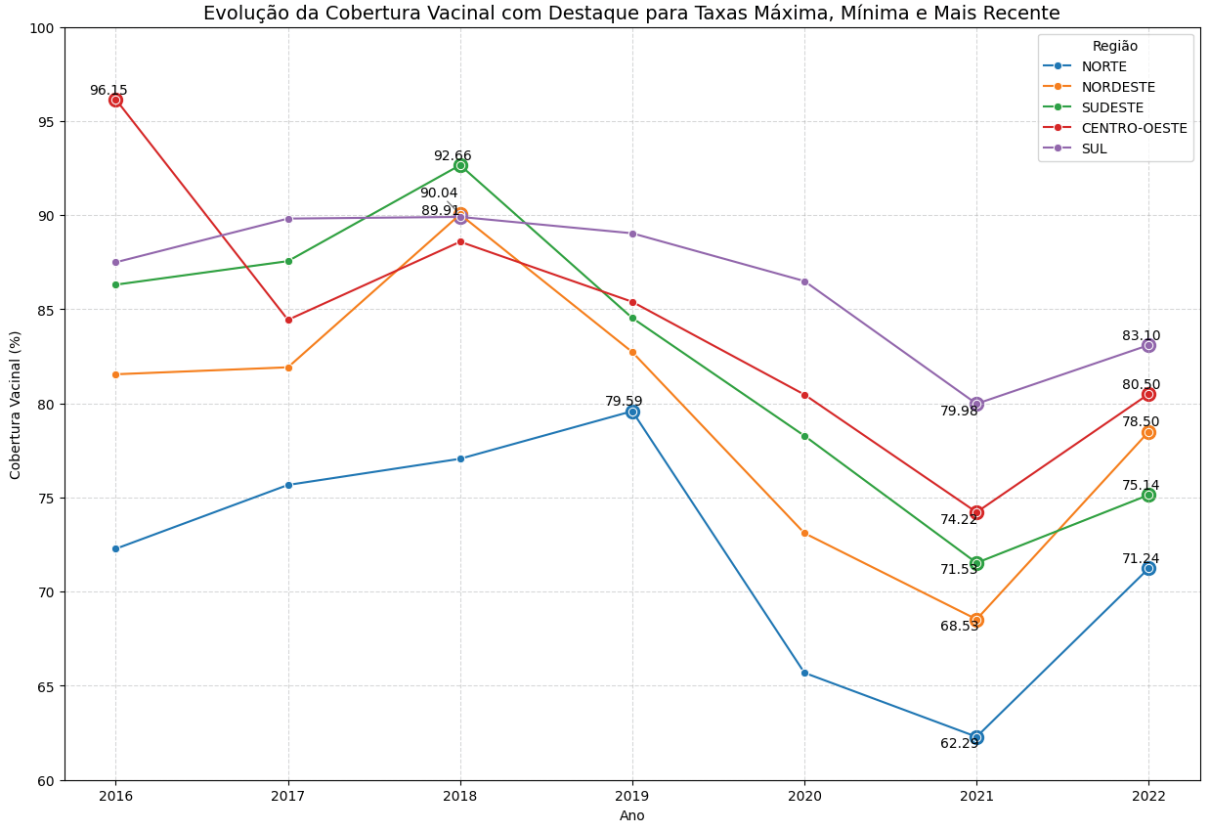
Uma melhoria implementada no gráfico de linhas unificado foi o de destacar os pontos máximos e mínimos de cada região.



Fonte: Elaboração própria.

Essa adição facilita a percepção da amplitude das variações, permitindo que o leitor compreenda com mais clareza a grandeza das diferenças entre os períodos analisados.

Por fim, também foi destacado o valor mais recente dentro do conjunto de dados, a ênfase no valor mais recente facilita a comparação com a maior taxa de cobertura vacinal já registrada, permitindo visualizar claramente o quanto a recuperação ainda está distante. Isso destaca a necessidade de intensificar ações para alcançar os níveis anteriores de vacinação e melhorar os indicadores nas regiões mais afetadas.



Fonte: Elaboração própria.

STORYTELLING

Nos últimos anos, o Brasil tem enfrentado desafios em relação à cobertura vacinal, com variações significativas entre as regiões. Dada a diversidade territorial do país, é essencial analisar a adesão à vacinação de forma dividida para entender onde as políticas públicas têm sido mais eficazes e onde é preciso intensificar os esforços. O ano de 2020 foi um marco, quando as taxas de vacinação caíram drasticamente, especialmente no Norte, que passou de 79,59% em 2019 para 65,69%, e no Sudeste, que desceu de 84,54% para 78,28% no mesmo período, refletindo os impactos da pandemia.

Apesar da queda brusca, a partir de 2020, algumas regiões começaram a mostrar sinais de recuperação. No Norte, por exemplo, a taxa subiu para 71,24% em 2022, mas ainda ficou distante da

meta de 95%[4] estabelecida pelo país. Embora a recuperação seja tímida, ela aponta a importância de continuar os esforços para restabelecer as metas de cobertura vacinal.

Essa análise permite que gestores de saúde identifiquem regiões mais vulneráveis e implementem estratégias para aumentar a adesão à vacinação. O gráfico de linhas unificado facilita essa compreensão, permitindo a formulação de políticas mais direcionadas.

[4] CONFEDERAÇÃO NACIONAL DE MUNICÍPIOS. *Avaliação das coberturas vacinais de rotina no Brasil, de 2009 a 2023, das crianças com até cinco anos de idade*. Brasília, DF: CNM, 2024. Disponível em: https://cnm.org.br/storage/biblioteca/2024/Estudos_tecnicos/202407_ET_SAU_Avaliacao_coberturas_vacinais_municipios_criancas_cinco_anos.pdf. Acesso em:15/02/2025.

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

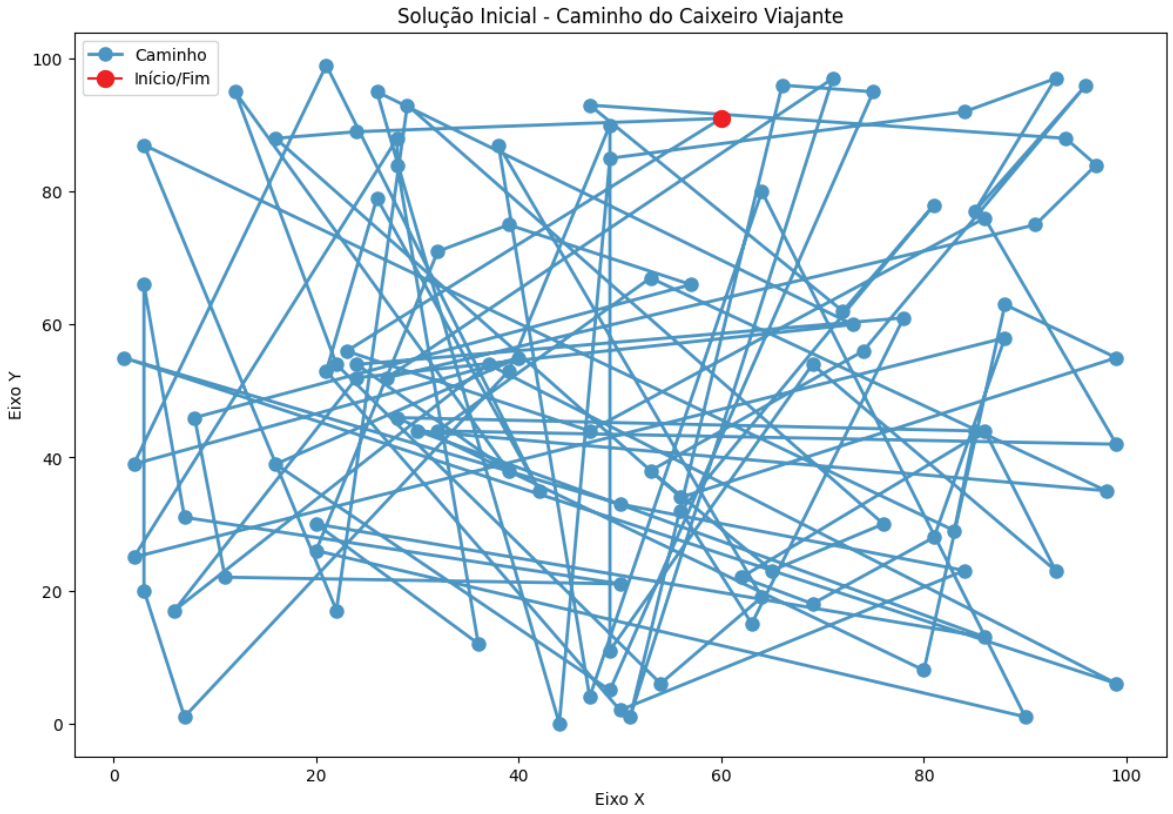
1.

```
# Bibliotecas
import matplotlib.pyplot as plt
import numpy as np
import random

# constantes

# número de cidades
QUANTIDADE_CIDADES = 100

# número de indivíduos
QUANTIDADE_INDIVIDUOS = 100

# referência para dimensão plano cartesiano
TAMANHO_ESPACO = 100

# taxa mutação
TAXA_MUTACAO = 0.01

# taxa cruzamento
TAXA_CRUZAMENTO = 0.9

# número de gerações
QUANTIDADE_GERACOES = 1000

# função para gerar cidades aleatórias
def gerar_cidades():
  cidades = []
  for _ in range(QUANTIDADE_CIDADES):
      x_coord, y_coord =
random.randint(0,TAMANHO_ESPACO),random.randint(0,TAMANHO_ESPACO)
```

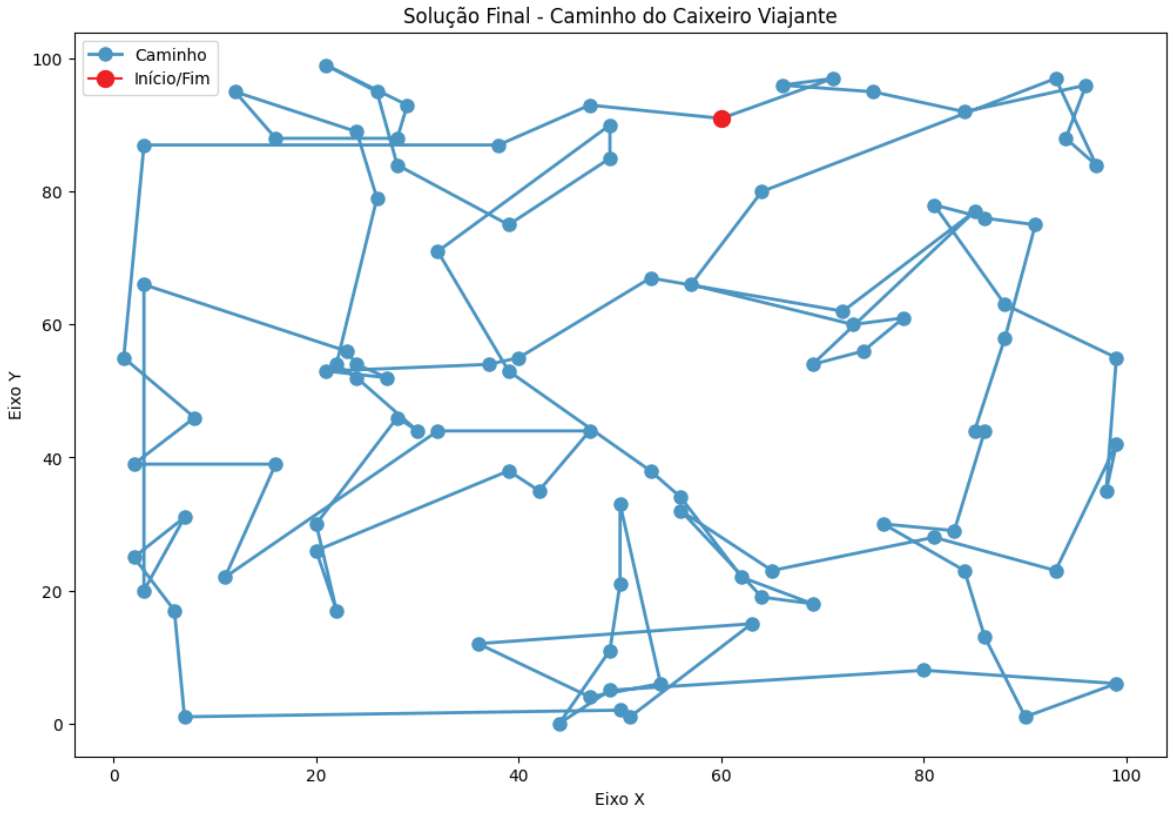
```
      cidades.append((x_coord, y_coord))
  # adiciona cidade inicial ao fim
  cidades.append(cidades[0])
  return cidades

# função para gerar população inicial
def gerar_populacao_inicial(cidades):
      indices = list(range(len(cidades)))
      populacao = []

      for _ in range(QUANTIDADE_INDIVIDUOS):
      # gera individuo com caminho aleatório
      individuo = random.sample(indices[1:QUANTIDADE_CIDADES],
k=QUANTIDADE_CIDADES-1)
      # insere cidade inicial/final no individuo
      individuo.insert(0, 0)
      individuo.append(100)
      populacao.append(individuo)

      return populacao

# função para calcular a distância euclidiana entre as cidades
# d = √((x2 - x1)² + (y2 - y1)²
def calc_distancia_euclidiana(x1, y1, x2, y2) -> float:
  distancia = ((x2 - x1) **2 + (y2 - y1) ** 2) ** 0.5
  return distancia

# função de avaliação
def calc_custo(caminho, cidades):
      custo = 0

      for i in range(len(caminho) - 1):
      x1, y1 = cidades[caminho[i]]
      x2, y2 = cidades[caminho[i+1]]
      custo += calc_distancia_euclidiana(x1, y1, x2, y2)

      return custo

# função para realizar seleção por torneio
def selecionar_pais_torneio(populacao, cidades, k= 5):
  escolhidos = random.sample(populacao, k)
  escolhidos.sort(key=lambda x: calc_custo(x, cidades))
  return escolhidos[0], escolhidos[1]

# função para realizar mutação
def mutar(individuo):
  if random.random() < TAXA_MUTACAO:
      i, j = sorted(random.sample(range(1,len(individuo)-1),2))
      individuo[i:j+1] = reversed(individuo[i:j+1])
  return individuo

# função para cruzamento crossover-ox
```

```
def cruzamento(pai1,pai2):
  tamanho = len(pai1)
  comeco, fim = sorted(random.sample(range(1, tamanho -1),2))
  filho = [-1] * tamanho

  filho[comeco:fim] = pai1[comeco:fim]
  genes_pai1 = set(filho[comeco:fim])
  restantes = [gene for gene in pai2 if gene not in genes_pai1]

  indice = 0
  for i in range(tamanho):
      if filho[i] == -1 and indice < len(restantes):
      filho[i] = restantes[indice]
      indice += 1

  filho[0] = pai1[0]
  filho[-1] = pai1[-1]

  return filho

def calcular_metricas_populacao(populacao, cidades):
  solucao = min(populacao, key=lambda x: calc_custo(x, cidades))
  custo_solucao = calc_custo(solucao,cidades)

  return solucao, custo_solucao

def plotar_caminho(cidades, caminho, titulo):
      plt.figure(figsize=(12, 8))
      # Extrai as coordenadas do caminho
      caminho_coords = [cidades[i] for i in caminho]
      caminho_x, caminho_y = zip(*caminho_coords)

      # Plota o caminho
      plt.plot(caminho_x, caminho_y, linestyle='-',
marker='o',color='#4c95c2', linewidth=2, markersize=8, label="Caminho")
      plt.plot(caminho_x[0], caminho_y[0], marker='o', color='red',
markersize=10, label="Início/Fim")

      plt.xlabel("Eixo X")
      plt.ylabel("Eixo Y")
      plt.title(titulo)
      plt.legend()

      plt.show()

# função principal
def algoritmo_genetico(cidades):
  populacao = gerar_populacao_inicial(cidades=cidades)

  melhor_solucao, melhor_custo =
calcular_metricas_populacao(populacao=populacao, cidades=cidades)
```

```
  solucao_inicial = melhor_solucao.copy()
  custo_inicial = float(melhor_custo)
  # Plotando gráfico da solução inicial
  print(f'Custo inicial da solução {custo_inicial}')
  plotar_caminho(cidades, melhor_solucao, "Solução Inicial - Caminho do
Caixeiro Viajante")

  for geracao in range(QUANTIDADE_GERACOES):
      nova_populacao = [melhor_solucao]
      melhores = sorted(populacao, key=lambda x: calc_custo(x, cidades))
      nova_populacao.extend(melhores[:])

      quantidade_individuo_cruzamento = int(QUANTIDADE_INDIVIDUOS *
TAXA_CRUZAMENTO)

      while len(nova_populacao) < quantidade_individuo_cruzamento:
      pai1, pai2 = selecionar_pais_torneio(populacao,cidades)
      filho = cruzamento(pai1,pai2)
      filho = mutar(filho)
      nova_populacao.append(filho)

      while len(nova_populacao) < QUANTIDADE_INDIVIDUOS:
      individuo = random.choice(populacao)
      filho = mutar(individuo)
      nova_populacao.append(individuo)

      populacao = nova_populacao.copy()
      melhor_solucao_atual, melhor_custo_atual =
calcular_metricas_populacao(populacao,cidades)

      if melhor_custo_atual < melhor_custo:
      melhor_solucao, melhor_custo = melhor_solucao_atual,
melhor_custo_atual

  return melhor_solucao, melhor_custo

# algoritmo genético
cidades = gerar_cidades()
melhor_solucao_encontrada, melhor_custo_encontrado =
algoritmo_genetico(cidades)

# Plotando solução inicial
Custo inicial da solução 4765.895726084021
```




```
# Plotando a solução final
plotar_caminho(cidades, melhor_solucao_encontrada, "Solução Final - Caminho do Caixeiro Viajante")
```

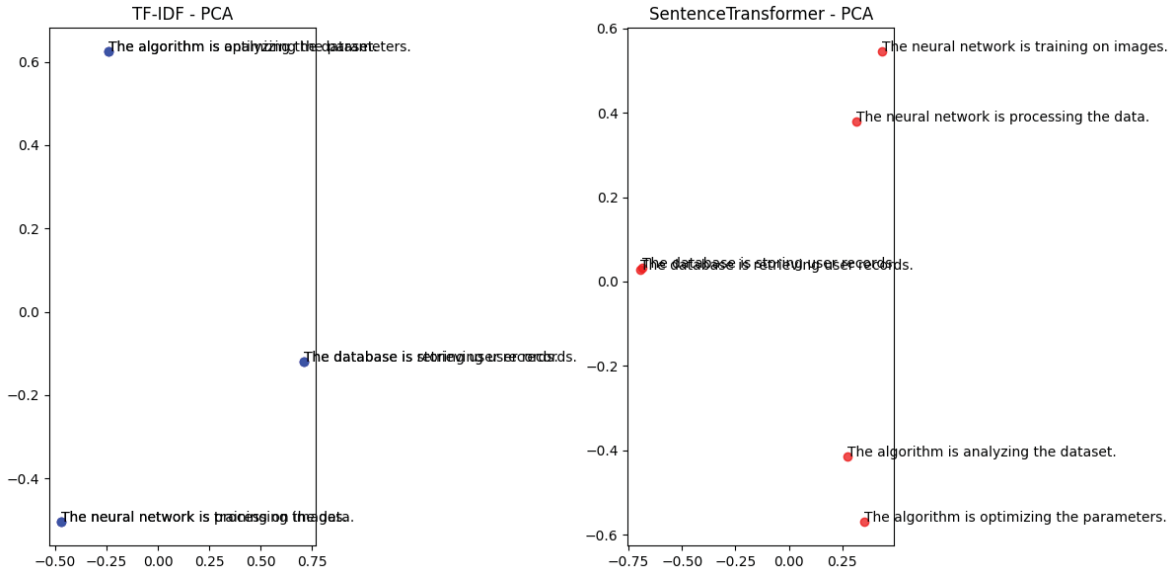


```
print(f'Custo final da solução {melhor_custo_encontrado}')
Custo final da solução 1375.535553111489
```

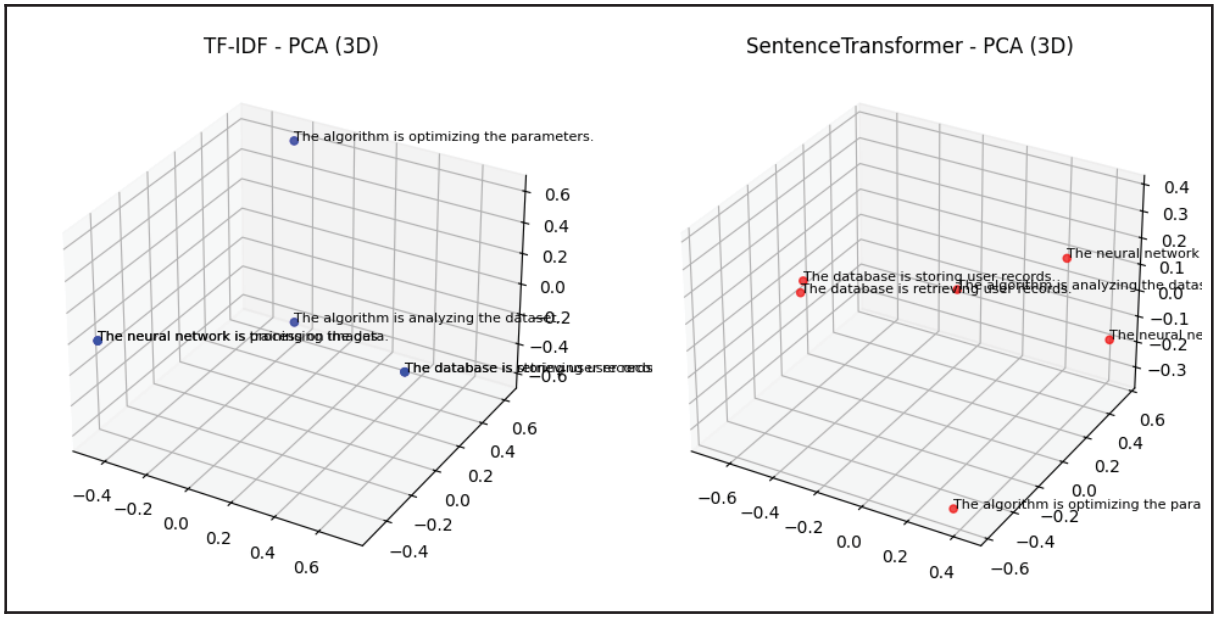
2.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer
from sentence_transformers import SentenceTransformer
from mpl_toolkits.mplot3d import Axes3D

texts = [
      "The algorithm is analyzing the dataset.",
      "The algorithm is optimizing the parameters.",
      "The neural network is processing the data.",
      "The neural network is training on images.",
      "The database is storing user records.",
      "The database is retrieving user records."
]

# PCA
def apply_pca(vectors, n_components = 3):
      pca = PCA(n_components)
      return pca.fit_transform(vectors)

# TF-IDF
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_vectors = tfidf_vectorizer.fit_transform(texts).toarray()

# SentenceTransformer
model = SentenceTransformer("all-MiniLM-L6-v2")
embeddings = model.encode(texts)

# Aplicando PCA
to_pca_tfidf = apply_pca(tfidf_vectors, n_components=2)
to_pca_bert = apply_pca(embeddings,n_components=2)

# Plotando 2D
plt.figure(figsize=(12, 6))

# TF-IDF
plt.subplot(1, 2, 1)
plt.scatter(to_pca_tfidf[:, 0], to_pca_tfidf[:, 1], color='blue',
label='TF-IDF',alpha=0.7)
for i, txt in enumerate(texts):
      plt.annotate(txt, (to_pca_tfidf[i, 0], to_pca_tfidf[i, 1]))
plt.title('TF-IDF - PCA')

# SentenceTransformer
plt.subplot(1, 2, 2)
```

```
plt.scatter(to_pca_bert[:, 0], to_pca_bert[:, 1], color='red',
label='SentenceTransformer',alpha=0.7)
for i, txt in enumerate(texts):
      plt.annotate(txt, (to_pca_bert[i, 0], to_pca_bert[i, 1]))
plt.title('SentenceTransformer - PCA')

plt.tight_layout()
plt.show()
```




```
# Plotando 3D
# Aplicando PCA
to_pca_tfidf_3d = apply_pca(tfidf_vectors)
to_pca_bert_3d = apply_pca(embeddings)

fig = plt.figure(figsize=(12, 6))

# TF-IDF
ax1 = fig.add_subplot(121, projection='3d')
ax1.scatter(to_pca_tfidf_3d[:, 0], to_pca_tfidf_3d[:, 1],
to_pca_tfidf_3d[:, 2], color='blue', alpha=0.7)
for i, txt in enumerate(texts):
      ax1.text(to_pca_tfidf_3d[i, 0], to_pca_tfidf_3d[i, 1],
to_pca_tfidf_3d[i, 2], txt, fontsize=8)
ax1.set_title('TF-IDF - PCA (3D)')

# SentenceTransformer
ax2 = fig.add_subplot(122, projection='3d')
ax2.scatter(to_pca_bert_3d[:, 0], to_pca_bert_3d[:, 1], to_pca_bert_3d[:,
2], color='red', alpha=0.7)
for i, txt in enumerate(texts):
      ax2.text(to_pca_bert_3d[i, 0], to_pca_bert_3d[i, 1],
to_pca_bert_3d[i, 2], txt, fontsize=8)
ax2.set_title('SentenceTransformer - PCA (3D)')

plt.show()
```


TF-IDF - PCA (3D)
The algorithm is optimizing the parameters.
The algorithm is analyzing the dataset.
The neural network is processing the data.
The database is storing user records
SentenceTransformer - PCA (3D)
The neural network
The database is storing user records
The database is retrieving user records
The algorithm is optimizing the para