

UNIVERSIDADE FEDERAL DO PARANÁ

CASSIANO KRUCHELSKI VIDAL

MEMORIAL DE PROJETOS: PROPOSTA DE APLICAÇÃO DE *CHATBOT*
UTILIZANDO IA GENERATIVA, *PROMPTS* DINÂMICOS E RAG

CURITIBA

2025

CASSIANO KRUCHELSKI VIDAL

MEMORIAL DE PROJETOS: PROPOSTA DE APLICAÇÃO DE *CHATBOT*
UTILIZANDO IA GENERATIVA, *PROMPTS* DINÂMICOS E RAG

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

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2025



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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **CASSIANO KRUCHELSKI VIDAL**, intitulada: **MEMORIAL DE PROJETOS: PROPOSTA DE APLICAÇÃO DE *CHATBOT* UTILIZANDO IA GENERATIVA, *PROMPTS* DINÂMICOS E RAG**, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua aprovação no rito de defesa.

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

Curitiba, 04 de Julho de 2025.

RAFAELA MANTOVANI FONTANA

Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO

Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

A utilização de IA generativa em aplicações de *chatbot* tem um grande potencial de trazer uma experiência de usuário muito mais elevada do ponto de vista de fluidez da conversa ao comparar com *chatbots* mais tradicionais que normalmente se atêm a um tipo de *script* rígido com respostas padronizadas. Ao utilizar um modelo de LLM, como por exemplo o GPT-4o-mini da *OpenAI*, é possível simular um tipo de conversação mais próximo da humana o que abre as possibilidades de aplicação desses *bots* em diversas áreas como suporte técnico e vendas de produtos. A combinação de uma LLM com um *chatbot* apresenta diversos desafios como ter um modelo treinado e com o devido conhecimento do negócio para que consiga, de fato, atingir o objetivo proposto que, de forma genérica, pode se resumir em atender à pessoa usuária que está o utilizando. Existem vários processos que podem ser aplicados para se ter um modelo devidamente apto para atendimento ao público, sendo que os principais são: treinamento (*fine tuning*), *prompt engineering* e *Retrieval-Augmented Generation*. Esse trabalho apresenta uma proposta de implementação de uma solução que combina alguns modelos (aqui chamados IAs auxiliares) com uma IA principal, aplicando *dynamic prompting* e RAG para geração de respostas mais precisas e contextualizadas para cada caso específico de uma pessoa usuária do *chatbot*.

Palavras-chave: LLM, IA generativa, *chatbot*, *prompt engineering*, RAG

ABSTRACT

The use of generative AI in chatbot applications has great potential to provide a much better user experience in terms of conversation fluidity when compared to more traditional chatbots that typically stick to a rigid script with standardized responses. By using an LLM model, such as OpenAI's GPT-4o-mini, it is possible to simulate a type of conversation that is closer to human, which opens up the possibilities of applying these bots in various areas such as technical support and product sales. The combination of an LLM with a chatbot presents several challenges, such as having a trained model with the necessary knowledge of the business so that it can, in fact, achieve the proposed objective, which, in general terms, can be summarized as serving the user who is using it. There are several processes that can be applied to have a model that is properly suited to customer service, the main ones being: training (fine tuning), prompt engineering and Retrieval-Augmented Generation. This work presents a proposal for implementing a solution that combines some models (here called auxiliary AIs) with a main AI, applying dynamic prompting and RAG to generate more precise and contextualized responses for each specific case of a person using the chatbot.

Keywords: LLM, generative AI, chatbot, prompt engineering, RAG

SUMÁRIO

1 PARECER TÉCNICO	6
1.1 APLICAÇÃO DE <i>CHATBOTS</i>	7
1.2 <i>PROMPTS</i> SIMPLES.....	8
1.3 <i>PROMPTS</i> DINÂMICOS E <i>RAG</i>	9
REFERÊNCIAS	12
APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL	13
APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA	20
APÊNDICE 3 – LINGUAGEM R.....	23
APÊNDICE 4 – ESTATÍSTICA APLICADA I	27
APÊNDICE 5 – ESTATÍSTICA APLICADA II	35
APÊNDICE 6 – ARQUITETURA DE DADOS	45
APÊNDICE 7 – APRENDIZADO DE MÁQUINA.....	47
APÊNDICE 8 – DEEP LEARNING	55
APÊNDICE 9 – BIG DATA	88
APÊNDICE 10 – VISÃO COMPUTACIONAL	91
APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	96
APÊNDICE 12 – GESTÃO DE PROJETOS DE IA	101
APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL	104
APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING	136
APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL	145

1 PARECER TÉCNICO

O conceito de *chatbot* – programas feitos para conversação de forma autônoma - não é algo recente. Embora seja algo que se tornou muito popular e amplamente utilizado na última década, o primeiro programa de *chatbot* desenvolvido que se tem conhecimento foi *Eliza* de 1966. “*Eliza*, desenvolvido por Joseph Weizenbaum no MIT em 1966, é considerado um dos primeiros *chatbots* criados para o público.” (Al-Amin et al., 2024, p.6)

Através de uma ótica bastante superficial, a ideia desse tipo de aplicação é simples: Uma pessoa manda uma mensagem para uma máquina, interagindo através de algum tipo de interface (como uma mensagem de texto digitada através de um teclado), e a máquina responde. Porém, a implementação desse tipo de sistema é bastante desafiadora, especialmente quando se considera o quão semelhante a um humano a máquina deve se parecer, ou, em outras palavras, o quão natural é a comunicação.

Os primeiros *chatbots* (como *Eliza*) utilizavam um sistema de *script* programado para realizar essa interação com a pessoa: “Verifica-se na entrada de texto a presença de palavras-chave. Quando tal tipo de palavra é encontrada a sentença é então transformada de acordo com uma regra associada com tal palavra.” (Weizenbaum, 1966, p.37).

Chatbots mais modernos, como a *Alexa* (lançada pela *Amazon* em 2014) que possui um sistema de conversação por voz e mais **natural**, utiliza técnicas avançadas como *Natural Language Processing* – NLP (Processamento Linguagem Natural), aprendizado de máquina e algoritmos de *Text to Speech* (Texto para voz) capazes de adicionar diferentes entonações e acentuações visando se aproximar da forma humana de conversar. De acordo com (Metallinou, 2020):

A arquitetura conversacional da *Alexa* aproveita componentes de simulação de diálogo e modelagem de conversação para oferecer suporte a padrões de conversas que incluem transferência de entidades, anáfora, confirmação de slots e APIs [...] os modelos levam em consideração a declaração atual da pessoa e o contexto de todo o histórico de conversas para prever as próximas ações ideais para *Alexa*.

Um dos exemplos mais recentes, e que foi um grande marco dentro do tema de *chatbots* foi o *ChatGPT*, apresentado pela *OpenAI* em 2021, ao implementar o uso de IA generativa; “Recentemente IA cresceu a ponto de influenciar diversos aspectos

da vida, com o ChatGPT da OpenAI se destacando como uma das ferramentas de maior impacto, com mais de 200 milhões de usuários semanais no mundo todo” (Hussein et al., 2025, p.1). O *ChatGPT* é uma interface de conversação construída em cima de um *Large Language Model – LLM* - (Modelo de Linguagem Grande) que foi previamente treinado a partir de uma gigantesca base de dados, possibilitando que seja possível gerar respostas muito similares à de um humano tendo em vista a entrada de texto da pessoa considerando todo o contexto ao gerar uma resposta que faça sentido.

O foco desse parecer técnico gira em torno de Inteligências Artificiais (IAs) generativas, apresentando uma proposta de aplicação de *chatbot* que utiliza uma combinação de modelos desenvolvidos e treinados pela *OpenAI* (como o *GPT-4o* e *GPT-4o-mini*), através de uma técnica de *prompts* dinâmicos “[...] prompts contextualmente dinâmicos melhoram a geração de resposta quando comparados com prompts fixos.” (Swamy et al., 2023, p.1), e da utilização de *Retrieval-Augmented Generation – RAG* (Geração Aumentada via Recuperação) “Exploramos modelos de RAG que utilizam a sequência de entrada para recuperar documentos em texto para serem utilizados como um contexto adicional ao gerar uma resposta” (Lewis et al., 2021, p.2), visando ter um produto capaz de se adaptar a diversos contextos de forma simples e rápida.

1.1 APLICAÇÃO DE *CHATBOTS*

Chatbots podem ser utilizados em diversos segmentos, como suporte para usuários e assistentes virtuais. Para fins didáticos, neste parecer será tratado de um *chatbot* aplicado a um negócio de venda de planos de consulta de crédito de uma empresa, na qual o *chatbot* deverá ser capaz de interagir com uma pessoa para tirar dúvidas e conduzi-la para a contratação de um dos possíveis planos. Nessa proposta de aplicação o *chatbot* será agnóstico à interface, ou seja, a plataforma a qual ele se conecta (*Whatsapp*, interface web ou *Telegram*, por exemplo) é indiferente e, portanto, não é o foco desse parecer.

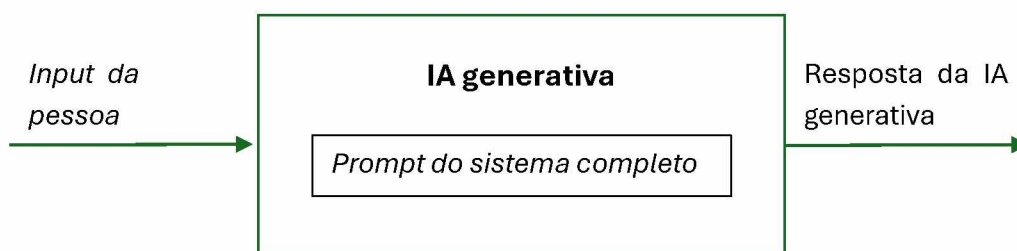
1.2 PROMPTS SIMPLES

Pode-se definir um *prompt* como um input discreto que pode ser utilizado pela pessoa usuária do *chatbot* como um guia de como as respostas devem ser geradas pelo sistema (por exemplo) e também como um input utilizado pela pessoa que desenvolve o *chatbot* para guiar o funcionamento geral do sistema. Através do *prompt* é possível descrever o objetivo do *chatbot* bem como outras informações, como restrições e a forma como deve se comunicar. "[...] *prompts* permitem integração perfeita entre modelos pré-treinados em tarefas subsequentes eliciando comportamentos desejados no modelo. *Prompts* podem ser linguagem natural que fornece context e guia o modelo." (Sahoo et al., 2025, p.1). Um exemplo simples desses *prompts* poderia ser o seguinte:

- *Prompt* do sistema: "Você é um vendedor para a empresa XYZ e seu papel é apresentar, explicar e vender planos de consulta de crédito. Você não deve responder perguntas que não estejam relacionadas à venda de planos. Utilize uma linguagem informal e responda em poucas palavras"
- *Prompt* da pessoa usuária: "Eu gostaria de saber sobre os planos"

Este é um exemplo bastante simples, mas que tem alguns elementos estruturantes de *prompts*: há uma descrição do papel e objetivo do *chatbot*, há uma instrução de restrição de assuntos que ele pode tratar e por fim uma instrução de como devem ser as respostas. Na figura é possível ver uma representação gráfica desse sistema de *chatbot*.

FIGURA 1 – REPRESENTAÇÃO DE APP DE CHATBOT COM PROMPT SIMPLES



FONTE: O autor (2025)

Esse sistema pode se tornar problemático conforme a complexidade do *chatbot* aumenta (como por exemplo adição de regras de negócio), pois *prompts* muito

extensos, com muitas regras podem interferir negativamente na geração da resposta “Em todos os experimentos realizados, a performance do modelo consistentemente se degrada conforme o aumento do comprimento do *input*.” (Hong et al., 2025). Em situações como a utilizada como exemplo nesse parecer (um *chatbot* que deve explicar e vender planos de consultas de crédito para uma empresa) pode ser uma boa solução a utilização de *prompts* dinâmicos e RAG.

1.3 PROMPTS DINÂMICOS E RAG

A ideia da utilização de *prompts* dinâmicos é separar todo o trabalho que apenas uma IA faria em diversos módulos de com responsabilidades menores e bem delimitadas bem como na orquestração das relações entre esses módulos, ou seja, na forma como o *prompt* da pessoa usuária passa pelos diversos módulos até chegar a uma resposta final. Dessa forma o *chatbot* que utiliza apenas uma IA generativa é transformado em um *chatbot* que utiliza um conjunto de várias IAs generativas.

Uma característica interessante de *prompts* dinâmicos é a possibilidade do *chatbot* ter mais noção de contextos, visto que os *prompts* podem ser construídos com base em informações novas que são descobertas ao longo da conversa, ou então se adaptar a situações diversas da pessoa usuária. "Busca-se que os prompts capturem e codifiquem diferentes sinais para cada momento do diálogo conforme o contexto [...] Assim espera-se extrair sinais úteis para serem utilizados com o modelo gerando uma resposta mais adequada " (Swamy et al., 2023, p.2).

Tendo como base o cenário hipotético descrito nesse parecer, uma proposta de aplicação de *prompts* dinâmicos seria a separação do trabalho de uma única IA nos seguintes módulos menores operados por IAs generativas:

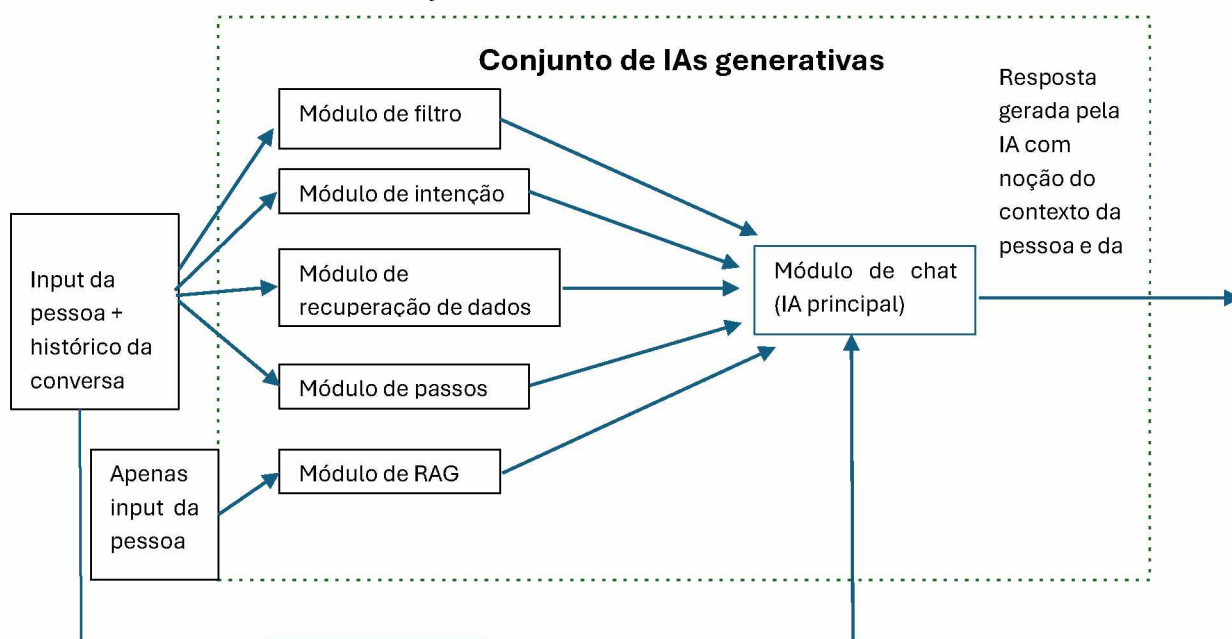
- Módulo de filtro: Contém como *prompt* uma série de restrições de assuntos que podem ser conversados. A sua entrada é o histórico de conversas junto com a pergunta da pessoa usuária e a saída é uma informação se a pergunta da pessoa está dentro das restrições;
- Módulo de intenção: Contém como *prompt* as possíveis intenções de uma pessoa (como dúvidas de acesso, problemas com pagamento ou interesse em comprar) e as regras para que a IA defina a intenção da

pessoa. A entrada é o histórico de conversas e a pergunta da pessoa. A saída é a intenção da pessoa;

- Módulo de recuperação de dados: Contém um *prompt* explicando os dados de interesse que a pessoa pode passar e o significado de cada um. A entrada é o histórico de conversas e a pergunta da pessoa. A saída são novos dados extraídos.
- Módulo de passos da conversa: Contém um *prompt* com os possíveis passos da conversa (exemplos: apresentação, tirar dúvidas, coletar dados e finalizar) e como cada passo é identificado. A entrada é o histórico da conversa mais a pergunta da pessoa. A saída é o passo da conversa em que a pessoa está, juntamente com um "micro objetivo" daquele passo;
- Módulo de RAG: Módulo que visa buscar em um banco de dados vetorial informações adicionais que sejam relevantes à pergunta da pessoa. Tem como entrada a pergunta que a pessoa fez e como saída uma série de afirmações relevantes à pergunta feita; e
- Módulo de conversa: Esse é o principal que tem como *prompt* o objetivo principal do *chatbot*. Recebe como entrada o histórico da conversa, a pergunta da pessoa juntamente com a saída dos outros módulos para fornecer a maior quantidade de informações contextuais para que a IA principal possa gerar a melhor resposta possível.

Utilizando esse sistema é possível escrever *prompts* de sistema pequenos para cada um dos módulos, fazendo com que as IAs generativas de cada módulo tenham uma funcionalidade pequena e simples, reduzindo a chance de alucinação. Por fim, a IA principal recebe uma série de informações que foram construídas conforme o contexto em que a pessoa se encontra na interação com o *chatbot*. Na figura é possível ver uma representação gráfica desse sistema.

FIGURA 2 - REPRESENTAÇÃO DO APP DE CHATBOT COM PROMPTS DINÂMICOS



FONTE: O autor (2025)

É interessante ressaltar que nessa solução é possível utilizar modelos de IA generativas menores e de baixo custo para realizar as tarefas de cada módulo, visto que são processamentos mais simples e diretos. Por fim vale lembrar que a gestão dos módulos e do histórico varia conforme a implementação da aplicação, sendo que existem diferentes formas de implementar e que não entram no escopo desse parecer técnico.

REFERÊNCIAS

AL-AMIN, M.; ALI, M.; SALAM, A.; KHAN, A.; ALI, A.; ULLAH, A.; ALAM, M.; CHOWDHURY, S. History of generative Artificial Intelligence (AI) chatbots: past, present, and future development. **arXiv preprint**, arXiv:2402.05122, 2024. Disponível em: <https://arxiv.org/abs/2402.05122>. Acesso em: 13 jul. 2025.

HONG, K.; TROYNIKOV, A.; HUBER, J. Context rot: how increasing input tokens impacts LLM performance. In: **Chroma Technical Report**. [S. l.], jul. 2025. Não paginado. Disponível em: <https://research.trychroma.com/context-rot>. Acesso em: 15 jul. 2025.

HUSSEIN, H.; GORDON, M.; HODGKINSON, C.; FOREMAN, R.; WAGAD, S. ChatGPT's impact across sectors: a systematic review of key themes and challenges. **Big Data and Cognitive Computing**, Basel, v. 9, n. 3, art. 56, 2025. Disponível em: <https://doi.org/10.3390/bdcc9030056>. Acesso em: 13 jul. 2025.

LEWIS, P.; PEREZ, E.; PIKTUS, A.; PETRONI, F.; KARPUKHIN, V.; GOYAL, N.; KUTLER, H.; LEWIS, M.; YIH, W.; ROCKTASCHEL, T.; RIEDEL, S.; KIELA, D. Retrieval-augmented generation for knowledge-intensive NLP tasks. **arXiv preprint**, arXiv:2005.11401, 2021. Disponível em: <https://arxiv.org/abs/2005.11401>. Acesso em: 13 jul. 2025.

METALLINO, A. Science innovations power Alexa Conversations dialogue management. In: **Amazon Science**. [S. l.], jul. 2020. Não paginado. Disponível em: <https://www.amazon.science/blog/science-innovations-power-alexa-conversations-dialogue-management>. Acesso em: 15 jul. 2025.

SAHOO, P.; SINGH, A.; SAHA, S.; JAIN, V.; MODAL, S.; CHADHA, A. A systematic survey of prompt engineering in large language models: techniques and applications. **arXiv preprint**, arXiv:2402.07927v2, 2025. Disponível em: <https://arxiv.org/abs/2402.07927>. Acesso em: 14 jul. 2025.

SWAMY, S.; TABARI, N.; CHEN, C.; GANGADHARAI, R. Contextual dynamic prompting for response generation in task-oriented dialog systems. **arXiv preprint**, arXiv:2301.13268, 2023. Disponível em: <https://arxiv.org/abs/2301.13268>. Acesso em: 15 jul. 2025.

WEIZENBAUM, J. ELIZA — a computer program for the study of natural language communication between man and machine. **Communications of the ACM**, New York, v. 9, n. 1, p. 36–45, jan. 1966. DOI: <https://doi.org/10.1145/365153.365168>. Disponível em: <https://dl.acm.org/doi/10.1145/365153.365168>. Acesso em: 15 jul. 2025.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

- (6,25 pontos)** Pergunte ao ChatGPT o que é Inteligência Artificial e cole aqui o resultado.
- (6,25 pontos)** Dada essa resposta do ChatGPT, classifique usando as 4 abordagens vistas em sala. Explique o porquê.
- (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.
- (6,25 pontos)** Entendendo o que é o ChatGPT, classifique o próprio ChatGPT usando as 4 abordagens vistas em sala. Explique o porquê.

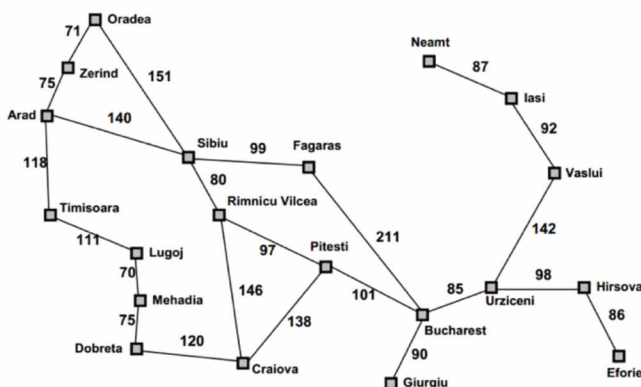
2 Busca Heurística

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.

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

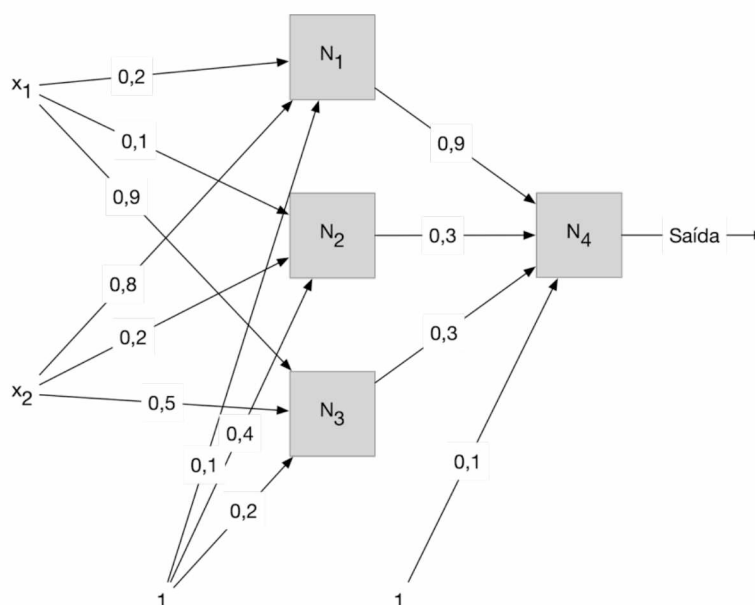
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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:

- (6,25 pontos)** Valor de saída do neurônio N_1
- (6,25 pontos)** Valor de saída do neurônio N_2
- (6,25 pontos)** Valor de saída do neurônio N_3
- (6,25 pontos)** Valor de saída da rede como um todo

B – RESOLUÇÃO

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

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

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

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

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

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

Agir como humanos: Apesar de o ChatGPT falar em uso de inteligência humana nas IAs, levando à abordagem de pensar como humanos *"(...) podem realizar tarefas que normalmente exigiriam inteligência humana"*, para que uma IA pense como humano, todo o processo de pensamento deve ser mapeado: introspecção, experimentos psicológicos, imagens cerebrais e ainda o fator pessoal. Quando

todo esse processo for determinado, então poderemos ter IAs com pensamento humano. Com isso, podemos dizer que as IAs existentes, por mais que consigam realizar tarefas humanas, não pensam como humanos, e sim imitam seu comportamento.

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

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

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

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

Referências:

GUINNESS, Harris. **How does ChatGPT work?**. 2023. Disponível em <<https://zapier.com/blog/how-does-chatgpt-work/>> Acesso em 20 fev. 2024

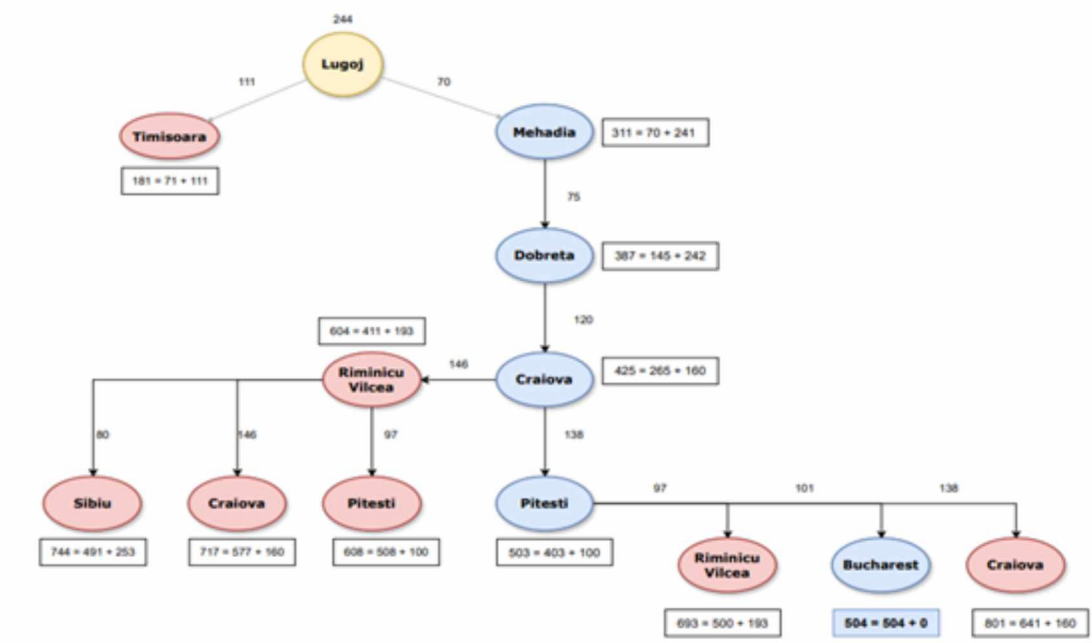
HETLER, Amanda. **Definition: ChatGPT**. 2023. Disponível em <<https://www.techtarget.com/whatis/definition/ChatGPT>> Acesso em 22 fev. 2024

OPENAI. **What is ChatGPT?** 2024. Disponível em <<https://help.openai.com/en/articles/6783457-what-is-chatgpt>> Acesso em 23 fev. 2024

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

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

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



3.a

Legenda:

p : Uvas caem

q : Raposa come as uvas

r : Uvas estão maduras

Base de Conhecimento (BC):

R1: $p \rightarrow q$ (Se as uvas caem, então a raposa as come)

R2: $q \rightarrow r$ (Se a raposa as come, então estão maduras)

R3: $\neg r \vee p$ (As uvas estão verdes ou as uvas caem)

R4: $r \rightarrow p$ COND, R3

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

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

R7: $q \leftrightarrow p$ BICOND, R6

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

$$BC \vdash q \leftrightarrow p$$

4.a

$$\sum x_i + w_i = -3 * 0.2 + 1 * 0.8 + 1 * 0.1 = 0.3$$

$$u = 0, 3$$

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

$$Saída N1 = 0.3$$

4.b

$$\sum x_i + w_i = -3 * 0.1 + 1 * 0.2 + 1 * 0.4 = 0.3$$

$$u = 0, 3$$

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

$$Saída N2 = 0.3$$

4.c

$$\sum x_i + w_i = -3 * 0.9 + 1 * 0.5 + 1 * 0.2 = -2$$

$$u = -2$$

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

$$Saída N3 = -2$$

4.d

$$\sum x_i + w_i = 0.3 * 0.9 + 0.3 * 0.3 + (-2) * 0.3 + 1 * 0.1 = -0.14$$

$$u = -0.14$$

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

$$Saída N4 = -0.1391$$

APÊNDICE 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

- Escolha as variáveis **numéricas** (modelos de Regressão) para serem as variáveis independentes do modelo. A variável target é **avg_price**. **Observação:** caso julgue necessário, faça a transformação de variáveis categóricas em variáveis numéricas para inputar no modelo. Indique **quais variáveis** foram transformadas e **como** foram transformadas
- Crie partições contendo 75% dos dados para treino e 25% para teste
- Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost (biblioteca XGBRegressor) para predição dos preços dos carros. **Observação:** caso julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique quais parâmetros foram inputados e indique o treinamento de cada modelo
- Grave os valores preditos em variáveis criadas
- Realize a análise de importância das variáveis para estimar a variável target, **para cada modelo treinado**
- Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
- Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R^2
- Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

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

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

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

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

3.h - Os três modelos tiveram bom ajuste aos dados, com R^2 sempre superior a 0,9. No entanto, a Random Forest sem parâmetros pré-definidos, apresentou MSE e MAE menores do que os demais modelos, e teve o melhor valor de R^2 , 0,98. Com isso, esse modelo, que teve como variáveis importantes `engine_size` (tamanho do motor) e `year_model` (ano do modelo) foi escolhido como o melhor.

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

Onde dap é o diâmetro na altura do peito (1,3metros), Ht é a altura total. Tem-se vários modelos alométricos, cada um com uma determinada característica, parâmetros, etc. Um modelo de regressão envolve aplicar os dados observados e encontrar b_0 e b_1 no modelo apresentado, gerando assim uma equação que pode ser usada para prever o volume de outras árvores.

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

Tarefas

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

- O modelo alométrico é dado por: $\text{Volume} = b_0 + b_1 * \text{dap}^2 * \text{Ht}$

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

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

- Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 – Na atividade, foram treinados três modelos: Random Forest, SVM (Support Vector Machine) e RNA (Rede Neural Artificial), utilizando os dados de treinamento. Cada modelo foi treinado para prever as classes com base nas características fornecidas pelo conjunto de dados.

Em seguida, avaliamos o desempenho de cada modelo utilizando os dados de teste. Calculamos métricas como precisão, recall e F1-Score. A precisão representa a proporção de exemplos previstos como positivos que são verdadeiramente positivos, o recall é a proporção de exemplos positivos corretamente identificados pelo modelo, e o F1-Score é a média harmônica entre precisão e recall, fornecendo uma medida geral do desempenho do modelo.

Posteriormente, calculamos os valores de F1-Score para cada modelo e identificamos aquele com o maior valor como o melhor modelo para esta tarefa. O modelo SVM obteve o maior F1-Score, seguido pelo modelo Random Forest.

Optamos pelo modelo SVM como o melhor devido ao seu alto F1-Score, indicando um bom equilíbrio entre precisão e recall. O SVM obteve um F1-Score superior ao do Random Forest, o que confirma sua capacidade na classificação das classes com base nos dados de satélite. Assim, considerando as métricas de desempenho, selecionamos o SVM como o modelo mais adequado para esta tarefa.

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

- Coeficiente de determinação (R^2)
- Erro padrão da estimativa (S_{yx})
- Porcentagem do erro padrão da estimativa ($S_{yx}\%$)

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

$$score = \frac{(R^2 + (1 - S_{yx}))}{2}$$

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

Modelo	R^2	S_{yx}	$S_{yx}\%$	Score
RNA	0,8867930	0,1354473	10,06305	0,8930813
Alométrico	0,8694429	0,1454567	10,80670	0,8806879
Random Forest	0,8486654	0,1566040	11,63489	0,8661582
SVM	0,7900779	0,1844433	13,70321	0,8265229

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

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

(15 pontos) Elaborar a tabela de frequências 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:

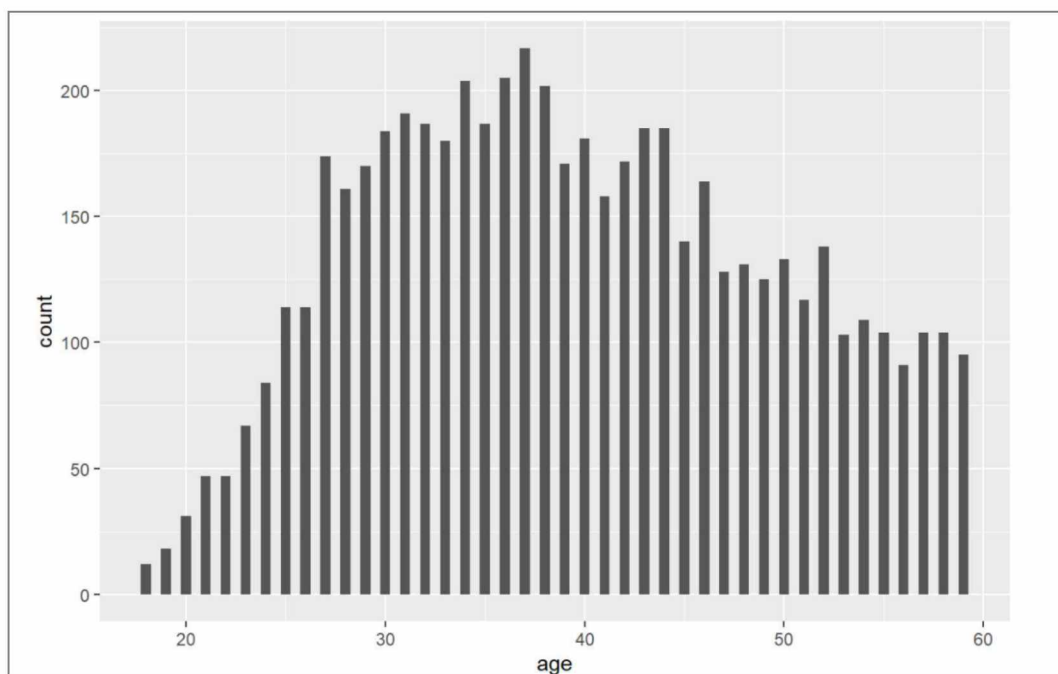
Você deve fazer os testes necessários (e mostra-los no documento pdf) para saber se você deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney (não paramétrico), justifique sua resposta sobre a escolha.

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

B – RESOLUÇÃO

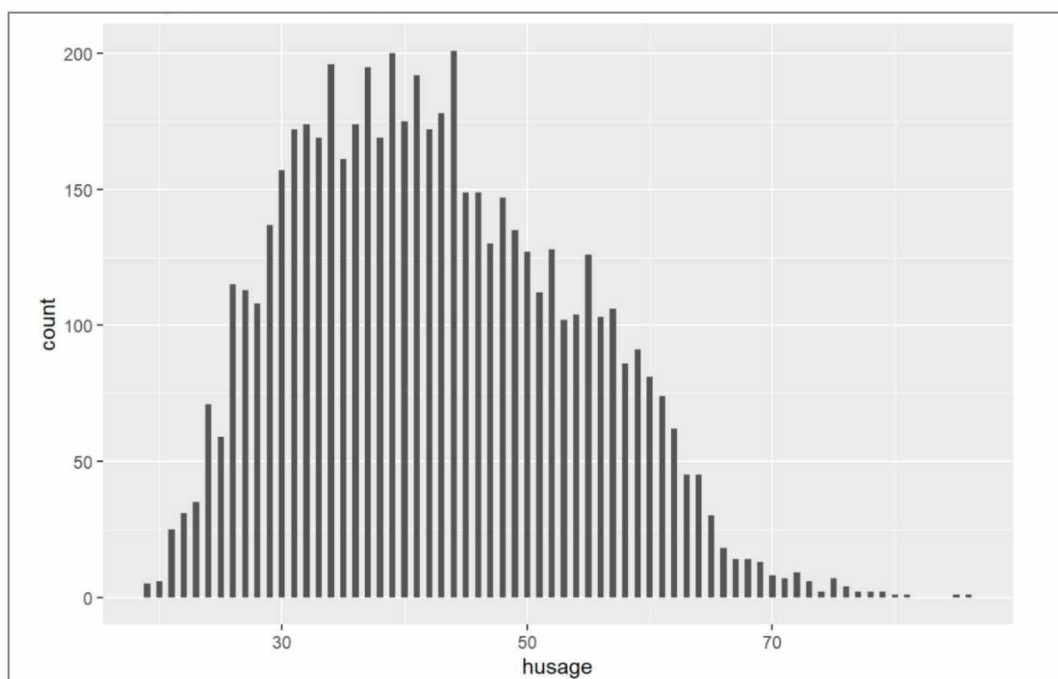
1.a - Gráficos histogramas e box-plot e comparação dos resultados

FIGURA 4.1 – HISTOGRAMA IDADE ESPOSAS



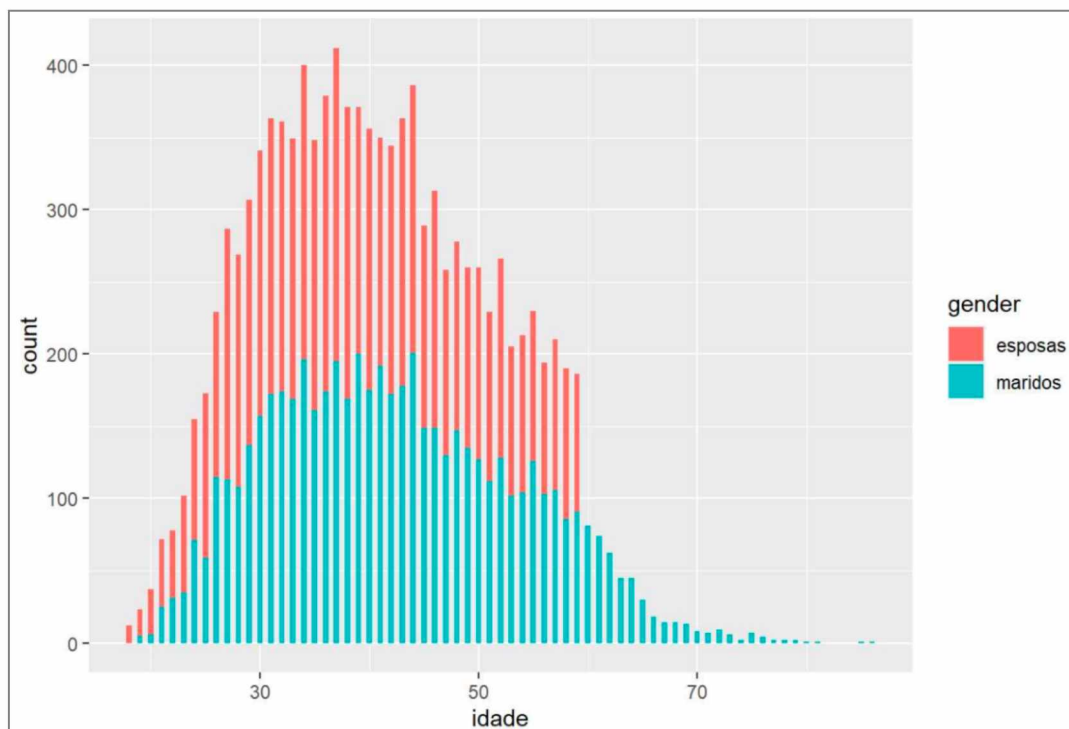
FONTE: O autor (2025)

FIGURA 4.2 – HISTOGRAMA IDADE MARIDOS



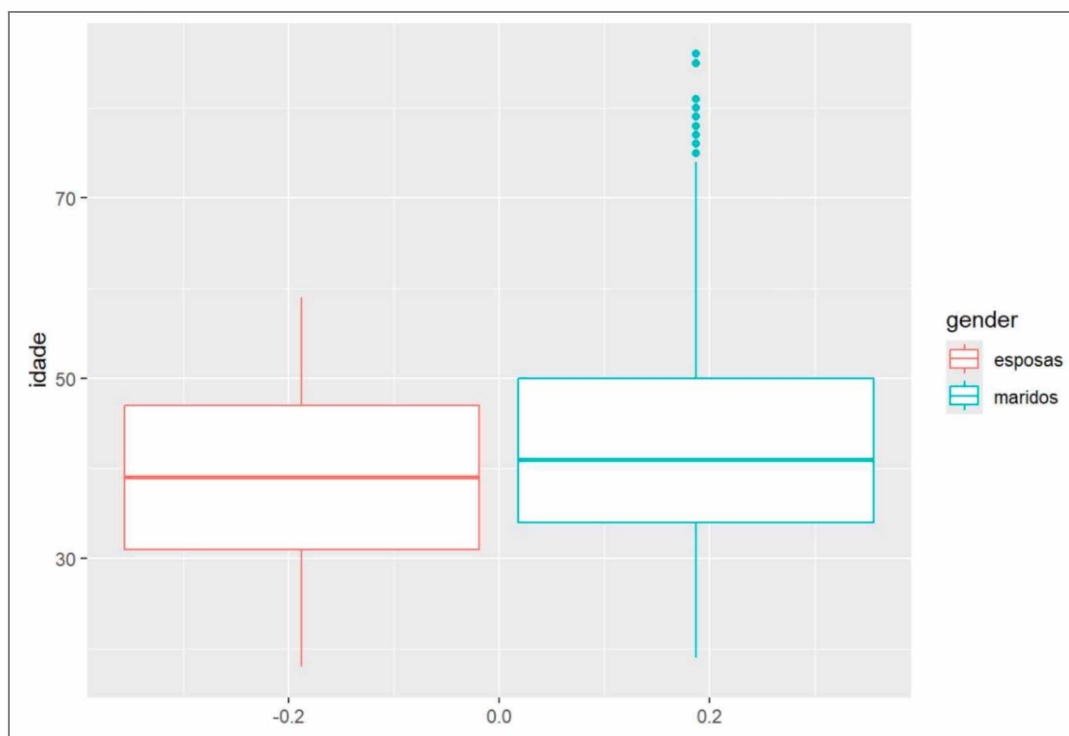
FONTE: O autor (2025)

FIGURA 4.3 – HISTOGRAMA IDADE ESPOSAS X HISTOGRAMA IDADE MARIDOS



FONTE: O autor (2025)

FIGURA 4.4 – BOXPLOT IDADE ESPOSAS X BOXPLOT IDADE MARIDOS



FONTE: O autor (2025)

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

2.a - Tabelas de frequências

Tabela de frequência da variável "age" (idade da esposa)

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

Proporção das frequências da variável "age" (idade da esposa)

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

Tabela de frequência da variável "husage" (idade do marido)

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

Proporção das frequências da variável "husage" (idade do marido)

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

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

2.a - Média, mediana e moda

Variável age

```
summary(dados$age)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    18.00   31.00   39.00   39.43   47.00   59.00

moda <- sort(table(dados$age),decreasing = T)[1]
moda
##    37
##   217
```

Variável husage

```
summary(dados$husage)
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    19.00   34.00   41.00   42.45   50.00   86.00

modah <- sort(table(dados$husage),decreasing = T)[1]
modah
##      44
##    201
```

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

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

2.b - Varância, desvio padrão e coeficiente da variação

Variável age

```
var(dados$age)
## [1] 00.75234

sd(dados$age)
## [1] 9.98761

sd(dados$age)/mean(dados$age)*100
## [1] 25.33153
```

Variável husage

```
var(dados$husage)
## [1] 126.0717

sd(dados$husage)
## [1] 11.22817
```

```
sd(dados$husage)/mean(dados$husage)*100
## [1] 26.44849
```

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

3 – Teste de médias/medianas

- Testando normalidade das variáveis age e husage através do teste de Kolmogorov

```
ks.test(dados$age, "pnorm", mean(dados$age), sd(dados$age))
## Warning in ks.test(dados$age, "pnorm", mean(dados$age), sd(dados$age)):
## default ks.test() cannot compute correct p-values with ties;
## see help page for one-sample Kolmogorov test for discrete distributions
##
## One-sample Kolmogorov-Smirnov test
## data: dados$age
## D = 0.058909, p-value < 2.2e-16
## alternative hypothesis: two-sided

ks.test(dados$husage, "pnorm", mean(dados$husage), sd(dados$husage))
## Warning in ks.test(dados$husage, "pnorm", mean(dados$husage),
## sd(dados$husage)): default ks.test() cannot compute correct p-values with
## ties;
## see help page for one-sample Kolmogorov test for discrete distributions
##
## One-sample Kolmogorov-Smirnov test
## data: dados$husage
## D = 0.059662, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

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

- Teste de Mann Whitney para comparar as medianas

H0: Não existe diferença entre os grupos

H1: Há diferença entre os grupos

```
# usando o teste de mann whitney
teste <- wilcox.test(idade~gender, data = dados_teste, exact=F, conf.int=T)
teste
##
## Wilcoxon rank sum test with continuity correction
##
## data: idade by gender
## W = 13619912, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
## 95 percent confidence interval:
## -3.000024 -2.000033
## sample estimates:
## difference in location
## -2.999966
```

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

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

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

- Modelo Ridge

Cálculo do valor ótimo de lambda

```

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

# Calculando o lambda por validação cruzada:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0, lambda = lambdas)

# Verificando qual o lambda ótimo
best_lambda_ridge <- ridge_lamb$lambda.min
best_lambda_ridge
[1] 0.02511886

```

Estimando o modelo: coeficientes

```

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

ridge_reg[["beta"]]

16 x 1 sparse Matrix of class "dgCMatrix"

husage      0.012320360
husearns    0.074994288
huseduc     0.026346992
hushrs     -0.038692294
earns       0.705154984
age         0.029864158
educ        0.112606137
exper      -0.005222281
husunion    0.010018714
husblck     -0.004937410
hushisp     0.025557764
kidge6      0.046700360
black       -0.035976462

```

```

hispanic    -0.100055093
union       0.136696857
kidlt6      0.139360366

```

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

Predição dos dados de treino

```

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

# As métricas da base de treinamento são:
metricas_ridge_treino <- eval_results(y_train, predictions_train,
dados_treino)
metricas_ridge_treino

      RMSE      Rsquare
      <dbl>      <dbl>
0.5595395  0.6867634

```

Predição dos dados de teste

```

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

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

      RMSE      Rsquare
      <dbl>      <dbl>
0.5375854  0.7007985

```

- Modelo Lasso

Cálculo do valor ótimo de lambda

```

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

```

```
# Verificando qual é o lambda ótimo
best_lambda_lasso <- lasso_lamb$lambda.min
best_lambda_lasso
[1] 0.006309573
```

Estimando o modelo: coeficientes

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

16 x 1 sparse Matrix of class "dgCMatrix"

```
husage      0.0066304451
husearns    0.0692790074
huseduc     0.0210733446
hushrs      -0.0313862324
earnings    0.7233054274
age         0.0201570373
educ        0.1087897978
exper       .
husunion    0.0007791477
husblack    -0.0026596447
hushisp     .
kidge6      0.0288263651
black       -0.0109971655
hispanic    -0.0618425974
union       0.1194218725
kidlt6      0.1103154935
```

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

Predição dos dados de treino

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

```
metricas_lasso_treino
```

RMSE	Rsquare
<dbl>	<dbl>
0.5596622	0.6866261

Predição dos dados de teste

```
# Predições e avaliações na base de teste
```

```
predictions_test_lasso <- predict(lasso_model, s = best_lambda_lasso, newx
= x_test)
```

```
# As métricas da base de teste são:
```

```
metricas_lasso_teste <- eval_results(y_test, predictions_test_lasso,
dados_teste)
```

```
metricas_lasso_teste
```

RMSE	Rsquare
<dbl>	<dbl>
0.5340347	0.7047378

- Modelo Elastic Net

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

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

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

```
elastic_reg$bestTune
```

	alpha	lambda
	<dbl>	<dbl>
9	0.8528671	0.005336688

Predição dos dados de treino

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

# Métricas de performance na base de treinamento
metricas_elastic_treino <- eval_results(y_train, predictions_train_elastic,
dados_treino)
metricas_elastic_treino
```

RMSE	Rsquare
<dbl>	<dbl>
0.5594508	0.686627

Predição dos dados de teste

```
# Predições e avaliações na base de teste
predictions_test_elastic <- predict(elastic_reg, x_test)

# As métricas de performance são:
metricas_elastic_test <- eval_results(y_test, predictions_test_elastic,
dados_teste)

métricas_elastic_teste
```

RMSE	Rsquare
<dbl>	<dbl>
0.5341944	0.7045612

Métricas de qualidade dos modelos

	RMSE	Rsquare
	<dbl>	<dbl>
RIDGE – TREINO	0.5595395	0.6867634
LASSO – TREINO	0.5596622	0.6866261
ELASTICNET – TREINO	0.5594508	0.6868627
RIDGE – TESTE	0.5375854	0.7007985
LASSO – TESTE	0.5340347	0.7047378
ELASTICNET – TESTE	0.5341944	0.7045612

Predição caso proposto

- Ridge

```

predict_our_ridge <- predict(ridge_reg, s = best_lambda_ridge, newx =
our_pred)
predict_our_ridge

[1,] 0.7051145

# 0 resultado é um valor padronizado que deve ser convertido ao valor
nominal
# consistente com o dataset original

wage_pred_ridge=(predict_our_ridge*pre_proc_val[["std"]][["lwage"]]) +
pre_proc_val[["mean"]][["lwage"]]

# 0 resultado é (ainda será aplicado exponencial)
wage_pred_ridge

[1,] 2.56323

```

Intervalo de confiança Ridge

```

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

[1,] 2.540604

CIupr_ridge

[1,] 2.585856
LS_RIDGE <- exp(CIupr_ridge)
LI_RIDGE <- exp(CIlwr_ridge)
ESTIMAD_RIDGE <- exp(wage_pred_ridge)

```

- Lasso

```

predict_our_lasso <- predict(lasso_model, s = best_lambda_lasso, newx =
our_pred)
predict_our_lasso

[1,] 0.7441195

# 0 resultado é um valor padronizado que deve ser convertido ao valor
nominal
# consistente com o dataset original

wage_pred_lasso=(predict_our_lasso*pre_proc_val[["std"]][["lwage"]]) +
pre_proc_val[["mean"]][["lwage"]]

# 0 resultado é (ainda será aplicado exponencial)
wage_pred_lasso

[1,] 2,583662

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

```

Intervalo de confiança Lasso

```

# 0 intervalo de confiança para o exemplo é:
m_lasso <- wage_pred_lasso # valor médio predito

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

[1,] 2.561036

CIupr_lasso

[1,] 2.606288

```



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

- Elastic Net

```
predict_our_elastic <- predict(elastic_reg, our_pred)
predict_our_elastic
```

```
[1] -0.03492759
```

```
# 0 resultado é um valor padronizado que deve ser convertido ao valor nominal
# consistente com o dataset original
```

```
wage_pred_elastic=(predict_our_elastic*pre_proc_val[["std"]][["lwage"]]) +
pre_proc_val[["mean"]][["lwage"]]
```

```
# 0 resultado é (ainda será aplicado exponencial)
```

```
wage_pred_elastic
```

```
[1] 2.175577
```

Intervalo de confiança Elastic Net

```
# 0 intervalo de confiança para o exemplo é:
```

```
m_elastic <- wage_pred_elastic # valor médio predito
```

```
CIlwr_elastic <- m_elastic + (qnorm(0.025))*dam # intervalo inferior
```

```
CIupr_elastic <- m_elastic + (qnorm(0.025))*dam # intervalo superior
```

```
CIlwr_elastic
```

```
[1] 2.152951
```

```
CIupr_elastic
```

```
[1] 2.198203
```

```
LI_ELASTIC <- exp(CIlwr_elastic)
```

```
LS_ELASTIC <- exp(CIupr_elastic)
```

```
ESTIMAD_ELASTIC <- exp(m_elastic)
```

Valores preditos para pessoa x e intervalos de confiança

```
estimacao_ridge<-cbind.data.frame(LI_RIDGE,ESTIMAD_RIDGE,LS_RIDGE)
estimacao_lasso<-cbind.data.frame(LI_LASSO,ESTIMAD_LASSO,LS_LASSO)
estimacao_elastic<-cbind.data.frame(LI_ELASTIC,ESTIMAD_ELASTIC,LS_ELASTIC)

pessoa_x <- rbind(estimacao_ridge, estimacao_lasso, estimacao_elastic)

colnames(pessoa_x) <- c('Limite inferior IC', 'Valor estimado', 'Limite
superior IC')
rownames(pessoa_x) <- c('Ridge', 'Lasso', 'ElasticNet')
```

	RMSE <dbl>	Rsquare <dbl>
Ridge	0.5595395	0.6867634
Lasso	0.5596622	0.6866261
ElasticNet	0.5594508	0.6868627

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

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

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

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1 - Resultados e considerações finais:

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

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

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

2 – Resultados e considerações finais:

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

Utilizamos um modelo SVM para fazer o treinamento e testes na base de dados. A acurácia do modelo de SVM aplicado sobre os dados originais foi de 88%. Com os tratamentos realizados, observou-se melhora de 6% nessa medida de qualidade, chegando à uma acurácia de 93%. Com isso chegamos ao objetivo deste trabalho que era obter de 2% a 5% de acréscimo de acurácia a uma base de dados desbalanceada.

APÊNDICE 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

1.a - Classificação - Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	Reference Prediction bus opel saab van bus 43 0 0 0 opel 0 42 0 0 saab 0 0 43 0 van 0 0 0 39
RNA – Hold-out	size=5 decay=0.1	85% (0.8503)	Reference Prediction bus opel saab van bus 41 0 1 2 opel 0 30 7 1 saab 1 9 35 0 van 1 3 0 36
SVM – Hold-out	C=1 Sigma= 0.07189928	85% (0.8502)	Reference Prediction bus opel saab van bus 43 0 0 1 opel 0 26 7 0 saab 0 12 35 0 van 0 4 1 38
SVM – CV	C=100 Sigma= 0.015	84% (0.8443)	Reference Prediction bus opel saab van bus 40 0 0 1 opel 1 34 12 2 saab 0 8 31 0 van 2 0 0 36

RNA – CV	size=11 decay=0.4	81% (0.8084)	Reference Prediction bus opel saab van bus 39 0 1 1 opel 0 27 9 1 saab 1 11 32 0 van 3 4 1 37
RF – CV	mtry=10	74% (0.7365)	Reference Prediction bus opel saab van bus 42 0 1 0 opel 0 18 13 2 saab 0 21 27 1 van 1 3 2 36
KNN	k=1	68% (0.6766)	Reference Prediction bus opel saab van bus 39 1 1 3 opel 1 17 16 1 saab 1 20 22 0 van 2 4 4 35

Novos casos:

a	Co mp	Circ	DCirc	Rad Ra	PrA xisRa	Max L Ra	Scat Ra	Elon g	PrA xisRect	Max L Rect	ScV arM axis	ScV arm axis	RaG yr	Ske wMa xis	Ske wm axis	Ku rt ma xis	Kurt Maxi s	H ol IR a	tip o
1	75	55	89	105	133	36	117	60	24	140	130	806	204	70	3	28	192	190	va n
2	89	42	56	241	124	7	200	34	28	148	200	450	261	92	1	29	184	208	bu s
3	115	37	107	315	55	48	251	28	20	125	320	210	139	100	9	37	181	201	op el

1.b - Classificação - Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
RF – Hold-out	mtry=2	100% (1)	Reference Prediction neg pos neg 100 0 pos 0 53
SVM – Hold-out	C=0.25 Sigma=0.1258432	78% (0.7843)	Reference Prediction neg pos neg 93 26 pos 7 27

RNA – Hold-out	size=1 decay=0.1	78% (0.7778)	Reference Prediction neg pos neg 90 24 pos 10 29
RF – CV	mtry=2	77% (0.7712)	Reference Prediction neg pos neg 89 24 pos 11 29
SVM – CV	C=2 Sigma= 0.015	76% (0.7582)	Reference Prediction neg pos neg 92 29 pos 8 24
RNA – CV	size=3 decay=0.1	76% (0.7581)	Reference Prediction neg pos neg 88 25 pos 12 28
KNN	k=9	73% (0.7255)	Reference Prediction neg pos neg 84 26 pos 16 27

Novos casos:

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

2.a - Regressão - Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.9458	0.03368	0.97462	0.03334	0.02296
RNA – Hold-out	size=41 decay=0.1	0.8341	0.05895	0.91351	0.058348	0.044075
RNA – CV	size=16 decay=0.1	0.8218	0.06110	0.90938	0.060474	0.048163

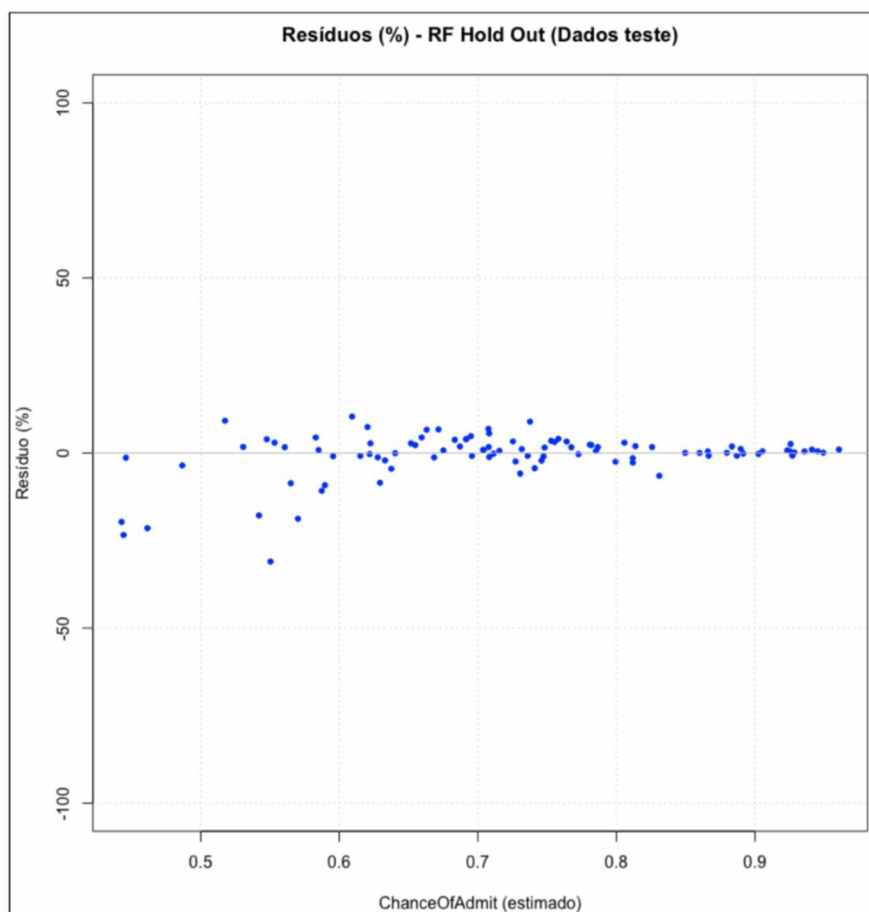
SVM – CV	C=50 Sigma=0.015	0.8209	0.61251	0.90913	0.60622	0.04388
RF – CV	mtry=2	0.8046	0.06398	0.89732	0.06332	0.04513
SVM – Hold out	C=0.5 Sigma= 0.1769097	0.8026	0.06430	0.89797	0.063643	0.045819
KNN	K=9	0.7883	0.06659	0.89068	0.065908	0.04751

Novos casos:

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

Gráfico de resíduos:

FIGURA 7.1 – GRÁFICO DE RESÍDUOS - ADMISSÃO



FONTE: O autor (2025)

2.b - Regressão - Biomassa

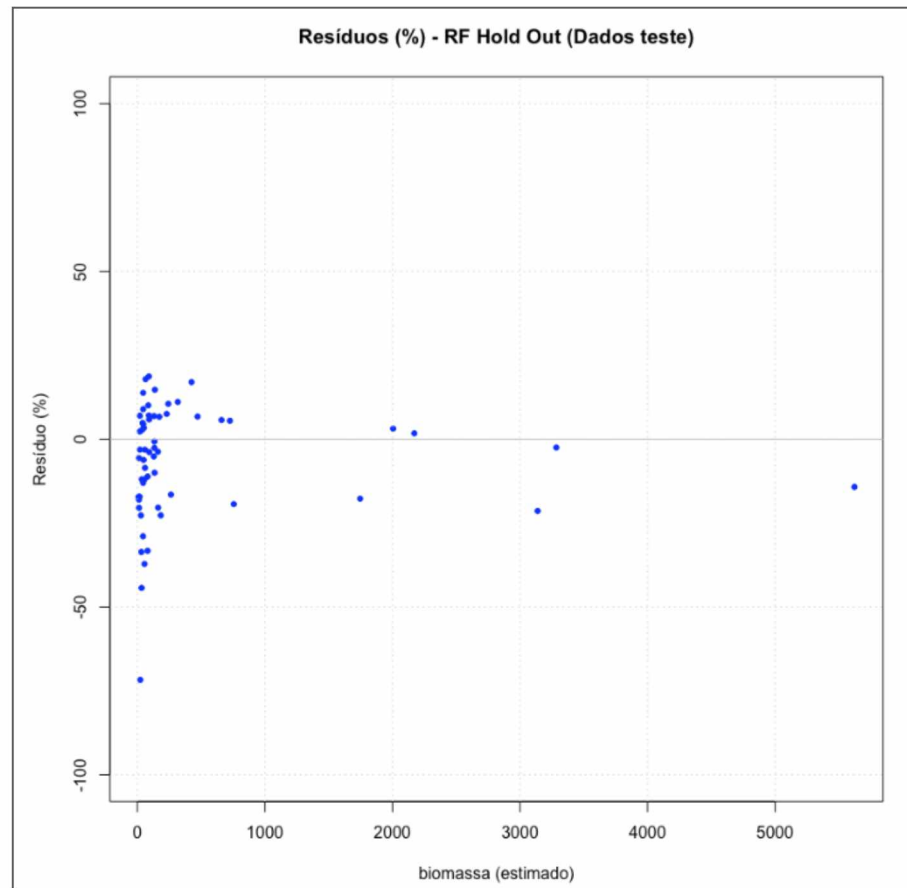
Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry=2	0.98	125.5789	0.9964	123.4682	41.8887
RF – CV	mtry=3	0.9702	153.4874	0.99179	150.9076	67.3541
RNA – CV	Size=36 decay=0.1	0.9397	218.1783	0.97679	214.5112	106.5668
KNN	K=1	0.9181	254.4146	0.95862	250.1385	101.6077
SVM – CV	C=100 Sigma=0.01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

Novos casos:

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

Gráfico de resíduos:

FIGURA 7.2 – GRÁFICO DE RESÍDUOS - BIOMASSA



FONTE: O autor (2025)

3 – Agrupamento - Veículo

Lista de Clusters gerados:

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

10 primeiras linhas do arquivo com o cluster correspondente.

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

Usa 10 clusters no experimento.

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

4 – Associação - Musculação

Regras geradas com uma configuração de Suporte e Confiança.

Visualizando os dados:

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

Obtendo a visão geral dos dados:

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

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

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

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

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

- Foram criadas 44 regras
- 1 regras com 1 itens
- 14 regras com 2 itens
- 23 regras com 3 itens
- 6 regras com 4 itens

```

> summary(rules)
set of 44 rules

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

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

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

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

```

Avaliação de regras:

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

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

APÊNDICE 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1 - Classificação de Imagens (CNN)

```
# Importar as bibliotecas necessárias
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras import datasets, layers, models
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

# Carregar o conjunto de dados CIFAR-10
```

```

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

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

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

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

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

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

# Construir a arquitetura da CNN

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

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

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

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

```

```

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

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

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

model.summary()

> Model: "sequential"
>
> _____
> Layer (type)                Output Shape          Param #
> =====
> conv2d (Conv2D)              (None, 15, 15, 32)    896
> conv2d_1 (Conv2D)            (None, 7, 7, 64)      18496
> conv2d_2 (Conv2D)            (None, 3, 3, 128)     73856
> flatten (Flatten)            (None, 1152)          0
> dropout (Dropout)            (None, 1152)          0
> dense (Dense)                (None, 1024)          1180672
> dropout_1 (Dropout)          (None, 1024)          0
> dense_1 (Dense)              (None, 10)            10250
> =====
> Total params: 1284170 (4.90 MB)
> Trainable params: 1284170 (4.90 MB)
> Non-trainable params: 0 (0.00 Byte)
> _____

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

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

```

```

r = model.fit(train_images, train_labels, epochs=15,
              validation_data=(test_images, test_labels))

> Epoch 1/15
> /usr/local/lib/python3.10/dist-packages/keras/src/backend.py:5727:
UserWarning: "`sparse_categorical_crossentropy` received
`from_logits=True`, but the `output` argument was produced by a Softmax
activation and thus does not represent logits. Was this intended?
>   output, from_logits = _get_logits(
> 1563/1563 [=====] - 14s 6ms/step - loss: 1.5768
- accuracy: 0.4250 - val_loss: 1.2754 - val_accuracy: 0.5340
> Epoch 2/15
> 1563/1563 [=====] - 8s 5ms/step - loss: 1.3006 -
accuracy: 0.5326 - val_loss: 1.1731 - val_accuracy: 0.5815
> Epoch 3/15
1563/1563 [=====] - 8s 5ms/step - loss: 1.1720 -
> ...
> 1563/1563 [=====] - 8s 5ms/step - loss: 1.0859 -
accuracy: 0.6110 - val_loss: 0.9983 - val_accuracy: 0.6441
> Epoch 5/15
> 1563/1563 [=====] - 7s 4ms/step - loss: 0.6774 -
accuracy: 0.7610 - val_loss: 0.8396 - val_accuracy: 0.7104
> Epoch 15/15
1563/1563 [=====] - 8s 5ms/step - loss: 0.6543 -
accuracy: 0.7687 - val_loss: 0.8297 - val_accuracy: 0.7162

# Visualizar o desempenho

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

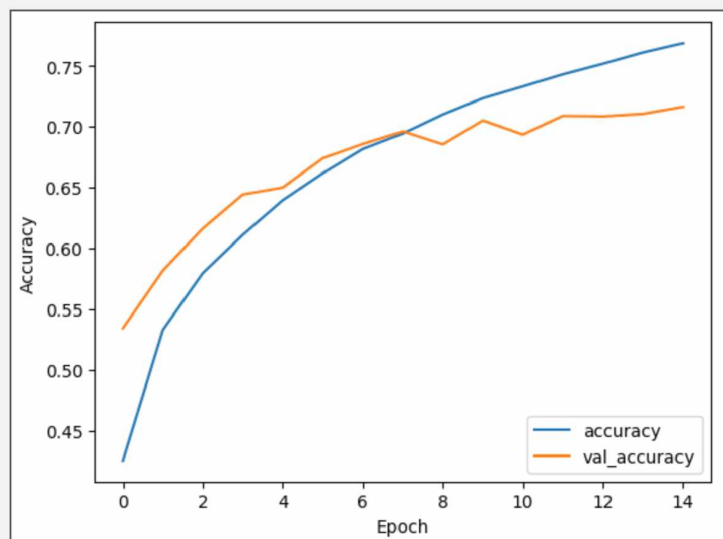
# Gráfico de perda
plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label = 'val_loss')

```



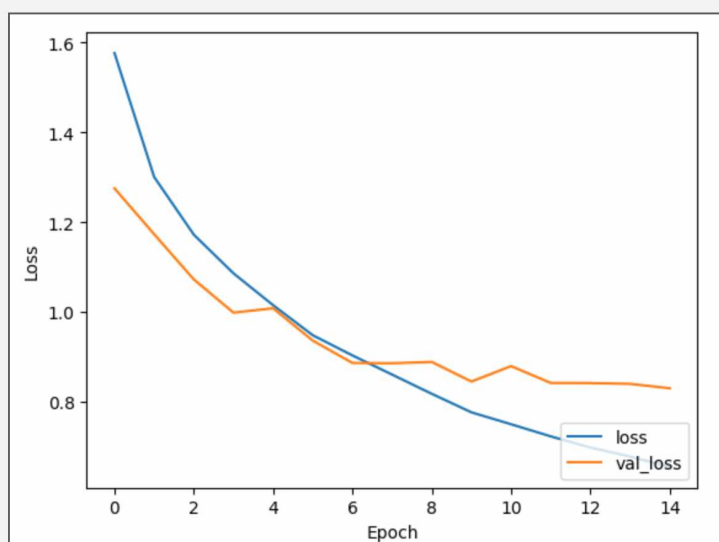
```
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='lower right')
plt.show()
```

FIGURA – 8.1 GRÁFICO DE ACURÁCIA X ÉPOCA PARA O MODELO CNN



FONTE: O autor (2025)

FIGURA 8.2 GRÁFICO DE PERDA X ÉPOCA PARA O MODELO CNN



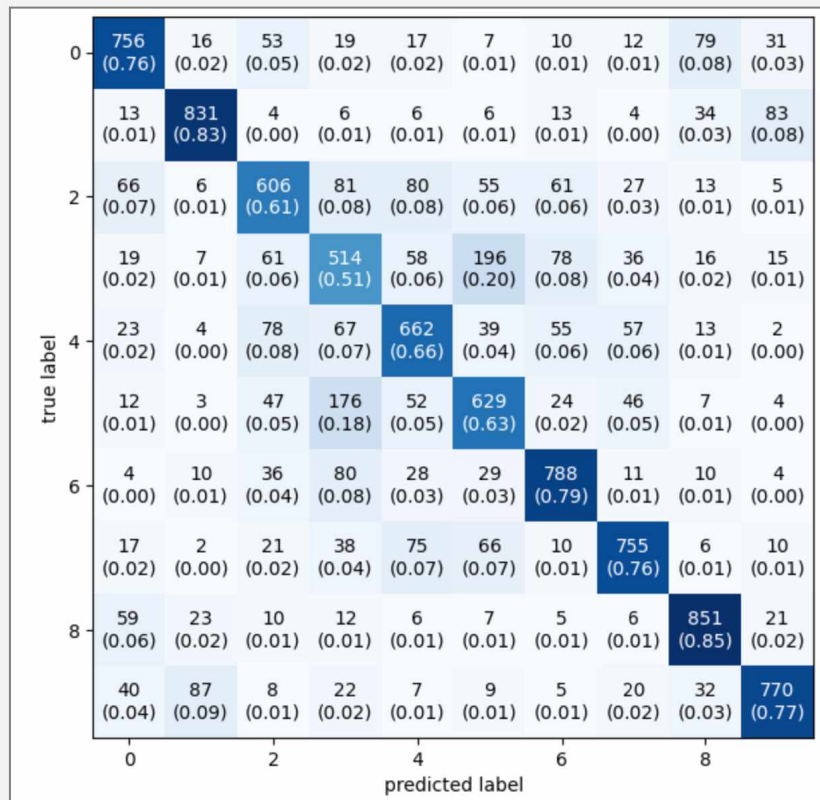
FONTE: O autor (2025)

Efetuando a predição

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

```
# Matriz de confusão
cm = confusion_matrix(test_labels, pred_labels)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7), show_normed=True)
> 313/313 [=====] - 1s 3ms/step
> (<Figure size 700x700 with 1 Axes>,
>  <Axes: xlabel='predicted label', ylabel='true label'>)
```

FIGURA 8.3 – MATRIZ DE CONFUSÃO PARA O MODELO CNN



FONTE: O autor (2025)

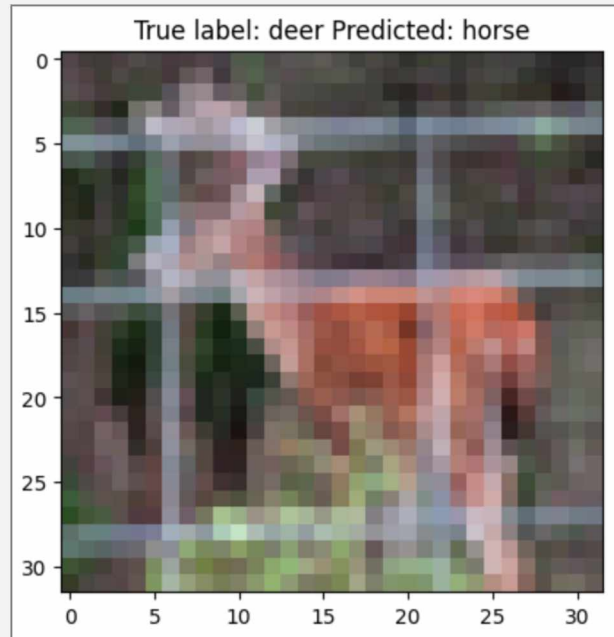
```
# Exibindo algumas classificações erradas
```

```
misclassified = np.where(pred_labels != test_labels)[0]
i = np.random.choice(misclassified)

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

> Text(0.5, 1.0, 'True label: deer Predicted: horse')
```

FIGURA 8.4 – EXEMPLO DE PREDIÇÃO ERRADA PELO MODELO CNN



FONTE: O autor (2025)

2 - Detector de SPAM (RNN)

Importação

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

Carregar base de dados

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

removendo algumas colunas

```
df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
```

```

df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1 })
y = df["b_labels"].values
print(df.head())

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

# separação das bases para treinamento e teste

x_train, x_test, y_train, y_test = train_test_split(df["data"], y,
test_size=0.33)

num_words = 20000
tokenizer = Tokenizer(num_words=num_words)

tokenizer.fit_on_texts(x_train)

sequences_train = tokenizer.texts_to_sequences(x_train)
sequences_test = tokenizer.texts_to_sequences(x_test)
word2index = tokenizer.word_index
V = len(word2index)
print(f"{V} tokens")

> 7126 tokens

# Acertar tamanho das sequências
data_train = pad_sequences(sequences_train)

T = data_train.shape[1]
data_test = pad_sequences(sequences_test, maxlen=T)
print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)
> data_train.shape: (3733, 189)
> data_test.shape: (1839, 189)

```

```
# Definição do modelo

D = 20
M = 5

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

model.summary()
```

FIGURA 8.5 – RESUMO DAS CAMADAS DO MODELO RNN

Model: "functional_1"		
Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 189)	0
embedding_6 (Embedding)	(None, 189, 20)	142,540
lstm_2 (LSTM)	(None, 5)	520
dense_6 (Dense)	(None, 1)	6
Total params: 143,066 (558.85 KB)		
Trainable params: 143,066 (558.85 KB)		
Non-trainable params: 0 (0.00 B)		

FONTE: O autor (2025)

```
# Compilar o modelo

model.compile(
    loss="binary_crossentropy",
    optimizer="adam",
    metrics=["accuracy"]
)

epochs = 5

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

```

> Epoch 1/5
> 117/117 _____ 6s 13ms/step - accuracy: 0.8393 - loss:
0.5503 - val_accuracy: 0.9255 - val_loss: 0.2909
> Epoch 2/5
> 117/117 _____ 1s 10ms/step - accuracy: 0.9512 - loss:
0.2453 - val_accuracy: 0.9734 - val_loss: 0.1685
> Epoch 3/5
> 117/117 _____ 2s 15ms/step - accuracy: 0.9829 - loss:
0.1474 - val_accuracy: 0.9788 - val_loss: 0.1199
> Epoch 4/5
> 117/117 _____ 2s 10ms/step - accuracy: 0.9915 - loss:
0.0982 - val_accuracy: 0.9842 - val_loss: 0.0916
> Epoch 5/5
> 117/117 _____ 1s 10ms/step - accuracy: 0.9948 - loss:
0.0721 - val_accuracy: 0.9869 - val_loss: 0.0773

```

```
# Visualizar accuracy e loss
```

```

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

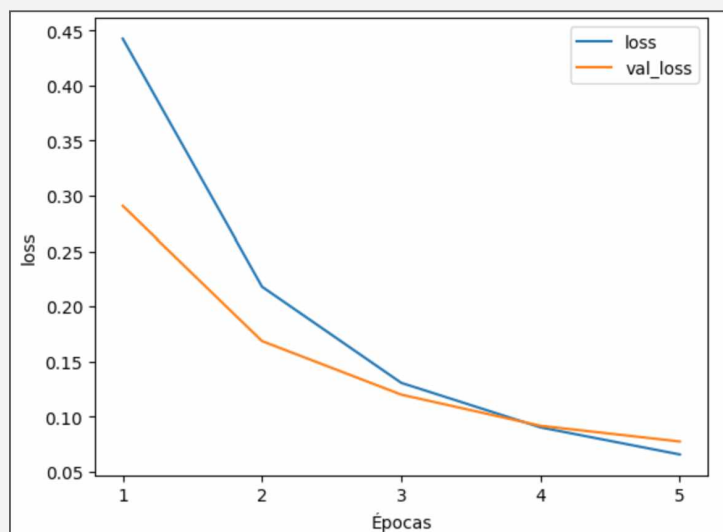
```

```

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

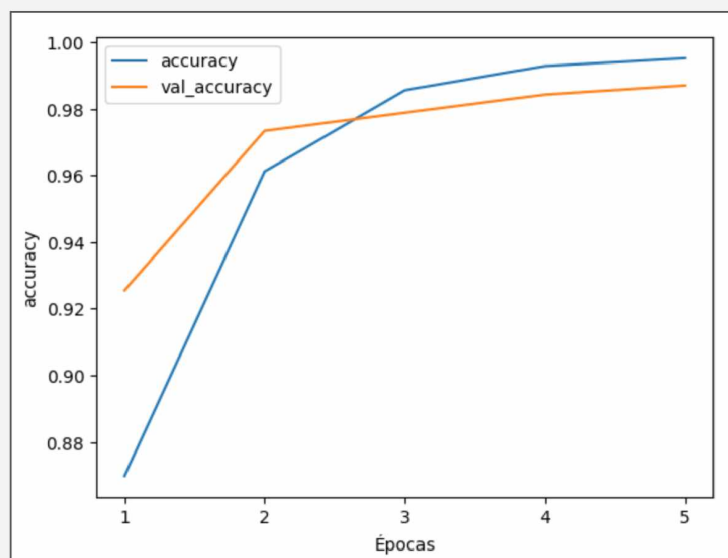
```

FIGURA 8.6 – GRÁFICO DE PERDA X ÉPOCA DO MODELO RNN



FONTE: O autor (2025)

FIGURA 8.7 – GRÁFICO DE ACURÁCIA X ÉPOCA DO MODELO RNN



FONTE: O autor (2025)

```
# Predição de um novo texto
```

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

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

```
pred = model.predict(data_texto)
```

```

print(pred)
print("SPAM" if pred >= 0.5 else "OK")
> 1/1 _____ 0s 102ms/step
> [[0.5963417]]
> SPAM

```

3 - Gerador de dígitos fake (GAN)

```

# Para Gerar os GIFs
!pip install imageio
!pip install git+https://github.com/tensorflow/docs

# Importações
import tensorflow as tf
import glob
import imageio
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
from tensorflow.keras import layers
import time
from IPython import display

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

```



```

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

# Cria o GERADOR
def make_generator_model():
    # Modelo sequencial
    # Camada de entrada
    # Camada de batch normalization
    # Camada de ativação
    model = tf.keras.Sequential()
    model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

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

    # Camada de convolução transposta
    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)

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

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

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

    return model

```

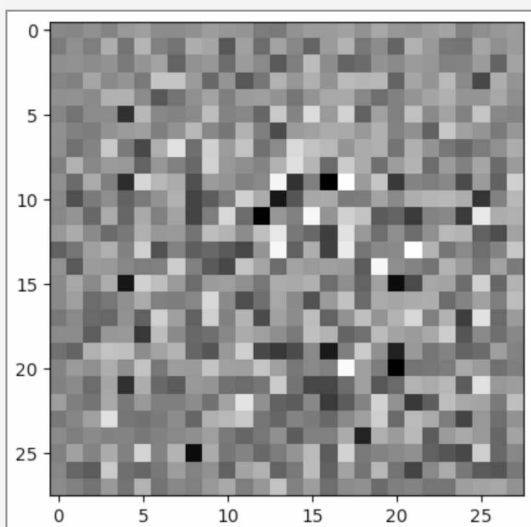
```
# Teste do GERADOR, ainda não treinado
# Cria um modelo
generator = make_generator_model()

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

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

> /usr/local/lib/python3.10/dist-
packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in the
model instead.
> super().__init__(activity_regularizer=activity_regularizer, **kwargs)
> <matplotlib.image.AxesImage at 0x7e2dc74ef550>
```

FIGURA 8.8 – IMAGEM FEITA PELO GERADOR SEM TREINAMENTO



FONTE: O autor (2025)

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

```

model.add(layers.Dropout(0.3))

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

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

return model

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

> tf.Tensor([[0.00129091]], shape=(1, 1), dtype=float32)
> /usr/local/lib/python3.10/dist-
packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do
not pass an `input_shape`/`input_dim` argument to a layer. When using
Sequential models, prefer using an `Input(shape)` object as the first layer
in the model instead.
> super().__init__(activity_regularizer=activity_regularizer, **kwargs)

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

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

# Função que calcula a perda do gerador
def generator_loss(fake_output):

```

```

# Calcula a perda do fake
return cross_entropy(tf.ones_like(fake_output), fake_output)

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

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

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

# Função que faz um passo de treinamento
# É uma `tf.function`, que compila essa função
# para um código mais rápido quando chamada
@tf.function
def train_step(images):
    # Gerar ruído
    noise = tf.random.normal([BATCH_SIZE, noise_dim])
    # Treinar
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        generated_images = generator(noise, training=True)
        # Treinar o discriminador
        real_output = discriminator(images, training=True)
        fake_output = discriminator(generated_images, training=True)
    # Calcular a perda

```

```

    gen_loss = generator_loss(fake_output)
    disc_loss = discriminator_loss(real_output, fake_output)
    # Calcular os gradientes
    gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
    gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
    # Aplicar os gradientes
    generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))

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

        for image_batch in dataset:
            train_step(image_batch)
        # Produce images for the GIF as you go
        display.clear_output(wait=True)
        generate_and_save_images(generator, epoch + 1, seed)
        # Save the model every 15 epochs
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)

        print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-
start))
    # Generate after the final epoch
    display.clear_output(wait=True)
    generate_and_save_images(generator, epochs, seed)

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

```

```

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

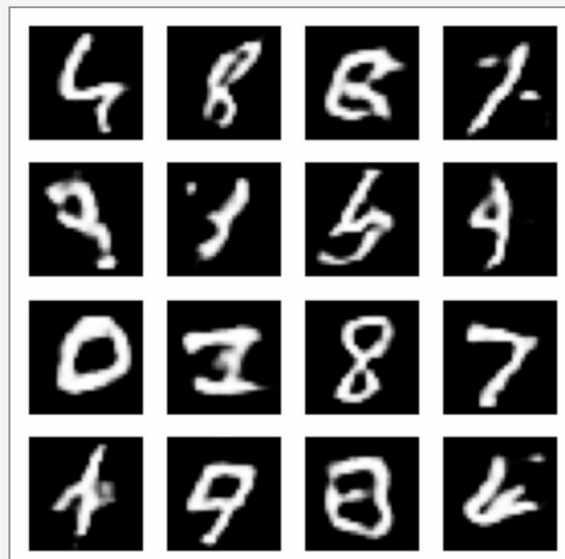
for i in range(predictions.shape[0]):
    plt.subplot(4, 4, i+1)
    plt.imshow(predictions[i, :, :, 0] * 127.5 + 127.5, cmap='gray')
    plt.axis('off')

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

# Treinar o modelo e restaurar o último ponto de verificação
train(train_dataset, EPOCHS)
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))

```

FIGURA 8.9 – IMAGEM CRIADA PELO GERADOR APÓS TREINAMENTO JUNTO COM O DISCRIMINADOR



FONTE: O autor (2025)

4 – Tradutor de textos (Transformer)

```

# Instalação de libs
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0

```

```

# Importaçãod de libs
import collections
import logging
import os
import pathlib
import re
import string
import sys
import time

import numpy as np
import matplotlib.pyplot as plt

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

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

# Carregamento 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']

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

    print()

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

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

```

```

> mas e se estes fatores fossem ativos ?
> mas eles não tinham a curiosidade de me testar .

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

# Tokenização e Destokenização
model_name = "ted_hrlr_translate_pt_en_converter"

tf.keras.utils.get_file(
    f"{model_name}.zip",
    f"https://storage.googleapis.com/download.tensorflow.org/models/{model_name}.zip",
    cache_dir='.',
    cache_subdir='',
    extract=True
)

tokenizers = tf.saved_model.load(model_name)

# Pipeline de entrada
def tokenize_pairs(pt, en):
    pt = tokenizers.pt.tokenize(pt)
    pt = pt.to_tensor()

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

BUFFER_SIZE = 20000
BATCH_SIZE = 64

def make_batches(ds):
    return (
        ds
        .cache()

```



```

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

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

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

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

# Máscara 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

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

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

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

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

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

        assert d_model % self.num_heads == 0

        self.depth = d_model // self.num_heads

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

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

```

```

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

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

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

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

    output = self.dense(concat_attention)

    return output, attention_weights

# 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)
    ])

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

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

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

```

```

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

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

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

    return out2

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

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

        self.ffn = point_wise_feed_forward_network(d_model, dff)

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

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

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

```

```

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

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

        return out3, attn_weights_block1, attn_weights_block2

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

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

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

```

```

        x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
        x += self.pos_encoding[:, :seq_len, :]
        x = self.dropout(x, training=training)
        for i in range(self.num_layers):
            x = self.enc_layers[i](x, training, mask)
        return x

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

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

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

        x = self.dropout(x, training=training)

```

```

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

    return x, attention_weights

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

```

```

        d_model,
        num_heads,
        dff,
        target_vocab_size,
        pe_target,
        rate
    )
    self.final_layer = tf.keras.layers.Dense(target_vocab_size)

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

        return final_output, attention_weights

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

        return enc_padding_mask, look_ahead_mask, dec_padding_mask

# Hiperparâmetros
num_layers = 4
d_model = 128

```



```

dff = 512
num_heads = 8
dropout_rate = 0.1

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

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

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

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

# Função de perda e métricas
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
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 = 25
train_step_signature = [
    tf.TensorSpec(shape = (None, None), dtype = tf.int64),
    tf.TensorSpec(shape = (None, None), dtype=tf.int64)
]

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

    with tf.GradientTape() as tape:
        predictions, _ = transformer([inp, tar_inp], training = True)
        loss = loss_function(tar_real, predictions)

    gradients = tape.gradient(loss, transformer.trainable_variables)
    optimizer.apply_gradients(zip(gradients,
transformer.trainable_variables))
    train_loss(loss)
    train_accuracy(accuracy_function(tar_real, predictions))

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

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

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

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

```

```

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

```

```

> Epoch 1 Batch 0 Loss 8.8324 Accuracy 0.0022
> Epoch 1 Batch 50 Loss 8.7743 Accuracy 0.0079
> Epoch 1 Batch 100 Loss 8.6878 Accuracy 0.0256
> ...
> Epoch 1 Batch 750 Loss 6.7963 Accuracy 0.1170
> Epoch 1 Batch 800 Loss 6.7057 Accuracy 0.1223
> Epoch 1 Loss 6.6897 Accuracy 0.1233
> Time taken for epoch 1: 171.75 secs
> ...
> Epoch 25 Batch 0 Loss 1.3378 Accuracy 0.6965
> Epoch 25 Batch 50 Loss 1.2698 Accuracy 0.7064
> Epoch 25 Batch 100 Loss 1.2818 Accuracy 0.7054
> ...
> Epoch 25 Batch 700 Loss 1.3153 Accuracy 0.7010
> Epoch 25 Batch 750 Loss 1.3201 Accuracy 0.7002
> Epoch 25 Batch 800 Loss 1.3243 Accuracy 0.6995
> Saving checkpoint for epoch 25 at ./checkpoints/train/ckpt-5
> Epoch 25 Loss 1.3248 Accuracy 0.6994
> Time taken for epoch 25: 97.99 secs

```

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

# Traduzindo
translator = Translator(tokenizers, transformer)
sentence = "vamos testar o tradutor."
translated_text, translated_tokens, attention_weights = translator(
    tf.constant(sentence)
)

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

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

```

APÊNDICE 9 – BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

RESUMO

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

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

ABSTRACT

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

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

1 INTRODUÇÃO

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

2 CARACTERIZAÇÃO DOS DADOS

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

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

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

3 FERRAMENTAS UTILIZADAS

1. NoSQL

1.1. MongoDB

Modelo de Dados: Documentos JSON.

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

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

1.2. Cassandra

Modelo de Dados: Colunas.

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

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

2. NewSQL

2.1. CockroachDB

Modelo de Dados: Relacional com suporte a SQL.

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

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

4 MODELAGEM DE DADOS

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

1. NoSQL (MongoDB e Cassandra):

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

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

2. NewSQL (CockroachDB):

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

5 CONSIDERAÇÕES FINAIS

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

APÊNDICE 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

1 - Extração de características

Métricas dos modelos:

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

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

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

Conclusão:

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

2 – Redes Neurais

Avaliação do VGG16 (sem augmentation):

Matriz de confusão:

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

Report de classificação:

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

Acurácia: 24.80%

Avaliação do VGG16 (com augmentation):

Matriz de confusão:

```

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

Report de classificação:

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

Acurácia: 26.15%

Avaliação do ResNet50 (sem augmentation):

Matriz de confusão:

```

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

```

Report de classificação:

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

Acurácia: 26.15%

Avaliação do ResNet50 (com augmentation):

Matriz de confusão:

```

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

```

Report de classificação:

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

Acurácia: 25.61%

Conclusão:

Após a avaliação dos modelos VGG16 e ResNet50, ambos apresentaram desempenhos semelhantes, com uma acurácia de 26.15% para o VGG16 e 25.61% para o ResNet50 com Data Augmentation. No entanto, o VGG16 teve um desempenho superior, especialmente na classe 0, com uma sensibilidade de 66%, enquanto a ResNet50 teve dificuldades em classificar corretamente a classe 1, resultando em um recall de 0%. Logo, considerando as métricas de precisão, recall e F1 score, o VGG16 se destaca como a melhor escolha devido à sua capacidade mais consistente de identificar as classes, especialmente a classe 0. Assim, o VGG16 é o modelo recomendado para a classificação das imagens.

APÊNDICE 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

Parâmetros para elaboração do Trabalho:

1. Relevância Ética: O trabalho deve abordar questões éticas significativas relacionadas ao uso da inteligência artificial, especialmente no contexto do ChatGPT. Os alunos devem identificar dilemas éticos relevantes e explorar como esses dilemas afetam diferentes partes interessadas, como usuários, desenvolvedores e a sociedade em geral.

2. Análise Crítica: Os alunos devem realizar uma análise crítica das implicações éticas do uso do ChatGPT em estudos de caso específicos. Eles devem examinar como o algoritmo pode influenciar a disseminação de informações, a privacidade dos usuários e a tomada de decisões éticas. Além disso, devem considerar possíveis vieses algorítmicos, discriminação e questões de responsabilidade.

3. Soluções Responsáveis: Além de identificar os desafios éticos, os alunos devem propor soluções responsáveis e éticas para lidar com esses dilemas. Isso pode incluir sugestões para políticas, regulamentações ou práticas de design que promovam o uso responsável da inteligência artificial. Eles devem considerar como essas soluções podem equilibrar os interesses de diferentes partes interessadas e promover valores éticos fundamentais, como transparência, justiça e privacidade.

4. Colaboração e Discussão: O trabalho deve envolver discussões em grupo e colaboração entre os alunos. Eles devem compartilhar ideias, debater diferentes pontos de vista e chegar a conclusões informadas através do diálogo e da reflexão mútua. O estudo de caso do ChatGPT pode servir como um ponto de partida para essas discussões, incentivando os alunos a aplicar conceitos éticos e legais aprendidos ao analisar um caso concreto.

5. Limite de Palavras: O trabalho terá um limite de 6 a 10 páginas teria aproximadamente entre 1500 e 3000 palavras.

6. Estruturação Adequada: O trabalho siga uma estrutura adequada, incluindo introdução, desenvolvimento e conclusão. Cada seção deve ocupar uma parte proporcional do total de páginas, com a introdução e a conclusão ocupando menos espaço do que o desenvolvimento.

7. Controle de Informações: Evitar incluir informações desnecessárias que possam aumentar o comprimento do trabalho sem contribuir significativamente para o conteúdo. Concentre-se em informações relevantes, argumentos sólidos e evidências importantes para apoiar sua análise.

8. Síntese e Clareza: O trabalho deverá ser conciso e claro em sua escrita. Evite repetições desnecessárias e redundâncias. Sintetize suas ideias e argumentos de forma eficaz para transmitir suas mensagens de maneira sucinta.

9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: <https://bibliotecas.ufpr.br/wp-content/uploads/2022/03/template-artigo-de-periodico.docx>

B – RESOLUÇÃO

RESUMO

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

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

ABSTRACT

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

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

1 INTRODUÇÃO

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

2 PRIVACIDADE E CONFIDENCIALIDADE

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

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

3 QUALIDADE DO ACONSELHAMENTO E RESPONSABILIDADE

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

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

4 VIÉS ALGORÍTMICO E DISCRIMINAÇÃO

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

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

4 TOMADA DE DECISÃO ÉTICA

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

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

5 PROPOSTA E SOLUÇÕES RESPONSÁVEIS

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

- 1. Políticas Claras de Privacidade e Segurança de Dados:** Desenvolver e aplicar políticas robustas de privacidade que garantam a proteção adequada dos dados dos usuários.
- 2. Diretrizes Éticas Específicas:** Estabelecer diretrizes éticas específicas para o uso de IA em aconselhamento psicológico, com ênfase na transparência, responsabilidade e respeito aos direitos dos usuários.
- 3. Mitigação de Viés Algorítmico:** Implementar medidas eficazes para identificar e mitigar vieses algorítmicos, incluindo revisão humana e diversificação dos conjuntos de dados de treinamento.
- 4. Educação e Conscientização dos Usuários:** Educar os usuários sobre as capacidades e limitações do ChatGPT, promovendo uma compreensão informada do uso de IA no suporte emocional.
- 5. Integração de Supervisão Humana:** Integrar supervisão humana qualificada para monitorar e revisar interações críticas, garantindo uma abordagem ética ao aconselhamento psicológico.

6 CONSIDERAÇÕES FINAIS

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

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

REFERÊNCIAS

- Floridi, L. (2020). "Soft Ethics, the Governance of the Digital and the General Data Protection Regulation: Developing a Data Ethics Framework." *Philosophy & Technology*, 33(2), 179-185.
- Turilli, M., & Floridi, L. (2009). "The Ethics of Information Transparency." *Ethics and Information Technology*, 11(2), 105-112.
- Bostrom, N., & Yudkowsky, E. (2014). "The Ethics of Artificial Intelligence." In E. Frankish & W. M. Ramsey (Eds.), *The Cambridge Handbook of Artificial Intelligence* (pp. 316-334). Cambridge University Press.
- Jobin, A., Ienca, M., & Vayena, E. (2019). "The Global Landscape of AI Ethics Guidelines." *Nature Machine Intelligence*, 1(9), 389-399.
- Bryson, J. J. (2018). "Patient data and artificial intelligence." *Nature Biomedical Engineering*, 2(5), 293-293.
- Taddeo, M., & Floridi, L. (2018). "How AI Can Be a Force for Good." *Science*, 361(6404), 751-752.
- Mittelstadt, B. D., et al. (2016). "The Ethics of Algorithms: Mapping the Debate." *Big Data & Society*, 3(2), 2053951716679679.
- Jobin, A., et al. (2019). "Artificial Intelligence: The Ambiguity of Ethics and Intelligence." *Nature*, 568(7750), 626-628.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Nome do artigo escolhido: Architecting ML-enabled systems: Challenges, best practices, and design decisions

Qual o **objetivo** do estudo descrito pelo artigo?

Os autores do artigo visam preencher uma lacuna nas pesquisas dentro da área de desenvolvimento de software, sendo mais especificamente no que diz respeito à elaboração de sistemas com componentes de Machine Learning – ML. O foco do estudo é identificar os desafios comuns, melhores práticas de design de software e principais decisões arquiteturais nesse tipo de sistema de um ponto de vista de pesquisadores e profissionais dentro dessa área. Para isso os autores elaboraram três perguntas para serem respondidas mediante o estudo dos dados levantados durante a pesquisa. Essas perguntas são:

1 – Quais são os desafios mais comuns no design de arquitetura em sistemas com componentes de Machine Learning?

2 – Quais são as melhores práticas de arquitetura em sistemas com componentes de Machine Learning?

3 – Quais são as principais decisões de design de arquitetura em sistemas com componentes de Machine Learning?

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

De acordo com os estudos apontados pelos autores no artigo, as empresas que produzem software têm testemunhado um aumento na demanda de soluções que exigem componentes de Machine Learning. A engenharia de software é um processo muito importante no desenvolvimento de qualquer tipo de aplicação e não é diferente no caso de sistemas com Machine Learning, especialmente considerando que esse tipo de sistema possui propriedades distintas e uma grande complexidade, fazendo-se necessário uma mudança substancial nos aspectos organizacionais e no processo de

desenvolvimento. Os autores mencionam que não existem estudos que analisem como profissionais percebem e usam decisões de design na arquitetura de sistemas com Machine Learning. Portanto, há uma lacuna no campo da pesquisa sobre esse assunto sendo que alguns pontos mencionados no artigo são: Manutenção e evolução durante o ciclo de vida, aspectos de qualidade, integração com outros componentes e gerenciamento de incertezas.

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

A pesquisa foi formulada tendo como objetivo responder à três perguntas relacionadas ao desenvolvimento de sistemas com Machine Learning sendo elas:

- 1 – Quais os desafios mais comuns no desenvolvimento desse tipo de sistema?
- 2 – Quais as melhores práticas no desenvolvimento desse tipo de sistema?
- 3 – Quais as principais decisões de design de arquitetura no desenvolvimento desse tipo de Sistema?

Para responder à essas perguntas, os pesquisadores conduziram uma pesquisa utilizando uma metodologia mista para extrair insights tanto do estado da arte e o estado atual da prática. Uma parte do processo consistiu em revisão de literatura proveniente de outros estudos e, para isso, foi consultado 4 das maiores bases de dados em engenharia de software. A outra parte consistiu em uma pesquisa online composta por 15 perguntas abertas e que foi direcionada para profissionais das redes de contatos dos pesquisadores. A partir da revisão literária e das entrevistas foram extraídos, de forma sistemática, dados para responder às perguntas.

Quais os **principais resultados** obtidos pelo estudo?

Os autores encontraram algumas correlações entre os desafios, melhores práticas e decisões de design, sendo que essas correlações foram agrupadas em seis categorias: arquitetura, dados, evolução, QA, modelo e ciclo de vida do desenvolvimento de software. Alguns dados que foram observados a partir da literatura e das entrevistas se enquadraram em outras categorias, mas que não foram observadas correlações. Em termos dos desafios os principais relatados foram a respeito da gestão de dados e arquitetura. Em termos de melhores práticas as principais relatadas nos estudos e entrevistas estão relacionadas com arquitetura, como o uso de padrões específicos e ferramentas, tendo em vista a baixa coesão por questões de manutenção. Em termos das decisões de design que impactam o desenvolvimento de sistemas com Machine Learning as principais decisões estão nos grupos de arquitetura, hardware e plataforma. De forma geral os dados encontrados nas pesquisas muitas vezes eram semelhantes aos encontrados nas entrevistas, porém em alguns casos houve certas divergências.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

Implementar o exemplo de Classificação usando a base de dados Fashion MNIST e a arquitetura RNA vista na aula **FRA - Aula 10 - 2.4 Resolução de exercício de RNA - Classificação**.

Além disso, fazer uma breve explicação dos seguintes resultados:

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

Informações:

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

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

2 Regressão (RNA)

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

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

Informações:

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

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

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

1 - Classificação (RNA)

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

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

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

# Definir o modelo da rede neural
model = tf.keras.Sequential([
```



```

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

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

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

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

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

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

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

# Plotando a perda de treino e validação
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Treinamento')
plt.plot(history.history['val_loss'], label='Validação')

```

```

plt.title('Perda durante o treinamento')
plt.xlabel('Épocas')
plt.ylabel('Perda')
plt.legend()

plt.show()

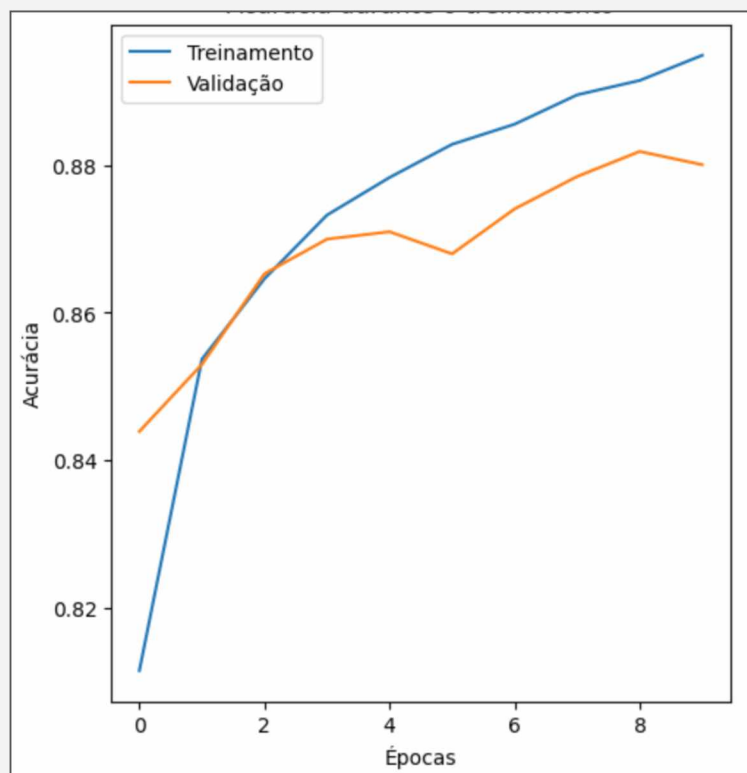
# Mostrar algumas classificações erradas
predictions = model.predict(x_test)
incorrect_indices = np.where(np.argmax(predictions, axis=1) != y_test)[0]

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

> Epoch 1/10
> 1875/1875 _____ 9s 4ms/step - accuracy: 0.7606 - loss: 0.6853 - val_accuracy: 0.8439 - val_loss: 0.4336
> Epoch 2/10
> 1875/1875 _____ 9s 4ms/step - accuracy: 0.8497 - loss: 0.4091 - val_accuracy: 0.8530 - val_loss: 0.3997
> Epoch 3/10
> 1875/1875 _____ 11s 4ms/step - accuracy: 0.8641 - loss: 0.3717 - val_accuracy: 0.8653 - val_loss: 0.3714
> ...
> Epoch 8/10
> 1875/1875 _____ 9s 4ms/step - accuracy: 0.8874 - loss: 0.3022 - val_accuracy: 0.8785 - val_loss: 0.3435
> Epoch 9/10
> 1875/1875 _____ 9s 5ms/step - accuracy: 0.8944 - loss: 0.2890 - val_accuracy: 0.8819 - val_loss: 0.3377
> Epoch 10/10
> 1875/1875 _____ 11s 5ms/step - accuracy: 0.8956 - loss: 0.2809 - val_accuracy: 0.8801 - val_loss: 0.3420
> 313/313 - 0s - 1ms/step - accuracy: 0.8801 - loss: 0.3420
> Test accuracy: 0.8801000118255615

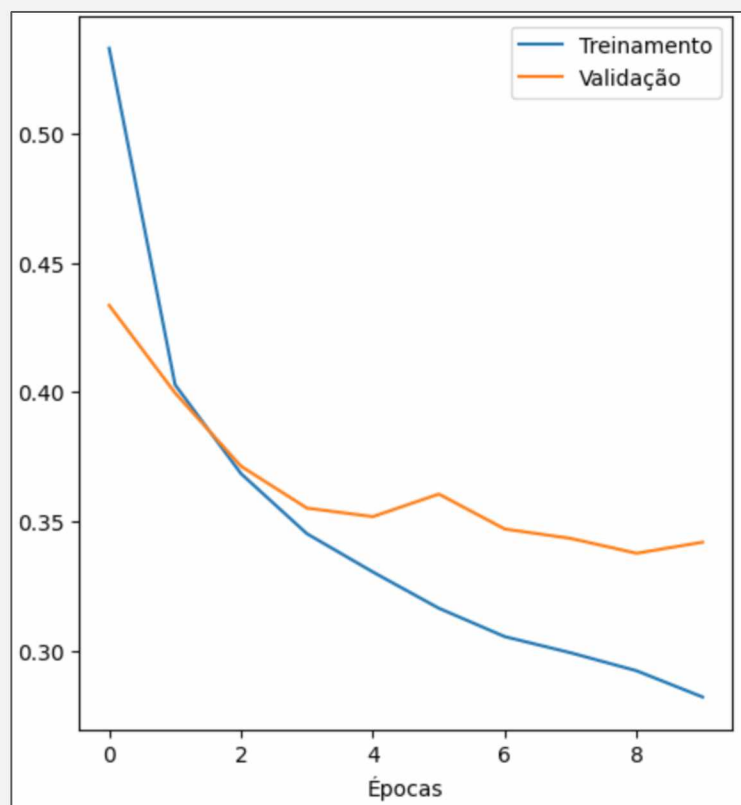
```

FIGURA 13.1 – GRÁFICO DE ACURÁCIA X ÉPOCA DURANTE O TREINAMENTO



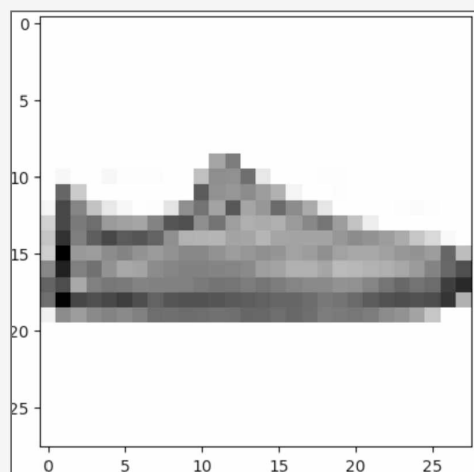
FONTE: O autor (2025)

FIGURA 13.2 – GRÁFICO DE PERDA X ÉPOCA DURANTE O TREINAMENTO



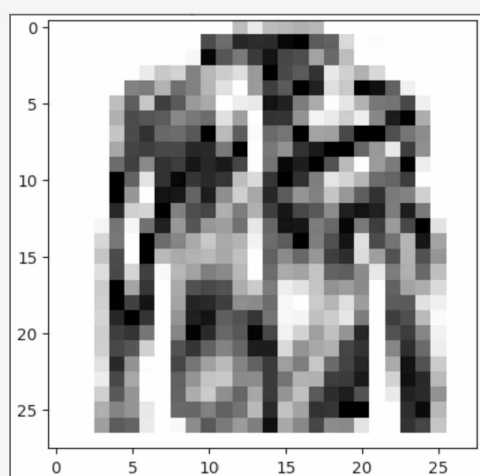
FONTE: O autor (2025)

FIGURA 13.3 – EXEMPLO DE PREDIÇÃO ERRADA 1 – PREDIÇÃO: 5, VERDADEIRO: 7



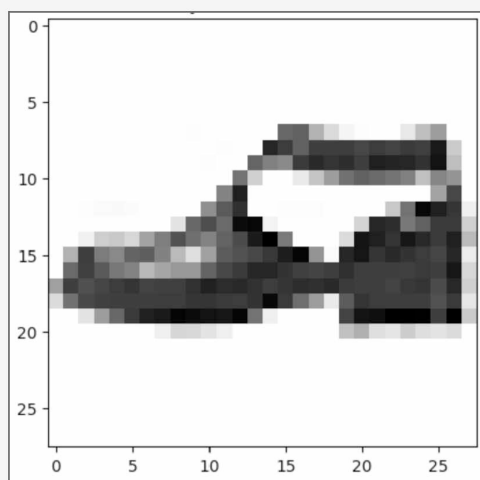
FONTE: O autor (2025)

FIGURA 13.4 – EXEMPLO DE PREDIÇÃO ERRADA 2 – PREDIÇÃO: 2, VERDADEIRO: 4



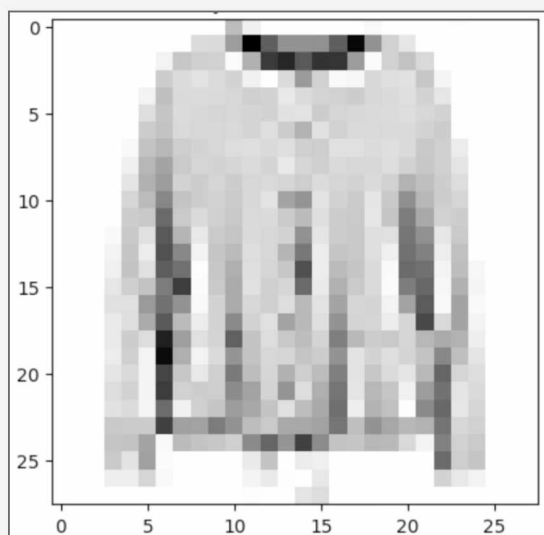
FONTE: O autor (2025)

FIGURA 13.5 – EXEMPLO DE PREDIÇÃO ERRADA 3 – PREDIÇÃO: 5, VERDADEIRO: 9



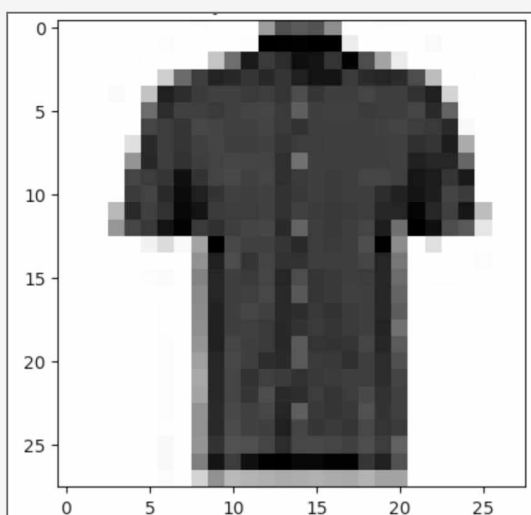
FONTE: O autor (2025)

FIGURA 13.6 – EXEMPLO DE PREDIÇÃO ERRADA 4 – PREDIÇÃO: 2, VERDADEIRO: 4



FONTE: O autor (2025)

FIGURA 13.7 – EXEMPLO DE PREDIÇÃO ERRADA 5 – PREDIÇÃO: 0, VERDADEIRO: 6



FONTE: O autor (2025)

Comentários

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

2 - Regressão (RNA)

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

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

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

FIGURA 13.8 – TABELA APRESENTANDO OS PRIMEIROS REGISTROS DO DATASET

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

FONTE: O autor (2025)

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

```
data.head()
```

FIGURA 13.9 – TABELA APRESENTANDO OS PRIMEIROS REGISTROS DO DATASET COM TÍTULOS DE COLUNAS ATUALIZADOS

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

FONTE: O autor (2025)

```
print(data.shape)
```

```
> (1599, 12)
```

```
# separa variáveis explicativas da variável resposta
```

```
x = data[['acidez_fixa',
'acidez_volatil',
'acido_citrico',
'acucar_residual',
'cloretos',
'dioxido_de_enxofre_livre',
'dioxido_de_enxofre_total',
'densidade',
'pH',
'sulfatos',
'alcool']].values.astype(float)
```

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

```
print(type(x))
```

```
print(type(y))
```

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

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

```
# verificando dados faltantes ou infinito
```

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

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

```
> False False
```

```
> False False
```

```

# normalização dos dados
from sklearn.preprocessing import StandardScaler

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

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

# separando base de treino e teste
x_treino, x_teste, y_treino, y_teste = train_test_split(x, y,
test_size=0.3, random_state=308)

x_teste

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

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

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

```



```

from keras import backend

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

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

# ajuste do modelo

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

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

# stops para epocas

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

# treinamento do modelo
treino_modelo =
modelo_1.fit(x_treino,y_treino,epochs=1000,validation_data=(x_teste,y_teste
),callbacks = [early_stops])

> Epoch 1/1000
> 35/35 _____ 1s 7ms/step - loss: 1.0810 - r2: -0.1394 -
rmse: 1.0190 - val_loss: 0.6992 - val_r2: 0.2907 - val_rmse: 0.8330
> Epoch 2/1000
> 35/35 _____ 0s 3ms/step - loss: 0.7095 - r2: 0.2355 -
rmse: 0.8326 - val_loss: 0.6565 - val_r2: 0.3392 - val_rmse: 0.8044
> Epoch 3/1000
> 35/35 _____ 0s 3ms/step - loss: 0.6401 - r2: 0.3561 -
rmse: 0.7913 - val_loss: 0.6649 - val_r2: 0.3262 - val_rmse: 0.8124

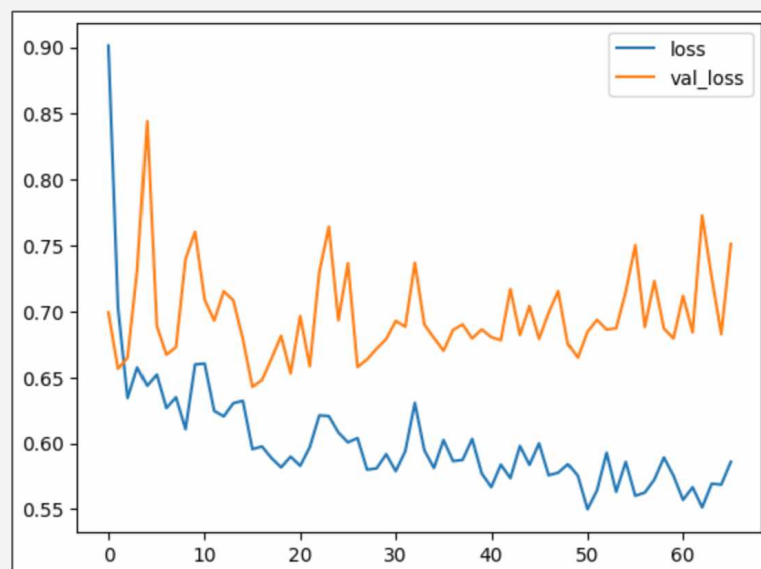
```

```
> ...
> Epoch 64/1000
> 35/35 _____ 0s 3ms/step - loss: 0.5693 - r2: 0.3528 -
rmse: 0.7489 - val_loss: 0.7257 - val_r2: 0.2646 - val_rmse: 0.8481
> Epoch 65/1000
> 35/35 _____ 0s 3ms/step - loss: 0.5788 - r2: 0.4041 -
rmse: 0.7543 - val_loss: 0.6827 - val_r2: 0.3114 - val_rmse: 0.8226
> Epoch 66/1000
> 35/35 _____ 0s 3ms/step - loss: 0.5569 - r2: 0.3755 -
rmse: 0.7383 - val_loss: 0.7511 - val_r2: 0.2405 - val_rmse: 0.8588
> ...
```

```
# avaliação do modelo
```

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

FIGURA 13.10 – GRÁFICO DE PERDA X ÉPOCA E PERDA DE VALIDAÇÃO X ÉPOCA



FONTE: O autor (2025)

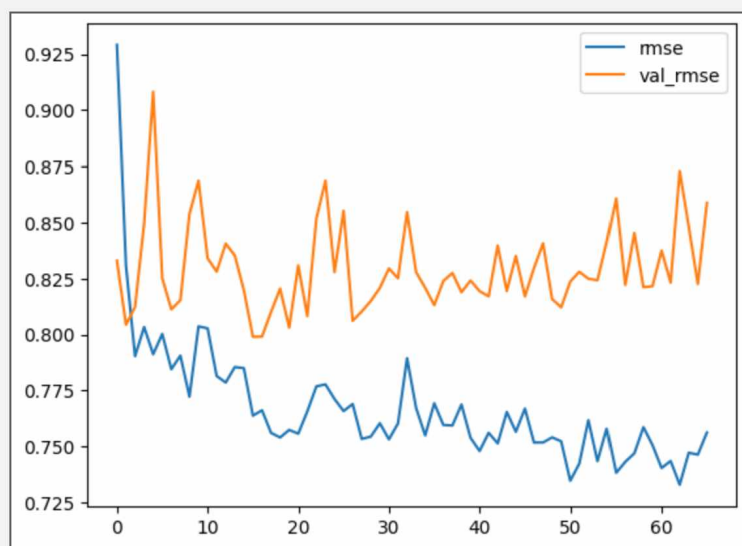
O gráfico acima representa a evolução da função de perda conforme o número de épocas aumenta. Observamos queda na função de perda conforme aumenta o número de épocas, o que é esperado, indicando que o modelo está aprendendo a minimizar o erro.

```
# rmse
```

```
plt.plot(modelo_1.history.history['rmse'], label='rmse')
plt.plot(modelo_1.history.history['val_rmse'], label='val_rmse')
```

```
plt.legend()
```

FIGURA 13.11 – GRÁFICO DE EVOLUÇÃO DO RMSE X ÉPOCA E RMSE DE VALIDAÇÃO X ÉPOCA



FONTE: O autor (2025)

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

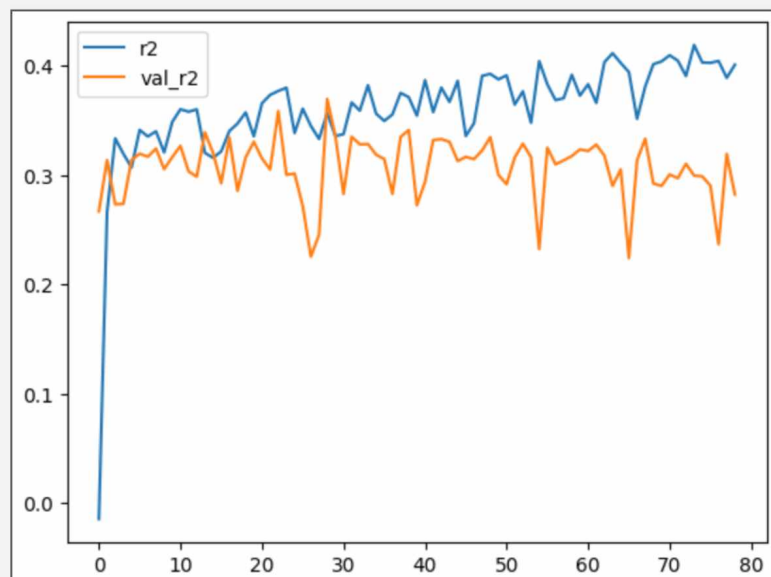
```
# plotando r2
```

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

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

```
plt.legend()
```

FIGURA 13.12 – GRFICO DE EVOLUÇÃO DE R2 X ÉPOCA E R2 DE VALIDAÇÃO X ÉPOCA



FONTE: O autor (2025)

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

```
y_hat = modelo_1.predict(x_teste).flatten()
mse = mean_squared_error(y_teste,y_hat)
rmse = sqrt(mse)
r2 = r2_score(y_teste,y_hat)
```

```
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'R2: {r2}')
```

```
> MSE: 0.6280485759170473
> RMSE: 0.7924951582924954
> R2: 0.3915046095974757
```

Comentários

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

3 - Sistema de recomendação

```
csv_path = '/content/Base_livros.csv'
K = 10
epochs = 50
batch_size = 1024

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

from sklearn.utils import shuffle
```

```

import numpy as np
import pandas as pd
# 2 Carregamento dos dados

!head '/content/Base_livros.csv'
> ISBN,Titulo,Autor,Ano,Editora,ID_usuario,Notas
> 2005018,Clara Callan,Richard Bruce Wright,2001,HarperFlamingo
Canada,276725,0
> 60973129,Decision in Normandy,Carlo D'Este,1991,HarperPerennial,276726,2
> 374157065,Flu: The Story of the Great Influenza Pandemic of 1918 and the
Search for the Virus That Caused It,Gina Bari Kolata,1999,Farrar Straus
Giroux,276727,6
> 393045218,The Mummies of Urumchi,E. J. W. Barber,1999,W. W. Norton &
Company,276729,1
> 399135782,The Kitchen God's Wife,Amy Tan,1991,Putnam Pub Group,276729,9
> 425176428,What If?: The World's Foremost Military Historians Imagine What
Might Have Been,Robert Cowley,2000,Berkley Publishing Group,276733,0
> 671870432,PLEADING GUILTY,Scott Turow,1993,Audioworks,276736,3
> 679425608,Under the Black Flag: The Romance and the Reality of Life Among
the Pirates,David Cordingly,1996,Random House,276737,9
> 074322678X,Where You'll Find Me: And Other Stories,Ann
Beattie,2002,Scribner,276744,4

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

```

FIGURA 13.13 – TABELA APRESENTANDO OS PRIMEIROS REGISTROS DO DATASET DA BASE DE LIVROS

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

FONTE: O autor (2025)

```

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

```

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

```
> ISBN          object
> Título         object
> Autor         object
> Ano           int64
> Editora       object
> ID_usuario    int64
> Notas        int64
> dtype: object
> -----
> Menor nota:  0
> Maior nota: 10
> -----
> Shape:  (128895, 7)
> -----
> Distribuição por Nota:
> Notas
> 0      11634
> 1      11780
> 2      11758
> 3      11773
> 4      11646
> 5      11739
> 6      11786
> 7      11653
> 8      11923
> 9      11436
> 10     11767
> Name: count, dtype: int64
```

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

FIGURA 13.14 – TABELA APRESENTANDO AMOSTRA DA QUANTIDADE DE TÍTULOS REPETIDOS

	count
Titulo	
Little Women	15
Wuthering Heights	14
Adventures of Huckleberry Finn	14
Dracula	14
The Secret Garden	13
...	...
Comparative cultural analysis: An introduction to anthropology	1
Yanomamo, the Fierce People (American Problem Studies)	1
Global Perspectives on Ecology	1
Man and environment	1
The Adventures of Lando Calrissian: Lando Calrissian and the Mindharp of Sharu/Lando Calrissian and the Flamewind of Oseon/Lando Calrissian and the	1

FONTE: O autor (2025)

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

FIGURA 13.15 – TABELA COM AMOSTRA DOS USUÁRIOS COM MAIS DE 20 LIVROS AVALIADOS

	count
ID_usuario	
11676	13602
278418	4533
16795	2948
23768	1708
11601	1571
...	...
13874	20
21231	20
277466	20
277272	20
20448	20

777 rows x 1 columns

FONTE: O autor (2025)

```
df.shape

> (98969, 7)

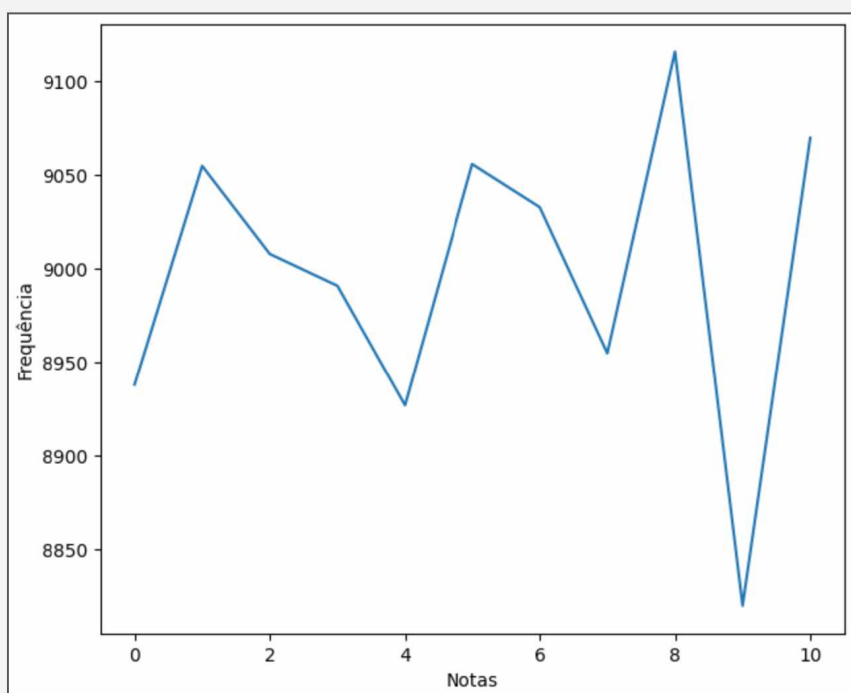
# 2.4 Verificação da distribuição do dataset
plt.figure(figsize=(16, 6))

plt.subplot(1, 2, 1)
score_counts = df['Notas'].value_counts().sort_index()
score_counts.plot()
plt.xlabel('Notas')
plt.ylabel('Frequência')
plt.xticks(rotation=0)

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

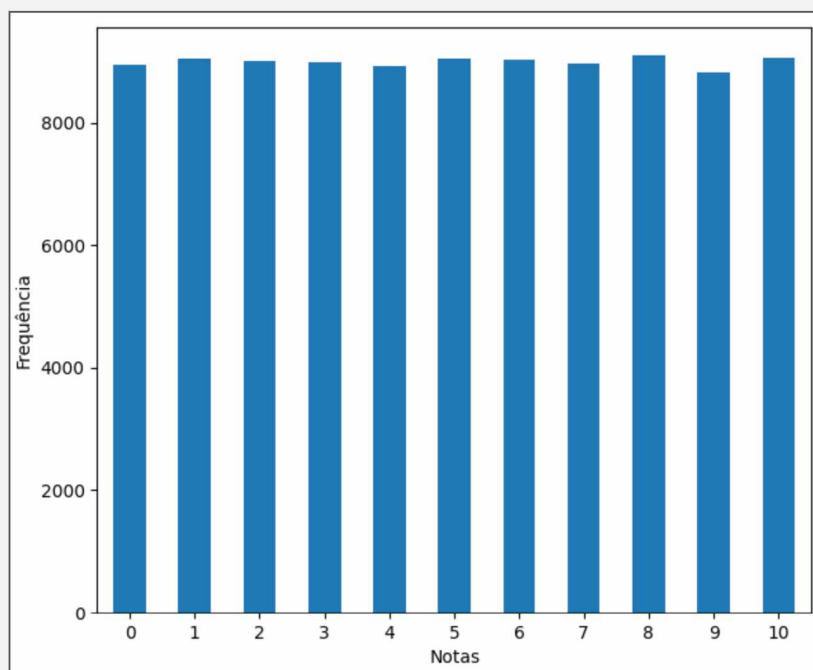
plt.show()
```

FIGURA 13.16 – GRÁFICO DE LINHA APRESENTANDO A FREQUÊNCIA DAS NOTAS



FONTE: O autor (2025)

FIGURA 13.17 – GRÁFICO DE BARRAS APRESENTANDO A FREQUÊNCIA DAS NOTAS



FONTE: O autor (2025)

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

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

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

print(df.dtypes)

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

```

# 3.2 Conversão de dimensões

# Obter tamanho das listas de Titulos e ID_usuario únicos

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

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

> Número de usuários únicos: 777
> Número de livros únicos: 91312

# 4.1 Criação de camadas referentes ao usuario

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

# 4.2 Criação das camadas referentes ao Titulo

i = Input(shape=(1, ))
i_emb = Embedding(input_dim=M, output_dim=K)(i)
i_emb = Flatten()(i_emb)

# Junção dos conjuntos de camadas

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

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

# 5.1 Compilar o modelo
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')

```

5.2 Sumário do modelo

```
model.summary()
```

FIGURA 13.18 – RESUMO DAS CAMADAS DO MODELO

Model: "functional"			
Layer (type)	Output Shape	Param #	Connected to
input_layer (InputLayer)	(None, 1)	0	–
input_layer_1 (InputLayer)	(None, 1)	0	–
embedding (Embedding)	(None, 1, 50)	38,850	input_layer[0][0]
embedding_1 (Embedding)	(None, 1, 50)	4,565,600	input_layer_1[0][0]
flatten (Flatten)	(None, 50)	0	embedding[0][0]
flatten_1 (Flatten)	(None, 50)	0	embedding_1[0][0]
concatenate (Concatenate)	(None, 100)	0	flatten[0][0], flatten_1[0][0]
dense (Dense)	(None, 256)	25,856	concatenate[0][0]
dense_1 (Dense)	(None, 1)	257	dense[0][0]
Total params: 4,630,563 (17.66 MB)			
Trainable params: 4,630,563 (17.66 MB)			
Non-trainable params: 0 (0.00 B)			

FONTE: O autor (2025)

6 Separação dos dados e pré-processamento

```
from sklearn.model_selection import train_test_split
```

```
train, test = train_test_split(df, test_size=0.2, random_state=42)
```

```
print(train.shape)
```

```
print(test.shape)
```

```
> (79175, 9)
```

```
> (19794, 9)
```

Verificação das notas de um usuário (id 64)

```
print(df.loc[df['id_usuario_cat_codes'] == 64, ['Titulo', 'ID_usuario',  
'Notas']])
```

	Titulo	ID_usuario	Notas
17582	A Time To Dance	3282	8.0
17583	The Crossroad	3282	9.0
17584	The Postcard	3282	6.0

17585	Promise, The	3282	2.0
17586	Chronicles of Fairacre, Comprising Village Sch...	3282	9.0
17587	The Last Silk Dress	3282	2.0
17588	Imagination: Embracing a theology of wonder	3282	4.0
17589	The Second Bend in the River (Point)	3282	5.0
17590	The American Country Inn And Bed & Breakfast C...	3282	1.0
17591	The Berenstain Bears and Too Much Junk Food (F...	3282	10.0
17592	Big Sarah's Little Boots (A Blue Ribbon Book)	3282	10.0
17593	Shark Swimathon (MathStart 3)	3282	2.0
17594	Five Silly Fishermen (Step-Into-Reading, Step 2)	3282	6.0
17595	The Gym Teacher from the Black Lagoon	3282	0.0
17596	Chato's Kitchen	3282	4.0
17597	Feelings	3282	0.0
17598	Zora Hurston and the Chinaberry Tree (Reading ...	3282	5.0
17599	The Bunyans	3282	8.0
17600	The Family Under the Bridge	3282	9.0
17601	Amber Brown Goes Fourth (Amber Brown (Paperback))	3282	2.0
17602	Polar Bears Past Bedtime (Magic Tree House 12,...	3282	9.0
17603	Lily and Miss Liberty	3282	9.0
17604	Amber Brown Wants Extra Credit (Amber Brown (P...	3282	3.0
17605	Dinosaurs Before Dark (Magic Tree House #1)	3282	10.0
17606	The Wind in the Willows	3282	6.0
17607	The Wind in the Willows	3282	4.0
17608	Stepping on the Cracks	3282	6.0
17609	Frog and Toad All Year (I Can Read Book 2)	3282	9.0
17610	The Giraffe and the Pelly and Me	3282	9.0
17611	How To Survive Third Grade : How To Survive Th...	3282	8.0
17612	The Castle in the Attic	3282	3.0
17613	Runaway Pony (Pony Pals (Paperback))	3282	2.0
17614	365 Easy Low-Calorie Recipes	3282	5.0

7. Treinar o modelo

Normalização

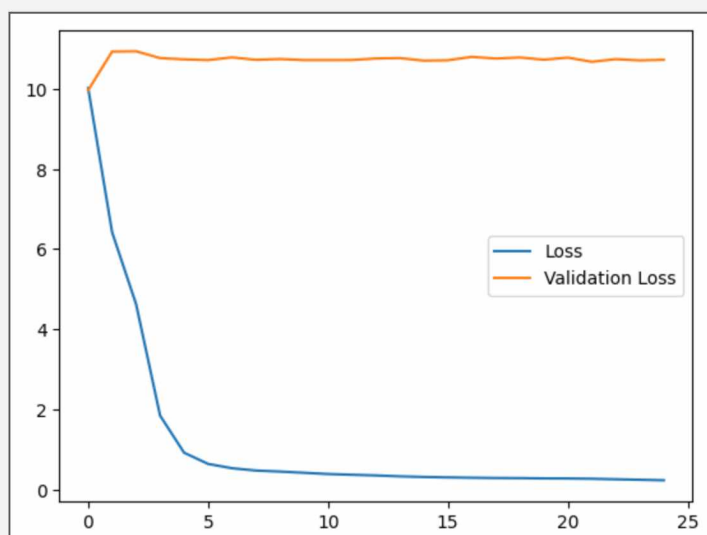
```
avg_notas = df.Notas.mean()
train_notas = train.Notas - avg_notas
test_notas = test.Notas - avg_notas
```

```
# Treinamento
r = model.fit(
    x=[train.id_usuario_cat_codes, train.titulo_cat_codes],
    y= train_notas,
    epochs=25,
    batch_size=256,
    verbose=2,
    validation_data=([test.id_usuario_cat_codes, test.titulo_cat_codes],
test_notas)

> Epoch 1/25
> 310/310 - 14s - 46ms/step - loss: 10.0177 - val_loss: 9.9655
> Epoch 2/25
> 310/310 - 14s - 47ms/step - loss: 6.4213 - val_loss: 10.9290
> ...
> Epoch 24/25
> 310/310 - 14s - 45ms/step - loss: 0.2496 - val_loss: 10.7093
> Epoch 25/25
> 310/310 - 21s - 68ms/step - loss: 0.2369 - val_loss: 10.7241

# 8. Plotar a função de perda
plt.plot(r.history['loss'], label='Loss')
plt.plot(r.history['val_loss'], label='Validation Loss')
plt.legend()
plt.show()
```

FIGURA 13.19 – GRÁFICO DE PERDA X ÉPOCA E PERDA DE VALIDAÇÃO X ÉPOCA



FONTE: O autor (2025)

```

# 9 Teste do sistema de recomendação
# 9.1 Gerar um array com usuário único

input_usuario = np.repeat(a=64, repeats=M) # 7346 -> 1120
books = np.array(list(set(df.titulo_cat_codes)))

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

> input_usuario:  [64 64 64 ... 64 64 64]
> books:  [    0     1     2 ... 91309 91310 91311]
> len input_user:  91312
> len books:  91312

# 9.2 Realiza a predição
preds = model.predict([input_usuario, books])

# 9.3 Tratamento da predição

notas_finais = preds.flatten() + avg_notas

max_idx = np.argmax(notas_finais)

result = df[df.titulo_cat_codes == books[max_idx]]

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

> Recomendação: Much Ado about Nothing por William Shakespeare. Nota:
11.199999809265137

```

Comentários

Com base no gráfico de perda, aparentemente pode estar ocorrendo overfitting (por volta da epoch 10, quando a função de perda dos dados de treino começa a cair, enquanto a função de perda dos dados de validação começa a subir). Foram realizados testes alterando diversos parâmetros, como o tamanho do embedding, batch size, learning rate, momentum, função de ativação e atributo utilizado

para o treino (título do livro em vez de ISBN). Em geral, o mesmo comportamento foi observado, sendo que, em alguns casos, foi necessário mais epochs para notar uma redução significativa na função de perda.

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

4 – Deepdream

1. Importação das bibliotecas

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

2. Importação da imagem

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

Download da imagem e gravação em array Numpy

```
def download(url, max_dim=None):
    name = url.split('/')[-1]
    image_path = tf.keras.utils.get_file(name, origin=url)
    img = PIL.Image.open(image_path)
    if max_dim:
        img.thumbnail((max_dim, max_dim))
    return np.array(img)
```

Normalização da imagem

```
def deprocess(img):
    img = 255*(img + 1.0)/2.0
    return tf.cast(img, tf.uint8)
```

Mostra a imagem

```
def show(img):
```

```
display.display(PIL.Image.fromarray(np.array(img)))

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

FIGURA 13.21 – IMAGEM BAIXADA PARA SER UTILIZADA NO DEEPPDREAM



FONTE: O autor (2025)

3. Preparar o modelo de extração de recursos

```
base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')

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

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


4. Cálculo da perda (loss)

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

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

    return tf.reduce_sum(losses)
```

5. Subida de gradiente (Gradient ascent)

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

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

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

```

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

        # Normalizacao dos gradintes
        gradients /= tf.math.reduce_std(gradients) + 1e-8

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

    return loss, img

deepdream = DeepDream(dream_model)

# 6. Circuito principal (Main Loop)

def run_deep_dream_simple(img, steps=100, step_size=0.01):

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

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

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

```

```

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

return result

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

```

FIGURA 13.21 – IMAGEM ALTERADA VIA DEEPDREAM



FONTE: O autor (2025)

```

# 7. Levando o modelo até um oitava

import time
start = time.time()

OCTAVE_SCALE = 1.30

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

```

```

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

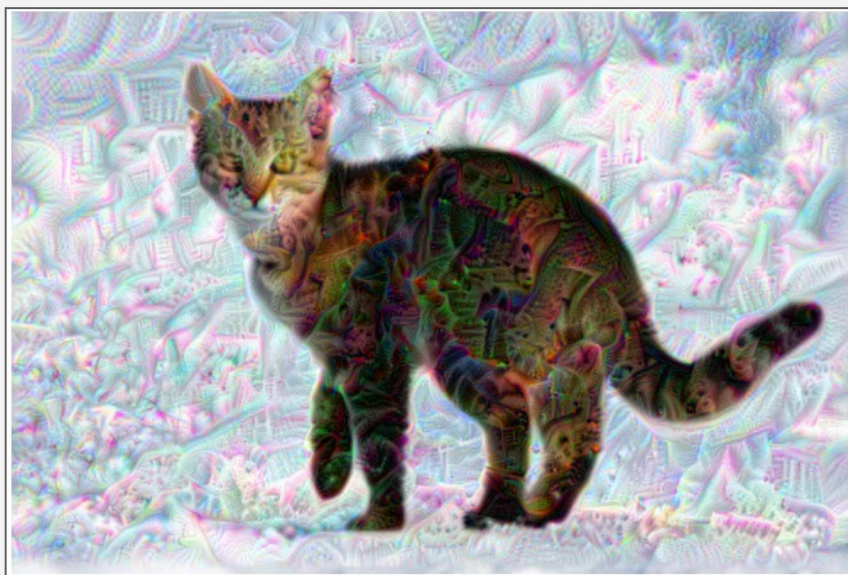
    img = tf.image.resize(img, new_shape).numpy()

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

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

```

FIGURA 13.22 – IMAGEM ALTERADA VIA DEEPPDREAM COM USO DE OITAVAS



FONTE: O autor (2025)

> 328.16870522499084

Imagem onírica obtida com Main Loop

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

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

Imagem onírica obtida ao levar o modelo até uma oitava

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

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

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

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

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

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

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

B – RESOLUÇÃO

1. CONJUNTO DE DADOS BRUTOS

FIGURA 14.1 – TABELA COM DADOS BRUTOS COLETADOS DURANTE TRÊS SEMANAS DE TESTES DO *CHATBOT* COM IA GENERATIVA

date	chat_start	interact_lead	interact_bot	chat_custom	success	obs
06/10/2025	14	42	55	31	2	
07/10/2025	25	78	112	68	3	
08/10/2025	29	78	109	94	2	deploy_fix
09/10/2025	28	83	121	99	3	
10/10/2025	31	94	118	56	3	deploy_rollback
13/01/2025	26	77	97	11	0	min_intervention
14/01/2025	28	81	104	13	1	min_intervention
15/01/2025	33	104	132	42	4	deploy_fix
16/01/2025	63	201	226	69	7	
17/01/2025	70	218	233	65	9	
20/01/2025	68	213	227	25	8	deploy_fix
21/01/2025	74	228	245	27	9	
22/01/2025	150	466	482	41	22	
23/01/2025	153	467	480	42	25	
24/01/2025	149	459	477	33	24	

FONTE: O autor (2025)

2. CONTEXTO E PÚBLICO ALVO

Uma determinada empresa de marketing e tecnologia - aqui nomeada como Empresa XYZ - está desenvolvendo uma solução de chatbot que utiliza um modelo de Inteligência Artificial - IA. A ideia da solução é que ela possa ser vendida para qualquer tipo de negócio que precise de um canal de contato com o consumidor final e/ou possíveis leads para a empresa. Como o chatbot utiliza modelos de IA previamente treinados, existe um trabalho de treinar e contextualizar o modelo para cada negócio que ele será “plugado”.

Como essa é a primeira iniciativa da Empresa XYZ nesse segmento e na utilização de um modelo de IA foi decidido utilizar esse chatbot por pelo menos três semanas em conjunto com um parceiro - uma empresa B2B que vende planos de consulta de crédito - em um modo que foi denominado assistido: ao invés do chatbot operar de forma completamente autônoma, uma Pessoa

Atendente - PA treinada pelo parceiro esteve acompanhando em tempo integral as mensagens enviadas pelo lead e as sugestões de respostas geradas pelo chatbot, através de uma interface desenvolvida especialmente para esse fim, e, estando a resposta de acordo, o PA a enviava. Caso a resposta não estivesse de acordo ou se por algum tipo de falha não fosse gerada uma resposta, o PA entrevistou customizando a sugestão gerada ou então escrevendo uma resposta do zero.

Com este parceiro, o objetivo do chatbot era apresentar os planos de consulta de crédito disponíveis, esclarecer quaisquer dúvidas com o lead e, por fim, realizar a venda do plano.

Para validar, junto à liderança da Empresa XYZ, a viabilidade e o valor entregue pela solução de chatbot com IA, foi necessário coletar dados que indicassem o quão assertivo nas conversas o chatbot é além de indicar o quanto de leads ele consegue converter. Para isso foi criada uma solução de tracking na qual determinadas ações que ocorrem na aplicação servem como gatilho para enviar eventos para um banco de dados para posterior análise. Os dados dos eventos coletados durante as três semanas de teste (excluindo finais de semana) podem ser vistos na Tabela 1. A seguir uma descrição do que cada um dos tipos de eventos representa:

- chat_start: Início de uma conversa, ou seja, quando um lead entra em contato com o chatbot
- Interact_lead: Mensagens enviadas pelo lead, ou seja, sempre que o lead envia uma mensagem para o chatbot, é enviado um evento deste tipo
- Interact_bot: Semelhante ao interact_lead mas agora para o chatbot, ou seja, sempre que uma mensagem é enviada para o lead (independente de ser uma mensagem gerada pelo chatbot ou escrita/customizada pelo PA) é enviado um evento deste tipo
- chat_custom: Toda sugestão de resposta gerada pelo chatbot que é alterada pelo PA ou qualquer mensagem escrita do zero pelo PA dispara um evento deste tipo
- success: Evento disparado toda vez que um lead converte, ou seja, quando compra um dos planos de consulta de crédito oferecidos pela empresa parceira durante a conversa com o chatbot

O objetivo é que, através desses dados, seja possível extrair pelo menos duas principais métricas que visam indicar se o chatbot em questão é um produto viável. Essas métricas são:

- Assertividade: O quanto as respostas geradas pelo chatbot são boas e estão de acordo com o contexto da conversa, ou seja, sugestões de resposta que não foram alteradas pelo PA
- Taxa de conversão: Porcentagem de leads convertidos

Além dos eventos disparados conforme determinadas ações ocorriam durante o ciclo de vida do chatbot, há também a coluna “obs” que diz respeito a situações excepcionais que ocorreram em determinados dias. Essas situações podem ser ações realizadas ou então políticas de uso do chatbot que foram empregadas durante tal dia. Com base na tabela, existem três situações distintas, sendo elas:

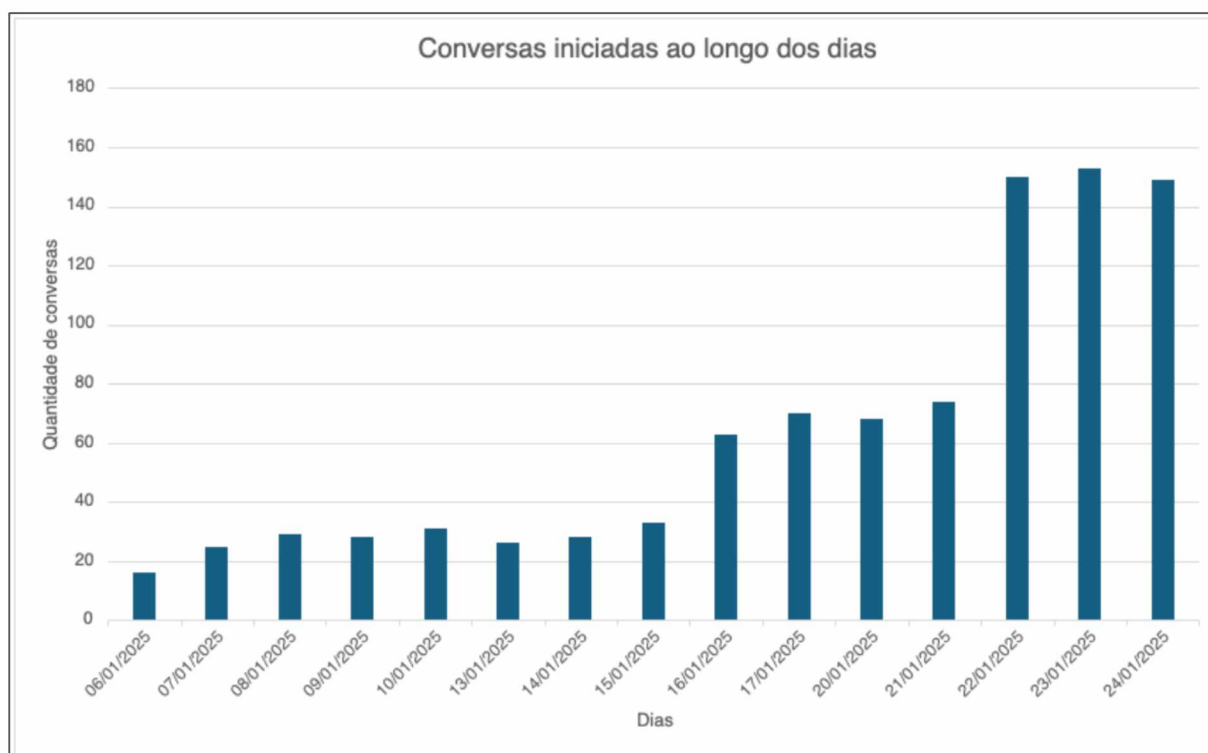
- `deploy_fix`: Alguma atualização desenvolvida na solução para corrigir algum problema ou então para melhorar algum comportamento visando aumentar assertividade e/ou conversão
- `deploy_rollback`: Retorno para uma versão anterior à alguma atualização que se mostrou problemática em algum sentido (introdução de bugs ou piora da performance do chatbot)
- `min_intervention`: Política para forçar a redução de intervenção do PA nas respostas sugeridas pelo chatbot, mesmo que isso pudesse resultar em redução de conversão

O público alvo dessa análise são líderes e diretores da Empresa XYZ que precisam desses insumos para avaliar se o faz sentido continuar o desenvolvimento do produto e oferecê-lo a potenciais clientes ou então se deve ser encerrado.

3. VISUALIZAÇÃO DOS DADOS

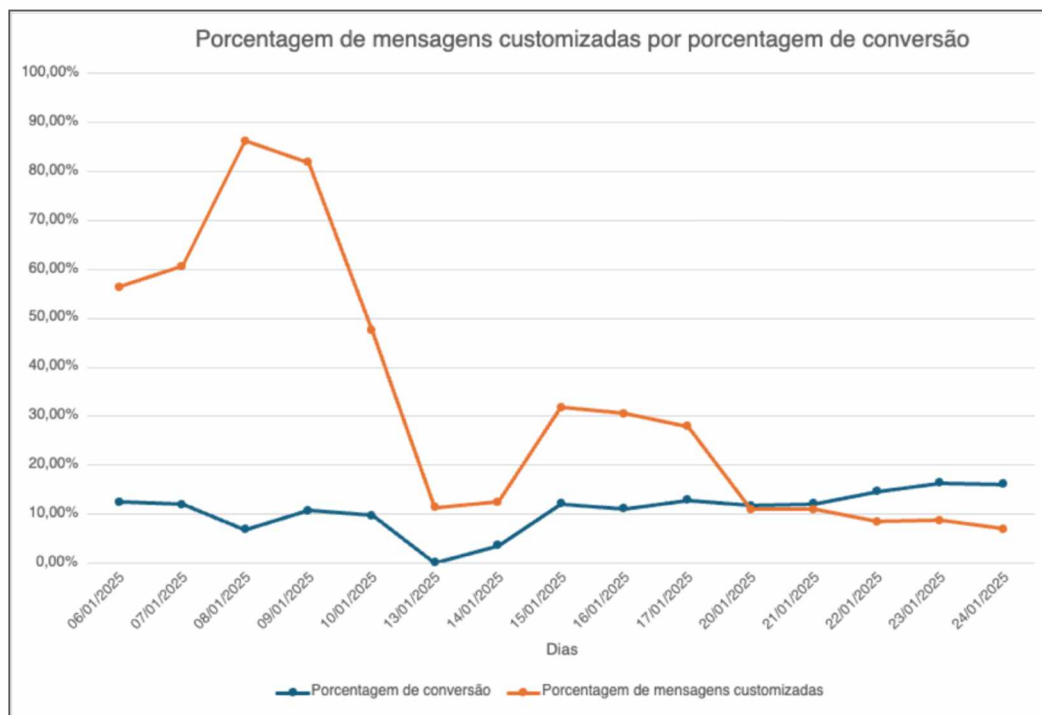
Os gráficos utilizados aqui para visualização dos dados foram feitos utilizando o Microsoft Excel, pois é uma ferramenta simples de utilizar e que apresenta as visualizações necessárias para explicar os dados obtidos.

FIGURA 14.2 – GRÁFICO DA QUANTIDADE DE CONVERSAS INICIADAS AO LONGO DOS DIAS



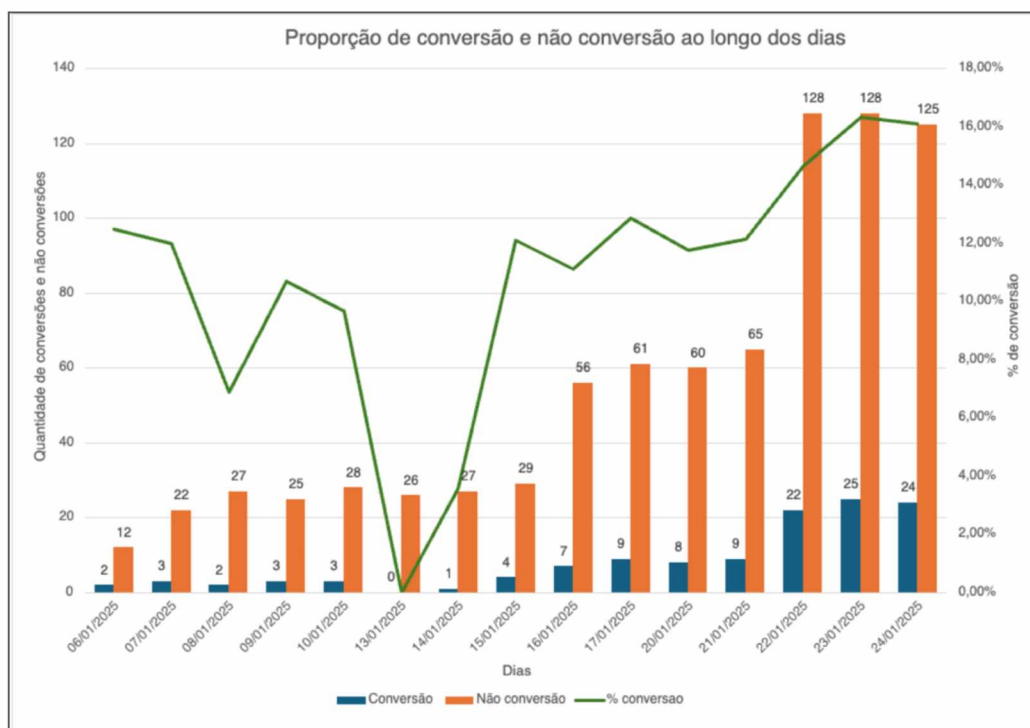
FONTE: O autor (2025)

FIGURA 14.3 – GRÁFICO DA RELAÇÃO ENTRE PORCENTAGEM DE MENSAGENS CUSTOMIZADAS E PORCENTAGEM DE CONVERSAS AO LONGO DOS DIAS



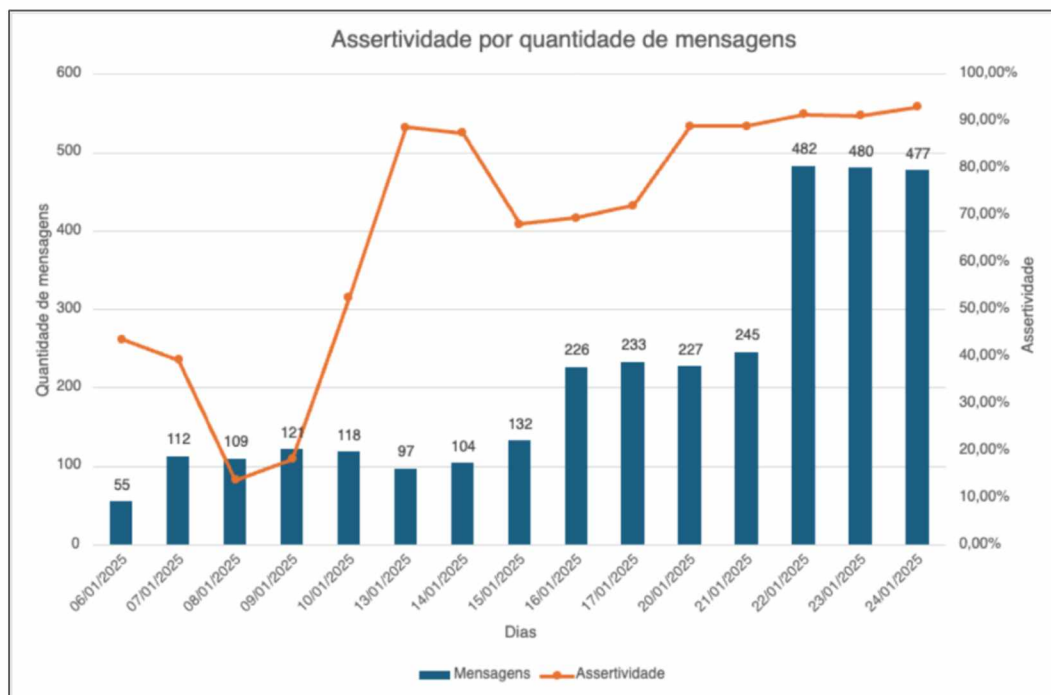
FONTE: O autor (2025)

FIGURA 14.4 – GRÁFICO DA PROPORÇÃO DE CONVERSAS QUE RESULTARAM EM CONVERSÃO E NÃO CONVERSÃO AO LONGO DOS DIAS



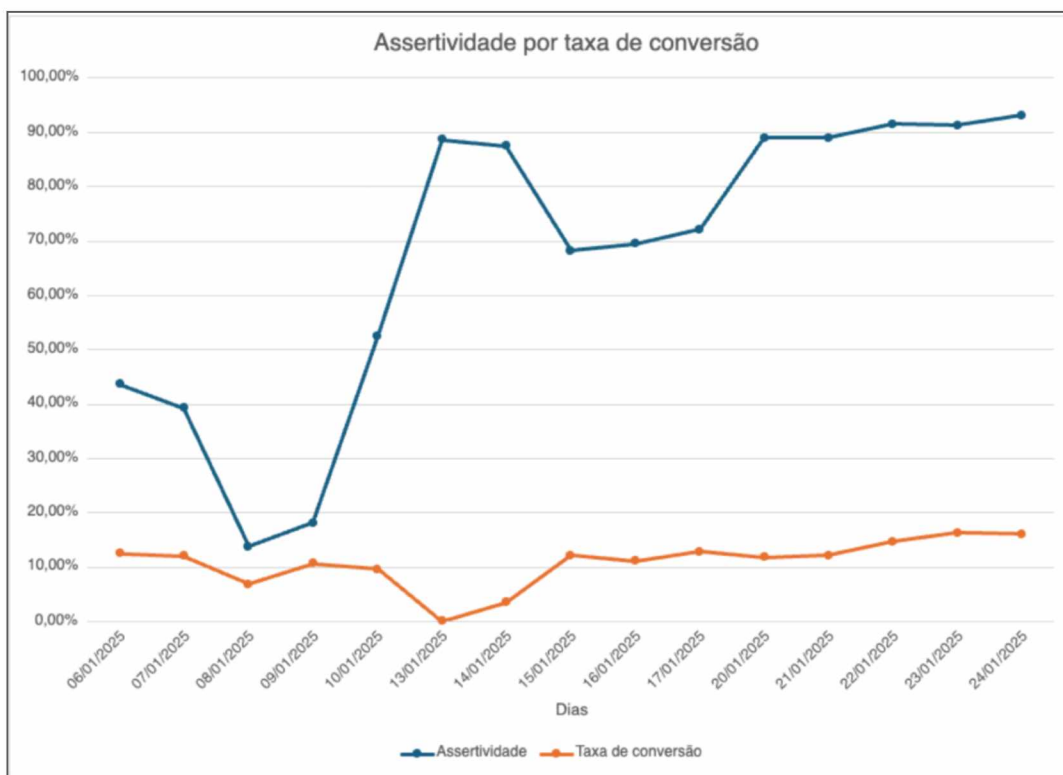
FONTE: O autor (2025)

FIGURA 14.5 – GRÁFICO DA RELAÇÃO DA ASSERTIVIDADE POR QUANTIDADE DE MENSAGENS AO LONGO DOS DIAS



FONTE: O autor (2025)

FIGURA 14.6 – GRÁFICO DA ASSERTIVIDADE POR TAXA DE CONVERSÃO AO LONGO DOS DIAS



FONTE: O autor (2025)

4. NARRATIVA

Através dos gráficos apresentados é possível entender como o teste do chatbot aconteceu ao longo das três semanas analisadas, bem como a forma como time de negócios atuou para aumentar as amostras de testes e o time de desenvolvimento trabalhou para corrigir problemas e melhorar as principais métricas estabelecidas (assertividade e conversão). Antes de iniciar a explicação, é importante ter em vista a motivação do teste e a forma ideal como o chatbot será utilizado e alguns valores considerados como “chaves” para tomada de decisão:

- Durante o teste o chatbot atuou como um “copiloto” do PA gerando respostas seguindo o contexto da conversa, reduzindo o esforço do PA em ter que fazer a gestão de múltiplas conversas simultâneas. Porém, idealmente o chatbot atuará de forma completamente autônoma, ou seja, sem intervenção humana, ou pelo menos com o mínimo de intervenção (por exemplo para confirmar a conclusão de uma venda)

- Os valores chaves para que o chatbot passe a ser considerado positivamente em uma tomada de decisão (ou seja, para que ele passe a ser considerado como um projeto minimamente viável) são que a taxa de conversão seja no mínimo igual a taxa de conversão de um humano que está na média de 5% e que a assertividade seja acima de 85%.

A primeira observação que pode ser feita a partir dos dados do Gráfico 1 - Quantidade de conversas iniciadas ao longo dos dias - é o aumento da quantidade de conversas iniciadas em momentos específicos. Primeiramente, a explicação sobre a diferença significativa da quantidade de conversas é que por se tratar de um teste sendo realizado com potenciais clientes, optou-se por direcionar apenas uma porcentagem dos clientes que chegavam nos canais de atendimento e venda dos planos, visto que o teste poderia comprometer a conversão geral da empresa parceira e até mesmo causar prejuízo a sua imagem. Outro fato que merece explicação são os dois aumentos significativos que ocorreram nos dias 16/01/2025 e 22/01/2025, sendo que conforme os testes foram ocorrendo o time de negócios concluiu que havia confiabilidade suficiente no chatbot para justificar um aumento na quantidade de leads e, conseqüentemente, um aumento nas amostras para avaliar a performance.

O Gráfico 2 - Relação entre porcentagem de mensagens customizadas e porcentagem de conversão ao longo dos dias - traz algumas informações muito importantes sobre a performance do chatbot, sobre a atuação do time de desenvolvimento para corrigir problemas e melhorar a qualidade através da análise das intervenções do PA com relação à conversão do chatbot. Antes de aprofundar a análise é necessário jogar luz ao fato de que a taxa de conversão da solução é muito boa (considerando o target de 5%). Inicialmente nota-se uma alta taxa de intervenção do PA (entre 40% e 50%) e uma boa conversão. Nos dias 08/01/2025 e 09/01/2025 há um aumento abrupto na taxa de intervenção do PA que coincide com um deploy realizado pelo time de desenvolvimento com a finalidade de melhorar as respostas do chatbot. Como pode ser visto, o deploy acabou causando um efeito contrário. Após dois dias (no dia 10/01/2025) é realizado um rollback voltando para a versão em que os testes foram iniciados. Importante notar que, embora houve uma flutuação na taxa de

conversão, ainda assim se manteve relativamente estável e em um bom patamar. Nos dias 13/01/2025 e 14/01/2025 a política de intervenção mínima (min_intervention) foi aplicada com a finalidade de validar se as intervenções realizadas pelo PA realmente impactam na conversão ou não. Nesse período foi estabelecido que o PA deveria atuar apenas em casos de erros extremos do chatbot. Nota-se aqui que a conversão, de fato, caiu chegando até mesmo em 0% no dia 13/01/2025, indicando que há espaço para melhoria do chatbot. No dia 15/01/2025 é removida a política de intervenção mínima e também é realizado um deploy que traz melhorias significativas no comportamento do chatbot. Essas duas alterações podem ser notadas pelo aumento de intervenções do PA (remoção da política de intervenção mínima), porém em uma taxa cerca de 50% menor do que estava no início dos testes, mantendo a conversão nos mesmos patamares. Por fim, no dia 20/01/2025 é realizado mais um deploy de melhoria do comportamento do chatbot que resulta em uma redução significativa da intervenção do PA e uma leve melhoria na conversão.

O Gráfico 3 - Proporção de conversas que resultaram em conversão e não conversão ao longo dos dias - mostra a proporção de conversão e não conversão do lead ao longo dos dias. Esse gráfico possui três pontos de interesse: O primeiro deles é a queda na taxa de conversão nos dias 08/01/2025 e 09/01/2025, coincidindo com o deploy realizado que acabou introduzindo algum possível bug na aplicação. O segundo ponto é entre os dias 13/01/2025 e 14/01/2025 em que foi introduzida a política de intervenção mínima para o PA. Nesses dias também é visível a queda de conversão, chegando a 0%, provando que, no estado em que chatbot se encontrava, a intervenção do PA era essencial para manter uma conversão aceitável e que, infelizmente, o chatbot não era um projeto minimamente viável. O terceiro ponto acontece no dia 21/01/2025, dia que sucede o último deploy de melhorias realizado durante os testes. Aqui é possível ver um aumento na taxa de conversão, saindo de uma faixa de 10% - 12% para 14% - 16%.

O Gráfico 4 - Relação da assertividade por quantidade de mensagens ao longo dos dias mostra a evolução da assertividade do chatbot ao longo dos testes após manutenção da aplicação por parte do time de desenvolvimento. Esse gráfico possui cinco pontos de interesse: O primeiro é logo nos dois primeiros dias do testes em que assertividade está entre 40% e 50% indicando, portanto, que há necessidade de melhoria para que a solução se torne um produto viável. O segundo ponto está entre os dias 08/01/2025 e 09/01/2025 em que foi realizado um deploy visando melhorar o chatbot mas que acabou causando o efeito contrário. O terceiro ponto está entre os dias 13/01/2025 e 14/01/2025 em que a assertividade deu um salto muito grande estando na faixa dos 90%. Esse ponto é muito importante observar com atenção, pois se não considerar o todo, é fácil se equivocar e achar que o chatbot está com uma performance excelente, porém é preciso lembrar que a assertividade é a taxa de mensagens enviadas sem intervenção do PA e que nesses dias em questão a política de intervenção mínima estava em vigor. Logo ao fazer a correlação com o Gráfico 2 fica claro que esse aumento na assertividade é um aumento “artificial” e que teve como consequência a redução da taxa de conversão. O quarto ponto está entre os dias 15/01/2025 e 17/01/2025 em que a taxa de assertividade voltou a ser uma indicação real, mas que passou da faixa dos 50% para 70% após um deploy de correção e melhoria. O quinto e último ponto está no dia 20/01/2025 em diante, em que houve o último deploy no

período de testes e que resultou em um aumento significativo da assertividade, subindo para a faixa de 90% dessa vez de forma “orgânica” sem perder a taxa de conversão.

Por fim, o Gráfico 5 - Assertividade por taxa de conversão ao longo dos dias - pode ser considerado como um compilado/resumo das principais métricas avaliadas no teste - assertividade e taxa de conversão - apresentando como esses valores evoluíram ao longo dos testes sob o efeito de manutenções e melhorias realizadas pelo time de desenvolvimento. Nesse gráfico é possível destacar quatro pontos. O primeiro é a queda de assertividade e uma pequena flutuação na conversão que ocorreram entre os dias 08/01/2025 e 09/01/2025 como resultado de um deploy que acabou introduzindo problemas na aplicação. O segundo ponto foi nos dias 13/01/2025 e 14/01/2025 em que uma assertividade alta aparece como consequência da política de intervenção mínima do PA, mas que junto com ela vem uma redução significativa da taxa de conversão, deixando claro que o chatbot naquele estado estava inviável de entrar em operação de forma autônoma. O terceiro ponto é entre os dias 15/01/2025 e 17/01/2025 em que foi realizado um deploy de correções e melhorias e a política de intervenção mínima foi removida. Nesse ponto, a taxa de conversão voltou para a faixa de 10% e o valor de assertividade subiu em relação ao valor do início do teste, passando de cerca de 50% para cerca de 70%. O último ponto foi do dia 20/01/2025 em diante em que foi realizado mais um deploy de correções e melhorias e que fez com que o valor de assertividade saltasse para a faixa de 90% e também causando um leve aumento na conversão de mais ou menos 12% para algo em torno de 16%.

Em conclusão, observa-se pelos gráficos que o chatbot se provou uma aplicação viável e que pode ser oferecida como um produto para outras empresas, sendo talvez apenas necessário realizar um período de testes semelhante ao que foi feito aqui para que seja feita a adaptação à realidade do negócio de cada potencial cliente.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

2) Compare a representação de dois modelos vetoriais

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

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

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

B – RESOLUÇÃO

1 - Algoritmo genético

[illegible]


```

    distancia += np.linalg.norm(cidades[percurso[-1]] -
                                cidades[percurso[0]]) # Retorno à cidade inicial
    return distancia

# Inicializar população aleatória
def inicializar_populacao():
    populacao = []
    for _ in range(POPULACAO_SIZE):
        percurso = list(range(NUM_CIDADES))
        random.shuffle(percurso)
        população.append(percurso)
    return populacao

# Função de seleção por torneio
def selecao(populacao):
    candidatos = random.sample(populacao, 5)
    return min(candidatos, key=calcular_distancia)

# Operador de crossover OX (Order Crossover)
def crossover(pai1, pai2):
    tamanho = len(pai1)
    inicio, fim = sorted(random.sample(range(tamanho), 2))
    filho = [-1] * tamanho
    filho[inicio:fim] = pai1[inicio:fim]
    ptr = fim
    for gene in pai2:
        if gene not in filho:
            if ptr >= tamanho:
                ptr = 0
            filho[ptr] = gene
            ptr += 1
    return filho

# Operador de mutação (swap entre duas cidades)
def mutacao(percurso):
    if random.random() < MUTACAO_RATE:
        i, j = random.sample(range(len(percurso)), 2)
        percurso[i], percurso[j] = percurso[j], percurso[i]
    return percurso

```

```

# Algoritmo Genético
def algoritmo_genetico():
    populacao = inicializar_populacao()
    melhor_percurso = min(populacao, key=calcular_distancia)
    melhor_distancia = calcular_distancia(melhor_percurso)

    for _ in range(GERACOES):
        nova_populacao = []
        for _ in range(int(POPULACAO_SIZE * CROSSOVER_RATE)):
            pai1, pai2 = selecao(populacao), selecao(populacao)
            filho = crossover(pai1, pai2)
            filho = mutacao(filho)
            nova_populacao.append(filho)
        while len(nova_populacao) < POPULACAO_SIZE:
            nova_populacao.append(selecao(populacao))
        populacao = nova_populacao
        melhor_atual = min(populacao, key=calcular_distancia)
        melhor_atual_distancia = calcular_distancia(melhor_atual)

        if melhor_atual_distancia < melhor_distancia:
            melhor_percurso, melhor_distancia = melhor_atual,
melhor_atual_distancia
    return melhor_percurso, melhor_distancia

# Executar o algoritmo
def plotar_percurso(percurso, titulo):
    plt.figure(figsize=(8, 8))
    caminho = cidades[percurso + [percurso[0]]]
    plt.plot(caminho[:, 0], caminho[:, 1], 'bo-')
    plt.title(titulo)
    plt.show()

# Primeira solução aleatória
populacao = inicializar_populacao()
solucao_inicial = populacao[0]
distancia_inicial = calcular_distancia(solucao_inicial)
plotar_percurso(solucao_inicial, f'Solução Inicial - Distância:
{distancia_inicial:.2f}')

```

```

# Melhor solução após evolução
melhor_percurso, melhor_distancia = algoritmo_genetico()
plotar_percurso(melhor_percurso, f'Melhor Solução - Distância:
{melhor_distancia:.2f}')

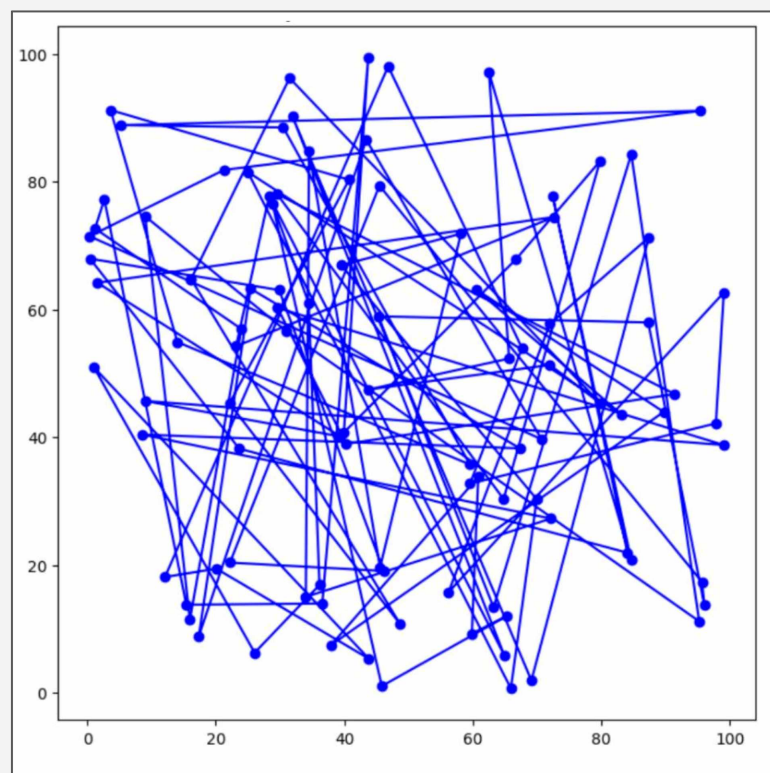
# Aplicação da PCA em modelos vetoriais de um texto
def aplicar_pca(modelo1, modelo2):
    dados = np.vstack((modelo1, modelo2))
    pca = PCA(n_components=2)
    resultado_pca = pca.fit_transform(dados)

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

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

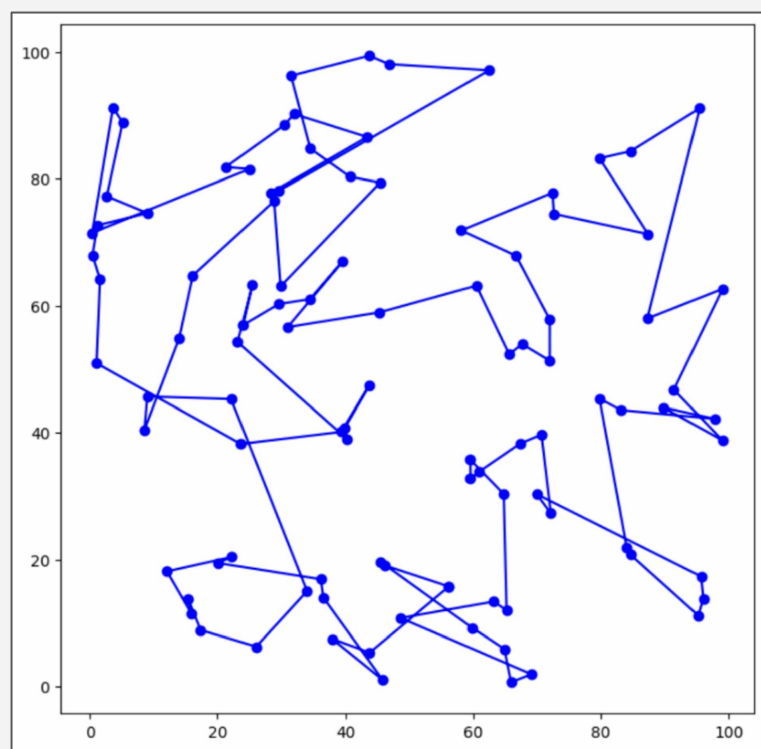
```

FIGURA 15.1 – SOLUÇÃO INICIAL DO ALGORITMO – DISTÂNCIA 1900.96



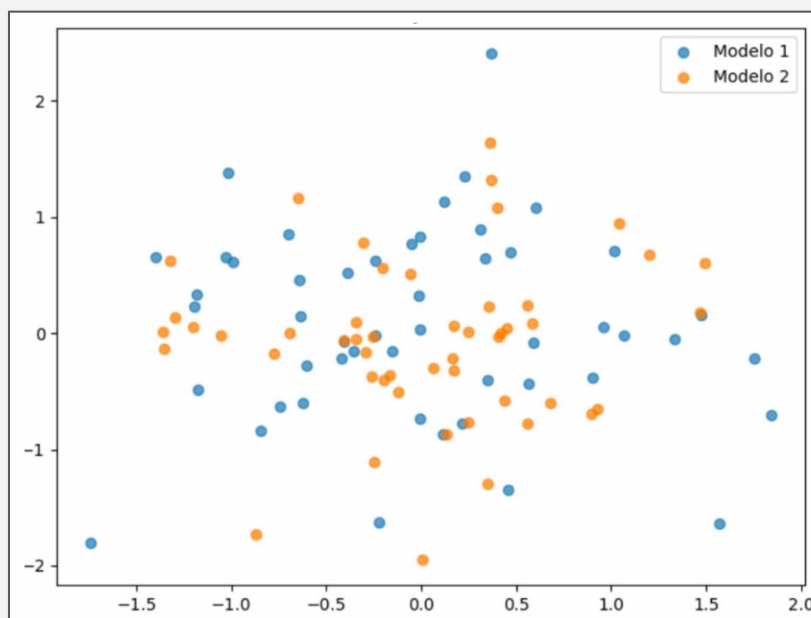
FONTE: O autor (2025)

FIGURA 15.2 – MELHOR SOLUÇÃO – DISTÂNCIA 1090.36



FONTE: O autor (2025)

FIGURA 15.3 – VISUALIZAÇÃO COM PCA



FONTE: O autor (2025)

2 – Compare a representação de dois modelos vetoriais

```
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.feature_extraction.text import TfidfVectorizer

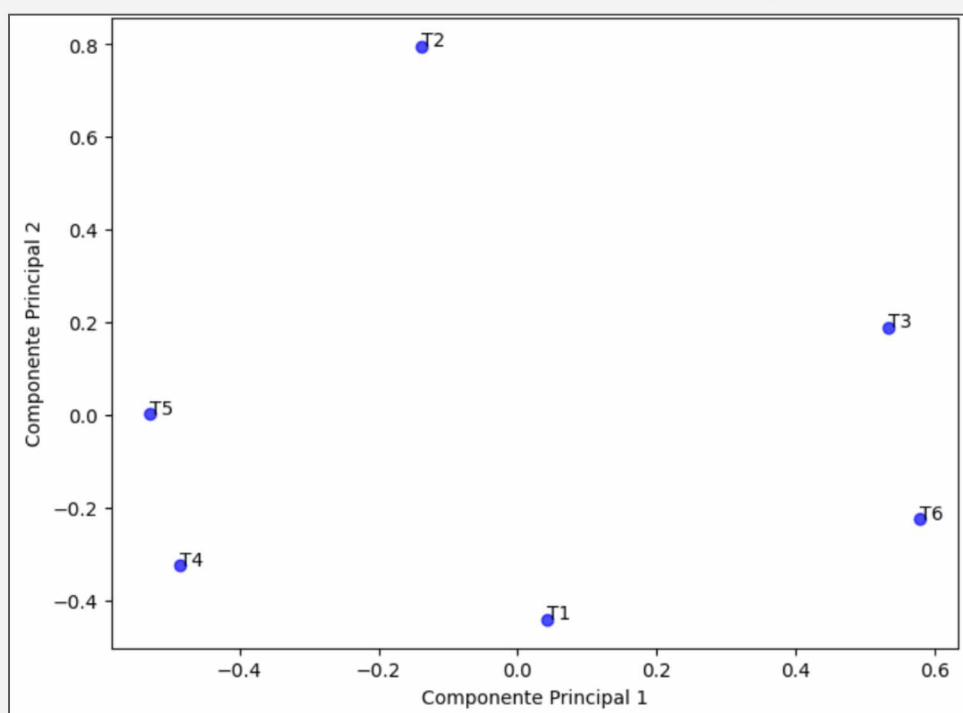
# Aplicação da PCA em representações vetoriais de textos
textos = [
    "O cachorro correu pelo parque e brincou com a bola.",
    "O gato dormiu no sofá durante a tarde inteira.",
    "As crianças brincaram no parque e correram felizes.",
    "O leão é um animal selvagem que vive na savana.",
    "O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

# Converter textos para vetores usando TF-IDF
vectorizer = TfidfVectorizer()
vetores_texto = vectorizer.fit_transform(textos).toarray()

# Aplicar PCA
pca = PCA(n_components=2)
resultado_pca = pca.fit_transform(vetores_texto)
```

```
# Plotar os vetores projetados
plt.figure(figsize=(8, 6))
plt.scatter(resultado_pca[:, 0], resultado_pca[:, 1], color='blue',
            alpha=0.7)
for i, txt in enumerate(textos):
    plt.annotate(f'T{i+1}', (resultado_pca[i, 0], resultado_pca[i, 1]))
plt.title("Visualização com PCA de Representações Textuais")
plt.xlabel("Componente Principal 1")
plt.ylabel("Componente Principal 2")
plt.show()
```

FIGURA 15.4 – VISUALIZAÇÃO COM PCA DE REPRESENTAÇÕES TEXTUAIS



FONTE: O autor (2025)