

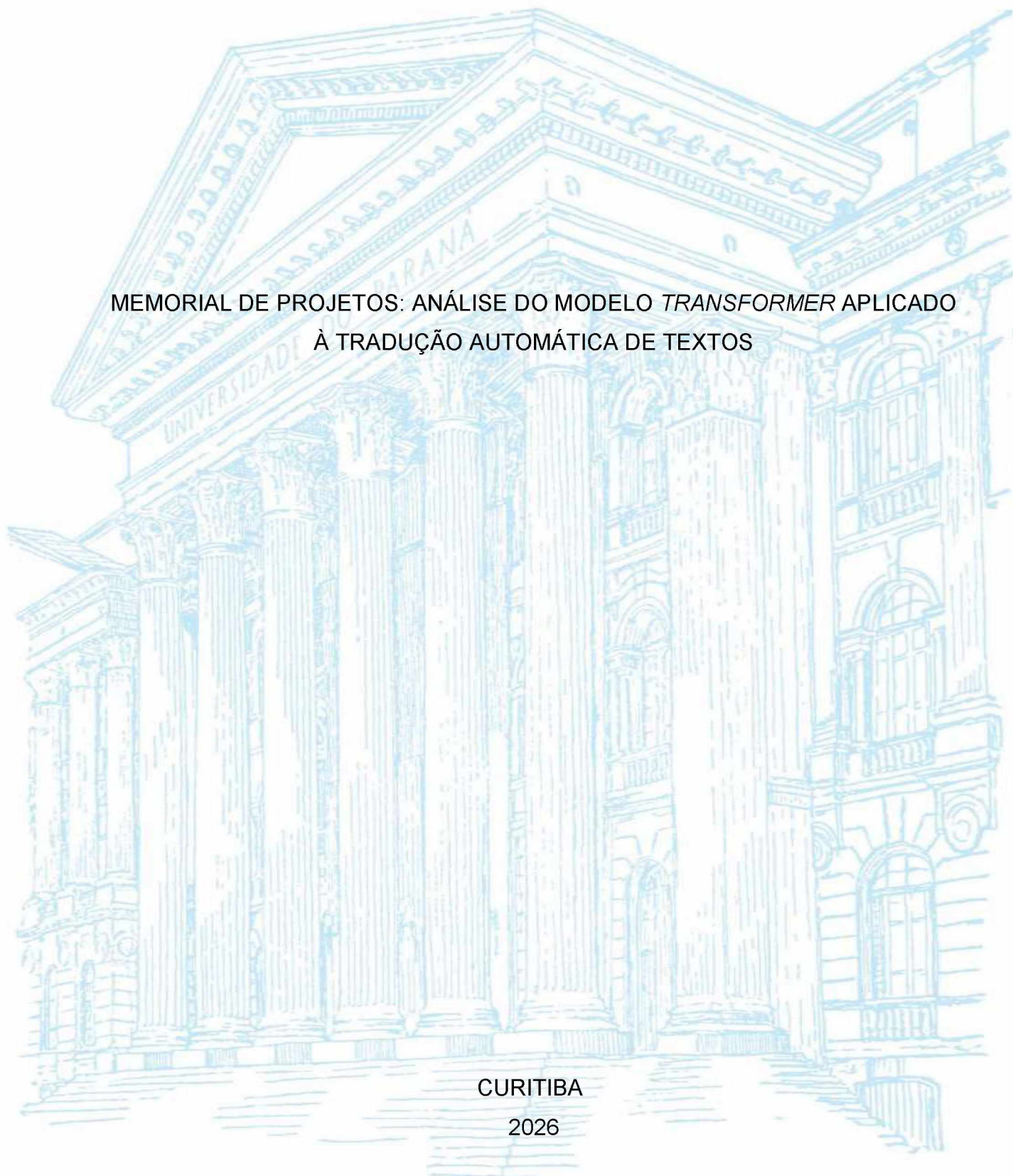
UNIVERSIDADE FEDERAL DO PARANÁ

AMAURI LEONEL KLAMAS JUNIOR

MEMORIAL DE PROJETOS: ANÁLISE DO MODELO *TRANSFORMER* APLICADO
À TRADUÇÃO AUTOMÁTICA DE TEXTOS

CURITIBA

2026



AMAURI LEONEL KLAMAS JUNIOR

MEMORIAL DE PROJETOS: ANÁLISE DO MODELO *TRANSFORMER* APLICADO
À TRADUÇÃO AUTOMÁTICA DE TEXTOS

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

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaña

CURITIBA

2026

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 **AMAURI LEONEL KLAMAS JUNIOR**, intitulada: **MEMORIAL DE PROJETOS: ANÁLISE DO MODELO TRANSFORMER APLICADO À TRADUÇÃO AUTOMÁTICA DE TEXTOS**, 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, 25 de Fevereiro de 2026.



RAZER ANTHON NIERWOJ S MONTANO
Presidente da Banca Examinadora



JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Neste trabalho apresenta-se um memorial técnico sobre a arquitetura de aprendizado de máquina conhecida como *Transformers*, considerada um dos avanços mais significativos no campo do Aprendizado Profundo nas últimas décadas. Proposta por Vaswani no artigo *Attention Is All You Need*, essa arquitetura representou uma mudança estrutural na forma como modelos computacionais processam sequências de dados. Diferentemente das abordagens tradicionais baseadas em recorrência, como as *Redes Neurais Recorrentes* (RNNs) e suas variações *Long Short-Term Memory* (LSTM) e *Gated Recurrent Units* (GRU), o *Transformer* fundamenta-se exclusivamente em mecanismos de atenção, eliminando a dependência sequencial e permitindo processamento paralelo dos dados. Essa inovação tornou possível maior eficiência computacional e melhor escalabilidade em cenários de grande volume de informações, além de mitigar problemas clássicos como o desaparecimento e explosão do gradiente em sequências longas. O mecanismo de autoatenção (*self-attention*) possibilita que cada elemento de uma sequência estabeleça relações diretas com todos os demais, capturando dependências de longo alcance e modelando contextos complexos de forma global. A introdução da codificação posicional permitiu preservar informações de ordem sem o uso de estruturas recorrentes, mantendo a integridade semântica das sequências. O impacto inicial dessa arquitetura foi observado no *Processamento de Linguagem Natural* (PLN), especialmente em tarefas como tradução automática, sumarização e geração de texto, mas sua aplicação rapidamente se expandiu para áreas como visão computacional, reconhecimento de fala, bioinformática e modelos multimodais. Atualmente, os *Transformers* constituem a base dos principais modelos de linguagem de larga escala e sistemas generativos contemporâneos. O objetivo deste estudo é analisar os fundamentos teóricos e matemáticos do Transformer, compreender as motivações que levaram ao seu desenvolvimento e examinar suas contribuições em relação aos modelos anteriores, bem como discutir os impactos científicos e tecnológicos decorrentes de sua ampla adoção na inteligência artificial moderna.

Palavras-chave: transformador; aprendizado profundo; aprendizado de máquina; redes neurais recorrentes; mecanismo de atenção.

ABSTRACT

This work presents a technical report on the machine learning architecture known as *Transformers*, considered one of the most significant advances in the field of Deep Learning in recent decades. Proposed by Vaswani in the article *Attention Is All You Need*, this architecture represented a structural shift in the way computational models process sequential data. Unlike traditional recurrence-based approaches, such as *Recurrent Neural Networks* (RNNs) and their *Long Short-Term Memory* (LSTM) and *Gated Recurrent Units* (GRU) variants, the *Transformer* is built entirely upon attention mechanisms, eliminating sequential dependency and enabling fully parallel data processing. This innovation allowed for greater computational efficiency and improved scalability in large-scale data scenarios, while mitigating classical issues such as vanishing and exploding gradients in long sequences. The *self-attention* mechanism enables each element of a sequence to establish direct relationships with all other elements, capturing long-range dependencies and modeling complex contextual interactions in a global manner. The introduction of positional encoding made it possible to preserve order information without relying on recurrent structures, maintaining the semantic integrity of sequences. The initial impact of this architecture was observed in *Natural Language Processing* (NLP), particularly in tasks such as machine translation, summarization, and text generation, but its application quickly expanded to areas including computer vision, speech recognition, bioinformatics, and multimodal models. Currently, *Transformers* form the foundation of state-of-the-art large-scale language models and contemporary generative systems. The objective of this study is to analyze the theoretical and mathematical foundations of the *Transformer*, understand the motivations behind its development, examine its contributions in comparison to previous models, and discuss the scientific and technological impacts resulting from its widespread adoption in modern artificial intelligence.

Keywords: transformers; deep learning; machine learning; recurrent neural networks; attention.

SUMÁRIO

1	PARECER TÉCNICO.....	6
2	FUNDAMENTAÇÃO TEÓRICA.....	7
2.1	ARQUITETURA MATEMÁTICA DO TRANSFORMER.....	7
2.2	CODIFICAÇÃO POSICIONAL.....	8
2.3	CUSTO COMPUTACIONAL.....	8
3	METODOLOGIA.....	9
4	RESULTADOS.....	9
5	CONCLUSÃO.....	10
	REFERÊNCIAS.....	11
	APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA.....	13
	APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA.....	21
	APÊNDICE C - LINGUAGEM R.....	39
	APÊNDICE D - ESTATÍSTICA APLICADA I.....	50
	APÊNDICE E - ESTATÍSTICA APLICADA II.....	59
	APÊNDICE F - ARQUITETURA DE DADOS.....	70
	APÊNDICE G - APRENDIZADO DE MÁQUINA.....	88
	APÊNDICE H - DEEP LEARNING.....	114
	APÊNDICE I - BIG DATA.....	138
	APÊNDICE J - VISÃO COMPUTACIONAL.....	142
	APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA.....	152
	APÊNDICE L - GESTÃO DE PROJETOS DE IA.....	162
	APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL.....	164
	APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING.....	188
	APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL.....	197

1 PARECER TÉCNICO

A introdução da arquitetura Transformer, proposta por *Vaswani et al.* (2017) no artigo *Attention Is All You Need*, representou uma mudança estrutural na forma de processar dados sequenciais. Diferentemente das Redes Neurais Recorrentes (RNNs) e de suas variações, como as *Long Short-Term Memory (LSTM)* e as *Gated Recurrent Units (GRU)* (*Vaswani et al.*, 2017), o *Transformer* é baseado inteiramente em mecanismos de atenção, eliminando a dependência sequencial entre os elementos de entrada.

Antes do surgimento dessa arquitetura, os modelos recorrentes apresentavam sérias limitações. A dependência temporal impedia a paralelização completa do treinamento, o que tornava o processo lento. Além disso, em sequências longas, havia a perda de gradiente — o fenômeno de *vanishing* e *exploding gradients*, no qual a informação se degradava à medida que a rede processava os passos da sequência.

Algumas limitações são a paralelização impossível, pois cada passo depende do anterior, o que faz o treinamento do modelo ser lento. Conhecidas como *Long-Term Dependencies*, onde quanto mais longa a sequência, mais difícil manter informação relevante. Os gradientes desaparecem ou explodem, ou seja, matematicamente o produto das derivadas tende a zero ou infinito (*Vaswani et al.*, 2017).

Por conta destas limitações os modelos anteriores acabavam priorizando informações recentes com um processamento lento e custoso. Sendo necessário encontrar uma forma para que os modelos conseguissem processar os dados sem que informações recentes tivessem uma relevância muito maior que as treinadas anteriormente.

O *Transformer* solucionou essas limitações ao introduzir o *mecanismo de atenção escalada (scaled dot-product attention)*, que permite o processamento de todos os elementos em paralelo. Essa abordagem tornou possível capturar dependências de longo alcance sem necessidade de recorrência, trazendo eficiência e precisão a tarefas de tradução automática, análise de texto, visão computacional e outras áreas do aprendizado profundo.

2 FUNDAMENTAÇÃO TEÓRICA

O Transformer é composto por duas partes principais: o *Encoder* e o *Decoder*. O *Encoder* processa a sequência de entrada e gera representações vetoriais para cada token, enquanto o *Decoder* utiliza essas representações, juntamente com as saídas anteriores, para gerar a próxima palavra na sequência traduzida. O processo é autorregressivo, isto é, cada *token* gerado é realimentado no modelo até que o *token* de fim de sequência seja atingido (Vaswani *et al.*, 2017).

Um dos principais componentes dessa arquitetura é o mecanismo de *autoatenção* (*Self-Attention*), que calcula a relação de cada elemento da sequência com todos os outros, atribuindo pesos que representam sua relevância contextual (Chollet, 2021) (Goodfellow; Bengio; Courville, 2016). Esse cálculo substitui a propagação sequencial de informação característica das *RNNs*, permitindo que o modelo capture o contexto global de forma simultânea.

2.1 ARQUITETURA MATEMÁTICA DO TRANSFORMER

O mecanismo central da arquitetura é o *Scaled Dot-Product Attention*, definido pela equação (Vaswani *et al.*, 2017):

Uma *RNN* calcula:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Em que:

- Q (*Queries*) representam o que está sendo buscado.
- K (*Keys*) são os índices de referência.
- V (*Values*) correspondem às informações associadas a cada token.

Cada palavra é transformada em três vetores Q_i , K_i , V_i , calculados por multiplicação da matriz de *embeddings* X pelas matrizes de pesos W_Q , W_K , W_V (Vaswani *et al.*, 2017):

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

Essas matrizes são aprendidas durante o treinamento. O resultado da operação QK^T mede a similaridade entre tokens; o fator de escala QK^T mede a similaridade entre tokens, o fator de escala $\sqrt{d_k}$ estabiliza os gradientes, e o *softmax* normaliza os pesos de atenção, garantindo que a soma de cada linha seja 1.

O vetor resultante é uma combinação ponderada dos valores V , na qual cada palavra é atualizada conforme sua importância em relação às demais. Esse processo é conhecido como *autoatenção*, e permite que o modelo capture relações sem depender da posição sequencial (Chollet, 2021) (Goodfellow; Bengio; Courville, 2016).

2.2 CODIFICAÇÃO POSICIONAL

Como o Transformer não utiliza recorrência, é necessário introduzir informações de ordem. Para isso, utiliza-se a codificação posicional, definida por funções senoidais (Vaswani *et al.*, 2017):

$$QPE_{pos,2i} = \sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{pos,2i+1} = \cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right)$$

Essas funções adicionam ao embedding de cada token uma componente dependente de sua posição na sequência, permitindo que o modelo aprenda relações baseadas na distância entre palavras.

2.3 CUSTO COMPUTACIONAL

O cálculo de atenção requer a multiplicação entre todas as combinações de tokens na sequência. Isso implica uma complexidade de tempo e memória proporcional a (Vaswani *et al.*, 2017):

$$O(n^2)$$

Onde n é o tamanho da sequência. Esse custo quadrático é o principal gargalo dos Transformers, motivando o desenvolvimento de variantes como *Longformer*, *Performer* e *FlashAttention*, que buscam reduzir a complexidade para $O(n \log n)$ ou inferior (Beltagy; Peters; Cohan, 2020); (Choromanska *et al.*, 2020); (Dao *et al.*, 2022)

3 METODOLOGIA

Para avaliar o desempenho da arquitetura *Transformer*, foi implementado um modelo de tradução automática português–inglês utilizando TensorFlow. A implementação foi baseada na base de dados TED HRLR *Translate*, que contém pares de sentenças paralelas em português e inglês (Tensorflow datasets, 2024).

O *pipeline* seguiu as seguintes etapas:

- **Carregamento e pré-processamento dos dados:** tokenização, remoção de símbolos e normalização de caracteres (Chollet, 2021).
- **Tokenização e criação dos pares de sequência:** conversão dos textos em tensores numéricos, aplicando ***padding*** para padronizar os comprimentos.
- **Definição da arquitetura Transformer:** construção das camadas Encoder e Decoder com atenção ***multi-cabeças*** e redes ***feed-forward*** (Vaswani *et al.*, 2017).
- **Aplicação de máscaras:** uso de ***padding mask*** para ignorar tokens nulos e ***look-ahead mask*** para impedir o acesso a tokens futuros durante o treinamento (Tensorflow, 2024).
- **Treinamento do modelo:** execução por múltiplas épocas, com ajuste automático dos pesos.

Durante o treinamento, o modelo aprendeu a mapear sequências de entrada em português para suas correspondentes em inglês, ajustando internamente os pesos de atenção e as representações vetoriais dos *tokens* (Chollet, 2021).

4 RESULTADOS

O modelo apresentou desempenho consistente ao traduzir sentenças curtas e médias, capturando a estrutura gramatical e o sentido geral das frases (Tensorflow, 2024).

A análise dos pesos de atenção demonstrou que o modelo aprendeu a correlacionar corretamente palavras semanticamente equivalentes entre os dois idiomas (Vaswani *et al.*, 2017).

Durante o processo de inferência, o tradutor foi capaz de gerar traduções compreensíveis e coerentes, evidenciando a capacidade da arquitetura de identificar padrões linguísticos complexos.

O uso de **atenção multi-cabeças** possibilitou que o modelo considerasse múltiplos contextos simultaneamente, resultando em uma tradução mais precisa (Chollet, 2021).

5 CONCLUSÃO

A arquitetura Transformer redefiniu os paradigmas do aprendizado profundo aplicado a tarefas de tradução automática (Vaswani *et al.*, 2017). Ao substituir a recorrência pelo mecanismo de atenção, tornou-se possível processar sequências inteiras em paralelo, com maior eficiência e desempenho.

A análise realizada demonstra como o Transformer supera as limitações das Redes Neurais Recorrentes (RNNs), permitindo capturar dependências de longo alcance e relações semânticas complexas (Goodfellow; Bengio; Courville, 2016).

A implementação prática evidenciou a capacidade do modelo em aprender padrões linguísticos e realizar traduções naturais entre português e inglês (CHOLLET, 2021). A eficiência, escalabilidade e flexibilidade da arquitetura consolidam o Transformer como a base dos modelos de linguagem mais avançados da atualidade, incluindo os sistemas generativos modernos (Vaswani *et al.*, 2017) (Dao *et al.*, 2022).

REFERÊNCIAS

CHARNIAK, Eugene. **Introduction to Deep Learning**. The MIT Press, 2018.

CHOLLET, François. **Deep Learning with Python**. 2^a ed. Manning Publications, 2021.

CHOROMANSKA, Anna *et al.* **Rethinking Attention with Performers**. arXiv:2009.14794, 2020.

DAO, Tri *et al.* **FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness**. In: **Advances in Neural Information Processing Systems**, 2022.

DATA SCIENCE ACADEMY. **Deep Learning Book**. Disponível em: <https://www.deeplearningbook.com.br>. Acesso em: 12 jul. 2025.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. Cambridge: The MIT Press, 2016.

MEEDENIYA, Dulani. **Deep Learning: A Beginners' Guide**. CRC Press, 2024.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial: Uma Abordagem Moderna**. 4. ed. Rio de Janeiro: LTC, 2022.

SEJNOWSKI, Terrence J. **A Revolução do Aprendizado Profundo**. Alta Books, 2020.

SKANSKI, Sandro. **Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence**. Springer, 2018.

TENSORFLOW DATASETS: **TED HRLR Translate**. Disponível em: https://www.tensorflow.org/datasets/catalog/ted_hrlr_translate. Acesso em: 27 out. 2025.

TENSORFLOW. **Transformer model for language understanding**. Disponível em: <https://www.tensorflow.org/text/tutorials/transformer>. Acesso em: 27 out. 2025.

VASWANI, Ashish *et al.* **Attention Is All You Need**. In: Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017). Long Beach, CA: Curran Associates, 2017. p. 5998–6008. Disponível em: <https://arxiv.org/abs/1706.03762>. Acesso em: 15 out. 2025.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA

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.

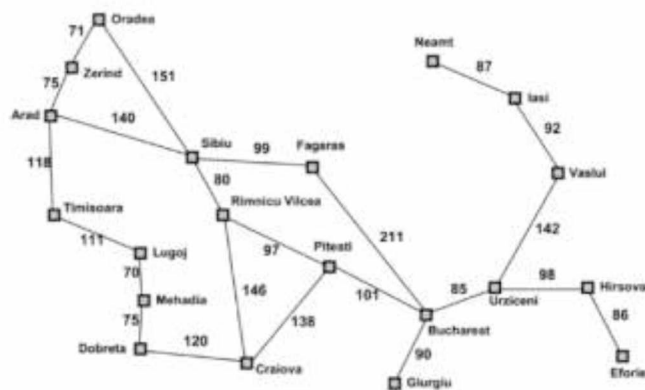
2 Busca Heurística

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 *ADLR* — 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: $\alpha \rightarrow \beta, \beta \rightarrow \gamma \vdash \alpha \rightarrow \gamma$

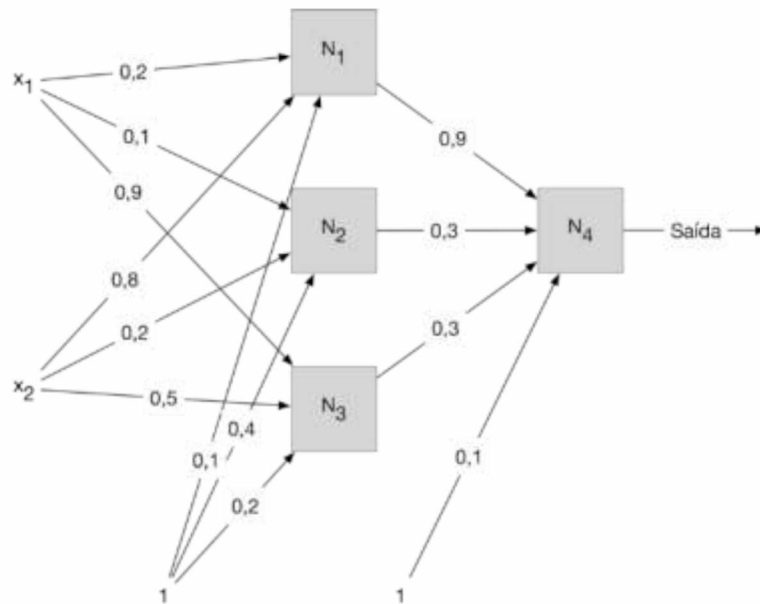
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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)

Prompt: Olá ChatGPT. O que é inteligência artificial?

Resposta: Olá! Inteligência artificial (IA) é um campo da ciência da computação que se concentra no desenvolvimento de sistemas e algoritmos capazes de realizar tarefas que normalmente exigiriam inteligência humana. Isso inclui uma variedade de habilidades, como reconhecimento de padrões, tomada de decisões, resolução de problemas, compreensão de linguagem natural e aprendizado. O objetivo da inteligência artificial é criar sistemas que possam perceber o ambiente ao seu redor, aprender com essas percepções e tomar decisões autônomas para alcançar objetivos específicos. Esses sistemas são frequentemente baseados em algoritmos de aprendizado de máquina e redes neurais, que podem analisar grandes conjuntos de dados, identificar padrões e fazer previsões. A inteligência artificial tem aplicações em uma ampla variedade de campos, incluindo medicina, finanças, transporte, manufatura, jogos, assistência virtual e muito mais.

b)

Inteligência Artificial (IA) como descrita pelo ChatGPT se encaixa no modelo “Agir Racionalmente”, a abordagem mais abrangente, visto que, como mencionado pelo Chat, a IA tem como objetivo ter consciência do seu ambiente, aprender com suas experiências e buscar tomar sempre a melhor decisão. Isso excluiria as abordagens “Agir como humano” e “Pensar como humano”, visto que o objetivo da IA descrita pelo chat não é imitar o comportamento humano nem replicar fielmente o modo como os humanos pensam, mas tomar decisões autônomas para atingir seus objetivos. Pela descrição fornecida, também é possível perceber que os objetivos de IA são mais abrangentes que a abordagem “Pensar racionalmente”, visto que essa linha tem como objetivo modelar um processo de raciocínio perfeito, mas nem sempre adaptável.

c)

O ChatGPT gera resposta baseada no contexto e na intenção por trás da questão do usuário. Seu poder está na habilidade de analisar perguntas e produzir respostas completas, detalhadas, e resultados baseados nas informações em texto

mais frequentes acessíveis digitalmente do mundo à época de seu treinamento. Em termos gerais, o ChatGPT funciona em duas fases principais.

A fase de coleta de dados é chamada de pré-treinamento, enquanto a fase de responsividade do usuário é chamada de inferência. A “magia” por trás da IA generativa e o motivo pelo qual ela explodiu de repente é que a maneira como o pré-treinamento funciona e ele ser enormemente escalável. Essa escalabilidade foi possibilitada por inovações recentes em tecnologia de hardware acessíveis e computação em nuvem. As IA's são treinadas usando dois principais enfoques: supervisionado e não supervisionado.

Para a maioria dos projetos de IA, antes do ChatGPT, o enfoque era supervisionado. Mas a abordagem supervisionada é difícil de ser escalonada, pois é impossível antecipar todas as perguntas que poderiam ser feitas à IA. Assim, o ChatGPT foi desenvolvido a partir de treinamento não supervisionado, ou seja, o modelo é treinado em dados nos quais nenhuma saída específica está associada a cada entrada. Na abordagem não supervisionada o modelo é treinado para aprender a estrutura subjacente e os padrões nos dados de entrada sem nenhuma tarefa específica definida, como em agrupamentos, detecção de anomalias e redução de dimensionalidade. No contexto da modelagem de linguagem, o treinamento não supervisionado foi usado para treinar o modelo a entender a sintaxe e semântica da linguagem natural, para que o ChatGPT pudesse gerar texto coerente e significativo em um contexto de conversação. Assim, o ChatGPT adquire seu aparente conhecimento ilimitado.

Como os desenvolvedores não precisam saber as saídas que vêm das entradas, tudo o que eles têm que fazer é despejar mais e mais informações no mecanismo de treinamento do ChatGPT, que é chamado de modelagem de linguagem baseada em transformadores. A arquitetura de transformadores é um tipo de rede neural usada para processar dados de linguagem natural, simulando o funcionamento de um cérebro humano processando informações por meio de camadas de nós interconectados. A arquitetura de transformadores processa sequências de palavras usando "auto atenção" para ponderar a importância de diferentes palavras em uma sequência ao fazer previsões.

A “auto atenção” é semelhante à maneira como um leitor pode voltar a uma frase ou parágrafo anterior para o contexto necessário para entender uma nova palavra em um livro. O transformador examina todas as palavras em uma sequência

para entender o contexto e as relações entre as palavras. O transformador é composto por várias camadas, cada uma com várias subcamadas. As duas principais subcamadas são a camada de “auto atenção” e a camada de “feedforward”.

A camada de “auto atenção” calcula a importância de cada palavra na sequência, enquanto a camada de “feedforward” aplica transformações não lineares aos dados de entrada. Essas camadas ajudam o transformador a aprender e entender as relações entre as palavras em uma sequência.

Durante o treinamento, o transformador recebe dados de entrada, como uma frase, e é solicitado a fazer uma previsão com base nessa entrada. O modelo é atualizado com base em quão bem sua previsão corresponde à saída real. Através desse processo, o transformador aprende a entender o contexto e as relações entre as palavras em uma sequência, tornando-se uma ferramenta poderosa para tarefas de processamento de linguagem natural, como tradução de idiomas e geração de texto. Há possibilidade da IA gerar conteúdo prejudicial ou tendencioso, pois podem aprender padrões e preconceitos presentes nos dados de treinamento. Por isso os desenvolvedores devem colocar limitações no conteúdo de treinamento e no aceite de resposta da IA, prevenindo e evitando questões sensíveis como racismo e extremismos.

Também é preciso considerar que nem todo conteúdo produzido pelo Chat GPT é integralmente confiável, a IA está sendo gradualmente aprimorada e ainda comete erros, ou os chamados “delírios” ao ter a capacidade de inventar informações a partir de inferências feitas do conteúdo de seu treinamento.

Fontes: (<https://www.zdnet.com/article/how-does-chatgpt-work/>); (<https://help.openai.com/en/articles/7842364-how-chatgpt-and-our-language-models-are-developed>)

d)

Pensar como os humanos: Considera-se que o ChatGPT não pensa como um humano, ChatGPT aprende a partir de vasta gama de dados, está limitado aos padrões e correlações do conjunto de dados de seu treinamento. Para pensar como humano ChatGPT precisaria de consciência, emoções e experiências subjetivas, assim a implementação de um programa de computador que leva em consideração aspectos cognitivos, introspectivos e empíricos é provavelmente inviável de ser implementado;

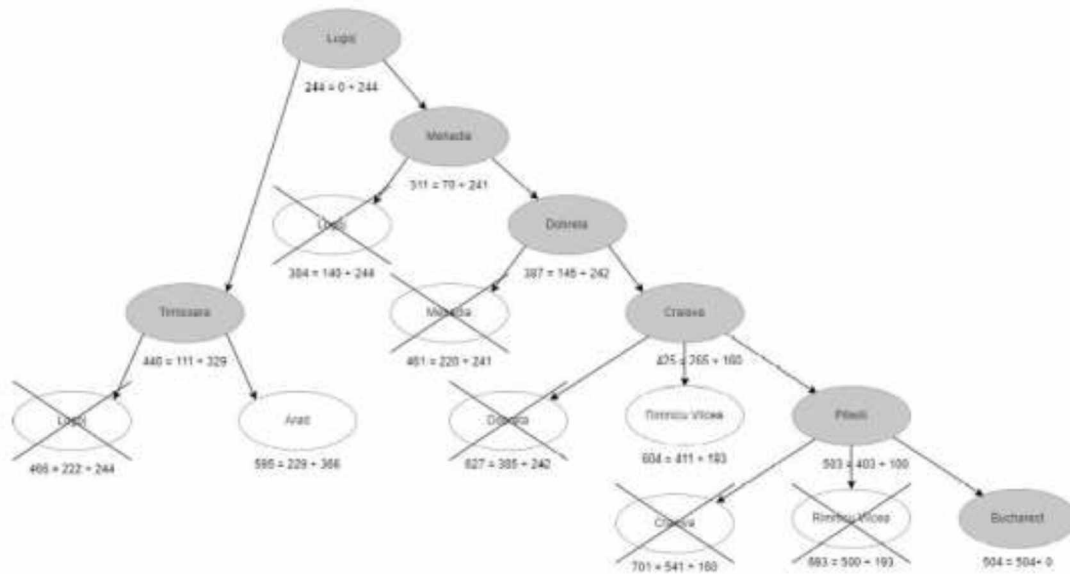
Pensar Racionalmente: Considera-se que o ChatGPT não é primariamente uma IA desenvolvida para pensar racionalmente, ou seja, seu objetivo não modelar o pensamento correto para atingir um resultado logicamente correto. Como é baseado em algoritmos e modelos estatísticos, ChatGPT apresenta respostas que são geralmente racionais no contexto a que foi exposto.

Agir como os Humanos: Considera-se que o ChatGPT atende parcialmente à essa abordagem. Ele consegue conversar em língua natural e responder como se fosse humano, por isso acredita-se que ele passaria no Teste de Turing atendendo os parâmetros da abordagem. Mas por outro lado ele é bastante formal e assertivo, ao passo que o vocabulário humano é marcado por gírias e erros de gramática e digitação, outro ponto seria que nessa abordagem não seria necessário verificar as respostas corretas, mas na maioria das vezes, com exceção de delírios, o ChatGPT preza por trazer a resposta certa.

Agir Racionalmente: Considera-se que ChatGPT não atende aos critérios dessa abordagem. Ele não incorpora agentes ou algoritmos de busca para encontrar o melhor resultado possível. Enquanto o ChatGPT é capaz de gerar respostas baseadas em padrões estatísticos aprendidos a partir de dados e retorna uma resposta racional ao input inicial, ele não realiza busca ativa autônoma em espaço de probabilidades para otimizar um objetivo específico.

FIGURA 1 – Busca Heurística, Lógica, Redes Neurais Artificiais

2. Busca Heurística



FONTE: Amauri (2025).

3. Lógica

R1: $p \rightarrow q$ R2: $q \rightarrow r$ R3: $\neg r \vee p$ R4: $r \rightarrow p$

Equivalência Condicional em R3

R5: $q \rightarrow p$

Silogismo Hipotético em R2 e R4

R6: $q \rightarrow p \wedge p \rightarrow q$

Conjunção em R5 e R1

R7: $q \leftrightarrow p$

Equivalência Bicondicional em R6

4. Redes Neurais Artificiais

a) 0,3

b) 0,3

c) -2

d) -0,1391

FONTE: Amauri (2025).

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

- 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 um breve explicação (máximo de quatro linhas) sobre os principais resultados encontrados na Análise Exploratória dos dados

2 Visualização dos dados

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

- 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**

- f. Dê uma breve explicação (máximo de quatro linhas) sobre os resultados encontrados na análise de importância de variáveis
- g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R²
- h. Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada

B - RESOLUÇÃO

1.a)

```
dados = pd.read_csv('D:/precos_carros_brasil.csv', decimal=',')

C:\Users\Luis Henrique
Paiva\AppData\Local\Temp\ipykernel_9644\3996436910.py:1: DtypeWarning:
Columns (1,2,3,4,5,6,7) have mixed types. Specify dtype option on import
or set low_memory=False.
    dados = pd.read_csv('D:/precos_carros_brasil.csv', decimal=',')
```

1.b)

```
# Verificação de dados faltantes
dados.isna().any()

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

# Verificação da quantidade de valores faltantes por coluna
dados.isna().sum()

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

```
dtype: int64
```

```
# Verificação de como os dados são apresentados
dados.head(10)
```

```
year_of_reference month_of_reference fipecode authentication \ 0 1 2 3
4 5 2021.0 2021.0 2021.0 2021.0 2021.0 January 004001-0 cfzltzfwrcp
January 004001-0 cdqwxwpw3y2p January 004001-0 cb1t3xwwj1xp January
004001-0 cb9gct6j65r0 January 004003-7 g15wg0gbz1fx 2021.0 January
004003-7 gv8bccsh1lp 2 6 7 8 9 2021.0 2021.0 2021.0 2021.0 brand 0 GM-
Chevrolet 1 GM- Chevrolet 2 GM- Chevrolet 3 GM- Chevrolet January
004004-5 chrppb85ytp January 004005-3 f9jrtb47t4fx January 004005-3
f5jtkzvb14p January 004006-1 g5wq00ps2pfx model fuel Corsa Wind 1.0 MPFI
/ EFI 2p Gasoline manual Corsa Wind 1.0 MPFI / EFI 2p Gasoline manual
Corsa Wind 1.0 MPFI / EFI 2p Gasoline manual Corsa Wind 1.0 MPFI / EFI
2p Alcohol manual 4 GM- Chevrolet Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI
Gasoline manual 5 GM- Chevrolet Corsa Pick-Up GL/ Champ 1.6 MPFI / EFI
Gasoline manual 6 GM- Chevrolet 7 GM- Chevrolet 8 GM- Chevrolet 9 GM-
Chevrolet gear \ Corsa GL 1.6 MPFI / 1.4 EFI 2p e 4p Gasoline manual
Corsa Sedan GL 1.6 MPFI 4p Gasoline manual Corsa Sedan GL 1.6 MPFI 4p
Gasoline manual Corsa Sedan GLS 1.6 16V MPFI 4p Gasoline manual
engine_size year_model avg_price_brl 0 1 2 3 4 5 6 7 8 9 1.0 1.0 1.0 1.0
1.6 1.6 1.6 1.6 1.6 1.6 [8]: dados.tail(10) 2002.0 2001.0 2000.0 2000.0
2001.0 2000.0 2000.0 2001.0 2000.0 2001.0 9162.0 8832.0 8388.0 8453.0
12525.0 12020.0 9632.0 10508.0 10167.0 12841.0
```

```
dados.tail(10)
```

```
year_of_reference month_of_reference fipecode authentication brand \
267532 267533 267534 267535 267536 267537 267538 267539 267540 267541
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN model fuel gear engine_size year_model avg_price_brl
267532 NaN NaN NaN NaN 267533 NaN NaN NaN 267534 NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN 3 267535
267536 267537 267538 267539 267540 267541 NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN NaN
```

```
# Como toda a linha está vazia, apaga-se essas linhas dados =
dados.dropna()
```

```
dados.tail(100)
```

```
year_of_reference month_of_reference fipecode authentication \ 202197
202198 202199 202200 202201 ... 202292 202293 202294 202295 202296 2023.0
```

```

2023.0 2023.0 2023.0 2023.0 ... 2023.0 2023.0 2023.0 2023.0 2023.0 brand
January 005505-0 jk8q7jz0lrdbm January 005506-9 qzcxgxbwjcb January
005506-9 qfs7gs1pppdrj January 005506-9 pt52qbsjb2dm9 January 005506-9
n2xcvjkqdtjdj2 ... .. January 005538-7 ccv3mvxnsz0dqw January 005539-5
January 005539-5 cdj27srtcvcdqw January 005540-9 January 005540-9
7hbnjnj9z5dqw 202197 VW- Volkswagen AMAROK Comfor. CD 2.0 TDI 4x4 Dies.
Aut. 202198 VW- Volkswagen AMAROK Extreme CD 3.0 4x4 TB Dies. Aut.
202199 VW- Volkswagen AMAROK Extreme CD 3.0 4x4 TB Dies. Aut. 202200 VW-
Volkswagen AMAROK Extreme CD 3.0 4x4 TB Dies. Aut. 202201 VW- Volkswagen
AMAROK Extreme CD 3.0 4x4 TB Dies. Aut. ... .. 202292 VW- Volkswagen 202293
VW- Volkswagen 202294 VW- Volkswagen 202295 VW- Volkswagen 202296 VW-
Volkswagen 202198 automatic 202199 automatic ... chmwfg3l5hbp 9w64fg6dhqp
model fuel \ Diesel Diesel Diesel Diesel Diesel ... .. Saveiro Robust 1.6
Total Flex 16V Gasoline Gol Last Edition 1.0 Flex 12V 5p Gasoline Gol
Last Edition 1.0 Flex 12V 5p Gasoline Polo Track 1.0 Flex 12V 5p
Gasoline Polo Track 1.0 Flex 12V 5p Gasoline gear engine_size year_model
avg_price_brl 202197 automatic 2.0 3.0 3.0 202200 automatic 202201
automatic ... 202292 manual 3.0 3.0 ... 1.6 2019.0 2023.0 2023.0 2022.0
2021.0 ... 2023.0 165417.0 334377.0 320244.0 304365.0 284509.0 ... 86038.0 4
202293 202294 202295 202296 manual manual manual manual [100 rows x 11
columns] 1.0 1.0 1.0 1.0 2023.0 2023.0 2023.0 2023.0 95997.0 87828.0
80845.0 74458.0

```

```

# Verificação se ainda existem dados faltantes
dados.isna().any()

```

```

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

```

```

dados.isna().sum()

```

```

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

```

1.c)

```

# Verificação se há dados duplicados
dados.duplicated().sum()

```

```

2

```

```

# Remoção dos dados duplicados
dados.drop_duplicates(inplace=True)

```

```

# Verificação se há dados duplicados
dados.duplicated().sum()

```

1.d)

```

0
# Verificação do tipo dos dados
dados.dtypes

year_of_reference month_of_reference fipecode authentication brand
model fuel gear engine_size year_model avg_price_brl float64 object
object object object object object object float64 float64 float64 dtype:
object

colunas_numericas = [col for col in dados.columns if dados[col].dtype
!='object']
colunas_categoricas = [col for col in dados.columns if dados[col].dtype
=='object']

dados[colunas_numericas].describe()

year_of_reference count engine_size year_model avg_price_brl
202295.000000 202295.000000 202295.000000 202295.000000 mean std min 25%
50% 75% max 2021.564695 0.571904 2021.000000 2021.000000 2022.000000
1.822302 0.734432 1.000000 2011.271514 52756.765713 6.376241
51628.912116 2000.000000 1.400000 6647.000000 2006.000000 22855.000000
1.600000 2022.000000 2023.000000 2.000000 6.200000 2012.000000
38027.000000 2016.000000 64064.000000 2023.000000 979358.000000

dados[colunas_categoricas].describe()

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

```

1.e)

```

dados['model'].value_counts()

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

dados['brand'].value_counts()

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

dados.dtypes

```

```

year_of_reference month_of_reference fipecode authentication brand
model fuel float64 object object object object object 7 gear
engine_size year_model avg_price_brl dtype: object object float64
float64 float64

```

1.f)

Primeiramente, notou-se que as últimas linhas apresentam todas as colunas como NaN, ou seja, não disponíveis. Desse modo, optou-se por excluir as linhas com essa característica. No que se refere a dados duplicados, observou-se duas ocorrências, esses registros também foram apagados. Por fim, foram criados dois grupos, um para as colunas com dados numéricos e outra para dados categóricos.

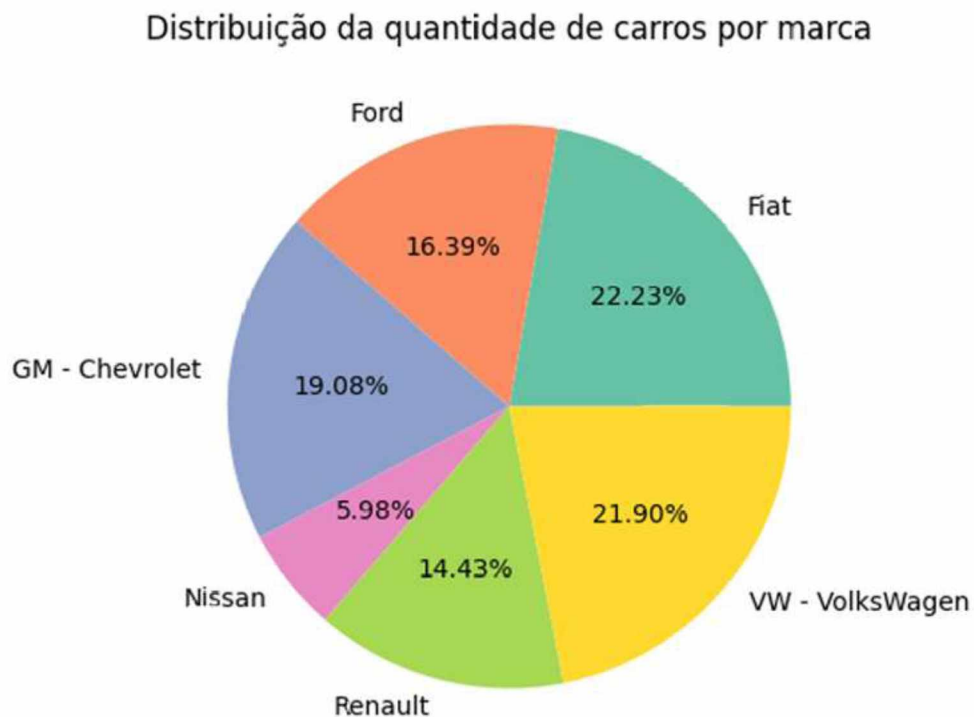
2.a)

```

carros_por_marca = dados['brand'].value_counts().sort_index()
plt.figure(figsize=(10,5))
plt.pie(carros_por_marca, labels=carros_por_marca.index,
autopct='%1.2f%%', colors=sns.color_palette('Set2'))
plt.title('Distribuição da quantidade de carros por marca')
Text(0.5, 1.0, 'Distribuição da quantidade de carros por marca')

```

GRÁFICO 1 – DISTRIBUIÇÃO DA QUANTIDADE DE CARROS



FONTE: Amauri (2025).

2.b)

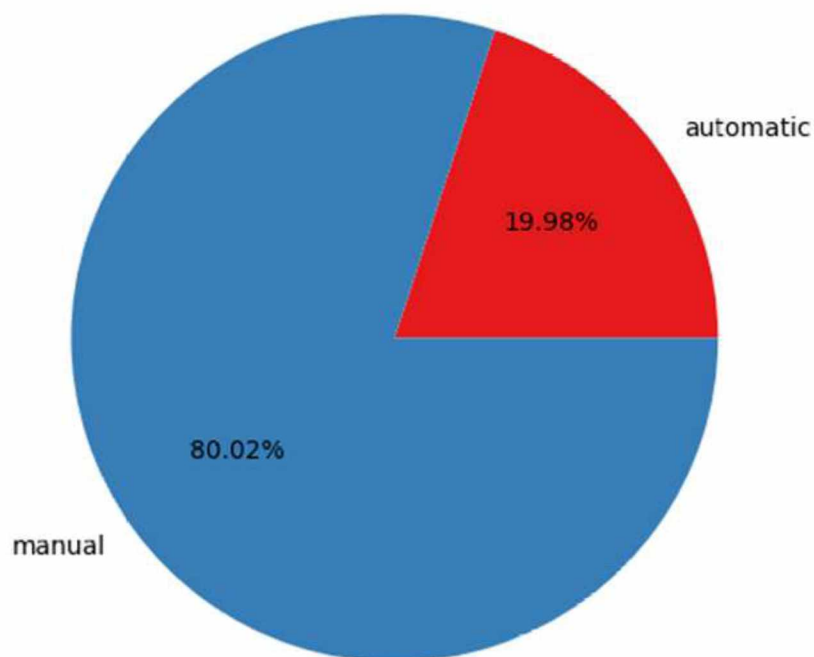
```

carros_por_cambio = dados['gear'].value_counts().sort_index()
plt.figure(figsize=(10,5))
plt.pie(carros_por_cambio, labels=carros_por_cambio.index,
autopct='%1.2f%%', colors=sns.color_palette('Set1'))
plt.title('Distribuição da quantidade de carros por tipo de câmbio')
plt.tight_layout()

```

GRÁFICO 2 – DISTRIBUIÇÃO DA QUANTIDADE DE CARROS POR TIPO DE CÂMBIO

Distribuição da quantidade de carros por tipo de câmbio



FONTE: Amauri (2025).

2.c)

```

# Função para converter o nome dos meses
def traduz_nome_mes(mes):
    match(mes):
        case 'January': return 'Janeiro'
        case 'February': return 'Fevereiro'
        case 'March': return 'Março'
        case 'April': return 'Abril'
        case 'May': return 'Maio'
        case 'June': return 'Junho'
        case 'July': return 'Julho'
        case 'August': return 'Agosto'
        case 'September': return 'Setembro'
        case 'October': return 'Outubro'

```

```

case 'November': return 'Novembro'
case 'December': return 'Dezembro'

# Adequação de algumas colunas
dados['mes_num'] = pd.to_datetime(dados.month_of_reference,
format='%B').dt.month.astype(int)
dados['mes'] = dados['month_of_reference'].astype(str)
dados['mes'] = dados['mes'].apply(traduz_nome_mes)
dados['ano'] = dados['year_of_reference'].astype(int)
dados['ano_modelo'] = dados['year_model'].astype(int)
dados['marca'] = dados['brand']
dados['cambio'] = dados['gear']
dados['combustivel'] = dados['fuel']
dados['preco_medio'] = dados['avg_price_brl'].astype(float)

# Filtrando apenas dados com ano de referência 2022
media_preco_mes_2022 = dados[dados.year_of_reference == 2022]

media_preco_mes_2022 =
media_preco_mes_2022.groupby(['mes_num', 'mes', 'ano'])['preco_medio'].mean().round(2)
media_preco_mes_2022.head(100)

mes_num mes 1 2 Janeiro ano 2022 Fevereiro 2022 3 4 5 6 7 8 9 10 11 12
Março Abril Maio Junho Julho Agosto Setembro Outubro Novembro 2022 2022
2022 2022 2022 2022 2022 54840.27 55824.52 56848.95 57150.04 57799.76
58065.61 57894.00 57923.54 58198.94 2022 2022 Dezembro 2022 58227.41
58215.63 57997.24 Name: preco_medio, dtype: float64

media_preco_mes_2022 =
media_preco_mes_2022.reset_index(name='preco_medio')
media_preco_mes_2022.head()

mes_num 0 1 1 mes ano preco_medio Janeiro 2022 2 Fevereiro 2022 2 3 4 3
4 5 Março 2022 Abril 2022 54840.27 55824.52 56848.95 57150.04 Maio 2022
57799.76

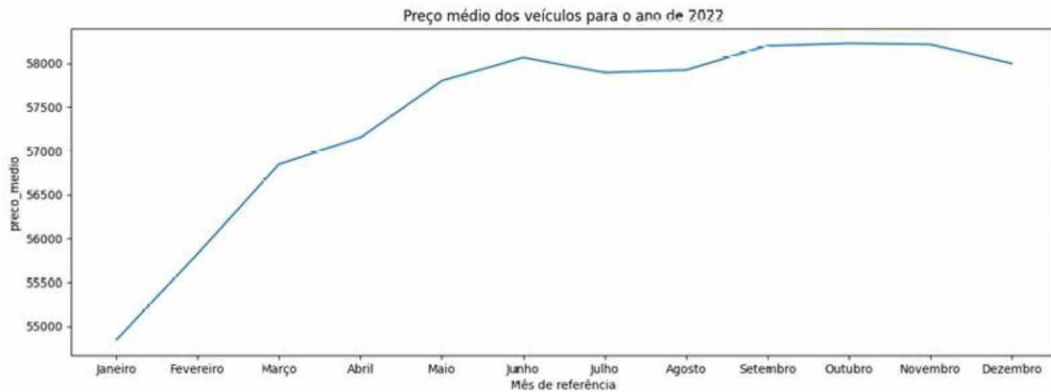
plt.figure(figsize=(15,5)) line =
sns.lineplot(data=media_preco_mes_2022, x="mes",
y="preco_medio", markers=True)

line.set_xlabel("Mês de referência") plt.title('Preço médio dos veículos
para o ano de 2022')

Text(0.5, 1.0, 'Preço médio dos veículos para o ano de 2022')

```

GRÁFICO 3 – PREÇO MÉDIO DOS VEÍCULOS POR MÊS

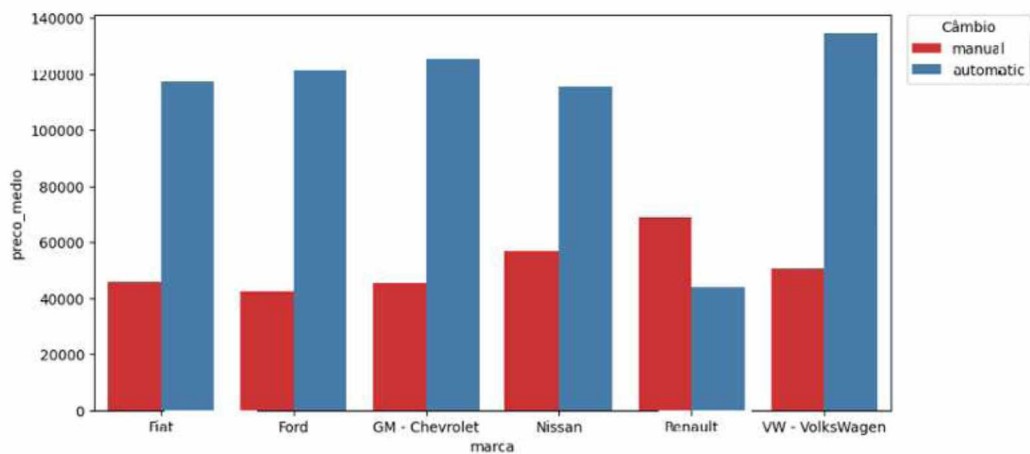


FONTE: Amauri (2025).

2.d)

```
media_preco_mes = dados.groupby(['mes_num', 'mes', 'marca', 'ano',
'cambio', 'combustivel'])['preco_medio'].mean().round(2)
media_preco_mes = media_preco_mes.reset_index(name='preco_medio')
media_preco_mes.head()
mes_num 0 mes marca ano cambio combustivel preco_medio
1 Janeiro Fiat 2021 automatic 1 2 3 4 1 Janeiro Fiat 2021 automatic 1 Janeiro Fiat 2021
1 Janeiro Fiat 2021 Diesel 124175.73 Gasoline manual manual 1 Janeiro
Fiat 2021 manual Alcohol Diesel Gasoline 59892.84 10518.67 73456.58
28911.03
plt.figure(figsize=(10,5)) sns.barplot(x='marca', y='preco_medio',
hue='cambio', data=media_preco_mes,palette='Set1', hue_order=['manual',
'automatic'], errorbar=None); plt.legend(title='Câmbio',
bbox_to_anchor=(1.02, 1), loc='upper left',borderaxespad=0)
```

GRÁFICO 3 – DISTRIBUIÇÃO MÉDIA DO PREÇO DE CARRO POR CÂMBIO



FONTE: Amauri (2025).

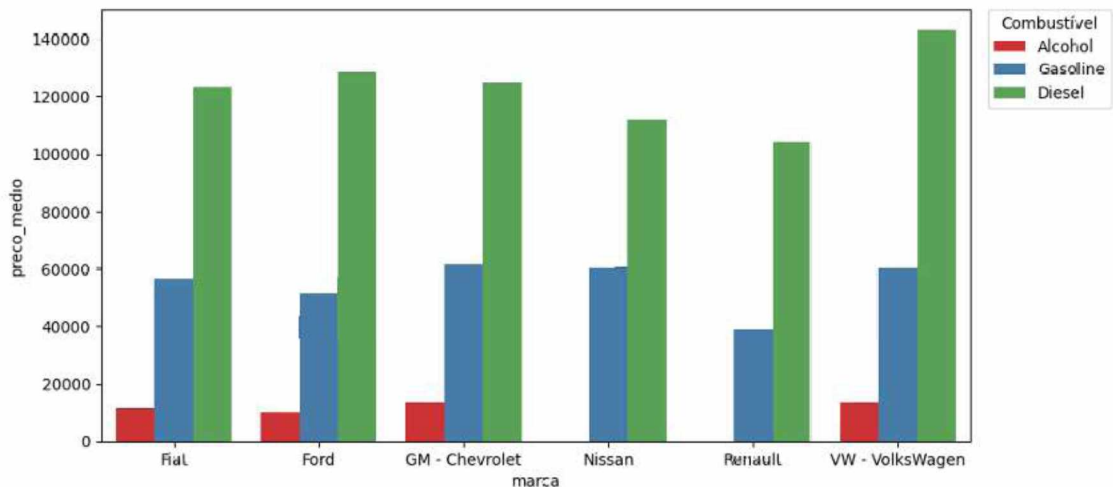
2.e)

Nota-se, pelo gráfico gerado, que, geralmente, carros com câmbio automático possuem um preço médio muito maior quando comparado com os carros com câmbio manual. A única exceção são os carros da marca Renault, que apresentam um valor médio maior para os modelos com câmbio manual.

2.f)

```
plt.figure(figsize=(10,5)) sns.barplot(x='marca', y='preco_medio',
hue='combustivel',data=media_preco_mes,
palette='Set1',hue_order=['Alcohol','Gasoline','Diesel'], errorbar=None,
color=sns.color_palette("hls", 8)); plt.legend(title='Combustível',
bbox_to_anchor=(1.02, 1), loc='upper left',borderaxespad=0)
```

GRÁFICO 4 – DISTRIBUIÇÃO MÉDIA DO PREÇO DE CARRO POR COMBUSTÍVEL



FONTE: Amauri (2025).

2.g)

Percebe-se que carros com combustível diesel possuem um preço médio bem maior que os modelos com outros tipos de combustíveis. Carros à gasolina possuem um preço médio em torno da metade dos modelos a diesel. Ainda, carros a álcool são os que possuem o menor valor, quando oferecidos pelas marcas. Nissan e Renault não oferecem modelos com esse tipo de combustível.

3.a)

```
#Para o caso em tela, considera-se importante o label-encoding das
colunas modelo, fuel e gear.
dados['modelo'] = LabelEncoder().fit_transform(dados['modelo'])
dados['combustivel'] =
LabelEncoder().fit_transform(dados['combustivel'])
dados['cambio'] = LabelEncoder().fit_transform(dados['cambio'])
```

```

# Como o model já indica implicitamente o fabricante, não é necessária a
coluna com a marca (brand).
dados = dados.drop(columns=['year_of_reference',
'month_of_reference', 'fipecode', 'authentication', 'avg_price_brl',
'brand', 'model', 'fuel', 'gear', 'mes', 'marca'], axis=1)

dados.columns

Index(['engine_size', 'year_model', 'mes_num', 'ano', 'ano_modelo',
'cambio', 'combustivel', 'preco_medio', 'modelo'], dtype='object')

dados.head()

engine_size year_model mes_num ano ano_modelo cambio combustivel \ 0 1 2
3 4 0 1 2 3 4 1.0 1.0 1.0 1.0 1.6 2002.0 2001.0 2000.0 2000.0 2001.0
preco_medio modelo 9162.0 8832.0 8388.0 297 297 1 2021 1 2021 1 2021 1
2021 1 2021 2002 2001 2000 2000 1 1 1 1 2001 297 8453.0 12525.0 297 260

# Mapa de correlação das variáveis numéricas com variável Target
dados.dtypes

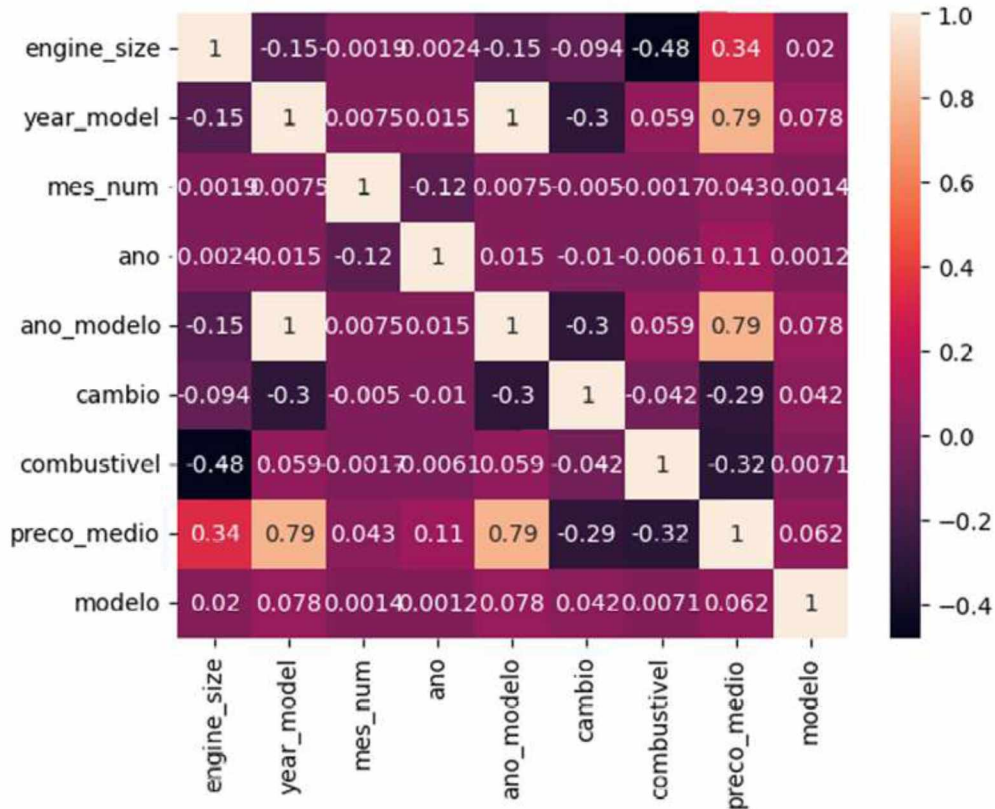
engine_size year_model mes_num ano ano_modelo cambio combustivel
preco_medio modelo float64 float64 int32 int32 int32 int32 int32 float64
int32 dtype: object

# Mapa de correlação das variáveis numéricas com variável Target
sns.heatmap(dados.corr("spearman"), annot = True) plt.title("Mapa de
Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()

```

GRÁFICO 5 – MAPA DE CORRELAÇÃO DAS VARIÁVEIS NUMÉRICAS

Mapa de Correlação das Variáveis Numéricas



FONTE: Amauri (2025).

```
# Variável X com as variáveis independentes
```

```
X = dados.drop(['preco_medio'],axis = 1) X.head()
```

```
engine_size year_model mes_num ano ano_modelo cambio combustivel \ 0 1 2
3 4 0 1 2 3 1.0 1.0 1.0 1.0 1.6 modelo 297 297 297 2002.0 2001.0 2000.0
2000.0 2001.0 1 2021 1 2021 1 2021 1 2021 1 2021 2002 2001 2000 1 1 1
2000 2001 297 4 260
```

```
# Variável Y com apenas a variável alvo
```

```
Y = dados['preco_medio']
```

```
Y.head()
```

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

3.b)

```
# Divisão dos dados em 75% para treinamento e 25% para teste
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size =
0.25,random_state = 42)
```

```
# Definição de uma semente (seed) para reprodução do experimento
random.seed(1234)
X_train

engine_size year_model mes_num ano ano_modelo cambio \ 156364 93758 2.3
1.6 2020.0 2011.0 8 2022 12 2021 43313 114972 69993 ... 119880 103695
131933 146868 121959 156364 93758 43313 114972 69993 ... 119880 103695 2.0
2.3 1.0 ... 1.6 1.6 2.3 1.3 1.8 2008.0 2013.0 2018.0 ... ... 2012.0 2013.0
2023.0 2003.0 2016.0 combustivel modelo 1 2 2 1 364 1099 6 2021 3 2022 9
2021 ... ... 3 2022 1 2022 5 2022 7 2022 2020 2011 2008 2013 2018 ... 1 1 1 1
1 2012 2013 2023 2003 4 2022 549 385 2 ... 131933 2 2 1 970 ... 1363 2036
365 146868 121959 2 2 358 1637
```

3.c)

```
# Random Forest, parâmetros default
model_rf = RandomForestRegressor()
# Ajuste do modelo
model_rf.fit(X_train, Y_train)

RandomForestRegressor()
```

3.d)

```
# Predição dos valores da média do preço dos carros com base nos dados
de teste
valores_preditos_rf = model_rf.predict(X_test)
# Valores preditos
valores_preditos_rf

array([ 42585.35, 11129.72, 9037.02, ..., 105492.71, 9386.1 , 25243.37])
```

3.e)

```
# Analise da importância das variáveis para estimar a varável alvo
model_rf.feature_importances_
feature_importances = pd.DataFrame(model_rf.feature_importances_, index
=X_train.columns, columns=['importance']).sort_values('importance',
ascending= False)
feature_importances

importance engine_size year_model ano_modelo modelo cambio combustivel
ano mes_num 0.449688 0.203383 0.188518 0.070706 0.036787 0.032890
0.012336 0.005691

# Mean Squared Error
mse = mean_squared_error(Y_test, valores_preditos_rf)
mse

6453899.269700155

# Mean Absolute Error
mae = mean_absolute_error(Y_test, valores_preditos_rf)
```

```

mae
1200.4943613975843
# R²
r2_score(Y_test, valores_preditos_rf)
0.9976019018269694
# Random Forest com parâmetros personalizados
(max_depth=25,min_samples_leaf=32, min_samples_split=28,
n_estimators=180, random_state=42)
model_rf_parametros = RandomForestRegressor(max_depth=25,
min_samples_leaf=32,min_samples_split=28, n_estimators=180,
random_state=42)
model_rf_parametros.fit(X_train, Y_train)

RandomForestRegressor(max_depth=25, min_samples_leaf=32,
min_samples_split=28,
n_estimators=180, random_state=42)

valores_preditos_rf_parametros = model_rf_parametros.predict(X_test)
model_rf_parametros.feature_importances_
feature_importances =
pd.DataFrame(model_rf_parametros.feature_importances_,index =
X_train.columns,
columns=['importance']).sort_values('importance',ascending = False)
feature_importances

importance engine_size ano_modelo year_model modelo combustivel cambio
ano mes_num 0.469388 0.207369 0.204471 0.047812 0.034912 0.023323
0.010392 0.002334

mse = mean_squared_error(Y_test, valores_preditos_rf_parametros)
mse
19
147755624.175803
mae = mean_absolute_error(Y_test, valores_preditos_rf_parametros)
mae
4591.705757877845
r2_score(Y_test, valores_preditos_rf_parametros)
0.9450979202519447
# XGBoost
model_xgboost = XGBRegressor()
model_xgboost.fit(X_train, Y_train)
XGBRegressor(base_score=None, booster=None, callbacks=None,
colsample_bylevel=None, colsample_bynode=None,
colsample_bytree=None, device=None, early_stopping_rounds=None,
enable_categorical=False, eval_metric=None, feature_types=None,
gamma=None, grow_policy=None, importance_type=None,
interaction_constraints=None, learning_rate=None, max_bin=None,
max_cat_threshold=None, max_cat_to_onehot=None,
max_delta_step=None, max_depth=None, max_leaves=None,
min_child_weight=None, missing=nan, monotone_constraints=None,

```

```

multi_strategy=None, n_estimators=None, n_jobs=None,
num_parallel_tree=None, random_state=None, ...)
valores_preditos_xgboost = model_xgboost.predict(X_test)
valores_preditos_xgboost
array([ 45716.93 , 11505.324, 9243.854, ..., 108816.414, 10348.15 ,
23879.635], dtype=float32)
model_xgboost.feature_importances_
feature_importances = pd.DataFrame(model_xgboost.feature_importances_,
index =X_train.columns,
columns=['importance']).sort_values('importance', ascending= False)
feature_importances
importance
engine_size
year_model
combustivel
cambio
modelo
ano
0.426666
0.235787
0.213182
0.071964
0.026653
0.019103
20
mes_num
ano_modelo
0.006645
0.000000
mse = mean_squared_error(Y_test, valores_preditos_xgboost)
mse
32224917.727042086
mae = mean_absolute_error(Y_test, valores_preditos_xgboost)
mae
3302.6570237540227
r2_score(Y_test, valores_preditos_xgboost)
0.988026073370855

```

3.f)

Tanto no modelo de Random Forest como com no de XGBoost, a principal variável foi o engine_size (tamanho do motor). A variável model_year (ano do modelo) teve grande impacto, no entanto, no modelo de Random Forest teve quase o mesmo impacto que a variável tamanho do motor, enquanto no XGBoost ficou com importância menor, dividindo o posto com outra variável, fuel (combustível).

3.g)

Os melhores resultados foram obtidos com o modelo Random Forest, parâmetros de default (mse próximo a 20180801, mae perto de 2340,6588 e r^2 score igual a 0,9925), o que se traduz em uma acurácia de mais de 99%. Para parâmetros personalizados do Random Forest, teve-se mse 150349810, mae 4738,7235 e r^2 de 0,944133. Para o XGBoost, mse 42868382, mae 3811,0364 e r^2 pouco acima de 0,984071.

3.h)

Dê uma breve explicação (máximo de quatro linhas) sobre qual modelo gerou o melhor resultado e a métrica de avaliação utilizada. Para se escolher o melhor resultado, toma-se como base a métrica R^2 , na qual, quanto maior o valor - de zero a um -, melhor o resultado. Para o teste de Random Forest com parâmetro default, o resultado foi de mais de 99%, valor considerado excelente. Os testes com parâmetros personalizado de Random Forest e XGBoost também tiveram resultados muito bons, acima dos 94%, mas não superior ao Random Forest default.

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

- 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;

- $S_{yx}\%$

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1.1)

```
#IAA003 - R - Trabalho#

#Equipe 8:
#Amauri Leonel Klamas Junior (amauri.klamas@ufpr.br)
#Diego Silva (diegofs@ufpr.br)
#Letícia Vieira da Rocha (leticia.rocha@ufpr.br)
#Luís Henrique Paiva (luis.paiva@ufpr.br)
#Ricardo Barbosa Ferreira (ricardo.ferreira1@ufpr.br)

#Libraries
library(caret)
library(mice)
library(mlbench)
library(RSNNS)
library(ggplot2)
library(dplyr)
library(viridis)

#####
# QUESTAO 1#
#####
```

```
#Q1_1)Carregue a base de dados Satellite
data("Satellite")
#head(Satellite)
dados<-Satellite[,c(17,18,19,20,37)]
head(dados)
```

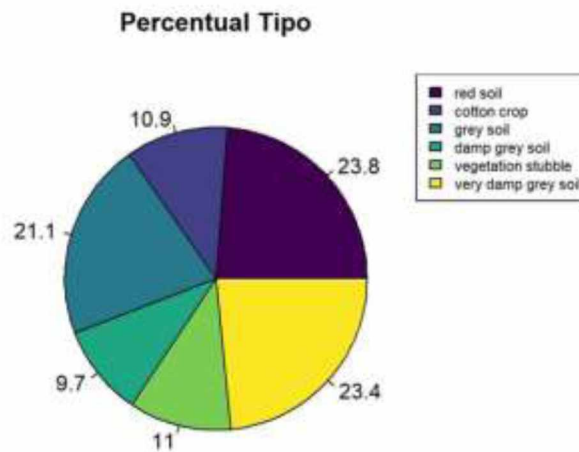
```
##   x.17 x.18 x.19 x.20  classes
## 1   92  112  118   85 grey soil
## 2   84  103  104   81 grey soil
## 3   84   99  104   78 grey soil
## 4   84   99  104   81 grey soil
## 5   76   99  104   81 grey soil
## 6   76   99  108   85 grey soil
```

1.2)

```
#Q1_2) Crie partições contendo 80% para treino 20% para teste:
set.seed(254)
indices<-createDataPartition(dados$classes, p=0.80, list=FALSE)
treino<-dados[indices,]
teste<-dados[-indices,]

t<-table(treino$classes)
porcentagens <-round(100*t/sum(t),1)
#cores<-c("red","cyan","gray","purple","green","yellow","pin")
cores <- viridis_pal(option = "D")(length(porcentagens))
pie(table(treino$classes), labels=porcentagens,main="Percentual Tipo",
col = cores)
legend("topright", legend=names(t), cex=0.7, fill = cores)
```

GRÁFICO 6 – DISTRIBUIÇÃO TIPO DE SOLO



FONTE: Amauri (2025).

1.3)

#Q1_33) Treine modelos RandomForest, SVM e RNA para predição destes dados:

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

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

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

1.4)

#Q1_4) Escolha o melhor modelo com base em suas matrizes de confusão

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

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

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

```
cm_rf<-confusionMatrix(predict.rf, teste$classes)
```

```
cm_svm<-confusionMatrix(predict.svm, teste$classes)
```

```
cm_rna<-confusionMatrix(predict.rna, teste$classes)
```

#Matrizes de confusão

```
cm_rf
```

```
##      predictions
## targets  1  2  3  4  5  6
##      1 301  0  2  1  8  0
##      2  0 128  0  3  5  1
##      3  2  0 248 28  0  8
##      4  0  1  14 58  1 46
##      5  3  8  0  2 116 10
##      6  0  3  7 33  11 236
```

```
cm_svm
```

```
##          predictions
## targets  1  2  3  4  5  6
##          1 302  0  1  1 11  0
##          2  0 126  0  0  4  1
##          3  2  0 260 27  0 12
##          4  0  1  9 68  1 34
##          5  2 11  0  2 113  8
##          6  0  2  1 27 12 246
```

```
cm_rna
```

```
##          predictions
## targets  1  2  3  4  5  6
##          1 299  0  7  1 15  0
##          2  1 118  0  0  1  0
##          3  2  0 258 44  0 22
##          4  1  0  5 23  1 14
##          5  3 19  0  3 111 14
##          6  0  3  1 54 13 251
```

1.5)

```
#Resultados Random Forest (RF)
tot_orig_rf <- apply(cm_rf,1,sum)
tot_pred_rf <- apply(cm_rf,2,sum)
sum_tot_rf <- sum(cm_rf)

#True Positive Rate = Taxa de Verdadeiro Positivo em Cada Categoria
# Do que originalmente era pertencente à categoria, quantos por cento o
modelo classificou corretamente como a categoria
tpr_rf = diag(cm_rf)/tot_orig_rf

#False Positive Rate = Taxa de Falso Positivo em cada Categoria
# Do que originalmente NÃO era a categoria, quantos por cento o modelo
classificou erroneamente como a categoria
fpr_rf = (tot_pred_rf-diag(cm_rf))/(sum_tot_rf - tot_orig_rf)

#Precisão ou Hit Rate = De tudo o que o modelo apontou como a categoria,
o que realmente acertou
hr_rf = diag(cm_rf) / tot_pred_rf

#F-Score
fsc_rf = 2*(hr_rf*tpr_rf)/(hr_rf+tpr_rf)

#Resultados SVM
tot_orig_svm <- apply(cm_svm,1,sum)
tot_pred_svm <- apply(cm_svm,2,sum)
sum_tot_svm <- sum(cm_svm)
```

```

#True Positive Rate = Taxa de Verdadeiro Positivo em Cada Categoria
# Do que originalmente era pertencente à categoria, quantos por cento o
modelo classificou corretamente como a categoria
tpr_svm = diag(cm_svm)/tot_orig_svm

#False Positive Rate = Taxa de Falso Positivo em cada Categoria
# Do que originalmente NÃO era a categoria, quantos por cento o modelo
classificou erroneamente como a categoria
fpr_svm = (tot_pred_svm-diag(cm_svm))/(sum_tot_svm - tot_orig_svm)

#Precisão ou Hit Rate = De tudo o que o modelo apontou como a categoria,
o que realmente acertou
hr_svm = diag(cm_svm) / tot_pred_svm

#F-Score
fsc_svm = 2*(hr_svm*tpr_svm)/(hr_svm+tpr_svm)

#Resultados Neural Network (RNA)
tot_orig_rna <- apply(cm_rna,1,sum)
tot_pred_rna <- apply(cm_rna,2,sum)
sum_tot_rna <- sum(cm_rna)

#True Positive Rate = Taxa de Verdadeiro Positivo em Cada Categoria
# Do que originalmente era pertencente à categoria, quantos por cento o
modelo classificou corretamente como a categoria
tpr_rna = diag(cm_rna)/tot_orig_rna

#False Positive Rate = Taxa de Falso Positivo em cada Categoria
# Do que originalmente NÃO era a categoria, quantos por cento o modelo
classificou erroneamente como a categoria
fpr_rna = (tot_pred_rna-diag(cm_rna))/(sum_tot_rna - tot_orig_rna)

#Precisão ou Hit Rate = De tudo o que o modelo apontou como a categoria,
o que realmente acertou
hr_rna = diag(cm_rna) / tot_pred_rna

#F-Score: melhor mais proximo de 1
fsc_rna = 2*(hr_rna*tpr_rna)/(hr_rna+tpr_rna)

#Comparacao de Acuracia entre os 3 Modelos
modelo <- c("Random Forest", "Support Vector Machine", "Neural Network")
acuracia <- c(sum(diag(cm_rf))/sum(cm_rf),
             sum(diag(cm_svm))/sum(cm_svm),
             sum(diag(cm_rna))/sum(cm_rna)
)

acuracia<-paste(round(acuracia,4)*100,"%")
df <- data.frame(modelo, acuracia)
df

```

```

#Comparação de sensibilidade, falso positivo, hit rate, F1Score por
grupo e modelo
modelos<-c(rep("RF",6),rep("SVM",6),rep("RNA",6))
grupo<-rep(c(1,2,3,4,5,6),6)
descricao<-rep(c("solo vermelho","colheita algodão","solo cinza","solo
cinza úmido","restolho de vegetação","solo cinza muito úmido"),6)
tpr<-round(c(tpr_rf, tpr_svm, tpr_rna),4)*100
fpr<-round(c(fpr_rf, fpr_svm, fpr_rna),4)*100
hr <-round(c(hr_rf, hr_svm, hr_rna),4)*100
fscore<-round(c(fsc_rf,fsc_svm,fsc_rna),4)
df2<-data.frame(modelos,grupo,descricao, tpr, fpr, hr, fscore)

sorted_df2 <- df2 %>%
  arrange(grupo, desc(tpr), fpr, desc(hr), fscore)

sorted_df2 <- distinct(sorted_df2)
sorted_df2

```

##	modelos	grupo	descricao	tpr	fpr	hr	fscore
## 1	RF	1	solo vermelho	96.47	0.51	98.37	0.9741
## 2	SVM	1	solo vermelho	95.87	0.41	98.69	0.9726
## 3	RNA	1	solo vermelho	92.86	0.73	97.71	0.9522
## 4	RNA	2	colheita algodão	98.33	1.89	84.29	0.9077
## 5	SVM	2	colheita algodão	96.18	1.21	90.00	0.9299
## 6	RF	2	colheita algodão	93.43	1.05	91.43	0.9242
## 7	RF	3	solo cinza	86.71	2.30	91.51	0.8905
## 8	SVM	3	solo cinza	86.38	1.12	95.94	0.9091
## 9	RNA	3	solo cinza	79.14	1.36	95.20	0.8643
## 10	SVM	4	solo cinza úmido	60.18	4.87	54.40	0.5714
## 11	RNA	4	solo cinza úmido	52.27	8.23	18.40	0.2722
## 12	RF	4	solo cinza úmido	48.33	5.76	46.40	0.4735
## 13	RF	5	restolho de vegetação	83.45	2.18	82.27	0.8286
## 14	SVM	5	restolho de vegetação	83.09	2.44	80.14	0.8159
## 15	RNA	5	restolho de vegetação	74.00	2.65	78.72	0.7629
## 16	SVM	6	solo cinza muito úmido	85.42	5.52	81.73	0.8353
## 17	RF	6	solo cinza muito úmido	81.38	6.54	78.41	0.7986
## 18	RNA	6	solo cinza muito úmido	77.95	5.20	83.39	0.8058

Com base nas matrizes de confusão, no geral, o melhor modelo foi o SVM com acurácia de 86%. Mas dependendo do interesse do pesquisador esse resultado pode ser alterado e avaliado conforme os seguintes indicadores:

- Acurácia Geral do Modelo = Quantos por cento do total da população o modelo classificou corretamente.

- True Positive Rate (TPR) = Sensibilidade = Do que originalmente era pertencente à categoria, quantos por cento o modelo classificou corretamente como a categoria;
- False Positive Rate (FPR) = Falso Positivo = Do que originalmente não era a categoria, quantos por cento o modelo classificou erroneamente como a categoria;
- Precisão ou Hit Rate = De tudo o que o modelo apontou como a categoria, o que realmente acertou;
- Fscore = F1Score

2.1)

```
#####
#QUESTAO 2 #
#####

#Q2_1)Carregar o arquivo Volumes.csv
(http://www.razer.net.br/datasets/Volumes.csv)

# Load necessary library for downloading data from URL
library(readr)

# URL of the CSV file
url <- "http://www.razer.net.br/datasets/Volumes.csv"

# Read CSV file from the URL
data <- read.csv2(url, sep = ";")

# Display the first few rows of the data
head(data)

##   NR  DAP   HT  HP     VOL
## 1  1 34.0 27.00 1.80 0.8971441
## 2  2 41.5 27.95 2.75 1.6204441
## 3  3 29.6 26.35 1.15 0.8008181
## 4  4 34.3 27.15 1.95 1.0791682
## 5  5 34.5 26.20 1.00 0.9801112
## 6  6 29.9 27.10 1.90 0.9067022
```

2.2)

```
#Q2_2)Eliminar a coluna NR, que só apresenta um número sequencia
df <- subset(data, select = -NR)
```

2.3)

```
#Q2_3)Criar partição de dados: treinamento 80%, teste 20%
```

```

set.seed(254)
indices<-createDataPartition(df$VOL, p=0.80, list=FALSE)
treino<-df[indices,]
teste<-df[-indices,]

```

2.4)

```

#Q2_4)Usando o pacote "caret", treinar os modelos: Random Forest (rf),
SVM (svmRadial), Redes Neurais (neuralnet) e o modelo alométrico de
SPURR
alom <- nls(VOL ~ b0 + b1*DAP*DAP*HT,df,start=list(b0=0.5, b1=0.5))
rf<-caret::train(VOL~., data=treino, method="rf")
svm<-caret::train(VOL~.,data=treino,method="svmRadial")
rna<-caret::train(VOL~.,data=treino,method="nnet", trace = FALSE)

```

2.5)

```

#Q2_5)Efetue as previsões nos dados de teste
predict.rf <-predict(rf,teste)
predict.svm<-predict(svm,teste)
predict.rna<-predict(rna,teste)
predict_alom <- exp(predict(alom, newdata = teste, interval =
"confidence"))

```

2.6)

```

#Q2_6)Crie suas próprias funções (UDF) e calcule as seguintes métricas
entre a previsão e os dados observados
# Definir a função para calcular o coeficiente de determinação
coeficiente_determinacao <- function(y_obs, y_pred) {
  # Calcular a média observada de y
  y_medio <- mean(y_obs)

  # Calcular a soma dos quadrados totais
  SST <- sum((y_obs - y_medio)^2)

  # Calcular a soma dos quadrados dos resíduos
  SSR <- sum((y_obs - y_pred)^2)

  # Calcular o coeficiente de determinação
  R2 <- 1 - (SSR / SST)

  return(R2)
}

r2rf <- coeficiente_determinacao(teste$VOL,predict.rf)
r2svm <- coeficiente_determinacao(teste$VOL,predict.svm)
r2rna <- coeficiente_determinacao(teste$VOL,predict.rna)
r2alom <- coeficiente_determinacao(teste$VOL,predict_alom)

erro_padrao_est<-function(y_obs,y_pred){

```

```

n<- length(y_obs)

# Calcular a soma dos quadrados dos resíduos
SSR <- sum((y_obs - y_pred)^2)

Sxy = sqrt(SSR/(n-2))

return(Sxy)
}

ep_rf <-erro_padrao_est(teste$VOL,predict.rf)
ep_svm <-erro_padrao_est(teste$VOL,predict.svm)
ep_rna <-erro_padrao_est(teste$VOL,predict.rna)
ep_alom <-erro_padrao_est(teste$VOL,predict_alom)

sxy_perc<-function(y_obs, sxy){
  y_bar <-mean(y_obs)
  sxy_perc <- sxy/y_bar*100
  return(sxy_perc)
}

sxyp_rf <-sxy_perc(teste$VOL,ep_rf)
sxyp_svm <-sxy_perc(teste$VOL,ep_svm)
sxyp_rna <-sxy_perc(teste$VOL,ep_rna)
sxyp_alom <-sxy_perc(teste$VOL,ep_alom)

```

2.7)

```

#Q2_7) Escolha o melhor modelo.
df3<-data_frame(modelos = c("RF","SVM","RNA","Alom"), Rsq =
c(r2rf,r2svm,r2rna,r2alom), EP =c(ep_rf,ep_svm,ep_rna,ep_alom),SxyPerc =
c(sxyp_rf,sxyp_svm,sxyp_rna,sxyp_alom) )

```

```

df3
## # A tibble: 4 × 4
##   modelos      Rsq    EP SxyPerc
##   <chr>      <dbl> <dbl>  <dbl>
## 1 RF         0.839 0.148   11.2
## 2 SVM         0.685 0.207   15.6
## 3 RNA        -0.861 0.503   38.0
## 4 Alom       -80.2  3.32   251.

```

O melhor modelo é o Random Forest por apresentar melhor coeficiente de determinação, ou seja, melhor acurácia. Além disso, apresenta menor erro padrão e menor percentual do erro.

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

(15 pontos) Elaborar a tabela de 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)

```
# TRABALHO DE IAA004 - Estatística

```{r libs, echo=FALSE, warning=FALSE}
#Pacotes:
library(readxl)
```

```

library(ggplot2)
library(tidyverse)
library(plotly)
library(car)
library(fdth)
library(dplyr)
library(stats)
library(coin)
library(dplyr)
library(ggpubr)
```



```

```{r}
#Diretório
setwd("C:/0LETICIA/IAA/IAA004_EST1/0_PRATICAS")
```

1 Gráficos e tabelas

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

```{r pressure, echo=FALSE}
load("salarios.RData")
summary(salarios)
```

```{r}
# Vamos desenhar o histograma com o metodo de sturges (para a quantidade
de quebras) da variavel "age" que eh a idade dos esposas
hist(salarios$age, xlab="Idade das Esposas", breaks = 8,
      ylab = "Frequência", main = "Histograma Idade das Esposas",
      col="pink")
```

```{r}
# Vamos desenhar o histograma com o metodo de sturges (para a quantidade
de quebras) da variavel "husage" que eh a idade dos maridos
hist(salarios$husage, xlab="Idade dos Maridos", ylab = "Frequência",
      main = "Histograma Idade dos Maridos", col = "lightblue")
```

```{r}
# Calculando as densidades kernel para cada grupo
densidade_esposas <- density(salarios$age)
densidade_maridos <- density(salarios$husage)

# Encontrando os limites do gráfico
limite_x <- range(c(densidade_esposas$x, densidade_maridos$x))

```


```

```

limite_y <- range(c(densidade_esposas$y, densidade_maridos$y))

Criando o gráfico vazio
plot(NULL, xlim = limite_x, ylim = limite_y,
 main = "Densidades de Idades Marido e Esposas",
 xlab = "Valores", ylab = "Densidade")

Adicionando as curvas de densidade de cada grupo ao gráfico
lines(densidade_esposas,lwd = 2, col = "pink")
lines(densidade_maridos,lwd = 2, col = "lightblue")

Adicionando a legenda
legend("topright", legend = c("Esposas", "Maridos"), fill = c("pink",
"lightblue"))
```


1.a): Na base salários a idade das esposas varia menos do que a idade dos esposos, as mulheres tem idades entre 18 e 59 anos enquanto os homens apresentam idade de 19 a 86 anos. Em média uma esposa tem 39 anos enquanto a média de idade dos homens é de 42,5 anos aproximadamente. Um quarto das mulheres tem até 31 anos, metade até 39 anos, 75% até 47 anos. Para o grupo masculino, um quarto tem até 34 anos, metade até 41 anos e 75% até 50 anos, homens com mais de 75 anos são considerados outliers.



```

```{r}
boxplot(list(esposas = salarios$age, maridos = salarios$husage),
        col = c("pink", "lightblue"),
        main = "Boxplot Idades Esposas e Maridos",
        ylab = "Idade")

# Adicionando uma legenda
legend("topright", legend = c("Esposas", "Maridos"), fill = c("pink",
"lightblue"))
```

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

```

```{r cars}
print("Idade dos Maridos")
idademarido <- fdt(salarios$husage)
print (idademarido)

print("Idade das Esposas")
idadeesposa <- fdt(salarios$age)
print (idadeesposa)
```

1.b): Na base salários a idade dos maridos varia de 18,81 a 86,86 anos, a faixa etária dos homens com maior frequência é a de 38 a 43 anos com

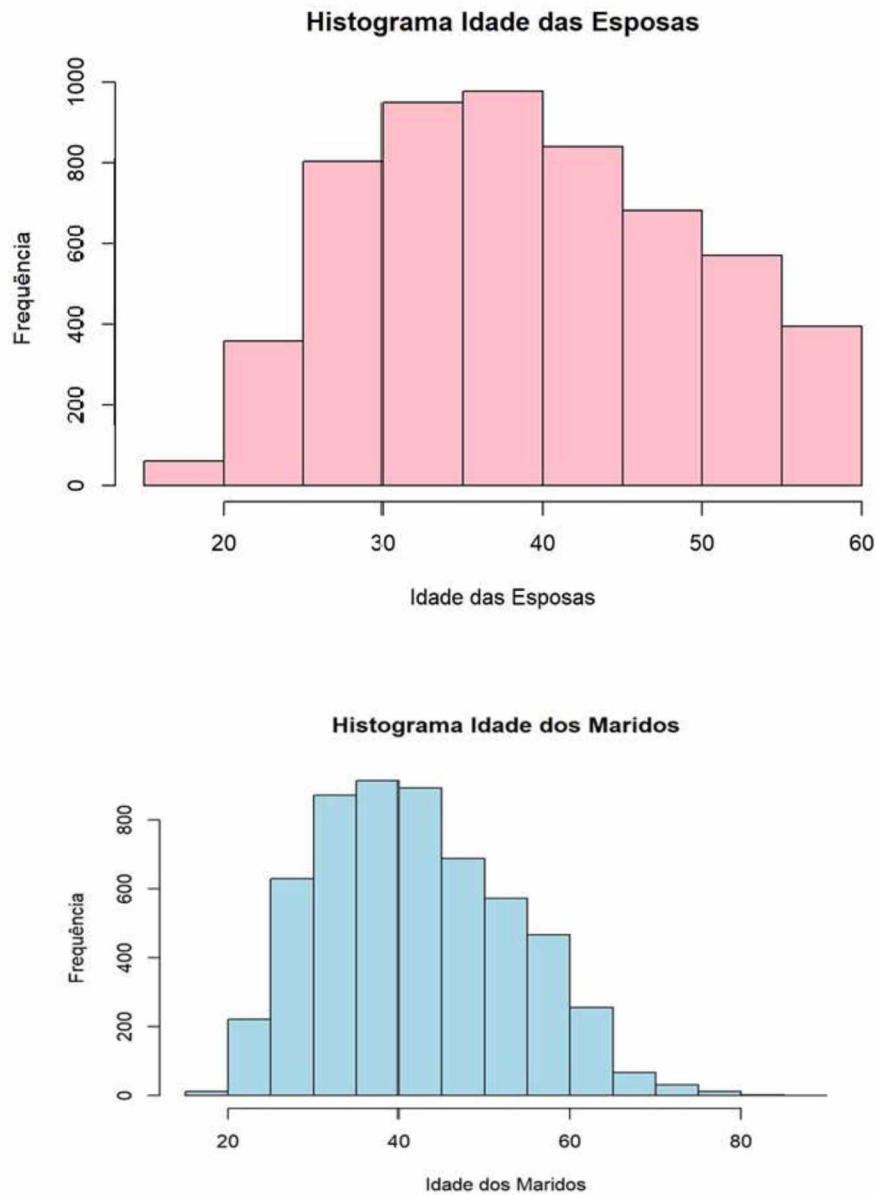
```


```


```

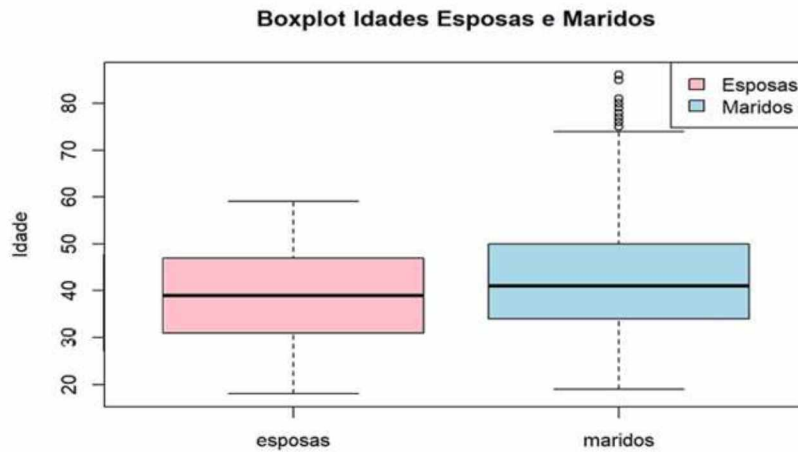
917 registros, é nesse intervalo que encontra-se a mediana do grupo, pois até o limite superior dessa classe acumula-se 56,6% das observações relacionadas aos maridos, há apenas 8 indivíduos homens com idade maior ou igual a 77,14 anos, demonstrando que seriam outliers. Quanto às esposas, menos de 10% delas tem idade inferior a 26,77 anos; a classe de maior frequência é a de 35,72 a 38,70 anos, a mediana está incluída na faixa etária de 38,71 a 41,68 anos.

GRÁFICO 7 – HISTOGRAMAS IDADES



FONTE: Amauri (2025).

GRÁFICO 8 – BOXPLOT IDADES



FONTE: Amauri (2025).

2)

#2.a): Todos os valores das medidas resumo das esposas são inferiores aos dos esposos. Enquanto a média de idade da esposa é de 39,43 anos a do marido é de 42,45 anos. Metade do grupo de esposas tem idade até 39 anos, já para o grupo de maridos a idade que divide o grupo em 50% é 41 anos. Entre as esposas a idade com maior frequência é 37 anos, entre os maridos a idade mais frequente é 44 anos.

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

```
``{r}
```

```
# Calculando variância, desvio padrão e coeficiente de variação para cada grupo
```

```
var_grupo1 <- var(salarios$age)
```

```
dp_grupo1 <- sd(salarios$age)
```

```
cv_grupo1 <- 100 * (dp_grupo1 / mean(salarios$age))
```

```
var_grupo2 <- var(salarios$husage)
```

```
dp_grupo2 <- sd(salarios$husage)
```

```
cv_grupo2 <- 100 * (dp_grupo2 / mean(salarios$husage))
```

```
# Criando uma tabela
```

```
tabela2 <- data.frame(
```

```
  Grupo = c("Esposas", "Maridos"),
```

```
  Variância = c(var_grupo1, var_grupo2),
```

```
  Desvio_Padrão = c(dp_grupo1, dp_grupo2),
```

```
  Coeficiente_de_Variação = c(cv_grupo1, cv_grupo2)
```

```
)
```

```

# Exibindo a tabela
tabela2
```

2.b): A variância da idade dos maridos é 1,26 vezes a variância da idade das esposas. Logo, a idade das esposas também apresenta um desvio padrão inferior ao da idade dos maridos. Enquanto as esposas apresentam um coeficiente de variação de 25,33% na idade, os maridos apresentam idade com coeficiente de variação de 26,45%.

3 Testes paramétricos ou não paramétricos
a) (40 pontos) Testar se as médias (se você escolher o teste paramétrico) ou as medianas (se você escolher o teste não paramétrico) das variáveis “age” (idade da esposa) e “husage” (idade do marido) são iguais, construir os intervalos de confiança e comparar os resultados.
Obs:
1) Você deve fazer os testes necessários (e mostra-los no documento pdf) para saber se você deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney (não paramétrico), justifique sua resposta sobre a escolha.
2) Lembre-se de que os intervalos de confiança já são mostrados nos resultados dos testes citados no item 1 acima.

```{r}
options(scipen = 999, digits = 10)

# Teste de normalidade para cada grupo usando Kolmogorov-Smirnov Test, pois n>5000
teste_grupo1 <- ks.test(salarios$age, "pnorm")
teste_grupo2 <- ks.test(salarios$husage, "pnorm")

# Criando uma tabela
tabela3 <- data.frame(
  Grupo = c("Esposas", "Maridos"),
  KS_p_value = c(teste_grupo1$p.value, teste_grupo2$p.value),
  Normalidade = c(ifelse(teste_grupo1$p.value > 0.05, "Normal", "Não normal"),
                 ifelse(teste_grupo2$p.value > 0.05, "Normal", "Não normal"))
)

# Exibindo a tabela
tabela3

##      Grupo      Média Mediana Moda
## 1 Esposas 39.42758      39    37
## 2 Maridos 42.45296      41    44

```

##	Grupo	Variância	Desvio_Padrão	Coeficiente_de_Variação
## 1	Esposas	99.75234	9.98761	25.33153
## 2	Maridos	126.07173	11.22817	26.44849

3)

```
### 3 Testes paramétricos ou não paramétricos
**a) (40 pontos) Testar se as médias (se você escolher o teste
paramétrico) ou as medianas (se você
escolher o teste não paramétrico) das variáveis "age" (idade da esposa)
e "husage" (idade do
marido) são iguais, construir os intervalos de confiança e comparar os
resultados.**
Obs:
1) Você deve fazer os testes necessários (e mostra-los no documento pdf)
para saber se você
deve usar o unpaired test (paramétrico) ou o teste U de Mann-Whitney
(não paramétrico),
justifique sua resposta sobre a escolha.
2) Lembre-se de que os intervalos de confiança já são mostrados nos
resultados dos testes
citados no item 1 acima.

```{r}
options(scipen = 999, digits = 10)

Teste de normalidade para cada grupo usando Kolmogorov-Smirnov Test,
pois n>5000
teste_grupo1 <- ks.test(salarios$age, "pnorm")
teste_grupo2 <- ks.test(salarios$husage, "pnorm")

Criando uma tabela
tabela3 <- data.frame(
 Grupo = c("Esposas", "Maridos"),
 KS_p_value = c(teste_grupo1$p.value, teste_grupo2$p.value),
 Normalidade = c(ifelse(teste_grupo1$p.value > 0.05, "Normal", "Não
normal"),
 ifelse(teste_grupo2$p.value > 0.05, "Normal", "Não
normal"))
)

Exibindo a tabela
tabela3
```

3.a) Nos testes de normalidade Kolmogorov-Smirnov para a idade das
esposas o p-valor obtido foi 0.000000000000000022204, ou seja, muito
menor do que 0.05 indicando a não normalidade dos dados. O mesmo se deu
para as idades dos homens, ou seja, para ambas as variáveis não há
normalidade, indicando necessidade de realizar teste não paramétrico U
de Mann-Whitney para comparação de medianas.
```

```

```{r}
age <- data.frame(
 group = rep(c("Esposa", "Marido"), each = 5634),
 age = c(salarios$age, salarios$husage)
)

Para evitar a notacao cientifica
options(scipen = 999)

Vamos fazer o teste se a idade mediana dos maridos eh igual a
idade mediana das mulheres

Hipoteses do teste:

H0: A idade mediana dos homens eh igual estatisticamente a idade
mediana das mulheres;
Ha: A idade mediana dos homens nao eh estatisticamente igual a
idade mediana das mulheres

O teste eh sempre feito com relacao a disposicao no vetor de
dados, do ultimo para o primeiro - No vetor: Man contra Woman
res <- wilcox.test(age ~ group, data = age,
 exact = TRUE, conf.int=TRUE)
res

O p-value do teste eh 0.00000000000000022204, que eh menor que o nivel
de
significancia 0,05. Podemos concluir que a idade mediana dos
maridos eh estatisticamente diferente da idade mediana das
mulheres (rejeitamos H0).
O intervalo de confianca da diferenca entre as medianas esta
entre -3,00 e -2.00, com uma mediana de -2,99.

...

3.a) Foi testada a hipótese nula das medianas das idades dos maridos e
esposas serem iguais, mas o teste U de Mann Whitney rejeitou a hipótese
nula apresentando um p-valor de 0.00000000000000022204, muito inferior
ao nivel de significância 0,05. Assim, podemos concluir que a idade
mediana dos
maridos é estatisticamente diferente da idade mediana da idade das
mulheres. O intervalo de confiança da diferenca entre as medianas está
entre -3,00 e -2.00, com uma mediana de -2,99.

Grupo KS_p_value Normalidade
1 Esposas 0 Não normal
2 Maridos 0 Não normal

```

```

Wilcoxon rank sum test with continuity correction

data: age by group
W = 13619912, p-value < 0.0000000000000022204
alternative hypothesis: true location shift is not equal to 0
95 percent confidence interval:
-3.000024415 -2.000032719
sample estimates:
difference in location
-2.99996616
```

## APÊNDICE E - ESTATÍSTICA APLICADA II

### A – ENUNCIADO

#### Regressões Ridge, Lasso e ElasticNet

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

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

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

```
#####
Unidade 2: Modelos de regularizacao ou penalidade
Secao 2.2: Regressao Ridge
#####

Instalando os pacotes necessarios
install.packages("plyr")
install.packages("readr")
install.packages("dplyr")
install.packages("caret")
install.packages("ggplot2")
install.packages("repr")
install.packages("glmnet")
```

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

setwd("C:/Users/Amauri/Desktop/IAA/Estatistica 2")

load("trabalhosalarios.RData")

Vamos guardar este dataset em um objeto chamaddo "dat"
dat <- trabalhosalarios

Vamos ver o dataset inteiro
View(dat)

Vamos visualizar parte do dataset
glimpse(dat)

Vamos ver como esta nossa memoria
gc()

set.seed(302)

index = sample(1:nrow(dat),0.8*nrow(dat))

Vamos criar a base de dados de treinamento
train = dat[index,]

Vamos criar a base de dados de teste
test = dat[-index,]

dim(train)
dim(test)

cols = c('husage', 'husearns', 'huseduc', 'hushrs', 'earns',
 'age', 'educ', 'exper', 'lwage')

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

Vamos ver o sumario estatistico das variaveis
padronizadas de cada dataset
```

```

summary(train)
summary(test)

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

A regressao Ridge "encolhe os valores dos coeficientes"

Vamos criar um objeto com as variaveis que usaremos no
modelo
cols_reg = c('husage', 'husunion', 'husearns', 'huseduc', 'hushrs',
 'earns', 'age', 'educ', 'exper', 'husblck',
 'hushisp', 'kidge6', 'black', 'hispanic',
 'union', 'kidlt6', 'lwage')

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

x = as.matrix(train_dummies)

y_train = train$lwage

x_test = as.matrix(test_dummies)

y_test = test$lwage

lambdas <- 10^seq(2, -3, by = -.1)
Calculando o lambda:
ridge_lamb <- cv.glmnet(x, y_train, alpha = 0,
 lambda = lambdas)
Vamos ver qual o lambda otimo
best_lambda_ride <- ridge_lamb$lambda.min
best_lambda_ride

Para contar o tempo de processamento do modelo usamos:
start <- Sys.time()
Estimando o modelo Ridge
ridge_reg = glmnet(x, y_train, nlambda = 25, alpha = 0,
 family = 'gaussian',
 lambda = best_lambda_ride)
end <- Sys.time()

```

```

difftime(end, start, units="secs")

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

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

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

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

As metricas da base de treinamento sao:
eval_results_ride(y_train, predictions_train, train)

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

As metricas da base de teste sao:
dt_results = eval_results_ride(y_test, predictions_test, test)

husage = (40-pre_proc_val[["mean"]][["husage"]])/
 pre_proc_val[["std"]][["husage"]]

husearns = 551 (rendimento do marido em US$)
husearns = (600-pre_proc_val[["mean"]][["husearns"]])/
 pre_proc_val[["std"]][["husearns"]]

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

#husunion

```

```

husunion = 0

husblck = 0 (o marido nao eh preto)
husblck = 1

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

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

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

earns = 355.5 (rendimento da esposa em US$)
earns = (355.5-pre_proc_val[["mean"]][["earns"]])/
 pre_proc_val[["std"]][["earns"]]

age = 37 anos (idade da esposa)
age = (38-pre_proc_val[["mean"]][["age"]])/
 pre_proc_val[["std"]][["age"]]

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

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

hispanic = 0 (esposa nao eh hispanica)
hispanic = 1

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

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

exper = 18 (experiencia)
exper = (18-pre_proc_val[["mean"]][["exper"]])/
 pre_proc_val[["std"]][["exper"]]

Vamos construir uma matriz de dados para a predicao
our_pred = as.matrix(data.frame(husage=husage,
 husearns=husearns,
 husunion=husunion,
 huseduc=huseduc,
 husblck=husblck,
 hushisp=hushisp,

```

```

 hushrs=hushrs,
 kidge6=kidge6,
 earns=earns,
 age=age,
 black=black,
 educ=educ,
 hispanic=hispanic,
 union=union,
 exper=exper,
 kidlt6=kidlt6))

Fazendo a predicao:
predict_our_ridge <- predict(ridge_reg,
 s = best_lambda_ridge,
 newx = our_pred)

O resultado da predicao eh:
predict_our_ridge

O resultado eh um valor padronizado, vamos converte-lo
para o valor nominal, consistente com o dataset original
wage_pred_ridge=(predict_our_ridge*
 pre_proc_val[["std"]][["lwage"]])+
 pre_proc_val[["mean"]][["lwage"]]

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

O intervalo de confianca para o nosso exemplo eh:
n <- nrow(train) # tamanho da amostra
m <- wage_pred_ridge # valor medio predito
s <- pre_proc_val[["std"]][["lwage"]] # desvio padrao
dam <- s/sqrt(n) # distribuicao da amostragem da media
CIlwr_ridge <- m + (qnorm(0.025))*dam # intervalo inferior
CIupr_ridge <- m - (qnorm(0.025))*dam # intervalo superior

Os valores sao:
CIlwr_ridge
CIupr_ridge

Entao, segundo as caracteristicas que atribuimos o
salario-hora da esposa eh em media US$2.12 e pode
variar entre US$2.10 e US$2.15

#####

Vamos atribuir alpha = 1 para implementar a regressao
lasso
lasso_lamb <- cv.glmnet(x, y_train, alpha = 1,

```

```

 lambda = lambdas,
 standardize = TRUE, nfolds = 5)

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

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

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

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

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

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

As metricas da base de treinamento sao:
eval_results_lasso(y_train, predictions_train, train)

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

As metricas da base de teste sao:
dt_results <- bind_rows(dt_results, eval_results_lasso(y_test,
predictions_test, test))

Vamos para a predicao
predict_our_lasso <- predict(lasso_model,

```

```

 s = best_lambda_lasso,
 newx = our_pred)
predict_our_lasso

Novamente, o resultado esta padronizado, nos temos de
converte-lo para valor compativel com o dataset original
wage_pred_lasso=(predict_our_lasso*
 pre_proc_val[["std"]][["lwage"]])+
 pre_proc_val[["mean"]][["lwage"]]

wage_pred_lasso
Portanto, o salario-hora da esposa eh US$2.15

Vamos criar o intervalo de confianca para o nosso
exemplo
n <- nrow(train)
m <- wage_pred_lasso
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_lasso <- m + (qnorm(0.025))*dam
CIupr_lasso <- m - (qnorm(0.025))*dam

O intervalo de confianca eh:
CIlwr_lasso
CIupr_lasso
Entao, o salario medio eh de US$2.15e pode variar
entre US$2.13 e US$2.18

#####

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

Nos nao temos o parametro "alpha", porque a regressao
ElasticNet vai encontra-lo automaticamente, cujo valor
estara entre 0 e 1 (para Ridge ==> alpha = 0; e
para Lasso ==> alpha = 1).
O parametro "lambda" tambem eh escolhido por
cross-validation

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

```

```

trControl = train_cont)

O melhor parametro alpha escolhido eh:
elastic_reg$bestTune

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

Vamos fazer as predicoes e avaliar a performance do
modelo

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

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

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

As metricas de performance na base de treinamento
sao:
eval_results_elastic(y_train, predictions_train, train)

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

As metricas de performance na base de teste sao:
dt_results <- bind_rows(dt_results, eval_results_elastic(y_test,
predictions_test, test))

predict_our_elastic <- predict(elastic_reg,our_pred)
predict_our_elastic

Novamente, o resultado eh padronizado, nos temos que
reverte-lo para o nivel dos valores originais do
dataset, vamos fazer isso:
wage_pred_elastic=(predict_our_elastic*
 pre_proc_val[["std"]][["lwage"]])+
pre_proc_val[["mean"]][["lwage"]]

```

```

wage_pred_elastic

Vamos criar o intervalo de confianca para o nosso
exemplo
n <- nrow(train)
m <- wage_pred_elastic
s <- pre_proc_val[["std"]][["lwage"]]
dam <- s/sqrt(n)
CIlwr_elastic <- m + (qnorm(0.025))*dam
CIupr_elastic <- m - (qnorm(0.025))*dam

Os valores minimo e maximo sao:
CIlwr_elastic
CIupr_elastic
Entao, o salario-hora medio da esposa eh de US$2.22
e pode variar entre US$2.20 e US$2.24

wage_pred = c(wage_pred_lasso, wage_pred_elastic)
dt_results <- cbind(dt_results, wage_pred)

Lrwage_pred = c(CIlwr_lasso, CIlwr_elastic)
dt_results <- cbind(dt_results, Lrwage_pred)

Uprwage_pred = c(CIupr_lasso, CIupr_elastic)
dt_results <- cbind(dt_results, Uprwage_pred)

library(gridExtra)
library(ggplot2)

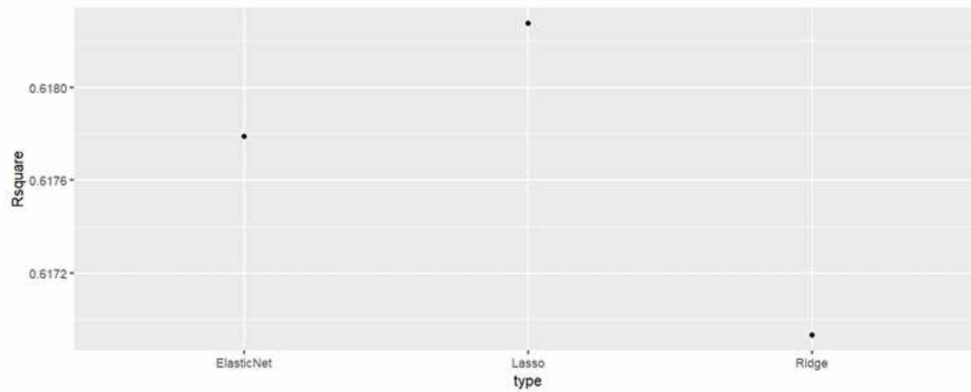
#define scatterplot
my_plot <- ggplot(dt_results, aes(x=type, y=Rsquare)) +
 geom_point()

#define table
my_table <- tableGrob(dt_results)

#create scatterplot and add table underneath it
grid.arrange(my_plot, my_table)

```

GRÁFICO 9 – RESOLUÇÃO EXERCÍCIO ESTATÍSTICA APLICADA II



	RMSE	Rsquare	type	wage_pred	Lwage_pred	Uprwage_pred
1	0.7113694	0.6169332	Ridge	2.132727	2.106823	2.154585
2	0.7101219	0.6182756	Lasso	2.171290	2.136468	2.193149
3	0.7105757	0.6177876	ElasticNet	2.186330	2.201853	2.208188

FONTE: Amauri (2025).

Baseado nos resultados observados e usando a medida Rsquare o modelo que demonstrou o melhor resultado foi o Lasso.

## APÊNDICE F - ARQUITETURA DE DADOS

### A – ENUNCIADO

#### 1 Construção de Características: Identificador automático de idioma

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

Parta do exemplo (notebook produzido no Colab) que foi 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)

```
O objetivo desse trabalho é demonstrar o processo de "construção de
atributos" e como ele é fundamental para o Reconhecimento de Padrões
(RP).
Primeiro um conjunto de "amostras" previamente conhecido (classificado)
amostras de texto em diferentes línguas

ingles = [
"Hello, how are you?", "I love to read books.", "The weather is nice
today.", "Where is the nearest restaurant?", "What time is it?", "I
enjoy playing soccer.", "Can you help me with this?"
]

espanhol = [
"Hola, ¿cómo estás?", "Me encanta leer libros.", "El clima está
agradable hoy.", "¿Dónde está el restaurante más cercano?", "¿Qué hora
es?", "Voy al parque todos los días.", "¿Puedes ayuda
]

portugues = [
"Estou indo para o trabalho agora.", "Adoro passar tempo com minha
família.", "Preciso comprar leite e pão.", "Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.", "O tr
]

A "amostras" de texto precisa ser "transformada" em padrões
Um padrão é um conjunto de características, geralmente representado por
um vetor e um conjunto de padrões no formato de tabela. Onde
cada linha é um padrão e as colunas as características e, geralmente, na
última coluna a classe

import random
pre_padroes = []

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

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

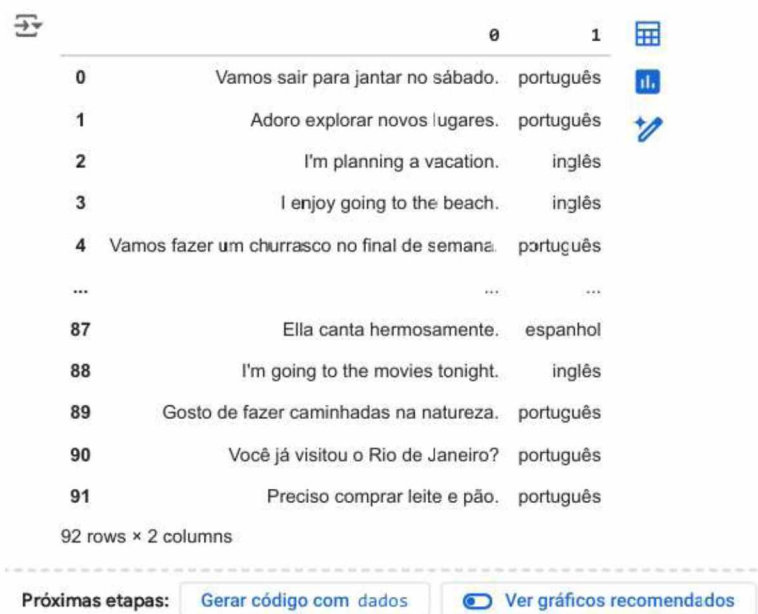
for frase in portugues:
pre_padroes.append([frase, 'português'])

random.shuffle(pre_padroes)
print(pre_padroes)
```

```
[['Vamos sair para jantar no sábado.', 'português'], ['Adoro explorar novos lugares.', 'português'], ["I'm planning a vacation.", 'inglês'], ['I enjoy going to the beach.', 'inglês'],
```

```
O DataFrame do pandas facilita a visualização.
import pandas as pd
dados = pd.DataFrame(pre_padroes)
dados
```

FIGURA 1 – VISUALIZAÇÃO DADOS EXERCÍCIO 1 ARQUITETURA DE DADOS



	0	1
0	Vamos sair para jantar no sábado.	português
1	Adoro explorar novos lugares.	português
2	I'm planning a vacation.	inglês
3	I enjoy going to the beach.	inglês
4	Vamos fazer um churrasco no final de semana.	português
...	...	...
87	Ella canta hermosamente.	espanhol
88	I'm going to the movies tonight.	inglês
89	Gosto de fazer caminhadas na natureza.	português
90	Você já visitou o Rio de Janeiro?	português
91	Preciso comprar leite e pão.	português

92 rows x 2 columns

Próximas etapas: [Gerar código com dados](#) [Ver gráficos recomendados](#)

FONTE: Amauri (2025).

```
#
import re
from collections import Counter
def tamanhoMedioFrases(texto):
 palavras = re.split("\\s", texto)
 tamanhos = [len(s) for s in palavras if len(s) > 0]
 soma = sum(tamanhos)
 return soma / len(tamanhos)
def funcao_caracteres(frase):
 # Frequência de caracteres específicos para idiomas
 caracteristicas = {}
 max_caracteres = 0
 caracteres_especificos = {
 'espanhol': 'ñáéíóúü',
 'portugues': 'çãõáéíóúà',
 'ingles': ''
```

```

 }
 for idioma, chars in caracteres_especificos.items():
 caracteristicas[idioma] = sum(frase.count(c) for c in chars) /
len(frase)
 if caracteristicas[idioma] > max_caracteres:
 max_caracteres = caracteristicas[idioma]
 return max_caracteres
def funcao_sufixos(texto):
 # Dicionário com sufixos comuns para cada idioma
 sufixos = {
 'portugues': ['ção', 'mente', 'idade', 'ência', 'dade', 'agem',
'al', 'eza', 'ia', 'ura', 'ar', 'ivo', 'oso', 'ante', 'ente', 'ável',
'ável', 'ico', 'ista', 'ístico', 'es', 'ísim
 'ingles': ['tion', 'ment', 'ness', 'ity', 'ing', 'ly', 'able',
'ible', 'ful', 'less', 'ism', 'ist', 'ance', 'ence', 'hood', 'ship',
'sion', 'ous', 'ive', 'al', 'en', 'er', 'ish',
 'espanhol': ['ción', 'ancia', 'encia', 'idad', 'tud', 'umbre',
'aje', 'dor', 'ero', 'ista', 'ura', 'mento', 'al', 'ar', 'il', 'ino',
'oso', 'ico', 'ivo', 'able', 'ible', 'nte']
 }

 # Dicionário para contar as frequências de sufixos encontrados para
cada idioma
 frecuencias_sufixos = {lang: 0 for lang in sufixos}
 max_sufixos = 0
 idioma_sufixo = ""
 valor = 0
 # Normalizar texto para minúsculas para evitar problemas de case-
sensitivity
 texto = texto.lower()

 # Calcular a frequência de cada sufixo no texto para cada idioma
 total_sufixos = 0 # Contador total de sufixos encontrados, para
calcular a frequência relativa
 for idioma, lista_sufixos in sufixos.items():
 for sufixo in lista_sufixos:
 count = len(re.findall(r'\b\w*' + re.escape(sufixo) + r'\b',
texto))
 frecuencias_sufixos[idioma] += count
 total_sufixos += count

 # Convertendo contagens em frequências relativas
 if total_sufixos > 0:
 for idioma in frecuencias_sufixos:
 frecuencias_sufixos[idioma] /= total_sufixos
 if frecuencias_sufixos[idioma] > max_sufixos:
 max_sufixos = frecuencias_sufixos[idioma]
 idioma_sufixo = idioma

 if idioma_sufixo == "portugues":

```

```

 valor = 0
 elif idioma_sufixo == "ingles":
 valor = 1
 else:
 valor = 2

 return valor

def extrai_ngrams(texto, n=3):
 # Limpeza básica do texto para remover pontuações e números
 texto = re.sub(r'^\w\s', '', texto).replace('_', '')

 # Criar n-grams de caracteres
 ngrams = [texto[i:i+n] for i in range(len(texto)-n+1)]
 return ngrams

def frequencia_ngrams(texto, n=3):
 # Extrair n-grams
 ngrams = extrai_ngrams(texto, n)

 # Contar frequência de cada n-gram
 frequencia = Counter(ngrams)

 # Total de n-grams para cálculo de frequências relativas
 total_ngrams = sum(frequencia.values())

 # Converter contagem em frequência
 if total_ngrams > 0:
 frequencia = {k: v / total_ngrams for k, v in
 frequencia.items()}

 return frequencia

Construir os perfis de n-grams para cada idioma
dados_treinamento = pre_padroes
perfis_idiomas = {}
n = 3 # Define o tamanho do n-gram
for texto, idioma in dados_treinamento:
 if idioma not in perfis_idiomas:
 perfis_idiomas[idioma] = Counter()
 perfil_ngrams = frequencia_ngrams(texto, n)
 perfis_idiomas[idioma] += Counter(perfil_ngrams)
Normalizar os contadores para frequências
for idioma in perfis_idiomas:
 total_ngrams = sum(perfis_idiomas[idioma].values())
 perfis_idiomas[idioma] = {k: v / total_ngrams for k, v in
 perfis_idiomas[idioma].items()}

def identifica_idioma(texto, perfis_idiomas, n=3):
 # Calcular o perfil de n-grams para o texto fornecido
 perfil_novo_texto = frequencia_ngrams(texto, n)

 # Comparar com cada perfil de idioma
 caracteristicas = {}

```

```


 max_similaridade = 0
idioma_predito = None
 for idioma, perfil in perfis_idiomas.items():
 # Interseção de n-grams
 intersecao = sum(min(perfil_novo_texto[ng], perfil.get(ng, 0))
for ng in perfil_novo_texto)
 # União de n-grams
 uniao = sum(perfil_novo_texto.get(ng, 0) + perfil.get(ng, 0) -
min(perfil_novo_texto.get(ng, 0), perfil.get(ng, 0)) for ng in
set(perfil_novo_texto) | set(perfil))
 # Cálculo da similaridade de Jaccard
 similaridade = intersecao / uniao if uniao != 0 else 0
 if similaridade > max_similaridade:
 max_similaridade = similaridade
 caracteristicas[idioma] = similaridade
 return max_similaridade

def extraiCaracteristicas(frase):
 texto = frase[0]
 pattern_regex = re.compile('[^\w+]', re.UNICODE)
 texto = re.sub(pattern_regex, ' ', texto)
 caracteristica1 = tamanhoMedioFrases(texto)
 caracteristica2 = funcao_caracteres(texto)
 caracteristica3 = funcao_sufixos(texto)
 # Normalize and clean the text
 texto = texto.lower() # Convert to lowercase
 texto = re.sub(r'^a-zA-Z\s]', '', texto) # Remove special characters
and punctuation
 texto = re.sub(r'\s+', ' ', texto) # Replace multiple spaces with a
single space
 texto = texto.strip() # Remove leading and trailing whitespace

 #print(frase)
 caracteristica4 = identifica_idioma(frase[0], perfis_idiomas)
 #print(caracteristica3, texto)
 padrao = [caracteristica1, caracteristica2, caracteristica3,
caracteristica4, texto, frase[1]]
 return padrao
def geraPadroes(frases):
 padroes = []
 for frase in frases:
 padrao = extraiCaracteristicas(frase)
 padroes.append(padrao)
 return padroes
padroes = geraPadroes(pre_padroes)
#print(padroes)
dados = pd.DataFrame(padroes)
#caracteristica4 = calcula_frequencia_ngrams_por_idioma([(texto, idioma)
for texto, idioma in pre_padroes])
dados

```

FIGURA 2 – VISUALIZAÇÃO DADOS EXERCÍCIO 1 ARQUITETURA DE DADOS



	0	1	2	3	4	5
0	4.500000	0.030303	0	0.053202	vamos sair para jantar no sbado	português
1	6.250000	0.000000	0	0.036642	adoro explorar novos lugares	português
2	3.800000	0.000000	1	0.047339	i m planning a vacation	inglês
3	3.500000	0.000000	1	0.073254	i enjoy going to the beach	inglês
4	4.500000	0.000000	1	0.064946	vamos fazer um churrasco no firal de semana	português
...	...	...	...	...	...	...
87	7.000000	0.000000	0	0.035736	ella canta hermosamente	espanhol
88	3.571429	0.000000	0	0.069904	i m going to the movies tonight	inglês
89	5.333333	0.000000	0	0.057682	gosto de fazer caminhadas na natureza	português
90	3.714286	0.030303	2	0.050638	voc j visitou o rio de janeiro	português
91	4.600000	0.035714	0	0.035483	preciso comprar leite e po	português

92 rows × 6 columns

Próximas etapas: [Gerar código com dados](#)  [Ver gráficos recomendados](#)

FONTE: Amauri (2025).

```
Treinando o modelo com SVM

Separando o conjunto de treinamento do conjunto de testes

from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
vet = np.array(padroes)
classes = vet[:, -1]
padroes_sem_classe = vet[:, 0:-1]
padroes_sem_classe = padroes_sem_classe[:, 0:-1]
padroes_sem_classe = pd.DataFrame(minmax_scale(padroes_sem_classe))
X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe,
classes, test_size=0.25, stratify=classes)
Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.
from sklearn import svm
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score, recall_score, f1_score
from sklearn.metrics import classification_report
treinador = svm.SVC() # algoritmo escolhido
modelo = treinador.fit(X_train, y_train)
#
score com os dados de treinamento
acuracia = modelo.score(X_train, y_train)
```

```

print("Acurácia nos dados de treinamento: {:.2f}%".format(accuracia *
100))
#
melhor avaliar com a matriz de confusão
y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print(cm)
print(classification_report(y_train, y_pred))
#
com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
print(cm)
print(classification_report(y_test, y_pred2))

```

Acurácia nos dados de treinamento: 72.46%

```

[[14 2 6]
 [1 20 2]
 [3 5 16]]

```

	precision	recall	f1-score	support
espanhol	0.78	0.64	0.70	22
inglês	0.74	0.87	0.80	23
português	0.67	0.67	0.67	24
accuracy			0.72	69
macro avg	0.73	0.72	0.72	69
weighted avg	0.73	0.72	0.72	69

métricas mais confiáveis

```

[[2 1 5]
 [1 6 0]
 [2 1 5]]

```

	precision	recall	f1-score	support
espanhol	0.40	0.25	0.31	8
inglês	0.75	0.86	0.80	7
português	0.50	0.62	0.56	8
accuracy			0.57	23
macro avg	0.55	0.58	0.55	23
weighted avg	0.54	0.57	0.54	23

2)

```

#
import numpy as np
import pandas as pd
base de dados disponível na UCI Machine Learning -
https://archive.ics.uci.edu/dataset/45/heart+disease
heart_colunas = ['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs',
 'restecg', 'thalach', 'exang',
 'oldpeak', 'slope', 'ca', 'thal', 'num']

```

```

heart = pd.read_csv('/content/hungarian.data',header=None,
 names=heart_colunas, lineterminator='\n',
na_values='?')
visualizar parte dos dados
print(heart.head())
caso queira usar do google-drive
#from google.colab import drive
#drive.mount('/content/drive')
#drive.mount("/content/drive", force_remount=True)

```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	\
0	1254	0	40	1	1	0	0	NaN	NaN
1	-9	2	140	0	289	-9	-9	-9	NaN
2	0	-9	-9	0	12	16	84	0	NaN
3	0	0	0	0	150	18	-9	7	NaN
4	172	86	200	110	140	86	0	0	NaN

```

 exang oldpeak slope ca thal num
0 NaN NaN NaN NaN NaN NaN
1 NaN NaN NaN NaN NaN NaN
2 NaN NaN NaN NaN NaN NaN
3 NaN NaN NaN NaN NaN NaN
4 NaN NaN NaN NaN NaN NaN

```

Para baixar direto da web e tratar arquivos compactados, sem o uso de arquivos locais.

```

import requests, zipfile, io
from io import BytesIO
import numpy as np
import pandas as pd
r =
requests.get('https://archive.ics.uci.edu/static/public/17/breast+cancer
+wisconsin+diagnostic.zip')
z = zipfile.ZipFile(io.BytesIO(r.content))
z.namelist()
dadosfp = z.open('wdbc.data')
dados = dadosfp.read()
cancer2 = pd.read_csv(io.BytesIO(dados),header=None,
 names=cancer_colunas, lineterminator='\n',
na_values='?')
print(cancer.head())
cancer.describe()

```

id	Diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	\
842302	M	17.99	10.38	122.80	1001.0	
0.11840						

```

842517 M 20.57 17.77 132.90 1326.0
0.08474
84300903 M 19.69 21.25 130.00 1203.0
0.10960
84348301 M 11.42 20.38 77.58 386.1
0.14250
84358402 M 20.29 14.34 135.10 1297.0
0.10030
 smoothness_mean compactness_mean concavity_mean
concave_mean \
842302 0.27760 0.3001 0.14710
0.2419
842517 0.07864 0.0869 0.07017
0.1812
84300903 0.15990 0.1974 0.12790
0.2069
84348301 0.28390 0.2414 0.10520
0.2597
84358402 0.13280 0.1980 0.10430
0.1809
 ... fractal_seradius_worst texture_worst perimeter_worst \
842302 ... 25.38 17.33 184.60
842517 ... 24.99 23.41 158.80
84300903 ... 23.57 25.53 152.50
84348301 ... 14.91 26.50 98.87
84358402 ... 22.54 16.67 152.20
 area_worst smoothness_worst compactness_worst
concavity_worst \
842302 2019.0 0.1622 0.6656
0.7119
842517 1956.0 0.1238 0.1866
0.2416
84300903 1709.0 0.1444 0.4245
0.4504
84348301 567.7 0.2098 0.8663
0.6869
84358402 1575.0 0.1374 0.2050
0.4000
 concave_worst symmetry_worst fractal_worst
842302 0.2654 0.4601 0.11890
842517 0.1860 0.2750 0.08902
84300903 0.2430 0.3613 0.08758
84348301 0.2575 0.6638 0.17300
84358402 0.1625 0.2364 0.07678

```

[5 rows x 31 columns]

```

Diagnosis radius_mean texture_mean perimeter_mean area_mean
smoothness_mean compactness_mean concavity_m

```

```

count 569.000000 569.000000 569.000000 569.000000 569.000000 569.000000
569.000000 569.000
mean 14.127292 19.289649 91.969033 654.889104 0.096360 0.104341 0.088799
0.048
std 3.524049 4.301036 24.298981 351.914129 0.014064 0.052813 0.079720
0.038
min 6.981000 9.710000 43.790000 143.500000 0.052630 0.019380 0.000000
0.000
25% 11.700000 16.170000 75.170000 420.300000 0.086370 0.064920 0.029560
0.020
50% 13.370000 18.840000 86.240000 551.100000 0.095870 0.092630 0.061540
0.033
75% 15.780000 21.800000 104.100000 782.700000 0.105300 0.130400 0.130700
0.074
max 28.110000 39.280000 188.500000 2501.000000 0.163400 0.345400
0.426800 0.201

```

```

from ucimlrepo import fetch_ucirepo

fetch dataset
heart_disease = fetch_ucirepo(id=45)

data (as pandas dataframes)
X = heart_disease.data.features
y = heart_disease.data.targets
Calculando a média de colunas específicas
mean_specific = X[['ca', 'thal']].mean()
Substituindo NaNs nas colunas específicas pela média
X[['ca', 'thal']] = X[['ca', 'thal']].fillna(mean_specific)

metadata
print(heart_disease.metadata)

variable information
print(heart_disease.variables)
{'uci_id': 45, 'name': 'Heart Disease', 'repository_url':
'https://archive.ics.uci.edu/dataset/45/heart+disease', 'data_url':
'https://archive.ics.uci.edu/static/public/45/data.csv',
 name role type demographic \
0 age Feature Integer Age
1 sex Feature Categorical Sex
2 cp Feature Categorical None
3 trestbps Feature Integer None
4 chol Feature Integer None
5 fbs Feature Categorical None
6 restecg Feature Categorical None
7 thalach Feature Integer None
8 exang Feature Categorical None
9 oldpeak Feature Integer None

```

```

10 slope Feature Categorical None
11 ca Feature Integer None
12 thal Feature Categorical None
13 num Target Integer None
 description units
missing_values
0 None years
no
1 None None
no
2 None None
no
3 resting blood pressure (on admission to the ho... mm Hg
no
4 serum cholestorl mg/dl
no
5 fasting blood sugar > 120 mg/dl None
no
6 None None
no
7 maximum heart rate achieved None
no
8 exercise induced angina None
no
9 ST depression induced by exercise relative to ... None
no
10 None None
no
11 number of major vessels (0-3) colored by flour... None
yes
12 None None
yes
13 diagnosis of heart disease None
no
<ipython-input-39-8cbaa159bf47>:14: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
X[['ca', 'thal']] = X[['ca', 'thal']].fillna(mean_specific)

realizar os tratamentos no exemplo,normalizar as colunas, remover a
coluna de identi cação e separar a classe dos atributos.

print(X.head())
Y = y
Y_orig = y
print(Y.head())

```

```

 age sex cp trestbps chol fbs restecg thalach exang oldpeak
slope \
0 63 1 1 145 233 1 2 150 0 2.3
3
1 67 1 4 160 286 0 2 108 1 1.5
2
2 67 1 4 120 229 0 2 129 1 2.6
2
3 37 1 3 130 250 0 0 187 0 3.5
3
4 41 0 2 130 204 0 2 172 0 1.4
1
ca thal
0 0.0 6.0
1 3.0 3.0
2 2.0 7.0
3 0.0 3.0
4 0.0 3.0
num
0 0
1 2
2 1
3 0
4 0

#X = X.dropna()
nan_columns = X.isna().any()
Imprime as colunas que contêm pelo menos um NaN
print(nan_columns)
Se você quiser listar o nome das colunas que têm valores NaN
nan_columns_list = nan_columns[nan_columns].index.tolist()
print("Colunas com NaN:", nan_columns_list)
age False
sex False
cp False
trestbps False
chol False
fbs False
restecg False
thalach False
exang False
oldpeak False
slope False
ca False
thal False
dtype: bool
Colunas com NaN: []

```

# realizar os tratamentos no exemplo, normalizar as colunas, remover a  
coluna de identi cação e separar a classe dos atributos.  
#Ndeverão ser realizada as tentativas de tratamento de dados, visando a  
melhoria no desempenho do classi cador (SVM).

```
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd
X_orig = X.copy()
print(X_orig.head())
Y_orig = Y_orig.squeeze()
#unique_values = Y_orig_series.unique()
print(Y_orig.unique())
normalização min-max,
X = pd.DataFrame(minmax_scale(X))
print(X_orig.head())
print(X.head())
A normalização pode ser manual, usando o pandas... ex:
#X['radius_mean'] = (X_orig['radius_mean']-X_orig['radius_mean'].mean())
/ X_orig['radius_mean'].std()
normalização min-max
#X['texture_mean'] = (X_orig['texture_mean']-
X_orig['texture_mean'].min()) / (X_orig['texture_mean'].max()-
X_orig['texture_mean'].min())
```

```

 age sex cp trestbps chol fbs restecg thalach exang oldpeak
slope \
0 63 1 1 145 233 1 2 150 0 2.3
3
1 67 1 4 160 286 0 2 108 1 1.5
2
2 67 1 4 120 229 0 2 129 1 2.6
2
3 37 1 3 130 250 0 0 187 0 3.5
3
4 41 0 2 130 204 0 2 172 0 1.4
1
 ca thal
0 0.0 6.0
1 3.0 3.0
2 2.0 7.0
3 0.0 3.0
4 0.0 3.0
[0 2 1 3 4]
 age sex cp trestbps chol fbs restecg thalach exang oldpeak
slope \
0 63 1 1 145 233 1 2 150 0 2.3
3
1 67 1 4 160 286 0 2 108 1 1.5
2
```

```

2 67 1 4 120 229 0 2 129 1 2.6
2
3 37 1 3 130 250 0 0 187 0 3.5
3
4 41 0 2 130 204 0 2 172 0 1.4
1
 ca thal
0 0.0 6.0
1 3.0 3.0
2 2.0 7.0
3 0.0 3.0
4 0.0 3.0
 0 1 2 3 4 5 6 7 8
\
0 0.708333 1.0 0.000000 0.481132 0.244292 1.0 1.0 0.603053 0.0
1 0.791667 1.0 1.000000 0.622642 0.365297 0.0 1.0 0.282443 1.0
2 0.791667 1.0 1.000000 0.245283 0.235160 0.0 1.0 0.442748 1.0
3 0.166667 1.0 0.666667 0.339623 0.283105 0.0 0.0 0.885496 0.0
4 0.250000 0.0 0.333333 0.339623 0.178082 0.0 1.0 0.770992 0.0
 9 10 11 12
0 0.370968 1.0 0.000000 0.75
1 0.241935 0.5 1.000000 0.00
2 0.419355 0.5 0.666667 1.00
3 0.564516 1.0 0.000000 0.00
4 0.225806 0.0 0.000000 0.00

```

#A próxima seção trata da construção do modelo, dos testes e das métricas #da matriz de confusão

```

from sklearn.model_selection import train_test_split
import numpy as np
com os dados originais
X_oring_train, X_orig_test, y_oring_train, y_orig_test =
train_test_split(X_orig,
 Y_orig, test_size=0.25,
 stratify=Y_orig,random_state=10)
com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X, Y,
test_size=0.25,
stratify=Y,random_state=10)
Treina o modelo com base nos dados originais (SVM).
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
treinador = svm.SVC() #algoritmo escolhido
modelo_orig = treinador.fit(X_oring_train, y_oring_train)
predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)

```

```

cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no
TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))
predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[123 0 0 0 0]
 [41 0 0 0 0]
 [27 0 0 0 0]
 [26 0 0 0 0]
 [10 0 0 0 0]]
 precision recall f1-score support

 0 0.54 1.00 0.70 123
 1 0.00 0.00 0.00 41
 2 0.00 0.00 0.00 27
 3 0.00 0.00 0.00 26
 4 0.00 0.00 0.00 10
 accuracy 0.54 0.54 0.54 227
 macro avg 0.11 0.20 0.14 227
 weighted avg 0.29 0.54 0.38 227
Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[41 0 0 0 0]
 [14 0 0 0 0]
 [9 0 0 0 0]
 [9 0 0 0 0]
 [3 0 0 0 0]]
 precision recall f1-score support

 0 0.54 1.00 0.70 41
 1 0.00 0.00 0.00 14
 2 0.00 0.00 0.00 9
 3 0.00 0.00 0.00 9
 4 0.00 0.00 0.00 3
 accuracy 0.54 0.54 0.54 76
 macro avg 0.11 0.20 0.14 76
 weighted avg 0.29 0.54 0.38 76
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no
 _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no

```

```

_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-
packages/sklearn/metrics/_classification.py:1344:
UndefinedMetricWarning: Precision and F-score are ill-defined and being
set to 0.0 in labels with no

```

#Como os dados cam após os processos de tratamento dos dados

```

from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
treinador = svm.SVC() #algoritmo escolhido
modelo = treinador.fit(X_train, y_train)
predição com os mesmos dados usados para treinar
y_pred = modelo.predict(X_train)
cm_train = confusion_matrix(y_train, y_pred)
print('Matriz de confusão - com os dados TRATADOS usados no
TREINAMENTO')
print(cm_train)
print(classification_report(y_train, y_pred))
predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_pred = modelo.predict(X_test)
cm_test = confusion_matrix(y_test, y2_pred)
print(cm_test)
print(classification_report(y_test, y2_pred))
Matriz de confusão - com os dados TRATADOS usados no TREINAMENTO
[[120 0 1 2 0]
 [16 15 3 7 0]
 [7 4 9 7 0]
 [3 4 4 15 0]
 [1 0 1 8 0]]
precision recall f1-score support

```

0	0.82	0.98	0.89	123
1	0.65	0.37	0.47	41
2	0.50	0.33	0.40	27
3	0.38	0.58	0.46	26
4	0.00	0.00	0.00	10
accuracy			0.70	227
macro avg	0.47	0.45	0.44	227
weighted avg	0.66	0.70	0.67	227

Matriz de confusão - com os dados ORIGINAIS usados para TESTES

```
[[40 1 0 0 0]
 [7 2 0 5 0]
 [3 3 2 1 0]
 [2 0 7 0 0]
 [0 3 0 0 0]]
```

	precision	recall	f1-score	support
0	0.77	0.98	0.86	41
1	0.22	0.14	0.17	14
2	0.22	0.22	0.22	9
3	0.00	0.00	0.00	9
4	0.00	0.00	0.00	3
accuracy			0.58	76
macro avg	0.24	0.27	0.25	76
weighted avg	0.48	0.58	0.52	76

/usr/local/lib/python3.10/dist-

packages/sklearn/utils/validation.py:1143: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of

```
y = column_or_1d(y, warn=True)
```

/usr/local/lib/python3.10/dist-

packages/sklearn/metrics/\_classification.py:1344:

UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no

```
_warn_prf(average, modifier, msg_start, len(result))
```

/usr/local/lib/python3.10/dist-

packages/sklearn/metrics/\_classification.py:1344:

UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no

```
_warn_prf(average, modifier, msg_start, len(result))
```

/usr/local/lib/python3.10/dist-

packages/sklearn/metrics/\_classification.py:1344:

UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no

```
_warn_prf(average, modifier, msg_start, len(result))
```

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/

classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no

## APÊNDICE G - APRENDIZADO DE MÁQUINA

### A – ENUNCIADO

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

### B – RESOLUÇÃO

#### RESOLUÇÃO CLASSIFICAÇÃO E REGRESSÃO

```
#
install.packages("e1071")
install.packages("caret")
library("caret")

#####
Classificação KNN Veículos
setwd("C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/06 - Veículos")
dados <- read.csv("6 - Veiculos - Dados.csv")

#retira o atributo 'a' que nesse database é um index
dados$a <- NULL

Cria um arquivo com treino com 80% e teste com 20% das linhas de
forma randomizada
set.seed(202412)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

Faz um grid com valores para K e
Executa o KNN
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202412)
knn <- train(tipo ~ ., data = treino, method = "knn", tuneGrid=tuneGrid)
knn

Faz a predição e mostra a matriz de confusão
predict.knn <- predict(knn, teste)
confusionMatrix(predict.knn, as.factor(teste$tipo))
```

```
#####
Classificação RNA Veículos

rna <- train(tipo~., data=treino, method="nnet",trace=FALSE)
rna

Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)

Matriz de confusão
confusionMatrix(predict.rna, as.factor(teste$tipo))

indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)
executa a RNA com esse ctrl
set.seed(202412)
rna <- train(tipo~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
rna
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))

size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202412)
rna <- train(
 form = tipo~. ,
 data = treino ,
 method = "nnet" ,
 tuneGrid = grid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
rna
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))

#####
Classificação SVM Veículos

Treinar SVM com a base de Treino
set.seed(202412)
svm <- train(tipo~., data=treino, method="svmRadial")
svm

Aplicar modelos treinados na base de Teste
```

```

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

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202412)

size, decay
set.seed(202412)
svm <- train(
 form = tipo~, ,
 data = treino ,
 method = "svmRadial" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
svm
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

#####
Classificação RF Veículos

Treinar SVM com a base de Treino
set.seed(202412)
rf <- train(tipo~., data=treino, method="rf")
rf

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

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid =expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202412)

size, decay
set.seed(202412)
rf <- train(
 form = tipo~, ,
 data = treino ,
 method = "rf" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
rf

```

```

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

PREDIÇÕES DE NOVOS CASOS -MELHOR PARA VEICULOS FOI SVM CV
dados_novos_casos <- read.csv("6 - Veiculos - NovosDados.csv")
dados_novos_casos$a <- NULL

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

#####
Classificação KNN Diabetes
setwd("C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/10 - Diabetes")
dados <- read.csv("10 - Diabetes - Dados.csv")

#retira o atributo 'num' que nesse database é um index
dados$num <- NULL

Cria um arquivo com treino com 80% e teste com 20% das linhas de
forma randomizada
set.seed(202412)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

Faz um grid com valores para K e
Executa o KNN
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
set.seed(202412)
knn <- train(diabetes ~ ., data = treino, method =
"knn",tuneGrid=tuneGrid)
knn

Faz a predição e mostra a matriz de confusão
predict.knn <- predict(knn, teste)
confusionMatrix(predict.knn, as.factor(teste$diabetes))

PREDIÇÕES DE NOVOS CASOS
dados_novos_casos <- read.csv("10 - Diabetes - NovosDados.csv")
dados_novos_casos$num <- NULL

predict.knn <- predict(knn, dados_novos_casos)
resultado <- cbind(dados_novos_casos, predict.knn)

```

```

resultado$diabetes <- NULL
View(resultado)

#####
Classificação RNA Diabetes

rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE)
rna

Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)

Matriz de confusão
confusionMatrix(predict.rna, as.factor(teste$diabetes))

indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)
executa a RNA com esse ctrl
set.seed(202412)
rna <- train(diabetes~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
rna
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202412)
rna <- train(
 form = diabetes~. ,
 data = treino ,
 method = "nnet" ,
 tuneGrid = grid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
rna
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

#####
Classificação SVM Diabetes

Treinar SVM com a base de Treino
set.seed(202412)
svm <- train(diabetes~., data=treino, method="svmRadial")
svm

```

```

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

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
set.seed(202412)

size, decay
set.seed(202412)
svm <- train(
 form = diabetes~. ,
 data = treino ,
 method = "svmRadial" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
svm
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$diabetes))

#####
Classificação RF Diabetes

Treinar SVM com a base de Treino
set.seed(202412)
rf <- train(diabetes~., data=treino, method="rf")
rf

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

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid =expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202412)

size, decay
set.seed(202412)
rf <- train(
 form = diabetes~. ,
 data = treino ,
 method = "rf" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,

```

```

 maxit = 2000,trace=FALSE)
rf
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$diabetes))

PREDIÇÕES DE NOVOS CASOS -MELHOR PARA VEICULOS FOI SVM CV
dados_novos_casos <- read.csv("10 - Diabetes - NovosDados.csv")
dados_novos_casos$num <- NULL

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

#####
Regressão KNN Admissão

#install.packages("Metrics")
library(Metrics)

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

calcular_syx <- function(predito, observado) {

 residuals <- observado - predito
 squared_residuals <- residuals^2
 mean_squared_residuals <- mean(squared_residuals)

 return(sqrt(mean_squared_residuals))
}

Leitura dos dados
setwd("C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/09 - Admissão")
dados <- read.csv("9 - Admissao - Dados.csv", header=T)
dados$num <- NULL

Cria arquivos de treino e teste
set.seed(202412)
ind <- createDataPartition(dados$ChanceOfAdmit, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]
Prepara um grid com os valores de k que
serão usados
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
Executa o Knn com esse grid

```

```

set.seed(202412)
knn <- train(ChanceOfAdmit ~ ., data = treino, method = "knn",
 tuneGrid=tuneGrid)
knn

Aplica o modelo no arquivo de teste
predict.knn <- predict(knn, teste)
Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.knn)

r2(predict.knn, teste$ChanceOfAdmit)

calcular_syx(predict.knn, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.knn, teste$ChanceOfAdmit)
print(pearson_corr)

#####
Regressão RNA Admissão

rna <- train(ChanceOfAdmit~., data=treino, method="nnet", , linout=T,
trace=FALSE)
rna

Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)

Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.rna)

r2(predict.rna, teste$ChanceOfAdmit)

calcular_syx(predict.rna, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.rna, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.rna, teste$ChanceOfAdmit)

indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)

```

```

executa a RNA com esse ctrl
set.seed(202412)
rna <- train(ChanceOfAdmit~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
rna
predict.rna <- predict(rna, teste)
Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.rna)

r2(predict.rna,teste$ChanceOfAdmit)

calcular_syx(predict.rna, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.rna, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.rna, teste$ChanceOfAdmit)

size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202412)
rna <- train(
 form = ChanceOfAdmit~. ,
 data = treino ,
 method = "nnet" ,
 tuneGrid = grid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
rna
predict.rna <- predict(rna, teste)
Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.rna)

r2(predict.rna,teste$ChanceOfAdmit)

calcular_syx(predict.rna, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.rna, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.rna, teste$ChanceOfAdmit)

```

```
#####
Regressão SVM Admissão

Treinar SVM com a base de Treino
set.seed(202412)
svm <- train(ChanceOfAdmit~., data=treino, method="svmRadial")
svm

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

Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.svm)

r2(predict.svm, teste$ChanceOfAdmit)

calcular_syx(predict.svm, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.svm, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.svm, teste$ChanceOfAdmit)

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))

size, decay
set.seed(202412)
svm <- train(
 form = ChanceOfAdmit~. ,
 data = treino ,
 method = "svmRadial" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
svm
predict.svm <- predict(svm, teste)

Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.svm)
```

```

r2(predict.svm, teste$ChanceOfAdmit)

calcular_syx(predict.svm, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.svm, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.svm, teste$ChanceOfAdmit)

#####
Regressão RF Admissão

Treinar SVM com a base de Treino
set.seed(202412)
rf <- train(ChanceOfAdmit~., data=treino, method="rf")
rf

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

Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.rf)

r2(predict.rf, teste$ChanceOfAdmit)

calcular_syx(predict.rf, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.rf, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.rf, teste$ChanceOfAdmit)

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid =expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202412)

size, decay
set.seed(202412)
rf <- train(
 form = ChanceOfAdmit~. ,
 data = treino ,

```

```

method = "rf" ,
tuneGrid = tuneGrid ,
trControl = ctrl ,
maxit = 2000,trace=FALSE)
rf
predict.rf <- predict(rf, teste)

Mostra as métricas

rmse(teste$ChanceOfAdmit, predict.rf)

r2(predict.rf, teste$ChanceOfAdmit)

calcular_syx(predict.rf, teste$ChanceOfAdmit)

pearson_corr <- cor(predict.rf, teste$ChanceOfAdmit)
print(pearson_corr)

MAE(predict.rf, teste$ChanceOfAdmit)

PREDIÇÕES DE NOVOS CASOS -MELHOR PARA VEICULOS FOI SVM CV
dados_novos_casos <- read.csv("9 - Admissao - NovosDados.csv")
dados_novos_casos$num <- NULL

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

#####
Regressão KNN Biomassa

Leitura dos dados
setwd("C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/05 - Biomassa")
dados <- read.csv("5 - Biomassa - Dados.csv", header=T)

Cria arquivos de treino e teste
set.seed(202412)
ind <- createDataPartition(dados$biomassa, p=0.80, list = FALSE)
treino <- dados[ind,]
teste <- dados[-ind,]
Prepara um grid com os valores de k que
serão usados
tuneGrid <- expand.grid(k = c(1,3,5,7,9))

```

```

Executa o Knn com esse grid
set.seed(202412)
knn <- train(biomassa ~ ., data = treino, method = "knn",
 tuneGrid=tuneGrid)
knn

Aplica o modelo no arquivo de teste
predict.knn <- predict(knn, teste)
Mostra as métricas

rmse(teste$biomassa, predict.knn)

r2(predict.knn, teste$biomassa)

calcular_syx(predict.knn, teste$biomassa)

pearson_corr <- cor(predict.knn, teste$biomassa)
print(pearson_corr)

MAE(predict.knn, teste$biomassa)

#####
Regressão RNA Biomassa

rna <- train(biomassa~., data=treino, method="nnet", , linout=T,
trace=FALSE)
rna

Predições dos valores do conjunto de teste
predict.rna <- predict(rna, teste)

Mostra as métricas

rmse(teste$biomassa, predict.rna)

r2(predict.rna, teste$biomassa)

calcular_syx(predict.rna, teste$biomassa)

pearson_corr <- cor(predict.rna, teste$biomassa)
print(pearson_corr)

MAE(predict.rna, teste$biomassa)

```

```

indica o método cv e numero de folders 10
ctrl <- trainControl(method = "cv", number = 10)
executa a RNA com esse ctrl
set.seed(202412)
rna <- train(biomassa~., data=treino, method="nnet",trace=FALSE,
trControl=ctrl)
rna
predict.rna <- predict(rna, teste)
Mostra as métricas

rmse(teste$biomassa, predict.rna)

r2(predict.rna,teste$biomassa)

calcular_syx(predict.rna, teste$biomassa)

pearson_corr <- cor(predict.rna, teste$biomassa)
print(pearson_corr)

MAE(predict.rna, teste$biomassa)

size, decay
grid <- expand.grid(size = seq(from = 1, to = 45, by = 10),decay =
seq(from = 0.1, to = 0.9, by = 0.3))
set.seed(202412)
rna <- train(
 form = biomassa~. ,
 data = treino ,
 method = "nnet" ,
 tuneGrid = grid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
rna
predict.rna <- predict(rna, teste)
Mostra as métricas

rmse(teste$biomassa, predict.rna)

r2(predict.rna,teste$biomassa)

calcular_syx(predict.rna, teste$biomassa)

```

```

pearson_corr <- cor(predict.rna, teste$biomassa)
print(pearson_corr)

MAE(predict.rna, teste$biomassa)

#####
Regressão SVM Biomassa

Treinar SVM com a base de Treino
set.seed(202412)
svm <- train(biomassa~., data=treino, method="svmRadial")
svm

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

Mostra as métricas

rmse(teste$biomassa, predict.svm)

r2(predict.svm, teste$biomassa)

calcular_syx(predict.svm, teste$biomassa)

pearson_corr <- cor(predict.svm, teste$biomassa)
print(pearson_corr)

MAE(predict.svm, teste$biomassa)

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid = expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))

size, decay
set.seed(202412)
svm <- train(
 form = biomassa~. ,
 data = treino ,
 method = "svmRadial" ,
 tuneGrid = tuneGrid ,
 trControl = ctrl ,
 maxit = 2000,trace=FALSE)
svm
predict.svm <- predict(svm, teste)

Mostra as métricas

```

```

rmse(teste$biomassa, predict.svm)

r2(predict.svm, teste$biomassa)

calcular_syx(predict.svm, teste$biomassa)

pearson_corr <- cor(predict.svm, teste$biomassa)
print(pearson_corr)

MAE(predict.svm, teste$biomassa)

#####
Regressão RF Biomassa

Treinar SVM com a base de Treino
set.seed(202412)
rf <- train(biomassa~., data=treino, method="rf")
rf

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

Mostra as métricas

rmse(teste$biomassa, predict.rf)

r2(predict.rf, teste$biomassa)

calcular_syx(predict.rf, teste$biomassa)

pearson_corr <- cor(predict.rf, teste$biomassa)
print(pearson_corr)

MAE(predict.rf, teste$biomassa)

ctrl <- trainControl(method = "cv", number = 10)
Vários C e sigma
tuneGrid =expand.grid(mtry=c(2, 5, 7, 9))
set.seed(202412)

size, decay
set.seed(202412)
rf <- train(
 form = biomassa~. ,

```

```

data = treino ,
method = "rf" ,
tuneGrid = tuneGrid ,
trControl = ctrl ,
maxit = 2000,trace=FALSE)
rf
predict.rf <- predict(rf, teste)

Mostra as métricas

rmse(teste$biomassa, predict.rf)

r2(predict.rf, teste$biomassa)

calcular_syx(predict.rf, teste$biomassa)

pearson_corr <- cor(predict.rf, teste$biomassa)
print(pearson_corr)

MAE(predict.rf, teste$biomassa)

PREDIÇÕES DE NOVOS CASOS -MELHOR PARA VEICULOS FOI SVM CV
dados_novos_casos <- read.csv("5 - Biomassa - NovosDados.csv")

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

```

FIGURA 9 – RESOLUÇÃO EXERCÍCIO APRENDIZADO DE MÁQUINA CLASSIFICAÇÃO

### Veículo

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	sigma = 0.015 and C = 100	0,9118	<pre> Confusion Matrix and Statistics  Reference Prediction bus opel saab van bus 46 0 0 0 opel 0 33 6 0 saab 0 9 36 0 van 0 0 0 40  Overall Statistics  Accuracy : 0.9118 </pre>

RNA – CV	size=31 decay=0.4	0,8824	<p>Confusion Matrix and Statistics</p> <pre> Reference Prediction bus opel saab van bus 46 0 0 0 opel 0 32 9 0 saab 0 10 33 1 van 0 0 0 39  Overall Statistics  Accuracy : 0.8824 95% CI : (0.8242, 0.9266) </pre>
SVM – Hold-out	sigma = 0.067603 3 and C = 1	0,7941	<pre> Reference Prediction bus opel saab van bus 46 0 0 0 opel 0 24 16 0 saab 0 17 25 0 van 0 1 1 40  Overall Statistics  Accuracy : 0.7941 95% CI : (0.7255, 0.8522) </pre>
RF – CV	mtry = 5	0.7706	<p>Confusion Matrix and Statistics</p> <pre> Reference Prediction bus opel saab van bus 46 0 0 0 opel 0 22 18 0 saab 0 19 23 0 van 0 1 1 40  Overall Statistics  Accuracy : 0.7706 95% CI : (0.7, 0.8315) </pre>

RF – Hold- out	mtry = 2	0.7647	<p>Confusion Matrix and Statistics</p> <pre> Reference Prediction bus opel saab van bus 46 0 0 0 opel 0 21 17 0 saab 0 20 23 0 van 0 1 2 40  Overall Statistics  Accuracy : 0.7647 95% CI : (0.6937, 0.8263) </pre>
RNA – Hold- out	size=3 decay=0. 1	0,7176	<pre> Reference Prediction bus opel saab van bus 43 0 0 0 opel 0 0 0 2 saab 2 42 41 0 van 1 0 1 38  Overall Statistics  Accuracy : 0.7176 95% CI : (0.6436, 0.7839) </pre>
KNN	k=3	0,7	<pre> Reference Prediction bus opel saab van bus 38 3 3 1 opel 5 17 12 1 saab 3 22 26 0 van 0 0 1 38  Overall Statistics  Accuracy : 0.7 95% CI : (0.6251, 0.7678) </pre>

## Classificação três novos casos

Comp	Circ	DCirc	RadRa	PrAsiRa	MaxRa	ScRa	Elong	PrAsiRect	MaxRect	ScVnMaxs	ScVnMaxs	Radgr	SkewMaxs	SkewMaxs	KurtMaxs	KurtMaxs	HullRa	predLam	
3	70	50	33	150	78	6	120	60	18	129	200	300	130	70	5	15	160	200	cpel
1	99	48	85	157	64	11	161	43	20	167	175	375	186	74	3	16	185	200	vaf
2	99	57	100	177	60	18	200	30	25	188	400	700	213	72	4	7	185	200	cpel

### Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV	sigma = 0.015 and C = 10	0.7987	<p>Confusion Matrix and Statistics</p> <p>Reference</p> <p>Prediction neg pos</p> <p>neg 96 21</p> <p>pos 10 27</p> <p>Accuracy : 0.7987</p> <p>95% CI : (0.7266, 0.8589)</p>
SVM – Hold-out	sigma = 0.134945 5 and C = 0.5	0.7727	<p>Confusion Matrix and Statistics</p> <p>Reference</p> <p>Prediction neg pos</p> <p>neg 92 21</p> <p>pos 14 27</p> <p>Accuracy : 0.7727</p> <p>95% CI : (0.6984, 0.8363)</p>
KNN	k=9	0.7727	<p>Reference</p> <p>Prediction neg pos</p> <p>neg 93 22</p> <p>pos 13 26</p> <p>Accuracy : 0.7727</p> <p>95% CI : (0.6984, 0.8363)</p>
RF – CV	mtry = 2	0.7597	<p>Confusion Matrix and Statistics</p> <p>Reference</p> <p>Prediction neg pos</p> <p>neg 88 19</p> <p>pos 18 29</p> <p>Accuracy : 0.7597</p> <p>95% CI : (0.6844, 0.8248)</p>
RF – Hold-out	mtry = 2	0.7532	<p>Confusion Matrix and Statistics</p> <p>Reference</p> <p>Prediction neg pos</p> <p>neg 87 19</p> <p>pos 19 29</p> <p>Accuracy : 0.7532</p>

RNA – CV	size=11 decay=0.4	0.7338	<pre> Confusion Matrix and Statistics  Reference Prediction neg pos neg 87 22 pos 19 26  Accuracy : 0.7338 95% CI : (0.6566, 0.8017) </pre>
RNA – Hold-out	size=3 decay=0.1	0.67639 20	<pre> Confusion Matrix and Statistics  Reference Prediction neg pos neg 94 38 pos 12 10  Accuracy : 0.6753 95% CI : (0.5953, 0.7485) </pre>

## Melhor resultado SVM-CV

## Classificação três novos casos

	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	predict.svm
1	10	100	84	33	105	200	170	55	pos
2	1	122	44	20	180	299	30	22	pos
3	2	150	40	50	500	44	245	30	pos

FONTE: Amauri (2025).

FIGURA 10 – RESOLUÇÃO EXERCÍCIO APRENDIZADO DE MÁQUINA REGRESSÃO

### Admissão

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
SVM – CV	sigma = 0.01 and C = 10	0.888294 3	0.0441364 3	0.942564 5	0.0441364 3	0.0327051 6
SVM – Hold-out	sigma = 0.2004666 and C = 1	0.867644 6	0.0480430 2	0.931743 5	0.0480430 2	0.0362442 1
RF – Hold-out	mtry = 2	0.86093	0.0492466	0.930875	0.0492466	0.0368191 4
SVM – Hold-out	sigma = 0.2004666 and C = 1	0.867644 6	0.0480430 2	0.931743 5	0.0480430 2	0.0362442 1
RF – CV	mtry = 2	0.859865 6	0.0494347	0.930446 3	0.0494347	0.0369878 3
RNA – Hold-out	size=5 decay=0.1	0.837802	0.0531841 6	0.918837 6	0.0531841 6	0.0438953
KNN	K=9	0.777338 7	0.0623134 7	0.883457	0.0623134 7	0.0586836 0
RNA – CV	size=41 decay=0.1	0.752869 6	0.0656481 7	0.875309 7	0.0656481 7	0.0540947 6

**Melhor resultado SVM-CV**

**Classificação três novos casos**

GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	predict.svm
300	122	4	20	15	500	1	0.5638051
313	100	2	40	35	999	0	0.5638051
200	50	5	20	30	700	1	0.5638051

### Biomassa

Técnica	Parâmetro	R2	Syx	Pearson	Rmse	MAE
RF – Hold-out	mtry = 2	0.8869383	403.3385	0.9852005	403.3385	102.0528
RF – CV	mtry = 2	0.8770836	420.5492	0.9846277	420.5492	101.737
SVM – CV	sigma = 0.01 and C = 100	0.8091402	524.0458	0.9605338	524.0458	154.5247
KNN	K=1	0.7776939	565.5718	0.9471209	565.5718	128.5158
RNA – Hold-out	size=5 decay=0.1	0.5949609	763.4146	0.8779725	763.4146	235.404
SVM – Hold-out	sigma = 0.864394 and C = 1	0.4750265	869.1218	0.8535094	869.1218	199.1633
RNA – CV	size=41 decay=0.1	- 0.0828944	1248.26	0.2379349	1248.26	345.362

### Melhor resultado RF-Hold-Out

### Classificação três novos casos

	dap	h	Me	predict.rf
1	250	155	67	12325.9269
2	15	30	9	1009.9459
3	18	7	61	59.2985

## RESOLUÇÃO AGRUPAMENTO

```

#
setwd("C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/06 - Veículos")
dados <- read.csv("6 - Veiculos - Dados.csv")

Load necessary libraries
install.packages("readr")
install.packages("dplyr")
install.packages("factoextra")
install.packages("ggplot2")

library(readr)
library(dplyr)
library(factoextra)
library(ggplot2)

para o kmodes
install.packages("klaR")
library(klaR)

file_path <- "C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/06 - Veículos/6
- Veiculos - Dados.csv"
df <- read_csv(file_path)

df_clustering <- df %>% select(-tipo)

df_scaled <- scale(df_clustering)

set.seed(202412)
kmeans_result <- kmeans(df_scaled, centers = 10, nstart = 25)

df$Cluster <- as.factor(kmeans_result$cluster)

head(df, n = 10)

print(kmeans_result)

fviz_cluster(kmeans_result, data = df_scaled, geom = "point", stand =
FALSE) +
 ggtitle("KMeans Clustering") +
 theme_minimal()

cluster.results <- kmodes(dados, 5, iter.max = 10, weighted = FALSE)
cluster.results

```

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

FIGURA 11 – RESOLUÇÃO EXERCÍCIO APRENDIZADO DE MÁQUINA AGRUPAMENTO

### Veículo

Lista de Clusters gerados:

10 primeiras linhas do arquivo com o cluster correspondente.

Usa 10 clusters no experimento.

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

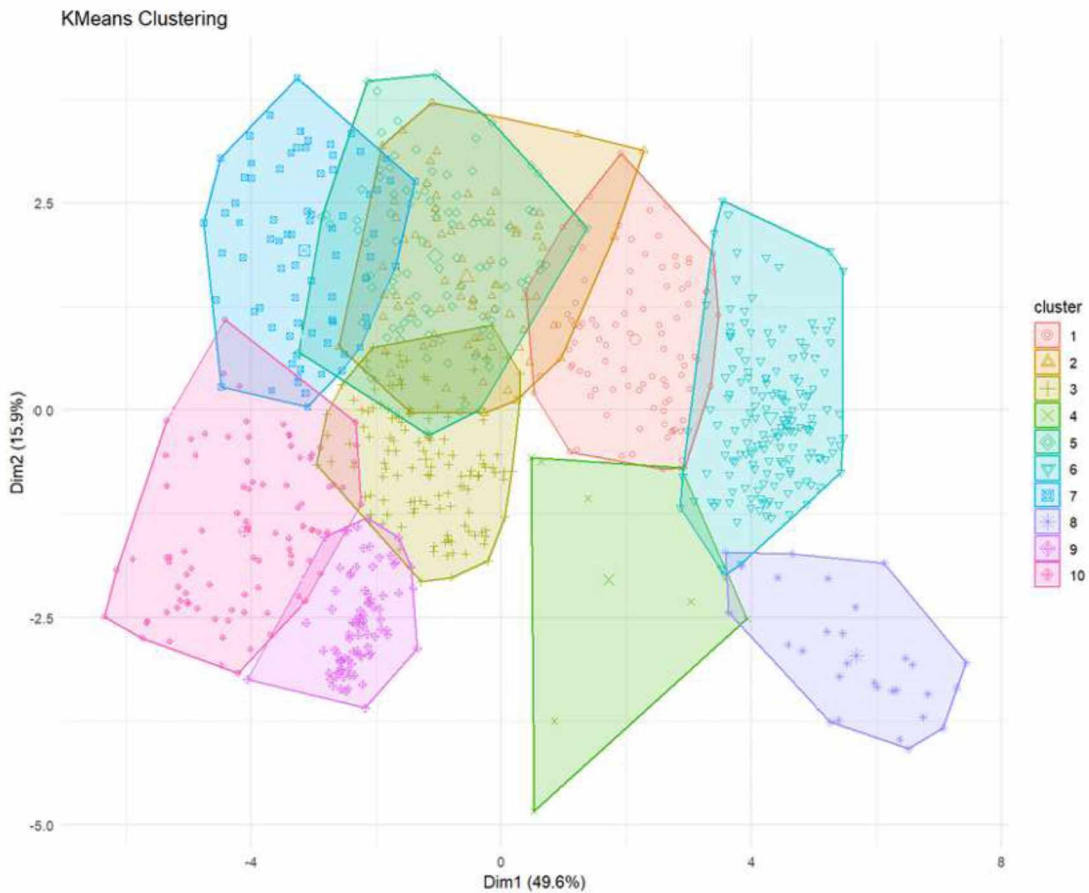
```

> head(df, n = 10)
A tibble: 10 x 23
 a Comp Circ DCirc RadRa PehisRa MaxRa ScatRa Elong PehisRect MaxRect ScVarMaxis ScVarmaxis RaGyr SkewMaxis Skewmaxis Wurtmaxis WurtMaxis HolRa Tipo Cluster
1 1 95 48 83 178 72 18 162 42 20 159 176 379 184 78 6 16 187 197 van 3
2 2 91 41 84 141 57 9 149 45 19 143 178 338 158 72 9 14 189 199 van 3
3 3 184 58 186 289 66 18 287 32 23 158 223 635 228 73 14 9 188 196 saab 6
4 4 93 41 82 159 63 9 148 46 19 143 168 389 127 63 6 18 199 287 van 7
5 5 85 44 78 285 181 82 149 45 19 144 241 325 188 127 9 11 188 183 bus 4
6 6 187 57 186 172 58 6 255 26 28 169 288 957 264 85 5 9 181 183 bus 8
7 7 97 43 73 173 65 6 153 42 19 143 176 361 172 66 13 1 288 284 bus 2
8 8 98 43 46 137 65 9 137 48 18 146 182 281 168 87 3 3 192 282 van 9
9 9 86 34 62 148 61 7 122 34 17 127 141 223 112 64 2 14 288 288 van 7
10 10 93 44 98 197 62 11 183 36 22 146 282 585 152 64 4 14 195 284 saab 1

```

FONTE: Amauri (2025).

GRÁFICO 10 – AGRUPAMENTO DE VEÍCULOS



FONTE: Amauri (2025).

## RESOLUÇÃO ASSOCIAÇÃO

```

Instalação dos pacotes necessários
install.packages("arules")
install.packages("arulesViz")
library(arules)
library(arulesViz)

file_path <- "C:/Users/Amauri/Desktop/IAA/aprendizadoMaq/12 -
Musculacao/2 - Musculacao - Dados.csv"
data <- read.csv(file_path, sep = ";", encoding = "latin1")

head(data)

set.seed(202412)
rules <- apriori(data, parameter = list(supp = 0.001, conf = 0.7,
minlen=2))
summary(rules)

Vamos ver as 5 primeiras regras ordenadas pela confiança:
options(digits=2)
inspect(sort(rules[1:20], by="confidence"))
```

## APÊNDICE G - 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

Especificações:

Implementar os modelos vistos em sala de aula e aplicar para os datasets especificados no enunciado.

Classificação de Imagens (CNN):

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

```

from sklearn.metrics import confusion_matrix

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

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

x_train.shape: (50000, 32, 32, 3)
y_train.shape: (50000,)
x_test.shape: (10000, 32, 32, 3)
y_test.shape: (10000,)

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

model = Model(i, x)

Relatório sobre a arquitetura da rede
model.summary()

#Compilar o modelo
model.compile(optimizer="adam",
loss="sparse_categorical_crossentropy", metrics=["accuracy"])

```

```

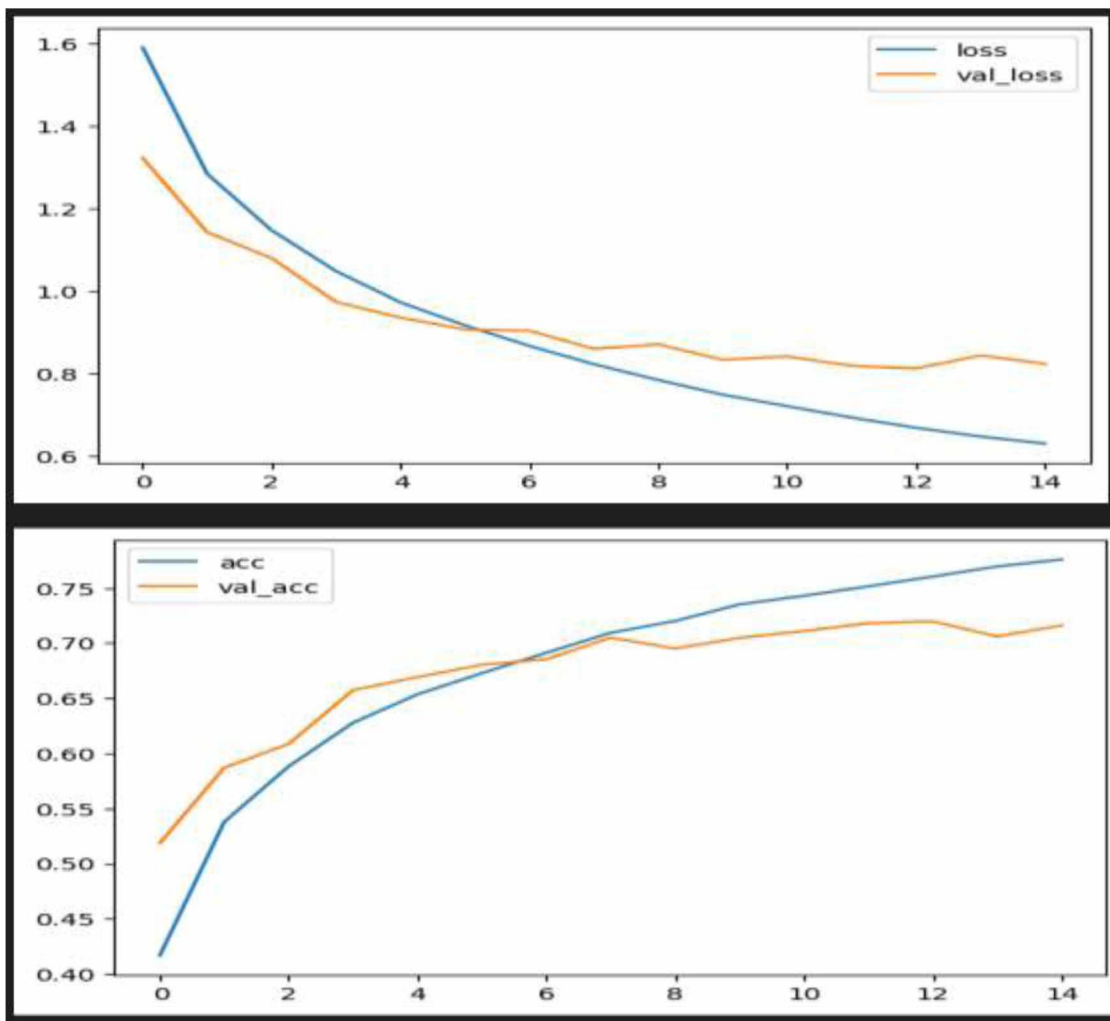
Treinar o modelo
r = model.fit(x_train, y_train, validation_data=(x_test, y_test),
epochs=15)

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

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

```

GRÁFICO 11 – FUNÇÃO DE PERDA, TREINO E ACURÁCIA CNN



FONTE: Amauri (2025).

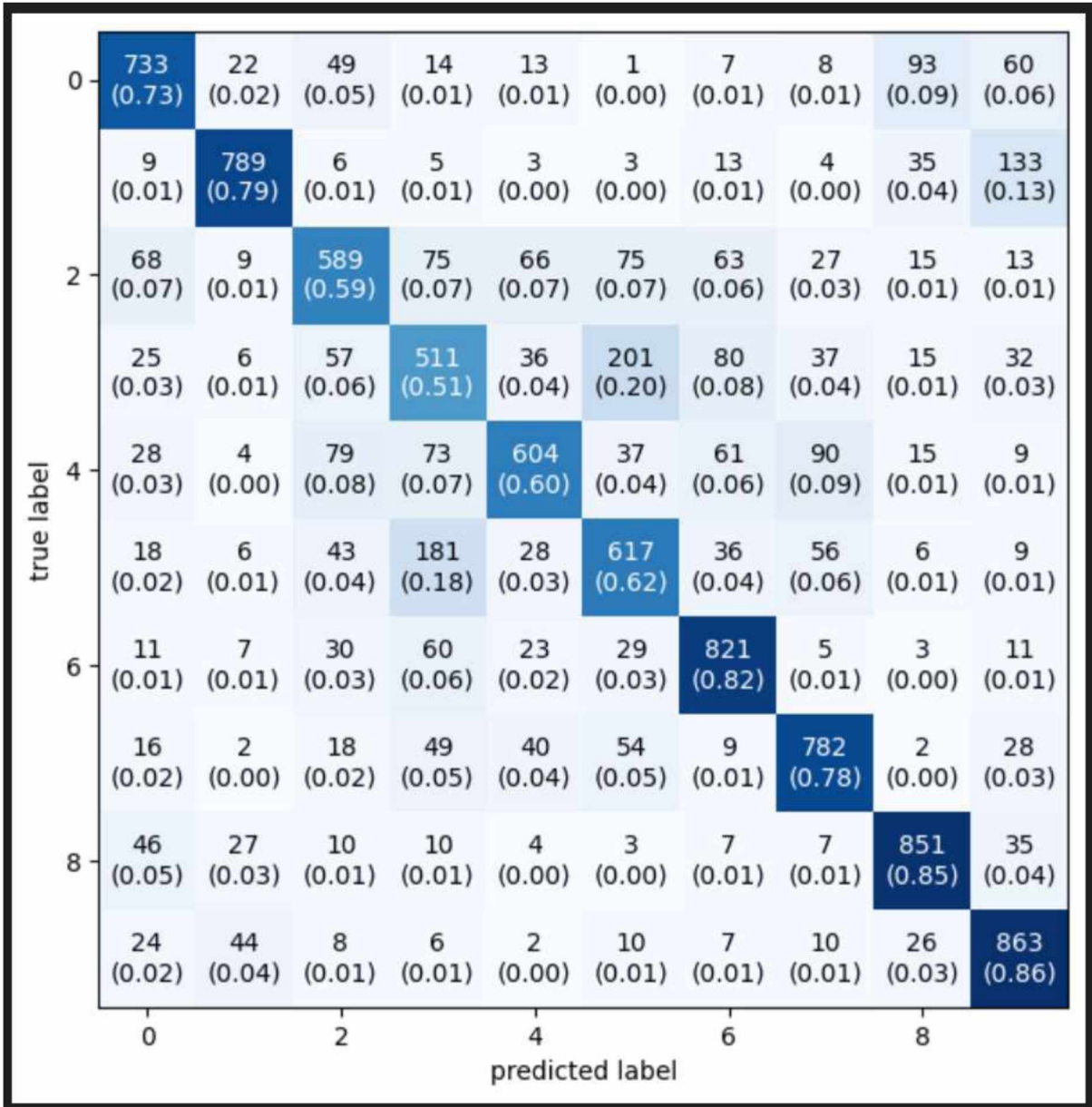
```

Efetuar predições na base de teste
argmax é usado pois a função de ativação da saída é softmax

```

```
argmax pega o neurônio que deu o maior resultado, isto é,
a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)
Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
show_normed=True)
```

GRÁFICO 12 – MATRIZ DE CONFUSÃO



FONTE: Amauri (2025).

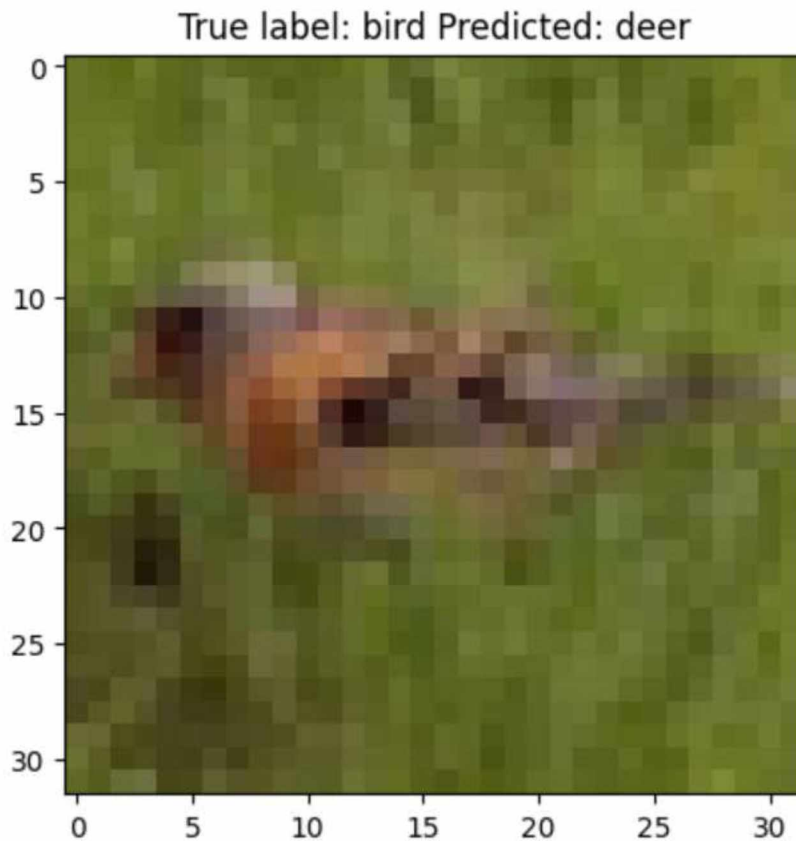
```
Mostrar algumas classificações erradas
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog",
```

```

"frog", "horse", "ship", "truck"]
misclassified = np.where(y_pred != y_test)[0]
i = np.random.choice(misclassified)
plt.imshow(x_test[i], cmap="gray")
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]],
labels[y_pred[i]]))

```

FIGURA 12 – IMAGEM DE CLASSIFICAÇÃO ERRADA



FONTE: Amauri (2025).

### Classificação Spam (RNN):

```

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

#carrega e arruma a base

```

```

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

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

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

7188 tokens

Acerta o tamanho das seqüências (padding)
data_train = pad_sequences(sequences_train) # usa o tamanho da maior
seq.
T = data_train.shape[1] # tamanho da seqüência
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)

Define o modelo
D = 20 # tamanho do embedding, hiperparâmetro que pode ser escolhido
M = 5 # tamanho do hidden state, quantidade de unidades LSTM
i = Input(shape=(T,)) # Entra uma frase inteira
x = Embedding(V+1, D)(i)
x = LSTM(M)(x)
x = Dense(1, activation="sigmoid")(x) # Sigmoide pois só tem 2 valores
model = Model(i, x)

model.summary()

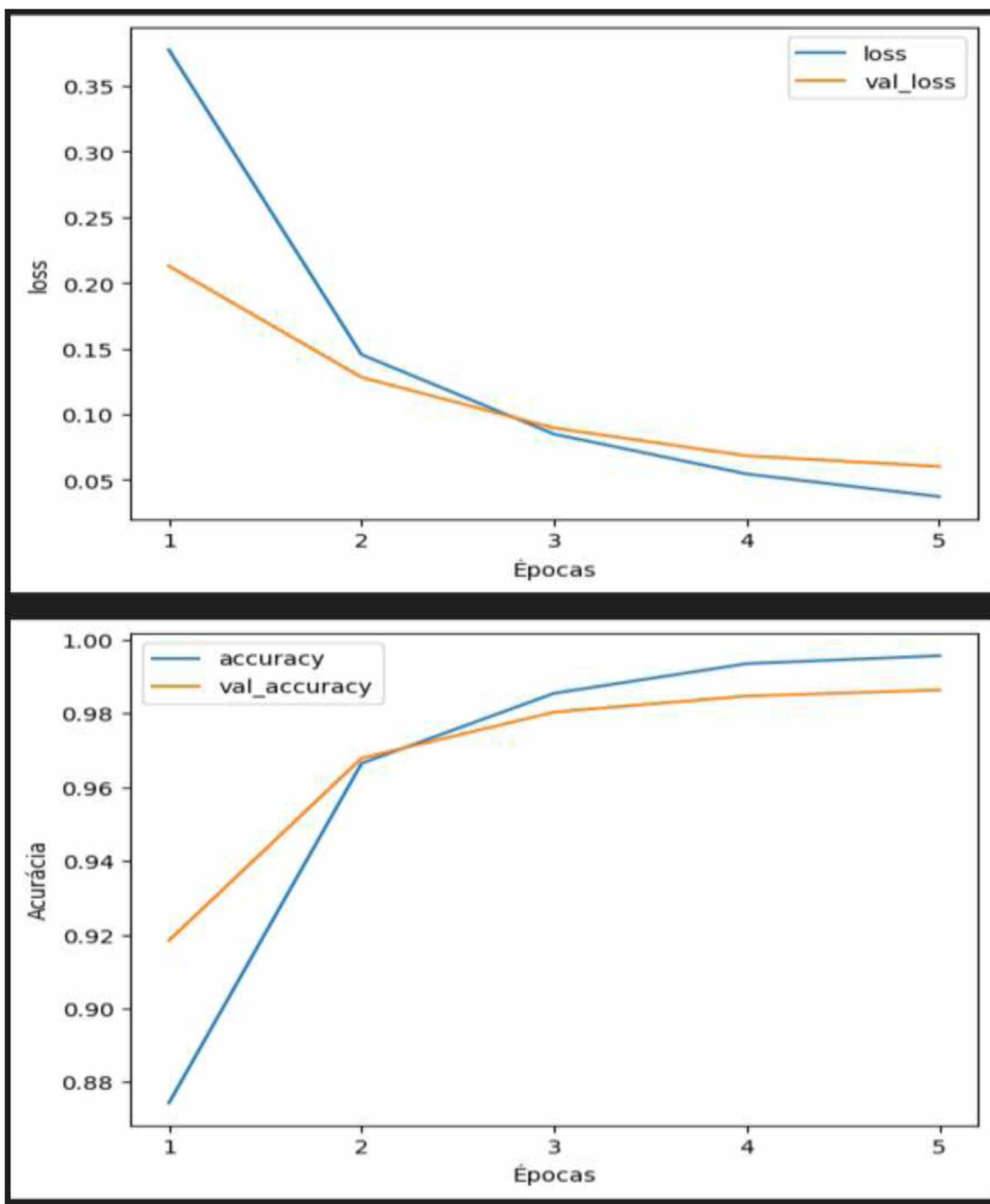
Compila e treina o modelo
model.compile(loss="binary_crossentropy", optimizer="adam",

```

```
metrics=["accuracy"])
epochs = 5
r = model.fit(data_train, y_train, epochs=epochs,
 validation_data=(data_test,
 y_test))

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

GRÁFICO 12 – FUNÇÃO DE PERDA, TREINO E ACURÁCIA RNN



FONTE: Amauri (2025).

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

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

Cria o GERADOR
def make_generator_model():
 model = tf.keras.Sequential()
 model.add(layers.Dense(7*7*256, use_bias=False, input_shape=(100,)))
 model.add(layers.BatchNormalization())
 model.add(layers.LeakyReLU())
 model.add(layers.Reshape((7, 7, 256)))
 assert model.output_shape == (None, 7, 7, 256)
 # Note: None is the batch size
 model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
padding='same', use_bias=False))
 assert model.output_shape == (None, 7, 7, 128)
 model.add(layers.BatchNormalization())
 model.add(layers.LeakyReLU())

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

```

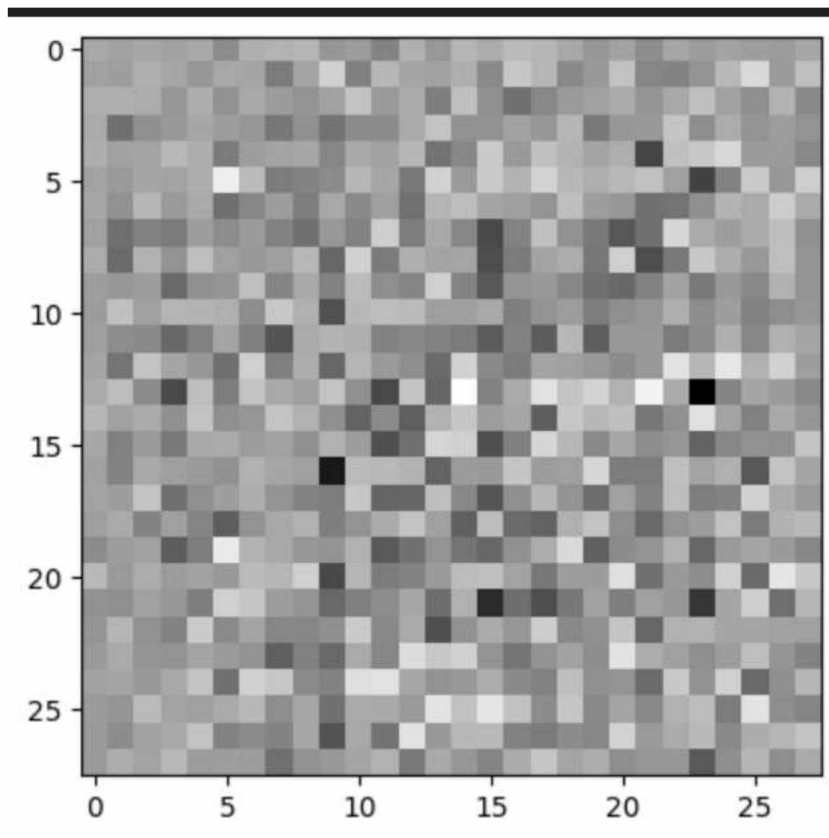
```

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

Teste do GERADOR, ainda não treinado
generator = make_generator_model()
noise = tf.random.normal([1, 100])
generated_image = generator(noise, training=False)
plt.imshow(generated_image[0, :, :, 0], cmap='gray')

```

FIGURA 13 – IMAGEM GERADA ANTES DE TREINO



FONTE: Amauri (2025).

```

Cria o DISCRIMINADOR
def make_discriminator_model():
 model = tf.keras.Sequential()
 model.add(layers.Conv2D(64, (5, 5), strides=(2, 2), padding='same',
input_shape=[28, 28, 1]))
 model.add(layers.LeakyReLU())
 model.add(layers.Dropout(0.3))
 model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
 model.add(layers.LeakyReLU())

```

```

model.add(layers.Dropout(0.3))
model.add(layers.Flatten())
model.add(layers.Dense(1))
return model

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

Define as funções de perda
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)
Perda do DISCRIMINADOR
def discriminator_loss(real_output, fake_output):
 real_loss = cross_entropy(tf.ones_like(real_output), real_output)
 fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
 total_loss = real_loss + fake_loss
 return total_loss
Perda do GERADOR
def generator_loss(fake_output):
 return cross_entropy(tf.ones_like(fake_output), fake_output)

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

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

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

Função que faz um passo de treinamento
É uma `tf.function`, que compila essa função

```

```

@tf.function
def train_step(images):
 noise = tf.random.normal([BATCH_SIZE, noise_dim])
 with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
 generated_images = generator(noise, training=True)
 real_output = discriminator(images, training=True)
 fake_output = discriminator(generated_images, training=True)
 gen_loss = generator_loss(fake_output)
 disc_loss = discriminator_loss(real_output, fake_output)
 gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
 gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)
 generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))
 discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator
, discriminator.trainable_variables))

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()

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

 for image_batch in dataset:
 train_step(image_batch)

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

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

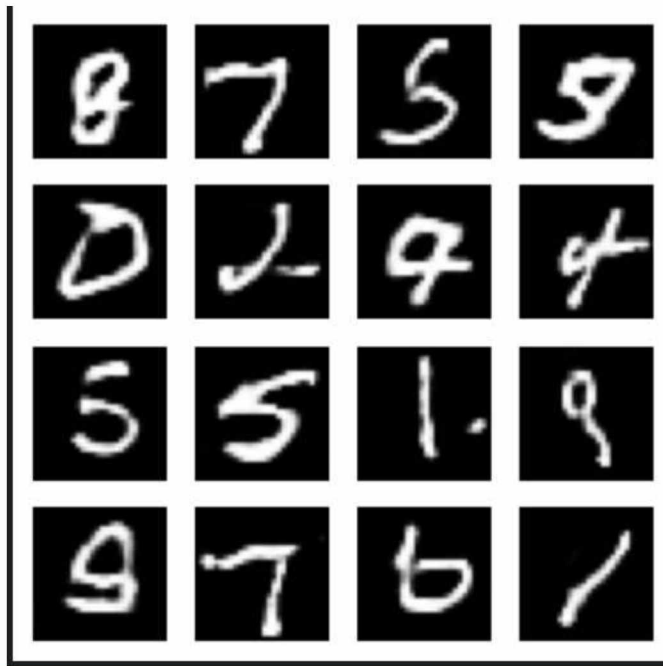
 print ('Time for epoch {} is {} sec'.format(epoch + 1, time.time()-
start))

```

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

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

FIGURA 14 – IMAGEM ÚLTIMO PONTO DE VERIFICAÇÃO



FONTE: Amauri (2025).

Tradutor de Textos (Transformer):

```
Instalação e importação
!pip uninstall tensorflow
!pip install tensorflow==2.15.0
!pip install tensorflow_datasets
!pip install -U tensorflow-text==2.15.0
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
import logging
logging.getLogger('tensorflow').setLevel(logging.ERROR) # suppress
warnings

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

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

 print()

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

e quando melhoramos a procura , tiramos a única vantagem da impressão ,
que é a serendipidade .
mas e se estes fatores fossem ativos ?
mas eles não tinham a curiosidade de me testar .

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

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

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

```

```

Faz padding com zeros.
pt = pt.to_tensor()
en = tokenizers.en.tokenize(en)
ragged -> dense
en = en.to_tensor()
return pt, en

Pipeline simples: processa, embaralha, agrupa os dados, prefetch
Datasets de entrada terminam com prefetch
BUFFER_SIZE = 20000
BATCH_SIZE = 64
def make_batches(ds):
 return (
 ds
 .cache()
 .shuffle(BUFFER_SIZE)
 .batch(BATCH_SIZE)
 .map(tokenize_pairs, num_parallel_calls=tf.data.AUTOTUNE)
 .prefetch(tf.data.AUTOTUNE))

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

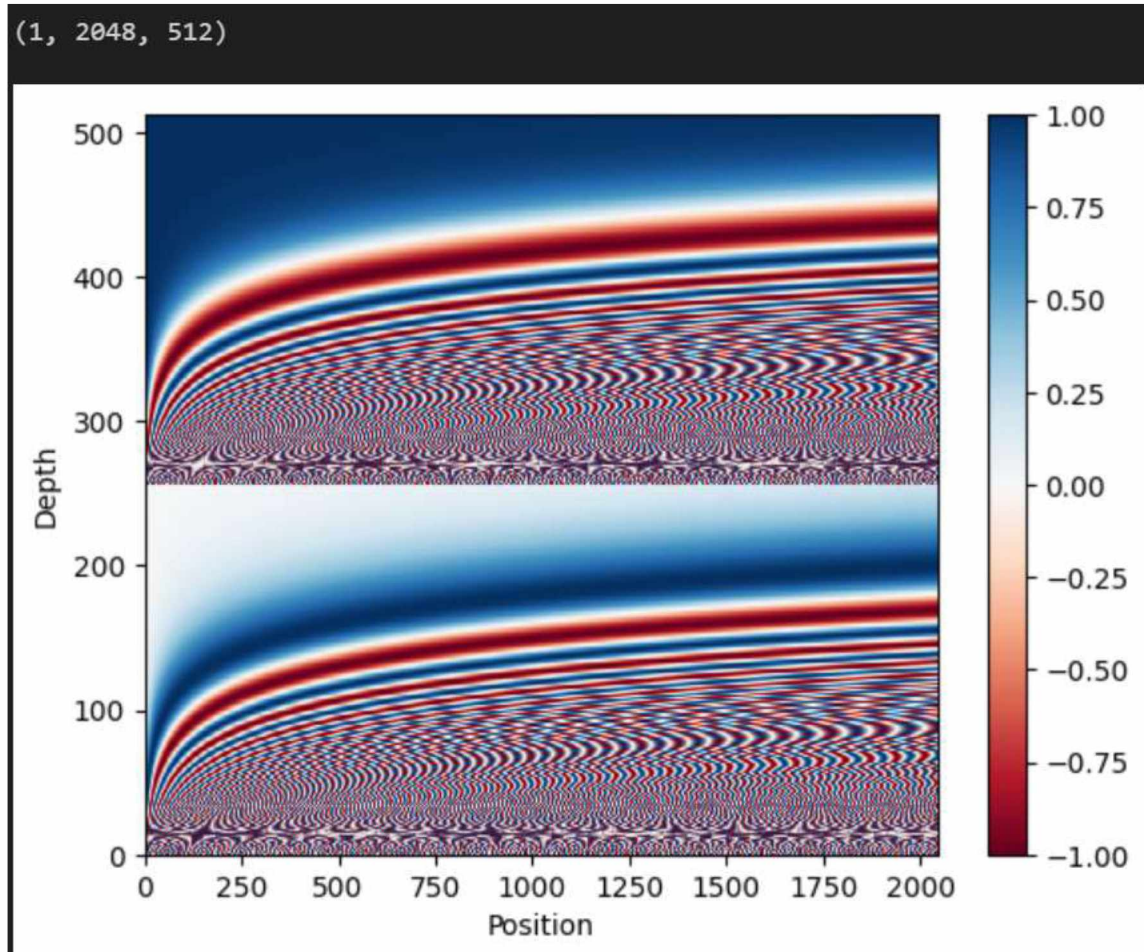
CODIFICAÇÃO POSICIONAL
def get_angles(pos, i, d_model):
 angle_rates = 1 / np.power(10000, (2 * (i//2)) / np.float32(d_model))
 return pos * angle_rates
def positional_encoding(position, d_model):
 angle_rads = get_angles(np.arange(position)[: , np.newaxis],
 np.arange(d_model)[np.newaxis, :],
 d_model)
 # sin em índices pares no array; 2i
 angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
 # cos em índices ímpares no array; 2i+1
 angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
 # newaxis, aumenta a dimensão [] -> [[]]
 pos_encoding = angle_rads[np.newaxis, ...]
 return tf.cast(pos_encoding, dtype=tf.float32)

CODIFICAÇÃO POSICIONAL
n, d = 2048, 512
pos_encoding = positional_encoding(n, d)
print(pos_encoding.shape)
pos_encoding = pos_encoding[0]
Arrumar as dimensões
pos_encoding = tf.reshape(pos_encoding, (n, d//2, 2))
pos_encoding = tf.transpose(pos_encoding, (2, 1, 0))
pos_encoding = tf.reshape(pos_encoding, (d, n))
plt.pcolormesh(pos_encoding, cmap='RdBu')

```

```
plt.ylabel('Depth')
plt.xlabel('Position')
plt.colorbar()
plt.show()
```

GRÁFICO 13 – GRÁFICO POSICIONAL DO TEXTO



FONTE: Amauri (2025).

```
Cria uma máscara de 0 e 1, 0 para quando há valor e 1 quando não há
def create_padding_mask(seq):
 seq = tf.cast(tf.math.equal(seq, 0), tf.float32)
 # add extra dimensions to add the padding
 # to the attention logits.
 return seq[:, tf.newaxis, tf.newaxis, :] # (batch_size, 1, 1, seq_len)
Máscara futura, usada no decoder
def create_look_ahead_mask(size):
 # zera o triângulo inferior
 mask = 1 - tf.linalg.band_part(tf.ones((size, size)), -1, 0)
 return mask # (seq_len, seq_len)
```

```

Função de Atenção
def scaled_dot_product_attention(q, k, v, mask):
 # Q K^T
 matmul_qk = tf.matmul(q, k, transpose_b=True) # (... , seq_len_q,
seq_len_k)
 # converte matmul_qk para float32
 dk = tf.cast(tf.shape(k)[-1], tf.float32)
 # divide por sqrt(d_k)
 scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
 # Soma a máscara, e os valores faltantes serão um número próximo a -
inf if mask is not None:
 scaled_attention_logits += (mask * -1e9)
 # softmax normaliza os dados, soman 1. // (... , seq_len_q, seq_len_k)
 attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)
 output = tf.matmul(attention_weights, v) # (... , seq_len_q, depth_v)
 return output, attention_weights

Atenção Multi-cabeças
class MultiHeadAttention(tf.keras.layers.Layer):

 def __init__(self, d_model, num_heads):
 super(MultiHeadAttention, self).__init__()
 self.num_heads = num_heads
 self.d_model = d_model
 assert d_model % self.num_heads == 0
 self.depth = d_model // self.num_heads
 self.wq = tf.keras.layers.Dense(d_model)
 self.wk = tf.keras.layers.Dense(d_model)
 self.wv = tf.keras.layers.Dense(d_model)
 self.dense = tf.keras.layers.Dense(d_model)

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

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

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

```

```

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

 # Calcula a atenção para cada cabeça (de forma matricial)
 # scaled_attention.shape == (batch_size, num_heads, seq_len_q,
depth)
 # attention_weights.shape == (batch_size, num_heads, seq_len_q,
seq_len_k)
 scaled_attention, attention_weights =
scaled_dot_product_attention(q, k, v, mask)
 # Troca a dimensão 2 com 1, para acertar o num_heads
 # (batch_size, seq_len_q, num_heads, depth)
 scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
 # Concatena os valores em: (batch_size, seq_len_q, d_model)
 concat_attention = tf.reshape(scaled_attention, (batch_size, -1,
self.d_model))
 output = self.dense(concat_attention) #(batch_size, seq_len_q,
d_model)
 return output, attention_weights

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

class EncoderLayer(tf.keras.layers.Layer):

 def __init__(self, d_model, num_heads, dff, rate=0.1):
 super(EncoderLayer, self).__init__()
 self.mha = MultiHeadAttention(d_model, num_heads)
 self.ffn = point_wise_feed_forward_network(d_model, dff)
 self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
 self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
 self.dropout1 = tf.keras.layers.Dropout(rate)
 self.dropout2 = tf.keras.layers.Dropout(rate)

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

```

```

 return out2

class DecoderLayer(tf.keras.layers.Layer):

 def __init__(self, d_model, num_heads, dff, rate=0.1):
 super(DecoderLayer, self).__init__()
 self.mha1 = MultiHeadAttention(d_model, num_heads)
 self.mha2 = MultiHeadAttention(d_model, num_heads)
 self.ffn = point_wise_feed_forward_network(d_model, dff)
 self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
 self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
 self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
 self.dropout1 = tf.keras.layers.Dropout(rate)
 self.dropout2 = tf.keras.layers.Dropout(rate)
 self.dropout3 = tf.keras.layers.Dropout(rate)

 def call(self, x, enc_output, training, look_ahead_mask,
padding_mask):
 # enc_output.shape == (batch_size, input_seq_len, d_model)
 # (batch_size, target_seq_len, d_model)
 attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
 attn1 = self.dropout1(attn1, training=training)
 out1 = self.layernorm1(attn1 + x)
 # (batch_size, target_seq_len, d_model)
 attn2, attn_weights_block2 = self.mha2(enc_output, enc_output, out1,
padding_mask)
 attn2 = self.dropout2(attn2, training=training)
 out2 = self.layernorm2(attn2 + out1) # (batch_size, target_seq_len,
d_model)
 ffn_output = self.ffn(out2) # (batch_size, target_seq_len, d_model)
 ffn_output = self.dropout3(ffn_output, training=training)
 out3 = self.layernorm3(ffn_output + out2) # (batch_size,
target_seq_len, d_model)
 return out3, attn_weights_block1, attn_weights_block2

class Encoder(tf.keras.layers.Layer):

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

```

```

def call(self, x, training, mask):
 seq_len = tf.shape(x)[1]
 # adding embedding and position encoding.
 x = self.embedding(x) # (batch_size, input_seq_len, d_model)
 x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
 x += self.pos_encoding[:, :seq_len, :]
 x = self.dropout(x, training=training)
 for i in range(self.num_layers):
 x = self.enc_layers[i](x, training, mask)
 return x # (batch_size, input_seq_len, d_model)

```

```

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

```

```

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

```

```

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

```

```

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

 def call(self, inputs, training):
 # Keras models prefer if you pass all your inputs in the first
 # argument
 inp, tar = inputs
 enc_padding_mask, look_ahead_mask, dec_padding_mask =
self.create_masks(inp, tar)
 # (batch_size, inp_seq_len, d_model)
 enc_output = self.encoder(inp, training, enc_padding_mask)
 # dec_output.shape == (batch_size, tar_seq_len, d_model)
 dec_output, attention_weights = self.decoder(
 tar, enc_output, training, look_ahead_mask, dec_padding_mask)
 # (batch_size, tar_seq_len, target_vocab_size)
 final_output = self.final_layer(dec_output)
 return final_output, attention_weights

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

Hiperparâmetros
num_layers = 4
d_model = 128
dff = 512
num_heads = 8
dropout_rate = 0.1

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

```

```

 super(CustomSchedule, self).__init__()
 self.d_model = d_model
 self.d_model = tf.cast(self.d_model, tf.float32)
 self.warmup_steps = warmup_steps
 def __call__(self, step):
 step = tf.cast(step, tf.float32) # Adicionado para evitar ERRO
 arg1 = tf.math.rsqrt(step)
 arg2 = step * (self.warmup_steps ** -1.5)
 return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)
learning_rate = CustomSchedule(d_model)
optimizer = tf.keras.optimizers.Adam(learning_rate, beta_1=0.9,
beta_2=0.98, epsilon=1e-9)

loss_object =
tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
 mask = tf.math.logical_not(tf.math.equal(real, 0))
 loss_ = loss_object(real, pred)
 mask = tf.cast(mask, dtype=loss_.dtype)
 loss_ *= mask
 return tf.reduce_sum(loss_)/tf.reduce_sum(mask)
def accuracy_function(real, pred):
 accuracies = tf.equal(real, tf.argmax(pred, axis=2))
 mask = tf.math.logical_not(tf.math.equal(real, 0))
 accuracies = tf.math.logical_and(mask, accuracies)
 accuracies = tf.cast(accuracies, dtype=tf.float32)
 mask = tf.cast(mask, dtype=tf.float32)
 return tf.reduce_sum(accuracies)/tf.reduce_sum(mask)
train_loss = tf.keras.metrics.Mean(name='train_loss')
train_accuracy = tf.keras.metrics.Mean(name='train_accuracy')

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

Checkpoint
checkpoint_path = "./checkpoints/train"
ckpt = tf.train.Checkpoint(transformer=transformer,
optimizer=optimizer)

```

```

ckpt_manager = tf.train.CheckpointManager(ckpt, checkpoint_path,
max_to_keep=5)
if a checkpoint exists, restore the latest checkpoint.
if ckpt_manager.latest_checkpoint:
 ckpt.restore(ckpt_manager.latest_checkpoint)
 print('Latest checkpoint restored!!')

EPOCHS = 20

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

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

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

for epoch in range(EPOCHS):
 start = time.time()
 train_loss.reset_state()
 train_accuracy.reset_state()
 #inp -> portuguese, tar -> english
 for (batch, (inp, tar)) in enumerate(train_batches):
 train_step(inp, tar)
 if batch % 50 == 0:
 print(f'Epoch {epoch + 1} Batch {batch} Loss
{train_loss.result():.4f} Accuracy {train_accuracy.result():.4f}')
 if (epoch + 1) % 5 == 0:
 ckpt_save_path = ckpt_manager.save()
 print(f'Saving checkpoint for epoch {epoch+1} at
{ckpt_save_path}')
 print(f'Epoch {epoch + 1} Loss {train_loss.result():.4f} Accuracy
{train_accuracy.result():.4f}')
 print(f'Time taken for 1 epoch: {time.time() - start:.2f} secs\n')

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

```

```

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

 for i in tf.range(max_length):
 output = tf.transpose(output_array.stack())
 predictions, _ = self.transformer([encoder_input, output],
training=False)
 predictions = predictions[:, -1:, :] # (batch_size, 1, vocab_size)
 predicted_id = tf.argmax(predictions, axis=-1)
 output_array = output_array.write(i+1, predicted_id[0])
 if predicted_id == end:
 break
 output = tf.transpose(output_array.stack())
 # output.shape (1, tokens)
 text = tokenizers.en.detokenize(output)[0]
 tokens = tokenizers.en.lookup(output)[0]
 _, attention_weights = self.transformer([encoder_input, output[:, :-
1]],
training=False)
 return text, tokens, attention_weights

translator = Translator(tokenizers, transformer)
sentence = "Eu li sobre triceratops na enciclopédia."
translated_text, translated_tokens, attention_weights = translator(
tf.constant(sentence))
print(f'{"Prediction":15s}: {translated_text}')

```

## APÊNDICE H - 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

#### Características dos Dados e os Vs do Big Data

- **Volume:** A aplicação financeira gera um grande volume de dados diariamente, desde transações financeiras até logs de acesso, cliques, e movimentação na interface do aplicativo.
- **Velocidade:** A captura dos dados ocorre em tempo real, uma vez que cada ação precisa ser registrada assim que ocorre para garantir a precisão e minimizar o tempo de resposta em disputas.
- **Variedade:** Os dados consistem de diferentes tipos, como logs de atividades, dados de transações financeiras, geolocalização, biometria comportamental (padrões de digitação, por exemplo) e metadados dos dispositivos usados.
- **Veracidade:** Os dados precisam ser precisos e confiáveis para serem usados como prova em disputas.
- **Valor:** O uso adequado dos dados trará valor à aplicação financeira, minimizando fraudes e garantindo a credibilidade das ações dos usuários.

Para lidar com esse cenário de Big Data, são utilizadas as seguintes ferramentas:

- **NoSQL:** O sistema usará uma base de dados NoSQL como MongoDB ou Cassandra para armazenar logs de atividades e eventos em tempo real. Esse tipo de banco de dados é ideal para armazenar dados não estruturados e semi-estruturados, como logs e informações sobre interações com o sistema.

- **Modelagem NoSQL:** Os dados serão organizados em documentos ou colunas, conforme necessário. Cada interação do usuário pode ser armazenada em um documento contendo detalhes como ID do usuário, ação realizada, data e hora, dispositivo utilizado, geolocalização, entre outros. Esse modelo permite uma rápida inserção e consulta de dados em grandes volumes.
- **NewSQL:** O NewSQL será usado para garantir que as transações financeiras sejam realizadas de maneira segura e consistente, preservando as propriedades ACID (Atomicidade, Consistência, Isolamento, Durabilidade).
- **Modelagem NewSQL:** As transações financeiras seguem um modelo de dados mais estruturado, onde cada transação é registrada com detalhes rigorosos, como IDs de contas, montantes transferidos, e timestamps. Esse tipo de banco garante que, mesmo em um sistema distribuído, os dados transacionais estejam corretos e disponíveis.

#### **Fluxo de Dados:**

- O usuário realiza uma ação na aplicação (ex.: login, transferência de dinheiro).
- A ação é registrada em um banco NoSQL como um log de evento contendo informações detalhadas sobre o contexto da ação (localização, dispositivo, hora, etc.).
- Se a ação envolver uma transação financeira, ela será registrada em um banco de dados NewSQL para garantir a consistência e segurança da operação.
- Os dados coletados são analisados continuamente em busca de padrões anômalos que possam indicar comportamentos fraudulentos ou maliciosos.

Esse sistema de monitoramento garante que todas as ações dos usuários sejam devidamente registradas, oferecendo um histórico confiável em disputas sobre envolvimento em transações. Ao combinar ferramentas de NoSQL e NewSQL, o sistema é capaz de lidar com grandes volumes de dados, além de garantir a segurança e integridade das transações financeiras.

## Modelo de Dados NoSQL (MongoDB)

O banco de dados NoSQL será responsável por armazenar logs de atividades dos usuários na aplicação financeira. Os dados são armazenados em documentos, com uma estrutura flexível e adaptável a diferentes tipos de ações.

### Estrutura de Campos:

- **\_id**: Identificador único para o log de atividade.
- **user\_id**: ID do usuário que realizou a ação.
- **timestamp**: Data e hora da atividade, em formato ISO 8601.
- **activity\_type**: Tipo de ação realizada, por exemplo, "login", "transferencia\_pix", etc.
- **details**: Informações específicas da atividade, como a quantia transferida e as contas envolvidas.
- **device\_info**: Informações sobre o dispositivo usado para realizar a ação, incluindo ID do dispositivo, tipo de dispositivo, sistema operacional e versão do aplicativo.
- **location**: Geolocalização no momento da ação, com latitude, longitude, cidade e país.
- **biometric\_info**: Dados biométricos comportamentais, como velocidade de digitação, padrões de toque e gestos no dispositivo.

## Modelo de Dados NewSQL

O banco NewSQL será utilizado para transações financeiras e armazenamento de dados com garantias ACID. Aqui temos uma tabela estruturada para registrar essas transações.

Tabela: Financial\_Transactions

Campos da Tabela:

- **transaction\_id**: Identificador único da transação.
- **user\_id**: ID do usuário que realizou a transação.

- source\_account: Conta de origem da transferência.
- destination\_account: Conta de destino da transferência.
- amount: Valor da transferência.
- currency: Moeda da transação, por exemplo, "USD".
- timestamp: Data e hora em que a transação foi realizada. • status: Status da transação, como "pending", "completed", "failed", etc.
- approval\_code: Código de aprovação da transação, gerado durante o processamento.
- device\_id: ID do dispositivo usado para a transação (para verificar padrões entre diferentes dispositivos).
- location: Campo JSON que armazena informações de localização, como latitude e longitude.
- metadata: Campo JSON para armazenar dados adicionais relacionados à transação (por exemplo, informações extras que possam ser úteis para auditoria ou verificação).

## **Integração dos Dados**

Logs NoSQL: Captura todas as ações do usuário e armazena em tempo real. Esse armazenamento flexível permite a adaptação a vários tipos de atividades e facilita a análise de comportamento.

Transações NewSQL: Armazena as transações financeiras com todas as garantias ACID, mantendo a integridade e consistência dos dados, o que é essencial para operações críticas como transferências de dinheiro.

Essa combinação de modelos NoSQL para atividades gerais e NewSQL para transações financeiras fornece um sistema robusto para monitorar e rastrear atividades de usuários, ajudando a resolver disputas com base em dados confiáveis

## APÊNDICE I - VISÃO COMPUTACIONAL

### A – ENUNCIADO

#### 1) Extração de Características

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

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

Tarefas:

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

#### 2) Redes Neurais

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

Tarefas:

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

## B – RESOLUÇÃO

```
import os
import pandas as pd
from sklearn.model_selection import train_test_split

Caminhos das pastas para cada classe
class_folders = {
 "0": "/content/train_w/0",
 "1+": "/content/train_w/1",
 "2+": "/content/train_w/2",
 "3+": "/content/train_w/3"
}

Listar todas as imagens de cada pasta e armazená-las junto com os
rótulos das classes
data = []
for label, folder in class_folders.items():
 for img in os.listdir(folder):
 if img.endswith(".png"):
 patient_id = img.split('_')[0] # Extrair ID do paciente
 # (primeiros dígitos)
 image_path = os.path.join(folder, img)
 data.append({"patient_id": patient_id, "image_path":
image_path, "class": label})

Converter os dados em um DataFrame
df = pd.DataFrame(data)

Agrupar por ID do paciente para dividir por pacientes
patients = df['patient_id'].unique()

Dividir pacientes em 80% treino e 20% validação
train_patients, val_patients = train_test_split(patients, test_size=0.2,
random_state=42)

Criar conjuntos de treino e validação
train_df = df[df['patient_id'].isin(train_patients)]
```

```

val_df = df[df['patient_id'].isin(val_patients)]

Exibir o resultado da divisão
print("Dados de Treinamento:")
print(train_df.head())

print("\nDados de Validação:")
print(val_df.head())

import cv2
import numpy as np
from skimage.feature import local_binary_pattern
from skimage.io import imread

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

def extract_lbp_features(image_path):
 # Ler a imagem em escala de cinza
 image = imread(image_path, as_gray=True)

 # Calcular a representação LBP da imagem
 lbp = local_binary_pattern(image, n_points, radius, METHOD)

 # Calcular o histograma do LBP
 (hist, _) = np.histogram(lbp.ravel(),
 bins=np.arange(0, n_points + 3),
 range=(0, n_points + 2))

 # Normalizar o histograma
 hist = hist.astype("float")
 hist /= (hist.sum() + 1e-6)

 return hist

Extrair características LBP para os dados de treino
train_df['lbp_features'] =
train_df['image_path'].apply(extract_lbp_features)

Extrair características LBP para os dados de validação
val_df['lbp_features'] =
val_df['image_path'].apply(extract_lbp_features)

Salvar características LBP em CSV
train_df[['image_path', 'class',
' lbp_features']].to_csv('train_lbp_features.csv', index=False)

```

```

val_df[['image_path', 'class',
'lbp_features']].to_csv('val_lbp_features.csv', index=False)

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

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

Carregar o modelo VGG16 pré-treinado e remover a última camada
base_model = VGG16(weights='imagenet', include_top=True)
model = Model(inputs=base_model.input,
outputs=base_model.get_layer('fc2').output)

def extract_vgg16_features(image_path):
 # Carregar e pré-processar a imagem
 img = image.load_img(image_path, target_size=(224, 224))
 x = image.img_to_array(img)
 x = np.expand_dims(x, axis=0)
 x = preprocess_input(x)

 # Obter o vetor de características da penúltima camada do VGG16
 features = model.predict(x)
 return features.flatten()

Extrair características VGG16 para os dados de treino
train_df['vgg16_features'] =
train_df['image_path'].apply(extract_vgg16_features)

Extrair características VGG16 para os dados de validação
val_df['vgg16_features'] =
val_df['image_path'].apply(extract_vgg16_features)

Salvar características VGG16 em CSV
train_df[['image_path', 'class',
'vgg16_features']].to_csv('train_vgg16_features.csv', index=False)
val_df[['image_path', 'class',
'vgg16_features']].to_csv('val_vgg16_features.csv', index=False)

print("Extração de características VGG16 concluída e salva em CSV!")

import numpy as np
import pandas as pd

```

```

Função auxiliar para converter a representação em string do array de
volta para um array NumPy
def convert_string_to_array(string):
 # Remover colchetes ao redor e dividir por espaços
 return np.array([float(x) for x in string.strip('[]').split()])

Carregar características LBP
train_lbp = pd.read_csv('train_lbp_features.csv')
val_lbp = pd.read_csv('val_lbp_features.csv')

Aplicar a função de conversão para transformar strings em arrays
train_lbp['lbp_features'] =
train_lbp['lbp_features'].apply(convert_string_to_array)
val_lbp['lbp_features'] =
val_lbp['lbp_features'].apply(convert_string_to_array)

Preparar dados para treinamento
X_train = np.stack(train_lbp['lbp_features'].values)
y_train = train_lbp['class']

X_val = np.stack(val_lbp['lbp_features'].values)
y_val = val_lbp['class']

Agora prosseguir com o treinamento dos modelos Random Forest, SVM e
MLP

Recarregar os CSVs de características VGG16 de treino e validação
train_vgg16 = pd.read_csv('train_vgg16_features.csv')
val_vgg16 = pd.read_csv('val_vgg16_features.csv')

Verificar as colunas e as primeiras linhas dos dados
print(train_vgg16.columns)
print(train_vgg16.head())

Verificar se a coluna image_path existe e contém dados válidos
if 'image_path' in train_vgg16.columns:
 print("A coluna image_path existe!")
 print(train_vgg16['image_path'].head())
else:
 print("A coluna image_path está ausente!")

Verificar se os arquivos CSV possuem conteúdo
import os

train_file = 'train_vgg16_features.csv'
val_file = 'val_vgg16_features.csv'

Verificar tamanho dos arquivos
print(f"Tamanho do arquivo CSV de treino: {os.path.getsize(train_file)}
bytes")

```

```
print(f"Tamanho do arquivo CSV de validação: {os.path.getsize(val_file)} bytes")

Exibir diretamente as primeiras linhas do arquivo CSV
with open(train_file, 'r') as f:
 print(f"Primeiras 10 linhas de {train_file}:")
 for _ in range(10):
 print(f.readline())
```

## TRABALHO DE IAA011 – Visão Computacional

### 1 Extração de Características

Com base nos resultados do notebook Jupyter, o modelo Random Forest obteve o seguinte desempenho no conjunto de teste:

- Acurácia geral: 79%

Precisão:

- Classe 0: 82%
- Classe 1: 61%
- Classe 2: 68%
- Classe 3: 100%

Recall:

- Classe 0: 99%
- Classe 1: 42%
- Classe 2: 74%
- Classe 3: 99%

F1-Score:

- Classe 0: 90%
- Classe 1: 50%
- Classe 2: 71%
- Classe 3: 99%

Sensibilidade (Recall para a Classe 1): 63,33%

Especificidade (Taxa de Verdadeiros Negativos para a Classe 0): 99%

#### Análise Detalhada:

Acurácia Geral (79%): Isso significa que, no total, o modelo classificou corretamente 79% das amostras no conjunto de teste. É um desempenho sólido, especialmente considerando que os dados são divididos em quatro classes.

#### Precisão e Recall por Classe:

- Classe 0: Apresenta uma precisão de 82% e um recall de 99%, indicando que o modelo é muito bom em prever corretamente quando uma imagem pertence à Classe 0 (baixa chance de falsos positivos) e em detectar quase todas as imagens dessa classe (alto recall).
- Classe 1: Tem uma precisão de 61%, mas um recall mais baixo de 42%. Isso indica que o modelo frequentemente classifica imagens como Classe 1, mas erra em detectar todas as imagens corretas dessa classe. O F1-Score de 50% reflete esse equilíbrio entre precisão e recall.
- Classe 2: O desempenho é razoável, com uma precisão de 68% e um recall de 74%. Isso significa que o modelo consegue detectar a maioria das imagens da Classe 2, com poucos falsos positivos.
- Classe 3: Este é o desempenho mais forte, com 100% de precisão e 99% de recall. O modelo detecta quase todas as imagens da Classe 3 corretamente e não comete erros ao classificá-las.
- F1-Score: O F1-Score é a média harmônica entre precisão e recall, e reflete a capacidade do modelo de lidar com classes desbalanceadas. A Classe 3 tem um F1-Score muito alto (99%), o que indica que o modelo lida muito bem com essa classe. Já a Classe 1 tem o F1-Score mais baixo (50%), o que reflete o desafio do modelo em equilibrar precisão e recall nessa classe.

Sensibilidade (Recall para a Classe 1): A sensibilidade de 63,33% indica que o modelo foi moderadamente eficaz em detectar corretamente as imagens da Classe 1, embora ainda tenha perdido algumas.

Especificidade (99%): A especificidade muito alta mostra que o modelo foi extremamente eficaz em evitar falsos positivos para a Classe 0, o que é um bom sinal de que ele não está classificando incorretamente imagens que não pertencem a essa classe.

Conclusão:

O modelo Random Forest mostrou um bom desempenho, com destaque para a precisão e recall elevados nas classes 0 e 3. No entanto, o desempenho para a Classe 1 (precisão de 61% e recall de 42%) foi mais fraco, indicando que o modelo pode ter dificuldades em distinguir algumas imagens dessa classe. No geral, o modelo é altamente confiável para detectar as classes 0 e 3, enquanto as classes 1 e 2 podem beneficiar de ajustes adicionais ou técnicas de balanceamento de dados para melhorar o desempenho.

Dentre os modelos avaliados, o Random Forest apresentou o melhor desempenho, especialmente em termos de acurácia e F1-Score, com destaque para a Classe 3, onde o desempenho foi quase perfeito.

## 2 Redes Neurais

Acurácia no Teste (VGG16 sem Data Augmentation): 89,76%

O modelo obteve uma acurácia relativamente alta, o que indica que ele é capaz de fazer previsões corretas em cerca de 90% das imagens no conjunto de teste.

Métricas do Relatório de Classificação:

Abaixo estão as métricas detalhadas de precisão, recall e F1-score para cada uma das quatro classes (0, 1, 2 e 3):

Classe	Precisão			Recall	F1-Score	Suporte
0	0.26	0.25	0.25	101		
1	0.24	0.27	0.25	90		
2	0.27	0.29	0.28	90		
3	0.21	0.18	0.19	90		
Macro Avg	0.24	0.25	0.24	371		
Weighted Avg	0.24	0.25	0.24	371		

#### Precisão (Precision):

A precisão variou entre 21% e 27%, o que é relativamente baixa.

#### Recall (Sensibilidade):

O recall foi um pouco melhor, variando de 18% a 29%, mas ainda está abaixo do ideal.

#### F1-Score:

As pontuações variaram de 19% a 28%, indicando que o modelo tem dificuldades para equilibrar ambas as métricas.

#### Matriz de Confusão:

Classe Real \ Predita	0	1	2	3
0 (101 imagens)	25	28	27	21
1 (90 imagens)	24	24	21	21
2 (90 imagens)	25	19	26	20
3 (90 imagens)	24	29	21	16

#### Pontos-Chave:

O modelo teve maior dificuldade em diferenciar as classes 0 e 1, com confusão significativa entre elas.

A classe 2 teve um desempenho ligeiramente melhor em termos de recall e precisão.

O modelo teve mais dificuldade em prever a classe 3, com o menor F1-score (19%).

**Conclusão:**

Embora a acurácia geral pareça elevada (cerca de 90%), as métricas de precisão, recall e F1-score para cada classe revelam que o modelo está com dificuldades em distinguir corretamente entre as diferentes classes. O modelo provavelmente está superajustando ou não captando bem as diferenças entre as classes, o que explica as baixas métricas nas classificações de classe individual.

## APÊNDICE J - 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**

## **ESTUDO DE CASO: IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT**

AMAURI LEONEL KLAMAS JUNIOR

### **RESUMO**

O ChatGPT, é uma ferramenta inovadora de inteligência artificial que possui grande potencial em vários campos devido à sua interface amigável e capacidade de resposta rápida. Apesar dos benefícios, o ChatGPT traz riscos de segurança e questões éticas inevitáveis. É fundamental analisar a tecnologia do ChatGPT, suas limitações, questões éticas, caminhos de governança e direções futuras, equipando os usuários com o conhecimento necessário para tomar decisões informadas e lidar com desafios potenciais. O uso crescente de ferramentas de inteligência artificial (IA) em diversos setores sendo um deles o setor educacional, ferramentas como o ChatGPT, destaca tanto oportunidades quanto desafios éticos. Este agente conversacional baseado em IA oferece respostas contextualmente relevantes em tempo real, reconfigurando o aprendizado personalizado e o engajamento dos estudantes. No entanto, o aumento da incorporação dessa tecnologia em ambientes acadêmicos exige uma navegação cuidadosa nas paisagens éticas emergentes.

Palavras-chave: Inteligência Artificial. ChatGPT. Ética.

### **ABSTRACT**

ChatGPT is an innovative artificial intelligence tool that holds great potential in various fields due to its user-friendly interface and quick response capabilities. Despite its benefits, ChatGPT brings inevitable security risks and ethical issues. It is crucial to analyze ChatGPT's technology, its limitations, ethical issues, governance paths, and future directions, equipping users with the necessary knowledge to make informed decisions and handle potential challenges. The growing use of artificial intelligence (AI) tools in various sectors, including the educational sector, highlights both opportunities and ethical challenges. This AI-based conversational agent offers contextually relevant responses in real time, reconfiguring personalized learning and student engagement. However, the increased incorporation of this technology in academic environments demands careful navigation of the emerging ethical landscapes.

Keywords: Artificial Intelligence. ChatGPT. Ethic.

## 1 INTRODUÇÃO

O desenvolvimento rápido da inteligência artificial (IA), especialmente ferramentas como o ChatGPT, tem o potencial de transformar muitos campos em diferentes áreas, com uma das mais afetadas sendo a própria educação. A governança efetiva e a regulação adequada são essenciais para mitigar os riscos e maximizar os benefícios dessa tecnologia emergente e tirar o máximo proveito da tecnologia mantendo um desenvolvimento e uso responsável da ferramenta, sendo assim alguns pontos precisam ser levantados.

## 2 HISTÓRIA E CAPACIDADES DO CHATGPT

Desenvolvido pela OpenAI, o ChatGPT evoluiu a partir dos modelos de arquitetura GPT (Generative Pretrained Transformer), sendo este um modelo de linguagem desenvolvido pela OpenAI. A evolução do GPT-1 para o GPT-3 trouxe melhorias significativas na capacidade do modelo de gerar texto humano, destacando-se como a tecnologia de linguagem mais avançada até o momento.

Com o GPT-3 sendo um dos mais avançados, contendo 175 bilhões de parâmetros. Suas capacidades incluem a compreensão e geração de texto natural, criação de conteúdo, instruções específicas para tarefas, escrita de código e suporte multilíngue. Essas ferramentas têm sido aplicadas em vários setores, incluindo a educação, o qual servirá de estudo de caso para este trabalho.

Sam Altman, CEO da OpenAI, compartilhou várias percepções sobre o futuro do ChatGPT e da IA em geral. Ele acredita que a IA tem o potencial de ser uma das tecnologias mais transformadoras da história humana, impactando todos os aspectos da vida, desde o trabalho até as interações sociais. Altman prevê que a IA resolverá alguns dos desafios mais significativos do mundo, como a pobreza e as mudanças climáticas, enquanto reconhece os riscos, incluindo o uso indevido e a perda de empregos. (Supply Chain Today - Homepage, 2024, <https://www.supplychaintoday.com/openai-ceo-sam-altman-on-the-future-of-ai-and-chatgpt/>).

A adoção do ChatGPT em ambientes educacionais está ligada à integração crescente da tecnologia nas salas de aula modernas. Ele tem sido utilizado para fornecer experiências de aprendizado personalizadas, responder a perguntas e facilitar interações semelhantes às de colegas, sem as limitações humanas. No entanto, a dependência excessiva dessa tecnologia pode levantar preocupações

sobre a privacidade dos dados, o viés inerente dos dados de treinamento e a originalidade nas produções dos estudantes.

### **3 CONSIDERAÇÕES ÉTICAS**

As considerações éticas sobre o uso do ChatGPT abrangem diversas áreas, refletindo as implicações e desafios associados ao desenvolvimento e implementação de tecnologias de IA. Aqui estão alguns pontos principais.

#### **3.1.1 CONJUNTOS DE DADOS E TREINAMENTO**

O ChatGPT processa grandes quantidades de dados para funcionar de forma eficiente, o que levanta questões sobre como esses dados são tratados e protegidos. Não é claro os métodos que são utilizados na captura e tratamento destes dados, sendo levantado por diversas organizações se a utilização de dados pelo OpenAI poderia ser considerada como plágio e violação de direitos autorais.

Além disso existe a preocupação com os dados inseridos pelos usuários ao utilizar a ferramenta. Segundo a empresa OpenAI, ela compromete-se a não usar os dados para construir perfis de usuários, mas a conscientização e as melhores práticas ao interagir com a IA são essenciais para evitar a divulgação involuntária de informações pessoais.

O ChatGPT foi treinado em vastos conjuntos de dados, incluindo livros, textos da web e conjuntos de dados coletivos como CommonCrawl. Métodos de pré-treinamento e afinação supervisionada foram usados para melhorar o desempenho do modelo, o pré-treinamento não supervisionado ajuda o modelo a entender padrões gerais de linguagem. A afinação supervisionada e o aprendizado por reforço com feedback humano (RLHF) são usados para melhorar a precisão e relevância das respostas do modelo, entretanto o treinamento em grandes volumes de dados disponíveis na web pode levar à reprodução de vieses presentes nesses dados.

A quantidade de fontes e a qualidade dos dados selecionados podem não ser a melhor escolha para o treinamento podendo enviesar decisões erradas pela ferramenta. O treinamento do ChatGPT em grandes volumes de dados da web pode incorporar vieses existentes, afetando negativamente grupos sub-representados, semelhante ao que ocorreu com a ferramenta Gemini do Google, onde usuários pediram para serem criadas imagens de soldados nazistas e a IA retornou soldados com uniformes nazistas com representações de diversas raças.

É essencial que os usuários estejam cientes dessas limitações e preparados para abordar informações enviesadas ou distorcidas que o modelo possa produzir.

### 3.1.2 ALUCINAÇÕES

O ChatGPT pode gerar textos que parecem corretos, mas são sem sentido ou incorretos. Essas alucinações podem resultar em desinformação, violação de privacidade e uso indevido para fins ilegais. É fundamental que os usuários verifiquem as informações fornecidas pelo ChatGPT, especialmente em contextos críticos como saúde, finanças e educação.

### 3.1.3 ORIGINALIDADE

A originalidade dos textos gerados é baixa, muitas vezes combinando ideias existentes sem inovação real. Portanto, problemas de plágio e dependência técnica são preocupações significativas na utilização da ferramenta. Instituições e usuários devem ser educados sobre a importância de usar o ChatGPT como uma ferramenta de apoio e não como um substituto para a criatividade e o pensamento original.

### 3.1.4 TOXICIDADE

O A coleta massiva de dados levanta questões sobre invasão de privacidade e segurança de dados. Há preocupações sobre o uso não autorizado de dados pessoais e vazamentos de dados. É essencial que as empresas de IA implementem rigorosos protocolos de segurança e garantam a transparência no uso dos dados dos usuários. modelo pode produzir conteúdo prejudicial ou discriminatório devido a dados de treinamento enviesados. A toxicidade pode ser descrita como a prática de preconceitos culturais, linguísticos, temporais e políticos. Desenvolvedores devem continuar a trabalhar em mecanismos para identificar e mitigar a toxicidade, garantindo que a IA promova interações positivas e inclusivas.

### 3.1.5 PRIVACIDADE

A coleta massiva de dados levanta questões sobre invasão de privacidade e segurança de dados. O ChatGPT coleta grandes quantidades de dados pessoais, levantando preocupações significativas de privacidade devido ao potencial de uso indevido de dados ou divulgação a terceiros sem o conhecimento ou consentimento dos usuários. Há preocupações sobre o uso não autorizado de dados pessoais e

vazamentos de dados. É essencial que as empresas de IA implementem rigorosos protocolos de segurança e garantam a transparência no uso dos dados dos usuários. Sendo necessário regulamentações de proteção de dados e restrições de usuários para melhorar a transparência e a responsabilidade.

### 3.1.6 USO MALICIOSO DO CHATGPT

Sistemas de IA como o ChatGPT podem ser usados maliciosamente em ataques cibernéticos, campanhas de desinformação e esquemas de phishing. Sua linguagem muito próxima de uma pessoa comum, pode ser usada para conversar com centenas de milhares de usuários para um esquema mais elaborado de phishing. Para mitigar esses riscos, recomenda-se uma maior regulamentação do desenvolvimento e implementação de IA, melhores medidas de ciber-segurança e maior conscientização pública sobre as ameaças habilitadas por IA.

### 3.1.7 IMPACTOS NA INTERAÇÃO HUMANA E HABILIDADES SOCIAIS

Embora os chatbots possam ajudar a reduzir a solidão entre idosos, há preocupações sobre os impactos negativos no uso de chatbots para comunicação, incluindo a diminuição da empatia e da satisfação com o apoio social. É necessário um foco maior nas implicações sociais e psicológicas dos sistemas de IA, bem como maior financiamento para o desenvolvimento de habilidades sociais e programas de apoio.

### 3.1.8 SUSTENTABILIDADE

O treinamento de modelos LLM como o ChatGPT consome grandes quantidades de energia, contribuindo para emissões de carbono e impactos ambientais. Pesquisadores e desenvolvedores devem buscar soluções mais eficientes em termos de energia para reduzir a pegada de carbono associada ao desenvolvimento e operação de IA.

## 3.2 IMPLICAÇÕES ÉTICAS DO USO DO CHATGPT NA EDUCAÇÃO

O uso do ChatGPT pode comprometer a integridade acadêmica, com estudantes potencialmente usando a IA para completar tarefas, desviando do esforço necessário para o aprendizado autêntico. Instituições educacionais estão desenvolvendo diretrizes sobre o uso ético dessas ferramentas, promovendo a

integridade acadêmica e incentivando métodos de avaliação que valorizem o pensamento crítico e a originalidade. Porém, a dependência excessiva de ferramentas de IA pode inibir o desenvolvimento de habilidades essenciais como o pensamento crítico e a resolução de problemas. É importante equilibrar o uso da IA com métodos tradicionais de ensino para garantir um aprendizado holístico.

Também deve-se levar em conta o acesso equitativo a ferramentas como o ChatGPT, isto pode ser fundamental para estudantes em certas realidades socioeconômicas, especialmente para estudantes de diferentes origens e com necessidades especiais. Iniciativas para fornecer internet de baixo custo, concessões tecnológicas e treinamento para educadores são essenciais para maximizar os benefícios dessas tecnologias.

#### **4 SOLUÇÕES RESPONSÁVEIS**

Vários países estão desenvolvendo regulamentos para a IA. Exemplos incluem o "Blueprint for an Artificial Intelligence Bill of Rights" dos EUA e o "Artificial Intelligence Act" da EU, por aqui temos o projeto de lei (PL) 2.338/2023, que regulamenta o desenvolvimento e o uso da inteligência artificial (IA) no Brasil. É necessário que demais instituições governamentais criem regulamentações do uso e desenvolvimento da IA em seus países. Sendo assim, a governança deve equilibrar inovação com segurança e responsabilidade.

Algumas práticas mais tecnológicas podem incluir a implementação de mecanismos de controle internos e externos para ajudar a mitigar os riscos associados à IA. Métodos como a adição de sinais de recompensa de segurança durante o treinamento do RLHF podem ser usados para reduzir saídas prejudiciais.

Talvez o mais importante seria a cooperação entre desenvolvedores, reguladores e a sociedade. Essa cooperação seria crucial para garantir o desenvolvimento seguro e benéfico da tecnologia de IA. Além disso, promover a educação e a conscientização sobre o uso ético e responsável da IA pode ajudar a criar uma cultura de inovação responsável, onde os benefícios da tecnologia são amplamente compartilhados e os riscos são minimizados.

#### 4.1 WOEBOT CASO DE SUCESSO

O chatbot de saúde mental Woebot e chatbots usados na campanha contra a COVID-19 são exemplos de implementações éticas bem-sucedidas. Woebot é um chatbot baseado em IA projetado para fornecer suporte em saúde mental. Desenvolvido pela Woebot Labs, Inc., ele utiliza princípios da Terapia Cognitivo-Comportamental (TCC) para ajudar os usuários a gerenciar suas emoções e pensamentos negativos.

A Woebot Labs garante que os dados dos usuários são mantidos em segurança e não são compartilhados com terceiros sem consentimento. Os desenvolvedores são transparentes sobre as capacidades e limitações do Woebot, garantindo que os usuários saibam que estão interagindo com uma IA e não com um humano.

### 5 CONSIDERAÇÕES FINAIS

O desenvolvimento rápido da inteligência artificial, especialmente ferramentas como o ChatGPT, tem o potencial de transformar significativamente diversas áreas, com um impacto notável na educação. Embora essas tecnologias tragam inúmeros benefícios, é crucial abordar e mitigar os riscos e desafios éticos associados ao seu uso. Para garantir o desenvolvimento seguro e benéfico da IA, é crucial a cooperação entre desenvolvedores, reguladores e a sociedade. Promover a educação e a conscientização sobre o uso ético e responsável da IA ajudará a criar uma cultura de inovação responsável, onde os benefícios da tecnologia são amplamente compartilhados e os riscos são minimizados.

Em resumo, a IA tem o potencial de transformar positivamente nossas vidas, mas esse potencial deve ser equilibrado com uma abordagem responsável e ética para garantir que seus benefícios sejam acessíveis a todos, sem comprometer a segurança, a privacidade e a integridade social.

## REFERÊNCIAS

HUA, SHANGYING & JIN, SHUANGCI & JIANG, SHENGYI. (2023). **The Limitations and Ethical Considerations of ChatGPT**. Data Intelligence. 6. 1-38.

MHLANGA, DAVID. **Open AI in Education, the Responsible and Ethical Use of ChatGPT Towards Lifelong Learning**. SSRN Electronic Journal, 2023.

FLAIH, TAWFEEQ & JASIM, YASER. (2023). **The Ethical Implications of ChatGPT AI Chatbot: A Review**. 2023. 49-57.

Supply Chain Today. **"OpenAI CEO Sam Altman on the Future of AI and ChatGPT"**. Supply Chain Today, 2024. Disponível em: <https://www.supplychaintoday.com/openai-ceo-sam-altman-on-the-future-of-ai-and-chatgpt>

**"Woebot - Your Self-Care Expert"**. Ithaca College. Disponível em: <https://www.ithaca.edu/woebot-your-self-care-expert>

## APÊNDICE K - 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

Aluno/a: Amauri Leonel Klomas Junior		Nome do artigo escolhido: Requirements practices and gaps when engineering human-centered Artificial Intelligence systems	
Qual o <b>objetivo</b> do estudo descrito pelo artigo?	Qual o <b>problema/opportunidade/situação</b> que levou à necessidade de realização desse estudo?	Qual a <b>metodologia</b> que os autores usaram para obter e analisar as informações do estudo?	Quais os <b>principais resultados</b> obtidos pelo estudo?
A engenharia de software para IA é uma área emergente com muitos desafios, práticas ainda não consolidadas e limitada por diretrizes específicas. O foco principal do estudo foi investigar como profissionais da indústria abordam o desenvolvimento de sistemas de IA centrados no humano durante a fase de RE. O objetivo é identificar práticas existentes e diretrizes que devem ser seguidas para melhor incorporar aspectos humanos nos sistemas de IA.	O estudo foi motivado pelos desafios e oportunidades associados à engenharia de sistemas de Inteligência Artificial (IA) centrados no ser humano, um campo emergente que ainda carece de práticas bem definidas e diretrizes consolidadas.  O estudo busca abordar essas lacunas e alinhar a pesquisa acadêmica com as práticas industriais para criar sistemas de IA que não só sejam tecnicamente eficazes, mas também socialmente responsáveis e centrados no humano.	Foi conduzida uma pesquisa com 29 profissionais para mapear diretrizes industriais existentes e esforços relatados na literatura e identificar lacunas entre a pesquisa e a prática industrial no contexto da RE voltada para IA.	Ferramentas e práticas inadequadas: A maioria usa ferramentas genéricas, como Word e UML, que não atendem às necessidades específicas da IA, como ética, dados e explicabilidade.  Desafios principais: Dificuldades em especificar requisitos relacionados à qualidade de dados, explicabilidade e limitações das capacidades da IA.  Lacunas entre teoria e prática: A pesquisa enfatiza aspectos éticos e humanos, enquanto a prática ainda foca mais em requisitos funcionais e ferramentas comerciais.  Importância dos aspectos humanos: Todas as áreas identificadas (dados, explicabilidade, erros, etc.) foram consideradas relevantes, mas feedback e erros ainda são subexplorados.  O estudo destaca a necessidade de novas ferramentas e métodos específicos para a RE de IA.

## APÊNDICE L - 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<sup>2</sup>).

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](https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg)
- **Importação da imagem:** Copiar código abaixo.

`url = "https://commons.wikimedia.org/wiki/Special:FilePath/Felis\_catus-cat\_on\_snow.jpg"`

Dica: Para exibir a imagem utilizando `display` (`display.html`) use o link [https://commons.wikimedia.org/wiki/File:Felis\\_catus-cat\\_on\\_snow.jpg](https://commons.wikimedia.org/wiki/File:Felis_catus-cat_on_snow.jpg)

## B – RESOLUÇÃO

### 1 Classificação (RNA)

```
!pip install tensorflow matplotlib numpy # (Descomente se estiver
rodando em ambiente sem as libs)
import os
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

SEED = 42
tf.keras.utils.set_random_seed(SEED)
np.random.seed(SEED)
tf.config.experimental.enable_op_determinism()
print(tf.__version__)

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

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

Flatten
x_train_flat = x_train.reshape((-1, 28*28))
x_test_flat = x_test.reshape((-1, 28*28))

class_names = [
 'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat',
 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'
]
len(class_names)

model = keras.Sequential([
 layers.Input(shape=(28*28,)),
 layers.Dense(256, activation='relu'),
 layers.Dropout(0.2),
 layers.Dense(128, activation='relu'),
 layers.Dropout(0.2),
 layers.Dense(10, activation='softmax')
])
```

```

model.compile(
 loss=keras.losses.SparseCategoricalCrossentropy(),
 optimizer=keras.optimizers.Adam(learning_rate=1e-3),
 metrics=['accuracy']
)
model.summary()

callbacks = [
 keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True,
monitor='val_loss')
]

history = model.fit(
 x_train_flat, y_train,
 validation_split=0.1,
 epochs=30,
 batch_size=128,
 callbacks=callbacks,
 verbose=1
)

test_loss, test_acc = model.evaluate(x_test_flat, y_test, verbose=0)
print(f"Loss (teste): {test_loss:.4f} | Acurácia (teste):
{test_acc:.4f}")

Loss (teste): 0.3238 | Acurácia (teste): 0.8901

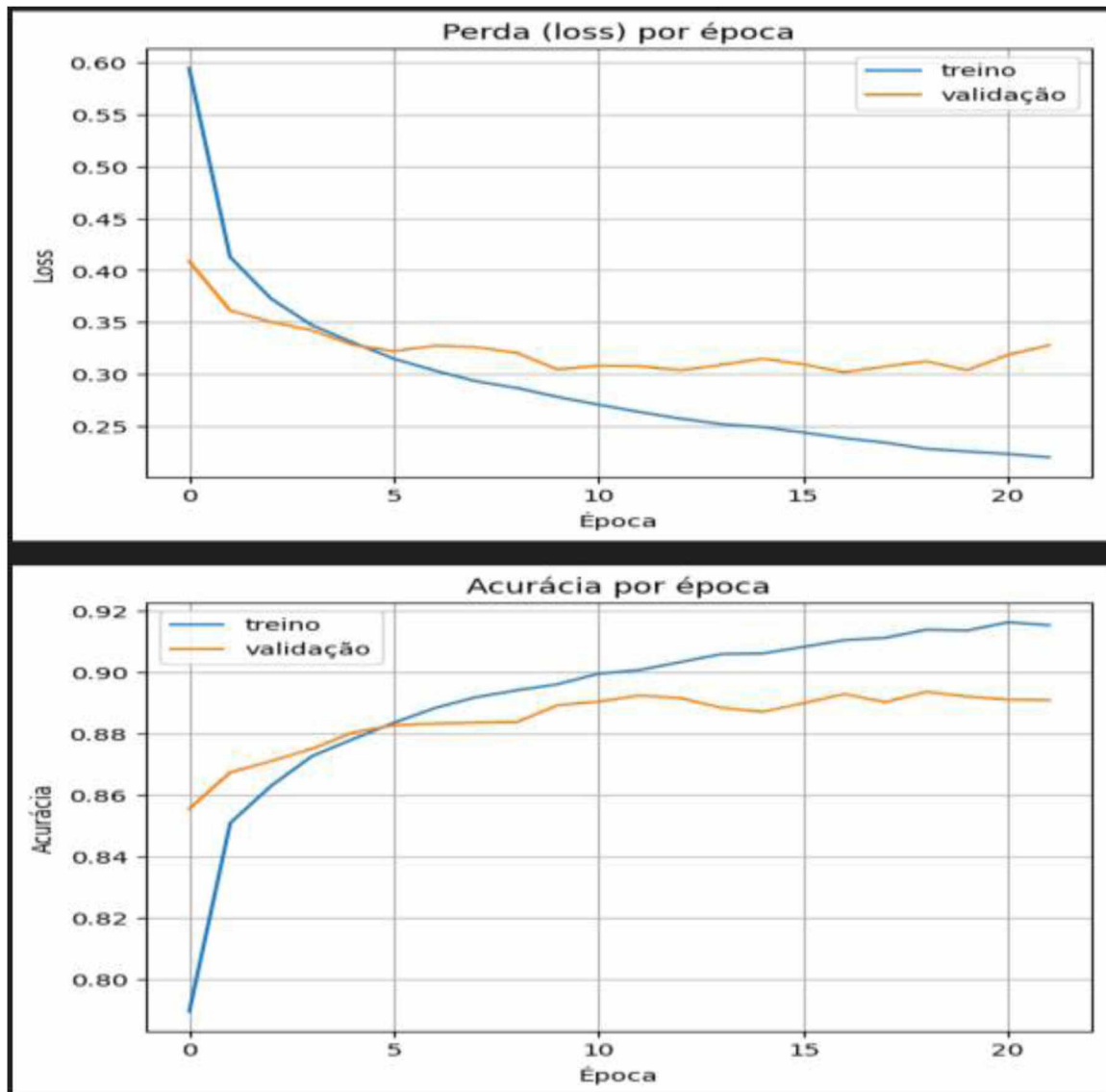
def plot_history(hist):
 # Perda
 plt.figure()
 plt.plot(hist.history['loss'], label='treino')
 plt.plot(hist.history['val_loss'], label='validação')
 plt.title('Perda (loss) por época')
 plt.xlabel('Época')
 plt.ylabel('Loss')
 plt.legend()
 plt.grid(True)
 plt.show()

 # Acurácia
 plt.figure()
 plt.plot(hist.history['accuracy'], label='treino')
 plt.plot(hist.history['val_accuracy'], label='validação')
 plt.title('Acurácia por época')
 plt.xlabel('Época')
 plt.ylabel('Acurácia')
 plt.legend()
 plt.grid(True)
 plt.show()

```

```
plot_history(history)
```

GRÁFICO 14 – FUNÇÃO DE PERDA, TREINO E ACURÁCIA RNA



FONTE: Amauri (2025).

```
probs = model.predict(x_test_flat, verbose=0)
y_pred = np.argmax(probs, axis=1)
errors_idx = np.where(y_pred != y_test)[0]
print(f"Total de erros no teste: {len(errors_idx)}")

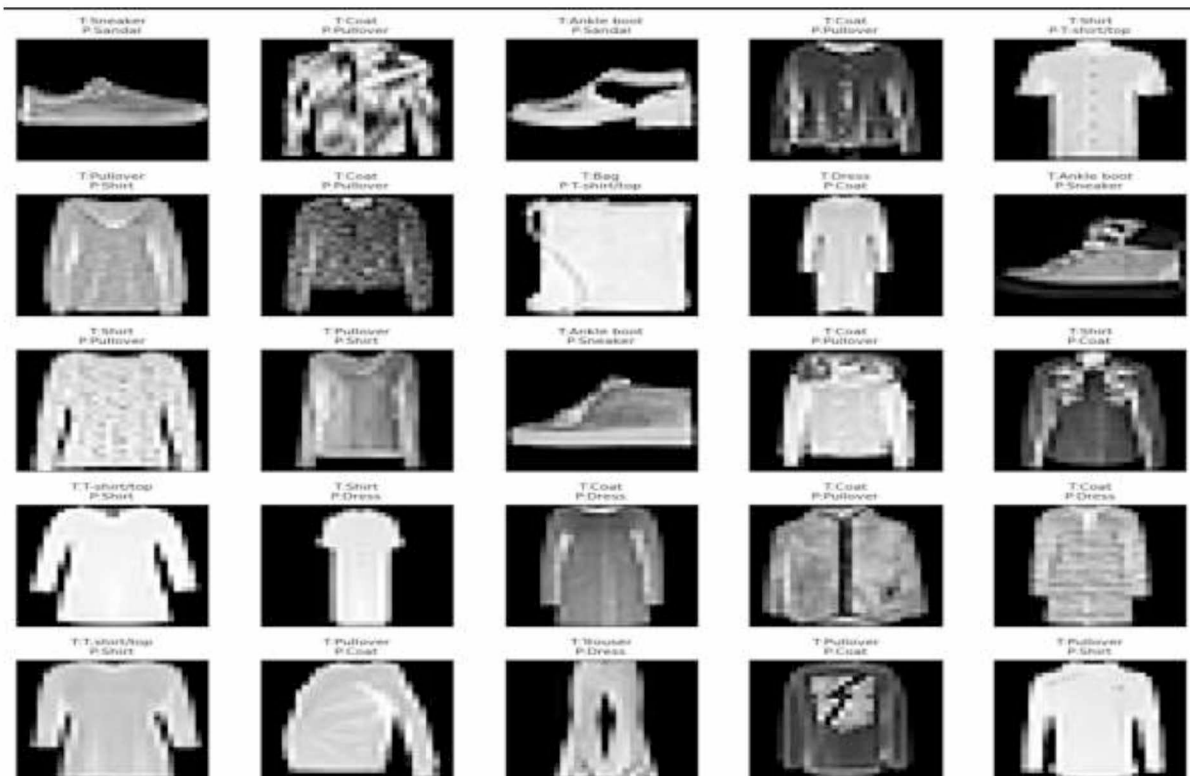
n_show = min(25, len(errors_idx))
idx_to_show = errors_idx[:n_show]

plt.figure(figsize=(10,10))
for i, idx in enumerate(idx_to_show):
 plt.subplot(5,5,i+1)
```

```
plt.imshow(x_test[idx], cmap='gray')
plt.xticks([]); plt.yticks([])
true_lbl = class_names[y_test[idx]]
pred_lbl = class_names[y_pred[idx]]
plt.title(f"T:{true_lbl}\nP:{pred_lbl}", fontsize=8)
plt.tight_layout()
plt.show()
```

Total de erros no teste: 1099

FIGURA 15 – IMAGEM CLASSIFICAÇÃO EXEMPLOS



FONTE: Amauri (2025).

```
try:
 from sklearn.metrics import confusion_matrix, classification_report
 import itertools

 cm = confusion_matrix(y_test, y_pred)
 print(classification_report(y_test, y_pred,
 target_names=class_names))

 plt.figure(figsize=(8,8))
 plt.imshow(cm, interpolation='nearest')
 plt.title('Matriz de Confusão')
 plt.colorbar()
 tick_marks = np.arange(len(class_names))
```

```

plt.xticks(tick_marks, class_names, rotation=45, ha='right')
plt.yticks(tick_marks, class_names)

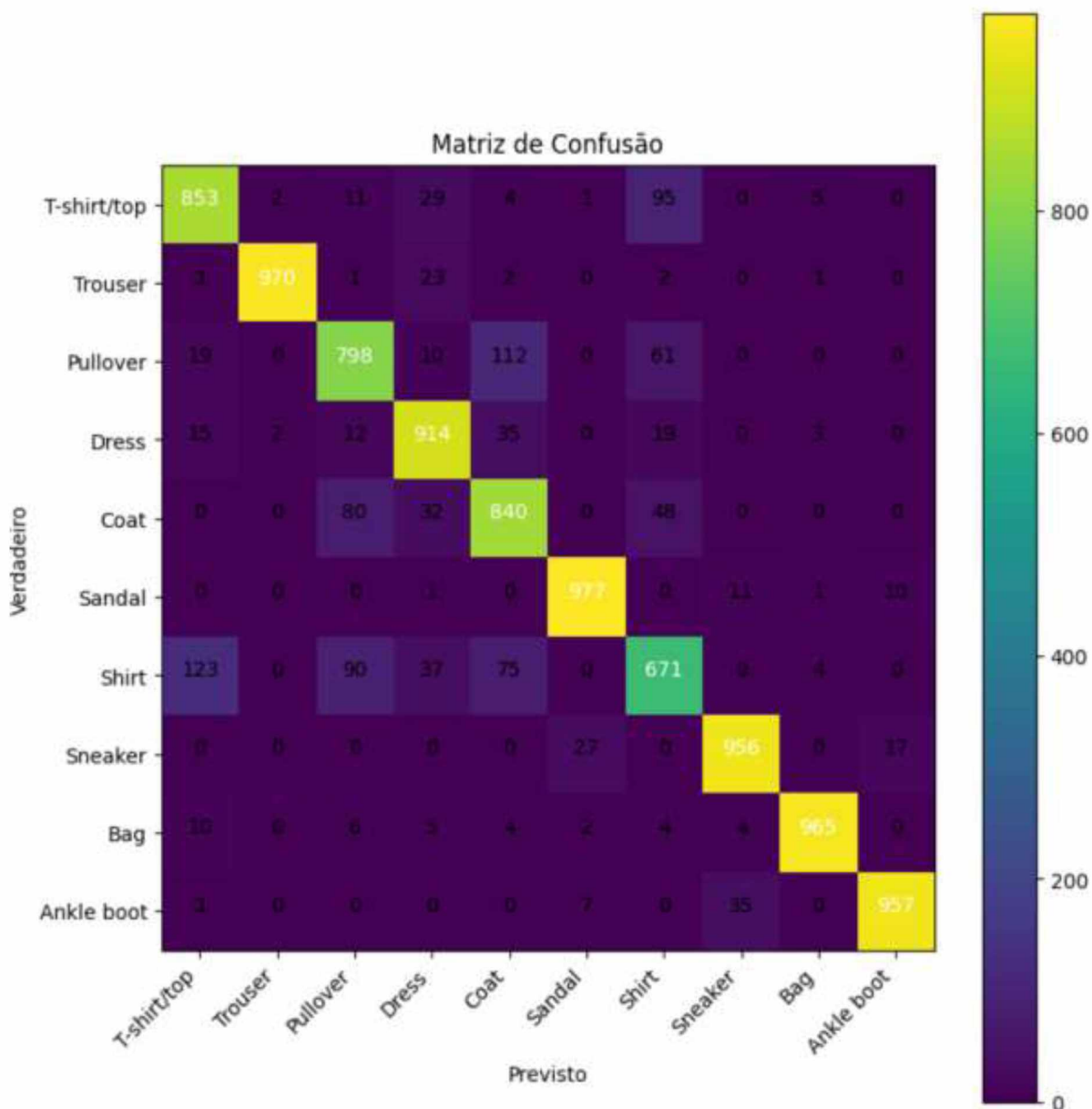
fmt = 'd'
thresh = cm.max() / 2.
for i, j in itertools.product(range(cm.shape[0]),
range(cm.shape[1])):
 plt.text(j, i, format(cm[i, j], fmt),
 horizontalalignment="center",
 color="white" if cm[i, j] > thresh else "black")

plt.ylabel('Verdadeiro')
plt.xlabel('Previsto')
plt.tight_layout()
plt.show()
except Exception as e:
 print('Scikit-learn ausente ou outro erro ao gerar métricas
opcionais:', e)

```

	precision	recall	f1-score	support
T-shirt/top	0.83	0.85	0.84	1000
Trouser	1.00	0.97	0.98	1000
Pullover	0.80	0.80	0.80	1000
Dress	0.87	0.91	0.89	1000
Coat	0.78	0.84	0.81	1000
Sandal	0.96	0.98	0.97	1000
Shirt	0.75	0.67	0.71	1000
Sneaker	0.95	0.96	0.95	1000
Bag	0.99	0.96	0.98	1000
Ankle boot	0.97	0.96	0.96	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

GRÁFICO 15 – MATRIZ DE CONFUSÃO RNA



FONTE: Amauri (2025).

## Breve explicação dos resultados

### Gráficos de perda (loss) e acurácia:

A loss de treino deve cair ao longo das épocas; a de validação também cai até certo ponto. Se a val\_loss voltar a subir enquanto a loss de treino ainda cai, é sinal de overfitting.

A acurácia (treino/validação) tende a subir com as épocas. Idealmente, as curvas de validação seguem as de treino de perto; grande divergência indica overfitting.

Com a arquitetura densa acima, é comum obter  $\sim 0.86\text{--}0.90$  de acurácia no teste (pode variar). Redes convolucionais (CNNs) geralmente performam melhor nesse dataset, mas aqui mantivemos o foco na RNA densa por alinhamento com a aula.

### **Classificações erradas:**

Erros frequentes ocorrem entre classes visualmente parecidas: ex.: Shirt vs T-shirt/top, Sandal vs Sneaker, Coat vs Pullover.

As imagens são pequenas ( $28\times 28$ ) e em tons de cinza; detalhes sutis se perdem. A RNA densa não explora estrutura espacial como uma CNN faria.

Para reduzir erros, tente: mais neurônios/camadas com regularização (Dropout/L2), data augmentation (quando aplicável em imagens 2D), ou migrar para CNN.

### **Métricas extras (opcional para relatório):**

Matriz de confusão por classe

Precision/Recall/F1 por classe

Curva de aprendizado alterando batch size e learning rate.

## 2 Regressão (RNA)

```

!pip install tensorflow pandas numpy scikit-learn matplotlib
Descomente se necessário
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score

SEED = 42
np.random.seed(SEED)
tf.keras.utils.set_random_seed(SEED)
tf.config.experimental.enable_op_determinism()
print(tf.__version__)

url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-
quality/winequality-red.csv"
data = pd.read_csv(url, delimiter=';')
data.columns = [
 'acidez_fixa',
 'acidez_volatil',
 'acido_citrico',
 'acucar_residual',
 'cloretos',
 'dioxido_de_enxofre_livre',
 'dioxido_de_enxofre_total',
 'densidade',
 'pH',
 'sulfatos',
 'alcool',
 'score_qualidade_vinho'
]
data.head()

X = data.iloc[:, :-1].values # todas menos a última
y = data.iloc[:, -1].values # última coluna
X.shape, y.shape

((1599, 11), (1599,))

X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.2, random_state=42
)

```

```

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
X_train_scaled.shape, X_test_scaled.shape

((1279, 11), (320, 11))

n_features = X_train_scaled.shape[1]
model = keras.Sequential([
 layers.Input(shape=(n_features,)),
 layers.Dense(64, activation='relu'),
 layers.Dense(32, activation='relu'),
 layers.Dense(1, activation='linear')
])

model.compile(
 loss='mse',
 optimizer=keras.optimizers.Adam(learning_rate=1e-3),
 metrics=['mae']
)
model.summary()

callbacks = [
 keras.callbacks.EarlyStopping(patience=20,
 restore_best_weights=True, monitor='val_loss')
]
history = model.fit(
 X_train_scaled, y_train,
 validation_split=0.2,
 epochs=500,
 batch_size=32,
 callbacks=callbacks,
 verbose=1
)

plt.figure()
plt.plot(history.history['loss'], label='treino')
plt.plot(history.history['val_loss'], label='validação')
plt.title('MSE (loss) por época')
plt.xlabel('Época')
plt.ylabel('MSE')
plt.legend()
plt.grid(True)
plt.show()

y_pred = model.predict(X_test_scaled, verbose=0).ravel()
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f"MAE: {mae:.4f}\nMSE: {mse:.4f}\nR²: {r2:.4f}")

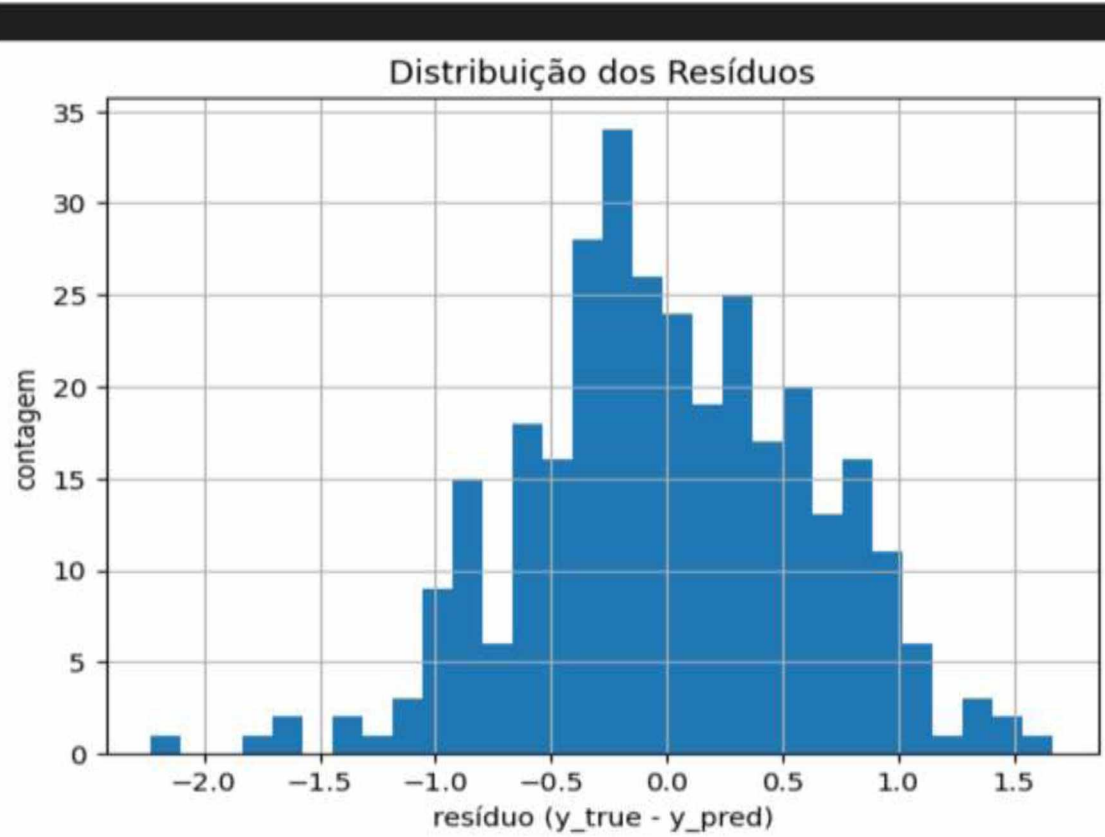
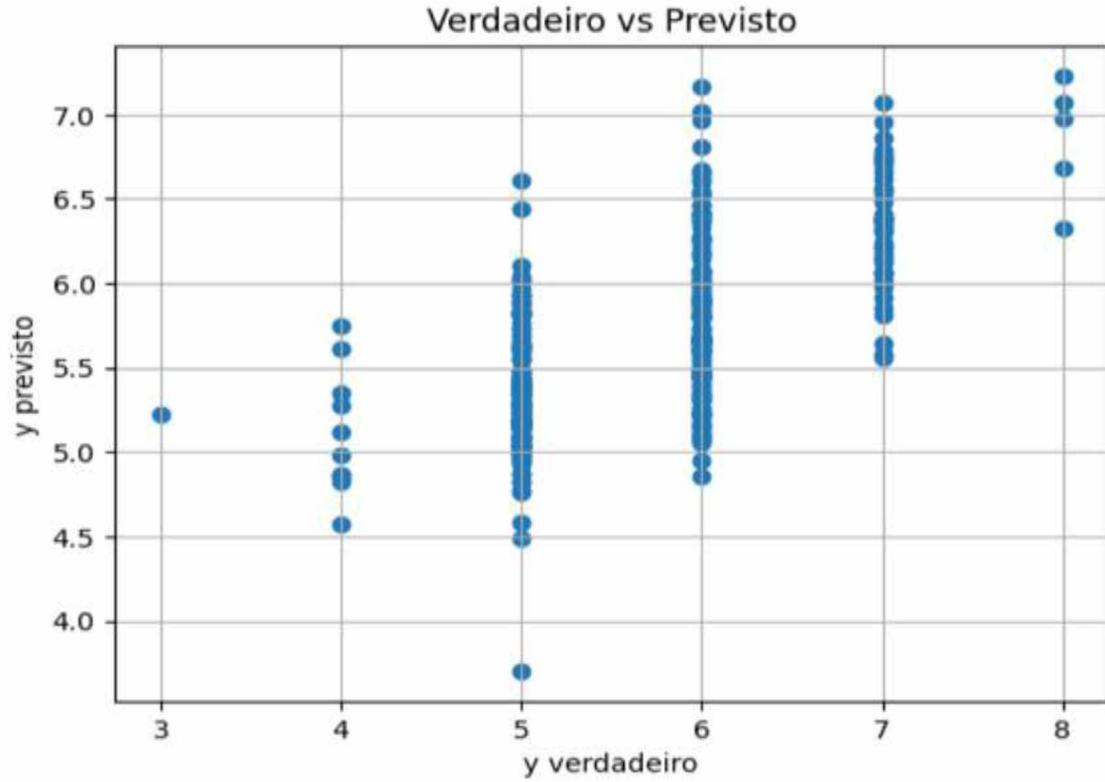
```

```
MAE: 0.4875
MSE: 0.3729
R2: 0.4294

Dispersão y_verdadeiro vs y_previsto
plt.figure()
plt.scatter(y_test, y_pred)
plt.title('Verdadeiro vs Previsto')
plt.xlabel('y verdadeiro')
plt.ylabel('y previsto')
plt.grid(True)
plt.show()

Resíduos
residuos = y_test - y_pred
plt.figure()
plt.hist(residuos, bins=30)
plt.title('Distribuição dos Resíduos')
plt.xlabel('resíduo (y_true - y_pred)')
plt.ylabel('contagem')
plt.grid(True)
plt.show()
```

GRÁFICO 16 – VERDADEIRO VS PREVISTO E DISTRIBUIÇÃO DE RESÍDUOS



FONTE: Amauri (2025).

Gráfico de loss (MSE):

- Espera-se que MSE de treino e validação diminuam ao longo das épocas. Se `val_loss` parar de cair ou começar a subir enquanto `loss` segue caindo, é `overfitting`. O `EarlyStopping` interrompe o treino antes de piorar.

Métricas (MAE, MSE,  $R^2$ ):

- MAE: erro médio absoluto – fácil de interpretar (em pontos de qualidade). Quanto menor, melhor.
- MSE: penaliza erros grandes de forma quadrática. Também quanto menor, melhor.
- $R^2$ : proporção da variância explicada (varia de  $-\infty$  a 1). Valores mais próximos de 1 indicam melhor ajuste.

Em geral, modelos lineares fortes (ex.: Ridge/Lasso) e árvores impulsionadas (Gradient Boosting/Random Forest) competem bem nesse dataset. A RNA densa captura não linearidades; o desempenho depende de hiperparâmetros (camadas, neurônios, learning rate, regularização) e do pré-processamento (padronização é essencial aqui).

### 3 Sistemas de Recomendação

```
!pip install pandas numpy matplotlib scikit-learn tensorflow
Descomente se precisar
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers

SEED = 42
np.random.seed(SEED)
tf.keras.utils.set_random_seed(SEED)
tf.config.experimental.enable_op_determinism()
```

```

print(tf.__version__)

csv_path = "/content/sample_data/Base_livros(in)2.csv" # caminho do
arquivo enviado
Try reading with a different separator, like semicolon, as comma might
be in data fields
try:
 df = pd.read_csv(csv_path, sep=';')
except Exception as e:
 print(f"Error reading CSV with semicolon separator: {e}")
 # If semicolon fails, try default comma and investigate specific
 error line if possible
 try:
 df = pd.read_csv(csv_path)
 print("Successfully read with comma separator, but there might
be tokenization issues.")
 except Exception as e:
 print(f"Error reading CSV with comma separator: {e}")
 print("Could not read the CSV file with either comma or
semicolon separators.")
 print("Please check the file content, especially around line
9928, for issues.")
 df = pd.DataFrame() # Create empty dataframe to avoid further
errors

if not df.empty:
 print(df.shape)
 display(df.head())

(128838, 13)
/tmp/ipython-input-3598655109.py:4: DtypeWarning: Columns (10) have
mixed types. Specify dtype option on import or set low_memory=False.
 df = pd.read_csv(csv_path, sep=';')

expected_cols = ['ID_usuario', 'ISBN', 'Titulo', 'Nota']
if list(df.columns) != expected_cols:
 # Tenta ajustar nomes comuns
 rename_map = {}
 for col in df.columns:
 lc = col.strip().lower()
 if 'usuario' in lc or 'user' in lc:
 rename_map[col] = 'ID_usuario'
 elif 'isbn' in lc:
 rename_map[col] = 'ISBN'
 elif 'titulo' in lc or 'title' in lc:
 rename_map[col] = 'Titulo'
 elif 'nota' in lc or 'rating' in lc or 'score' in lc:
 rename_map[col] = 'Nota'
 df = df.rename(columns=rename_map)
 missing = set(expected_cols) - set(df.columns)

```

```

 if missing:
 raise ValueError(f"Colunas faltando após tentativa de renomear:
{missing}")

Garante tipos
df['ID_usuario'] = df['ID_usuario'].astype(str)
df['ISBN'] = df['ISBN'].astype(str)
df['Titulo'] = df['Titulo'].astype(str)
df['Nota'] = pd.to_numeric(df['Nota'], errors='coerce')
df = df.dropna(subset=['Nota'])
df.head()

user_enc = LabelEncoder()
item_enc = LabelEncoder()

df['user_id'] = user_enc.fit_transform(df['ID_usuario'])
df['item_id'] = item_enc.fit_transform(df['ISBN'])

n_users = df['user_id'].nunique()
n_items = df['item_id'].nunique()
n_users, n_items

(11111, 108556)

Do encoded item_id para ISBN e Titulo
id_to_isbn = {i: isbn for i, isbn in enumerate(item_enc.classes_)}
isbn_to_title = df.groupby('ISBN')['Titulo'].agg(lambda x:
x.mode().iat[0] if not x.mode().empty else x.iloc[0]).to_dict()

def item_title_from_id(item_id):
 isbn = id_to_isbn.get(item_id, None)
 return isbn_to_title.get(isbn, f"ISBN {isbn}") if isbn is not None
else str(item_id)

item_title_from_id(0)

"Monk's-hood"

X = df[['user_id', 'item_id']].values
y = df['Nota'].values.astype('float32')

X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.2, random_state=SEED
)
X_train.shape, X_test.shape

((86845, 2), (21712, 2))

k = 32 # dimensão dos embeddings

```

```

user_in = layers.Input(shape=(1,), name='user')
item_in = layers.Input(shape=(1,), name='item')

user_emb = layers.Embedding(input_dim=n_users, output_dim=k,
name='user_emb')(user_in)
item_emb = layers.Embedding(input_dim=n_items, output_dim=k,
name='item_emb')(item_in)

user_vec = layers.Flatten()(user_emb)
item_vec = layers.Flatten()(item_emb)

x = layers.Concatenate()([user_vec, item_vec])
x = layers.Dense(64, activation='relu')(x)
x = layers.Dense(32, activation='relu')(x)
out = layers.Dense(1, activation='linear')(x)

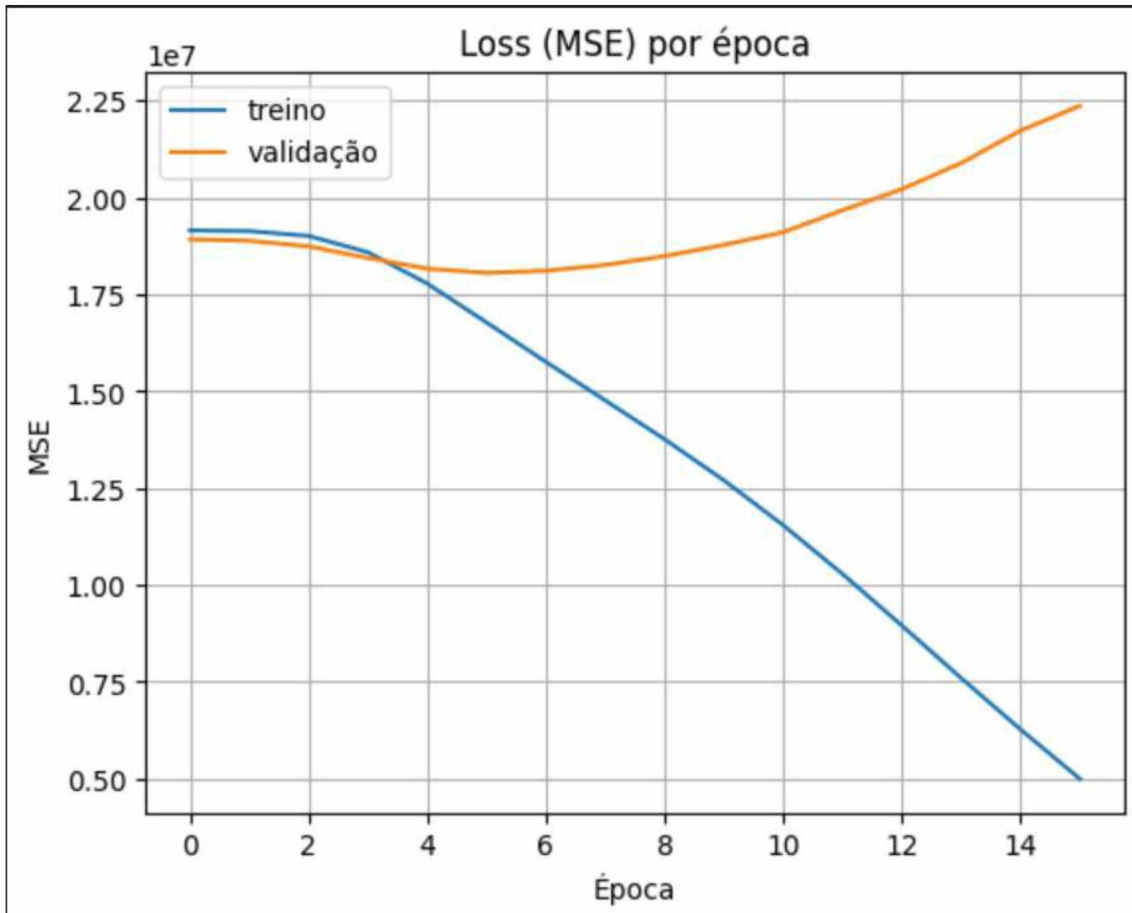
model = keras.Model(inputs=[user_in, item_in], outputs=out)
model.compile(optimizer=keras.optimizers.Adam(learning_rate=1e-3),
loss='mse')
model.summary()

callbacks=[keras.callbacks.EarlyStopping(patience=10,
restore_best_weights=True, monitor='val_loss')]
history = model.fit(
 [X_train[:,0], X_train[:,1]], y_train,
 validation_split=0.2,
 epochs=100,
 batch_size=256,
 callbacks=callbacks,
 verbose=1
)

Plot loss
plt.figure()
plt.plot(history.history['loss'], label='treino')
plt.plot(history.history['val_loss'], label='validação')
plt.title('Loss (MSE) por época')
plt.xlabel('Época')
plt.ylabel('MSE')
plt.legend(); plt.grid(True)
plt.show()

```

GRÁFICO 17 – FUNÇÃO DE PERDA POR ÉPOCA



FONTE: Amauri (2025).

Gráfico de Loss (MSE):

- Espera-se que o loss de treino e de validação diminuam com as épocas.
- Se o val\_loss começar a subir enquanto o loss de treino cai, indica overfitting; o EarlyStopping ajuda a interromper no ponto ótimo.

Recomendações:

- Para um usuário, o modelo gera uma pontuação prevista (ScorePrevisto) para cada livro não avaliado.
- Os Top-N livros com maior pontuação prevista são sugeridos como recomendações.

- Dependendo da escala de Nota (ex.: 0–10 ou 1–5), pode ser útil clpear/normalizar a saída do modelo.

Melhorias possíveis:

- Adicionar viés de usuário e item (camadas Embedding separadas para biases ou camadas Embedding e Dot).
- Balancear dados muito esparsos com amostragem negativa (para feedback implícito) ou ponderar perdas.
- Incluir metadados (gênero, autor, ano) e/ou usar arquiteturas neurais mais profundas (NCF, NeuMF).

## 4 Deepdream

```
!pip install tensorflow pillow numpy matplotlib requests # Descomente
se necessário
import io, os, math, requests
import numpy as np
import PIL.Image as Image
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from IPython.display import display, HTML

SEED = 42
tf.keras.utils.set_random_seed(SEED)
np.random.seed(SEED)
print(tf.__version__)

Use a local file path instead of downloading
img_path = "/content/sample_data/Felis_catus-cat_on_snow.jpg"

try:
 orig = Image.open(img_path).convert('RGB')
 orig = orig.resize((800, int(800 * orig.height / orig.width))) #
 redimensiona para ~800px na largura
 display(orig)

except FileNotFoundError:
 print(f"Error: The file {img_path} was not found.")
except Exception as e:
 print(f"An error occurred while opening the image: {e}")
```

FIGURA 16 – IMAGEM DE EXEMPLO PARA USO NO DEEPPDREAM



FONTE: Amauri (2025).

```
def pil_to_tensor(img):
 arr = np.array(img)
 return tf.convert_to_tensor(arr[None, ...], dtype=tf.float32)

def tensor_to_pil(t):
 arr = tf.clip_by_value(t[0], 0., 255.).numpy().astype(np.uint8)
 return Image.fromarray(arr)

def deprocess(img_tensor):
 return tensor_to_pil(img_tensor)

def random_jitter(img, max_jitter=8):
 ox, oy = np.random.randint(-max_jitter, max_jitter+1, size=2)
 return tf.roll(tf.roll(img, shift=ox, axis=1), shift=oy, axis=2)

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

Selecione algumas camadas em diferentes profundidades
layer_names = [
 'mixed3', # texturas mais simples
 'mixed5', # padrões intermediários
 'mixed7' # padrões mais complexos
]
layers_out = [base_model.get_layer(name).output for name in layer_names]
```

```

dream_model = keras.Model(inputs=base_model.input, outputs=layers_out)

for layer in base_model.layers:
 layer.trainable = False

def calc_loss(img_tensor):
 # InceptionV3 espera entradas pré-processadas de uma forma
 específica
 x = tf.keras.applications.inception_v3.preprocess_input(img_tensor)
 activations = dream_model(x)
 if not isinstance(activations, (list, tuple)):
 activations = [activations]
 losses = [tf.reduce_mean(act) for act in activations]
 return tf.add_n(losses)

@tf.function
def gradient_ascent_step(img, step_size):
 with tf.GradientTape() as tape:
 tape.watch(img)
 loss = calc_loss(img)
 grads = tape.gradient(loss, img)
 # normaliza gradientes para passo estável
 grads /= (tf.math.reduce_std(grads) + 1e-8)
 img = img + step_size * grads
 return img, loss

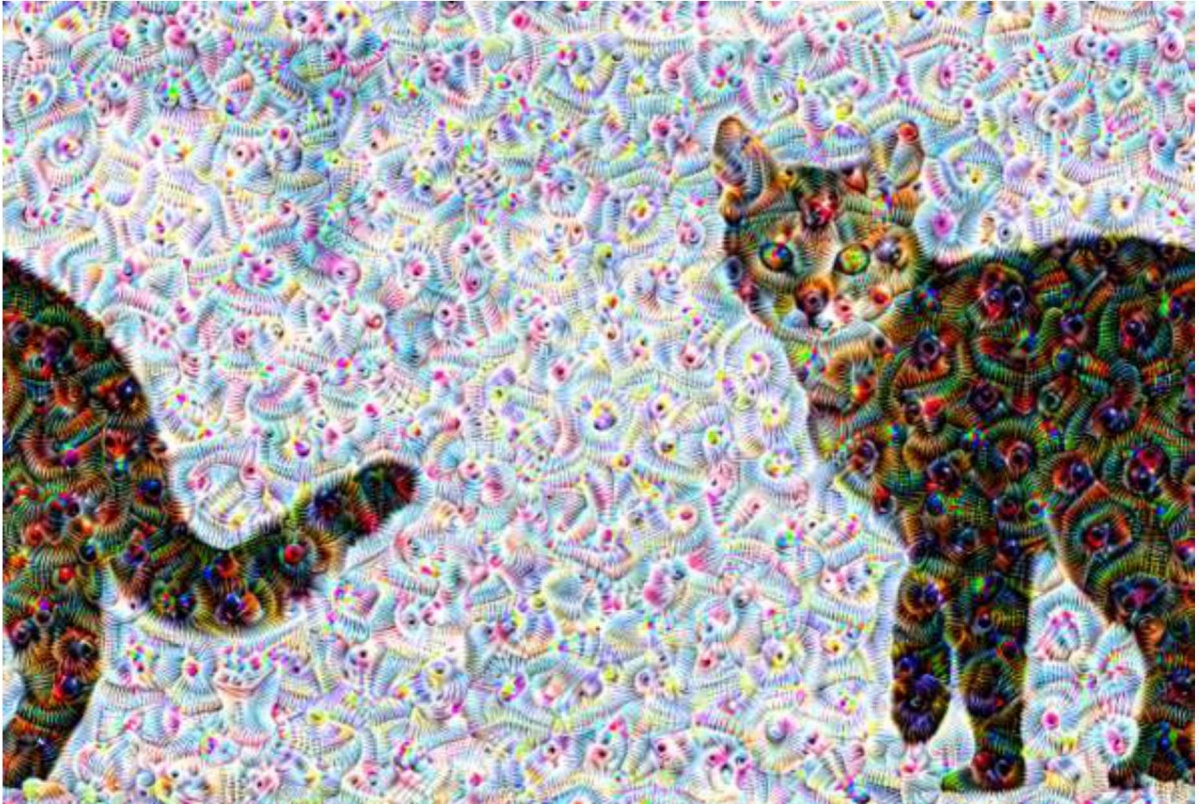
def deepdream_main_loop(img_pil, steps=60, step_size=10.0, jitter=8):
 img = pil_to_tensor(img_pil)
 for i in range(steps):
 if jitter:
 img = random_jitter(img, max_jitter=jitter)
 img, loss = gradient_ascent_step(img, tf.constant(step_size))
 if jitter:
 img = tf.roll(tf.roll(img, shift=-jitter, axis=1), shift=-
jitter, axis=2)
 if (i+1) % 20 == 0:
 print(f"Passo {i+1}/{steps} | Loss: {loss.numpy():.2f}")
 return img

dream_main = deepdream_main_loop(orig, steps=60, step_size=8.0,
jitter=8)
img_main = deprocess(dream_main)
display(img_main)

Passo 20/60 | Loss: 1.49
Passo 40/60 | Loss: 2.09
Passo 60/60 | Loss: 2.28

```

FIGURA 17 – IMAGEM DE EXEMPLO COM MAIN LOOP



FONTE: Amauri (2025).

```
def resize_img(img_pil, size):
 return img_pil.resize(size, Image.LANCZOS)

def deepdream_octaves(img_pil, steps_per_octave=40, step_size=6.0,
jitter=6, octaves=3, scale=1.4):
 # Cria lista de tamanhos (da menor para a maior)
 w, h = img_pil.size
 octave_sizes = []
 for i in range(octaves):
 ow = int(w / (scale ** (octaves - 1 - i)))
 oh = int(h / (scale ** (octaves - 1 - i)))
 octave_sizes.append((max(64, ow), max(64, oh)))

 base = img_pil
 for idx, sz in enumerate(octave_sizes):
 base = resize_img(base, sz)
 print(f"Octave {idx+1}/{len(octave_sizes)} | tamanho:
{base.size}")
 dreamed = deepdream_main_loop(base, steps=steps_per_octave,
step_size=step_size, jitter=jitter)
 base = deprocess(dreamed)
 return base
```

```
img_oct = deepdream_octaves(orig, steps_per_octave=40, step_size=6.0,
jitter=6, octaves=3, scale=1.4)
display(img_oct)
```

```
Octave 1/3 | tamanho: (408, 271)
Passo 20/40 | Loss: 1.51
Passo 40/40 | Loss: 1.86
Octave 2/3 | tamanho: (571, 380)
Passo 20/40 | Loss: 1.74
Passo 40/40 | Loss: 2.25
Octave 3/3 | tamanho: (800, 533)
Passo 20/40 | Loss: 1.79
Passo 40/40 | Loss: 2.25
```

FIGURA 18 – IMAGEM DE EXEMPLO COM OCTAVES

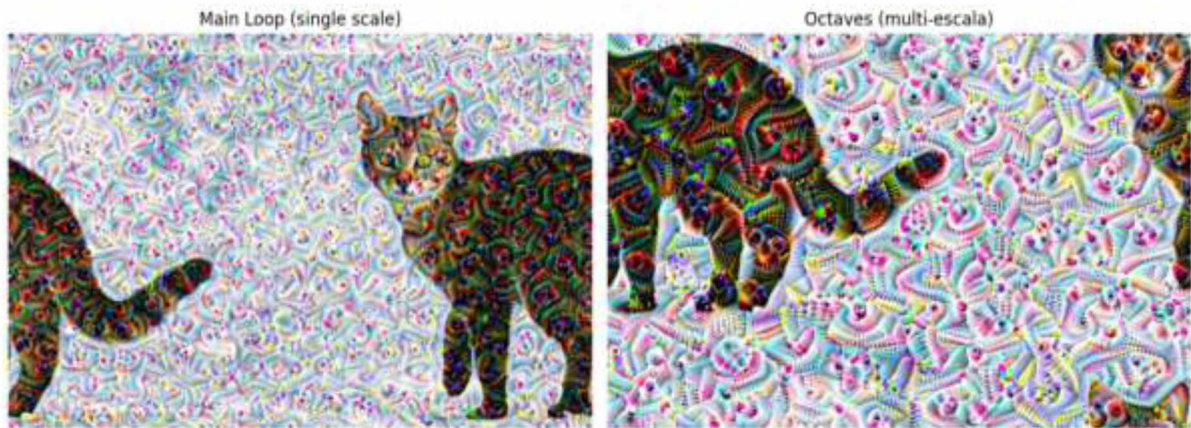


FONTE: Amauri (2025).

```
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.imshow(img_main); plt.title('Main Loop (single scale)')
plt.axis('off')

plt.subplot(1,2,2)
plt.imshow(img_oct); plt.title('Octaves (multi-escala)')
plt.axis('off')
plt.tight_layout()
plt.show()
```

FIGURA 19 – COMPARAÇÃO IMAGENS DE MAIN LOOP E OCTAVES



FONTE: Amauri (2025).

Imagem onírica por Main Loop (uma única escala):

- O gradiente de ativações amplifica padrões que a rede (InceptionV3) reconhece.
- Os detalhes tendem a surgir de forma mais local e às vezes com artefatos de alta frequência.

Imagem onírica por Octaves (multi-escala):

- Ao processar da escala menor para a maior, os padrões globais são inseridos em baixa resolução e depois refinados.
- O resultado costuma ter texturas mais coerentes e com detalhes multiescala (menos ruído fino).

Diferenças principais:

- Main Loop: pode enfatizar detalhes finos rapidamente, mas sem estrutura global consistente.
- Octaves: promove padrões coerentes em várias escalas, resultando em imagens oníricas visualmente mais estáveis.
- Dicas: Ajuste `layer_names`, `steps`, `step_size`, `octaves` e `scale` para controlar intensidade/estilo. Camadas mais profundas → padrões mais complexos; camadas rasas → texturas simples.

## APÊNDICE M - 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

## TRABALHO DE IAA007 – Visualização de Dados e Storytelling

### 1. Conjunto de Dados Brutos

O conjunto de dados utilizado contém informações sobre preferências de investimento, fatores considerados ao investir, fontes de informação e expectativas de retorno de um grupo de 40 investidores. O dataset inclui variáveis como:

- Demográficas: Gênero e idade.
- Comportamentais: Frequência de monitoramento de investimentos e expectativa de retorno.
- Preferências de Investimento: Tipos de ativos escolhidos e ordenados por prioridade.
- Fontes de Informação: Como os investidores obtêm conhecimento sobre o mercado.
- Durante a análise, o conjunto de dados foi limpo e melhorado:

Padronização de categorias para evitar inconsistências (remoção de espaços extras em valores e colunas). Criação da variável Most Preferred Investment, identificando o ativo mais preferido por cada investidor. Agrupamento de Idades em Faixas Etárias para facilitar a análise do comportamento de diferentes grupos.

### 2. Contexto

O objetivo desta análise é compreender o comportamento e as preferências dos investidores. Como:

- Quais são os ativos mais preferidos?
- Como gênero e idade influenciam as decisões financeiras?
- Quais fatores mais impactam as escolhas de investimento?

#### 4. Storytelling da Visualização

A análise conta uma história sobre o perfil do investidor, evidenciando:

Quem são os investidores?

- A maioria dos investidores são homens entre 25 e 40 anos.
- A faixa etária predominante está entre 26-35 anos, indicando um público jovem e economicamente ativo.

Como eles investem?

- Mutual Funds e Equity Market são os investimentos mais populares.
- Investidores mais jovens preferem ações (Equity Market), enquanto os mais velhos optam por investimentos seguros como Government Bonds e Fixed Deposits.

O que influencia suas decisões?

- Retorno sobre o investimento é o fator mais considerado, seguido pelo risco.
- A maioria monitora investimentos mensalmente, sugerindo um perfil moderado de acompanhamento.

Como eles se informam?

- Consultores financeiros e revistas especializadas são as principais fontes.
- A internet ainda não é a principal escolha para aprendizado financeiro, mostrando oportunidades para educação digital.

- Como os investidores se informam sobre o mercado?

A análise pode ser útil para os seguintes públicos alvos:

- Instituições financeiras: Criar produtos alinhados aos interesses dos investidores.
- Consultores financeiros: Auxiliar clientes na escolha de ativos compatíveis com seus perfis.
- Novos investidores: Entender o comportamento de outros investidores e tomar decisões mais informadas.

### 3. Visualizações de Dados

As visualizações foram feitas usando Python, Seaborn e Matplotlib, pois essas ferramentas permitem gráficos interativos e personalizáveis para análise exploratória.

Gráficos Criados:

- Distribuição de Idade por Gênero → Mostra a faixa etária predominante entre investidores.
- Preferências de Investimento → Exibe os ativos mais populares, com base nos rankings individuais.
- Fatores Considerados ao Investir (Pizza Chart) → Identifica os critérios mais importantes para os investidores.
- Expectativa de Retorno por Tipo de Investimento (Boxplot) → Compara retornos esperados com os ativos escolhidos.
- Fontes de Informação para Investimentos → Revela onde os investidores buscam conhecimento sobre o mercado.
- Investimento mais Preferido por Gênero → Compara a escolha de ativos entre homens e mulheres.
- Investimento mais Preferido por Faixa Etária → Mostra padrões de investimento por idade.

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

◇ Carregar os dados
```

```

df = pd.read_csv("/content/sample_data/Original_data.csv")

◇ Renomeando colunas para facilitar a análise (removendo espaços extras)
df = df.rename(columns=lambda x: x.strip())

◇ Padronizando os valores categóricos (removendo espaços extras e deixando em formato uniforme)
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)

◇ Criando uma coluna que indica o investimento mais preferido de cada investidor
investment_cols = [
 "What do you think are the best options for investing your money? (Rank in order of preference) [Mutual Funds]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Equity Market]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Debentures]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Government Bonds]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Fixed Deposits]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Public Provident Fund]",
 "What do you think are the best options for investing your money? (Rank in order of preference) [Gold]"
]

Criar nova coluna com a opção mais preferida (menor ranking)
df["Most Preferred Investment"] =
df[investment_cols].idxmin(axis=1).str.extract(r'\[(.*?)\]')

◇ Criar Faixas Etárias
df['Age Group'] = pd.cut(df['AGE'], bins=[18, 25, 35, 45, 55, 65], labels=["18-25", "26-35", "36-45", "46-55", "56-65"])

◇ Configuração de estilo dos gráficos
sns.set(style="whitegrid")

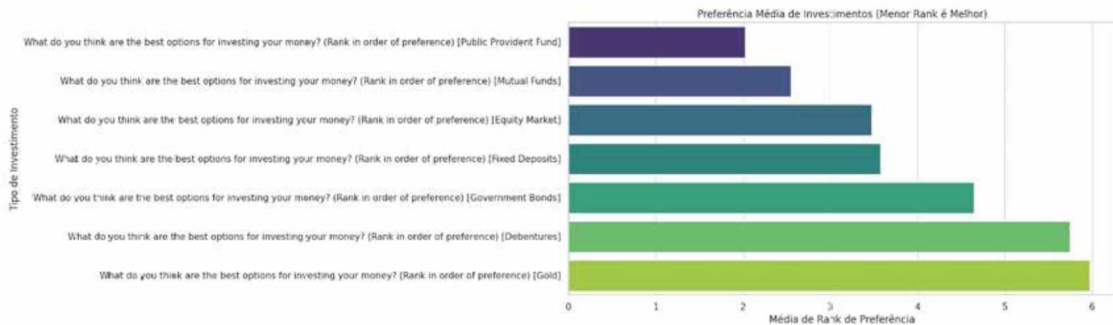
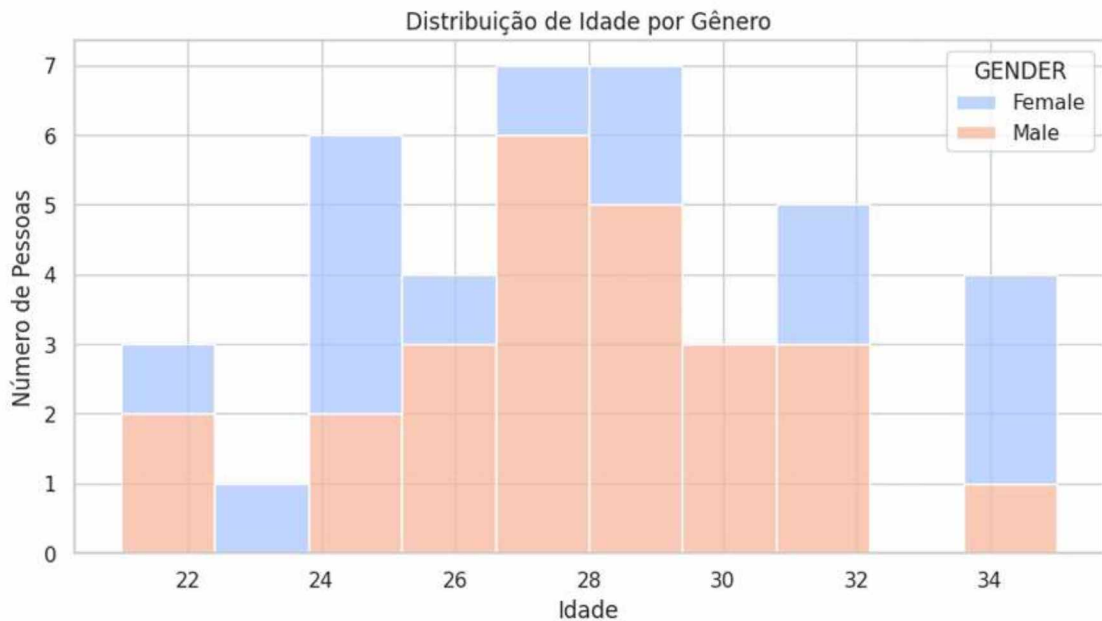
1 Distribuição de Idade por Gênero
plt.figure(figsize=(10,5))
sns.histplot(data=df, x='AGE', hue='GENDER', multiple='stack', bins=10, palette='coolwarm')
plt.title("Distribuição de Idade por Gênero")
plt.xlabel("Idade")
plt.ylabel("Número de Pessoas")
plt.show()

2 Preferências de Investimento (Rank médio por tipo de investimento)

```

```
investment_means = df[investment_cols].mean().sort_values()
plt.figure(figsize=(12,6))
sns.barplot(x=investment_means.values, y=investment_means.index,
palette='viridis')
plt.title("Preferência Média de Investimentos (Menor Rank é Melhor)")
plt.xlabel("Média de Rank de Preferência")
plt.ylabel("Tipo de Investimento")
plt.show()
```

GRÁFICO 18 – DISTRIBUIÇÃO DE IDADE POR GÊNERO

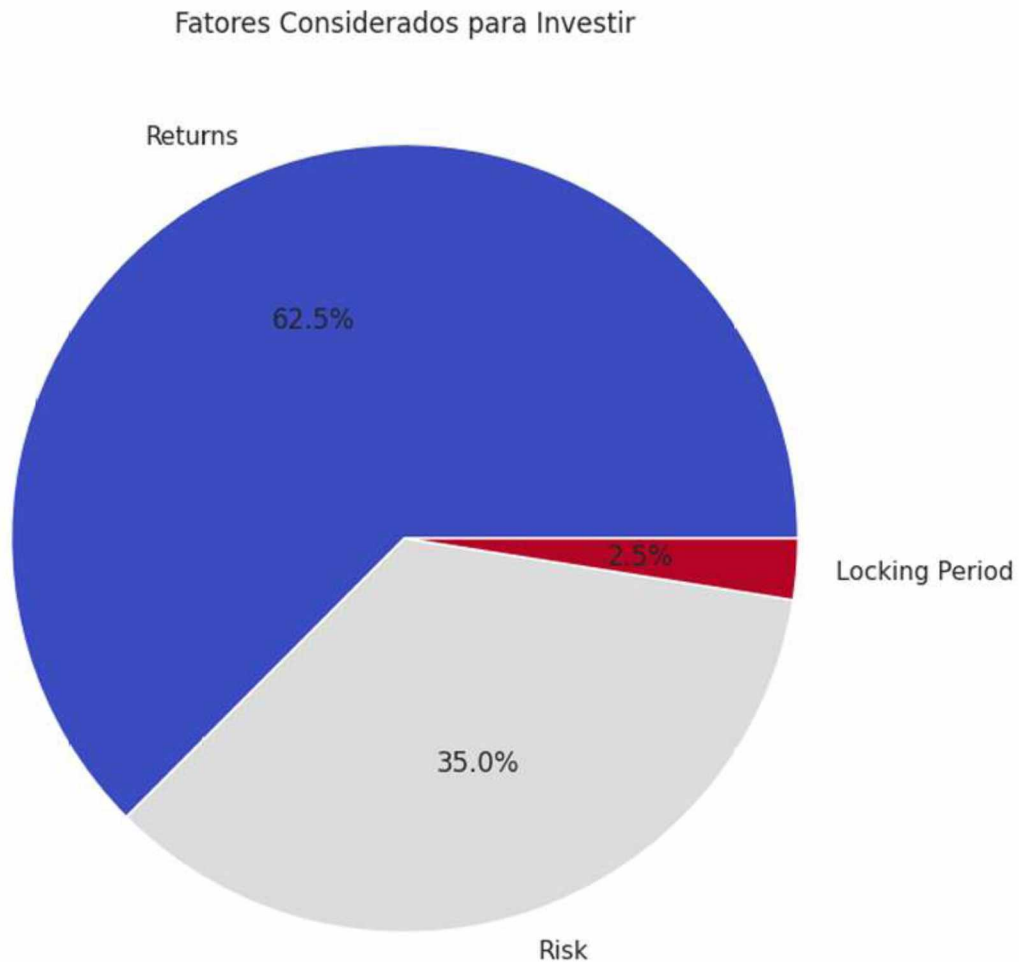


FONTE: Amauri (2025).

```
3 Fatores Considerados para Investimento (Gráfico de Pizza)
plt.figure(figsize=(8,8))
df["What are the factors considered by you while investing in any instrument?"].value_counts().plot(kind='pie', autopct='%1.1f%%',
cmap='coolwarm')
plt.title("Fatores Considerados para Investir")
plt.ylabel("")
```

```
plt.show()
```

GRÁFICO 19 – DISTRIBUIÇÃO DE FATORES PARA INVESTIMENTO

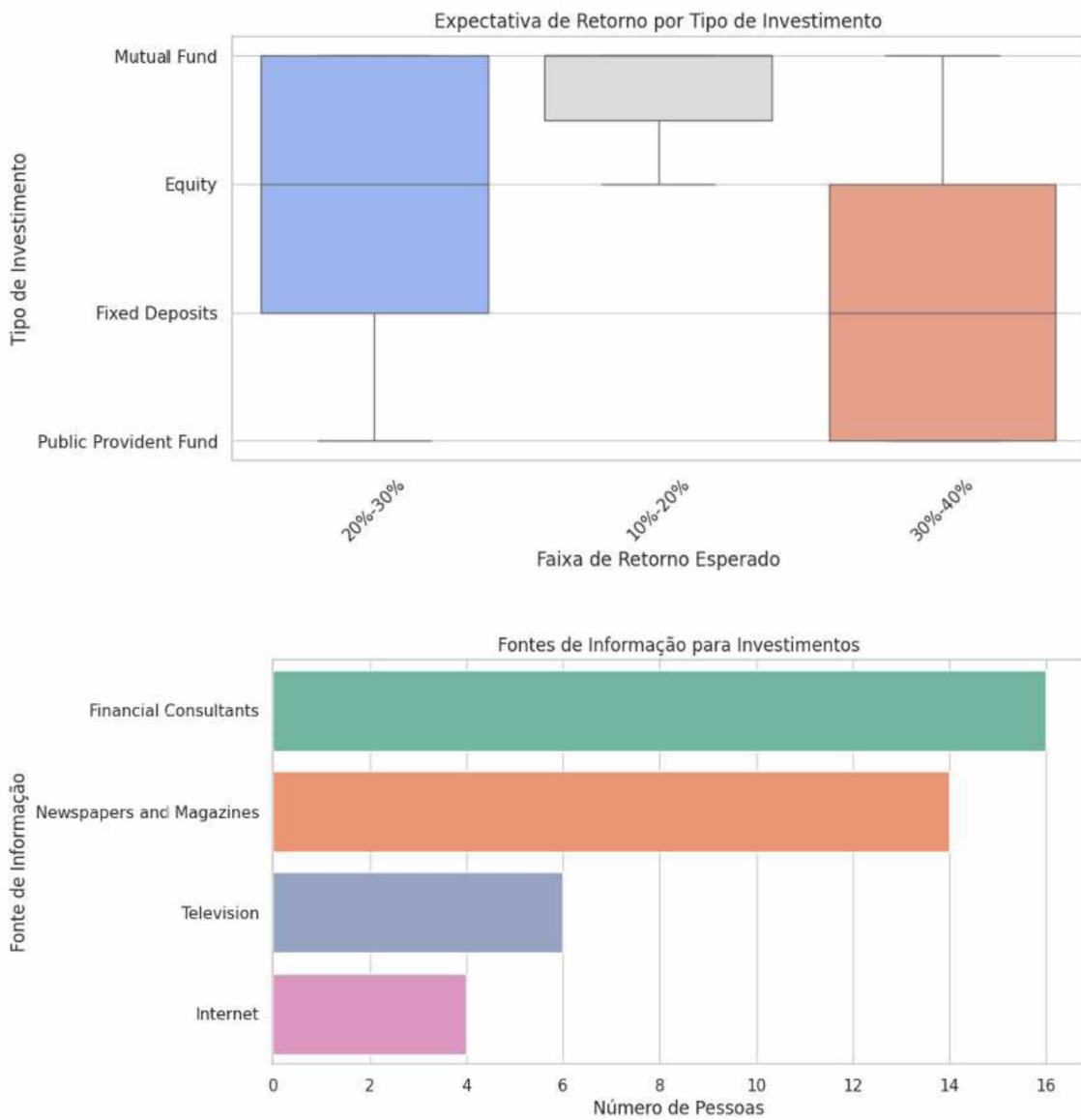


FONTE: Amauri (2025).

```
4 Expectativa de Retorno vs Tipo de Investimento
plt.figure(figsize=(10,5))
sns.boxplot(x="How much return do you expect from any investment
instrument?",
 y="Which investment avenue do you mostly invest in?",
 data=df, palette='coolwarm')
plt.title("Expectativa de Retorno por Tipo de Investimento")
plt.xlabel("Faixa de Retorno Esperado")
plt.ylabel("Tipo de Investimento")
plt.xticks(rotation=45)
plt.show()
```

```
5 Fonte de Informação para Investimentos
plt.figure(figsize=(10,5))
sns.countplot(y="Your sources of information for investments is",
 data=df, palette='Set2',
 order=df["Your sources of information for investments
is"].value_counts().index)
plt.title("Fontes de Informação para Investimentos")
plt.xlabel("Número de Pessoas")
plt.ylabel("Fonte de Informação")
plt.show()
```

GRÁFICO 20 – EXPECTATIVA DE RETORNO POR TIPO DE INVESTIMENTO

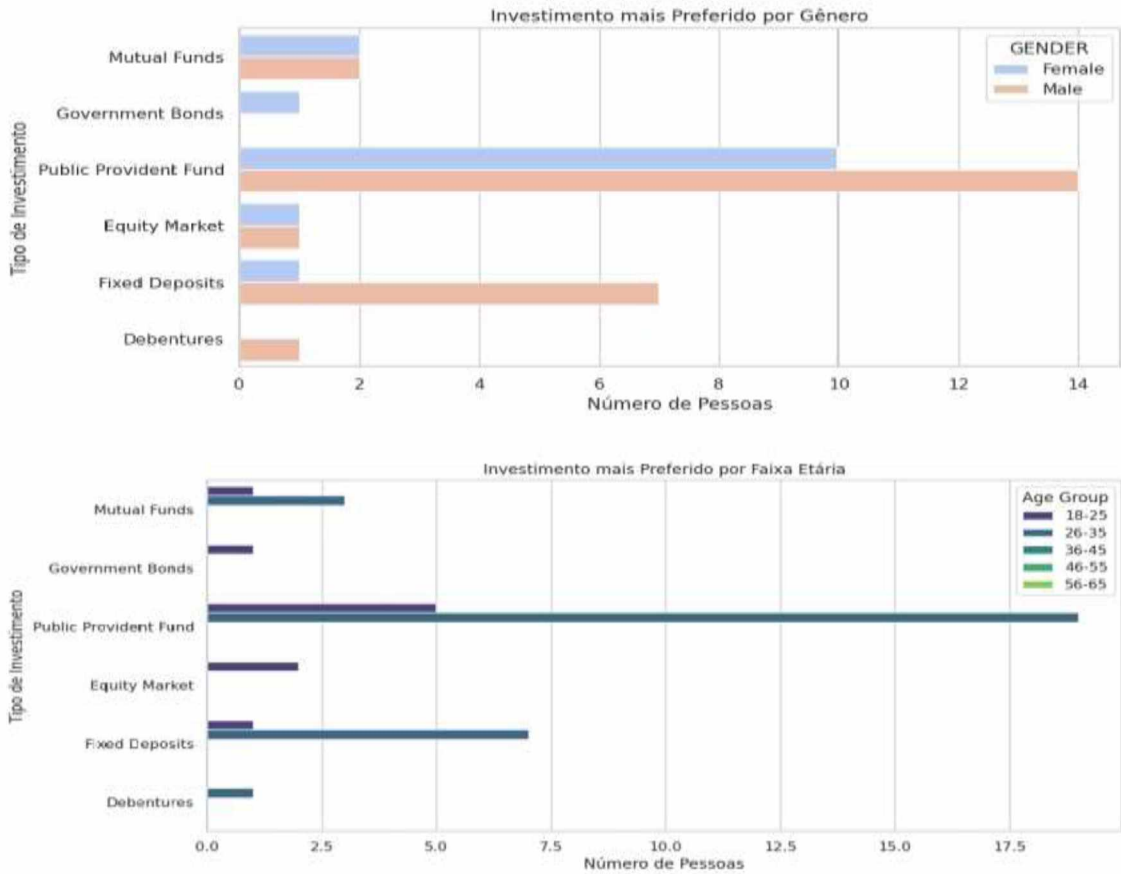


FONTE: Amauri (2025).

```
6 Investimento mais Preferido por Gênero
plt.figure(figsize=(10,5))
sns.countplot(y="Most Preferred Investment", hue="GENDER", data=df,
palette='coolwarm')
plt.title("Investimento mais Preferido por Gênero")
plt.xlabel("Número de Pessoas")
plt.ylabel("Tipo de Investimento")
plt.show()

7 Investimento mais Preferido por Faixa Etária
plt.figure(figsize=(12,6))
sns.countplot(y="Most Preferred Investment", hue="Age Group", data=df,
palette='viridis')
plt.title("Investimento mais Preferido por Faixa Etária")
plt.xlabel("Número de Pessoas")
plt.ylabel("Tipo de Investimento")
plt.show()
```

GRÁFICO 21 – DISTRIBUIÇÃO POR TIPO DE INVESTIMENTO (GÊNERO E FAIXA ETÁRIA)



FONTE: Amauri (2025).

## APÊNDICE N - 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**

## TRABALHO DE IAA015 – Tópicos de Inteligência Artificial

### 1) Algoritmo Genético

Melhor Solução Inicial (População Aleatória):

Distância total aproximada: ex: **4212.73** unidades (valor depende da geração aleatória). O caminho foi visualizado graficamente, destacando a distribuição aleatória e a falta de otimização.

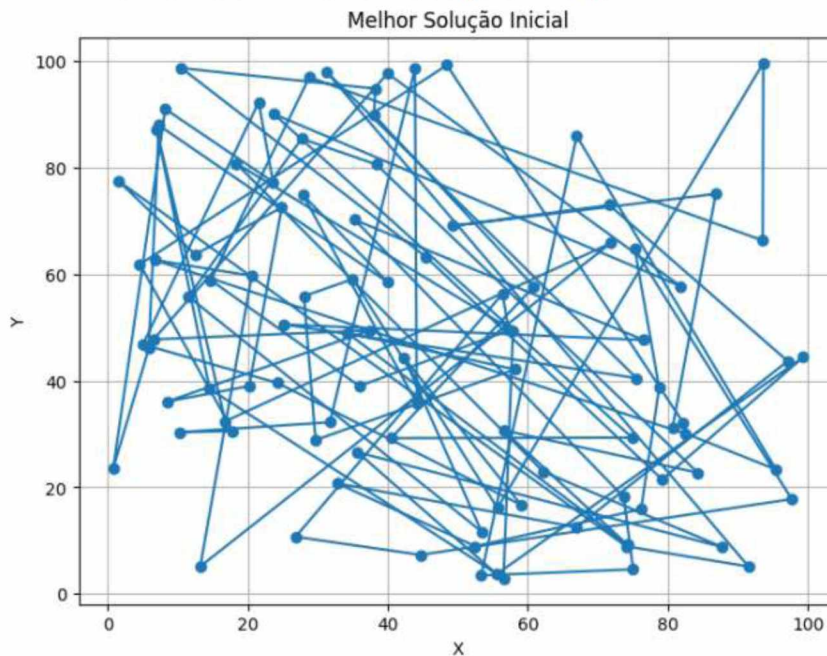
Melhor Solução após 1000 Gerações:

- Distância total aproximada: ex: **2004.12** unidades.

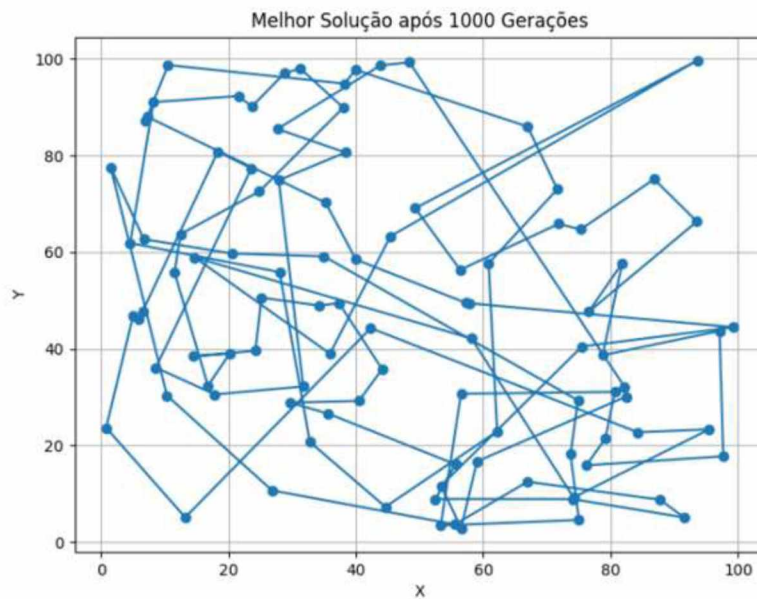
O percurso final demonstrou uma organização mais eficiente e menor sobreposição de rotas, com uma rota mais curta e otimizada.

Foram geradas duas figuras:

- Caminho da melhor solução inicial.



- Caminho otimizado após 1000 gerações.



Essas imagens ilustram claramente a melhora significativa na estrutura do caminho após o processo evolutivo.

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

Parâmetros do algoritmo
NUM_CIDADES = 100
TAMANHO_POPULACAO = 100
NUM_GERACOES = 1000
TAXA_MUTACAO = 0.01
TAXA_CROSSOVER = 0.9

Gerar coordenadas aleatórias das cidades
coordenadas = np.random.rand(NUM_CIDADES, 2) * 100

Função de avaliação (distância total)
def calcular_distancia(caminho):
 dist = 0
 for i in range(len(caminho) - 1):
 dist += np.linalg.norm(
 coordenadas[caminho[i]] - coordenadas[caminho[i + 1]]
)
 return dist

Inicializar população
def gerar_populacao():
 populacao = []
```

```

for _ in range(TAMANHO_POPULACAO):
 meio = list(range(1, NUM_CIDADES))
 random.shuffle(meio)
 caminho = [0] + meio + [0] # cidade inicial é a 0
 populacao.append(caminho)
return populacao

Crossover OX (Order Crossover)
def crossover_ox(pai1, pai2):
 start, end = sorted(random.sample(range(1, NUM_CIDADES), 2))
 meio = pai1[start:end]
 resto = [c for c in pai2 if c not in meio and c != 0]
 novo = [0] + resto[:start - 1] + meio + resto[start - 1:] + [0]
 return novo

Mutação por troca de posição
def mutacao(caminho):
 if random.random() < TAXA_MUTACAO:
 i, j = sorted(random.sample(range(1, NUM_CIDADES), 2))
 caminho[i], caminho[j] = caminho[j], caminho[i]
 return caminho

Seleção por torneio
def selecao_torneio(populacao):
 competidores = random.sample(populacao, 5)
 return min(competidores, key=calcular_distancia)

Evolução
def evoluir_populacao(populacao):
 nova_populacao = []

 melhor = min(populacao, key=calcular_distancia)
 nova_populacao.append(melhor) # elitismo

 while len(nova_populacao) < TAMANHO_POPULACAO:
 if random.random() < TAXA_CROSSOVER:
 pai1 = selecao_torneio(populacao)
 pai2 = selecao_torneio(populacao)
 filho = crossover_ox(pai1, pai2)
 else:
 filho = copy.deepcopy(selecao_torneio(populacao))

 filho = mutacao(filho)
 nova_populacao.append(filho)

 return nova_populacao

Função para plotar caminho
def plotar_caminho(caminho, titulo):
 x = [coordenadas[i][0] for i in caminho]

```

```
y = [coordenadas[i][1] for i in caminho]

plt.figure(figsize=(8, 6))
plt.plot(x, y, marker='o')
plt.title(titulo)
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True)
plt.show()

Execução
def main():
 populacao = gerar_populacao()

 melhor_inicial = min(populacao, key=calcular_distancia)
 print(f"Distância inicial:
{calcular_distancia(melhor_inicial):.2f}")
 plotar_caminho(melhor_inicial, "Melhor Solução Inicial")

 for geracao in range(NUM_GERACOES):
 populacao = evoluir_populacao(populacao)

 melhor_final = min(populacao, key=calcular_distancia)
 print(f"Distância final após {NUM_GERACOES} gerações: "
 f"{calcular_distancia(melhor_final):.2f}")

 plotar_caminho(
 melhor_final,
 f"Melhor Solução após {NUM_GERACOES} Gerações"
)

if __name__ == "__main__":
 main()
```

## 2) Compare a representação de dois modelos vetoriais

### Visualização com Count Vectorizer (PCA):

- Textos com vocabulário semelhante, como os que mencionam “gato”, “cachorro” e “sofá”, foram projetados próximos no espaço vetorial.
- Frases com conteúdo distinto, como “flores coloridas” ou “entrega cartas”, apareceram mais afastadas.

### Visualização com TF-IDF (PCA):

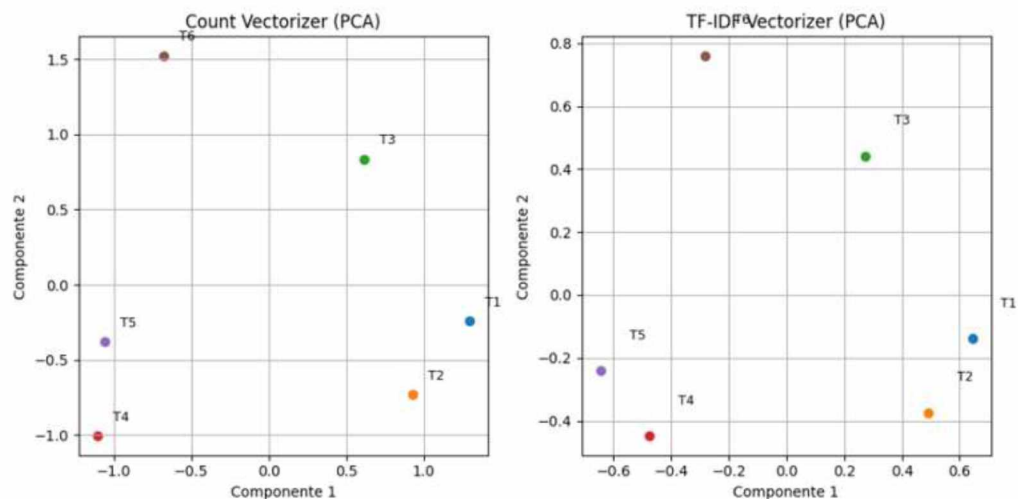
- A separação entre textos foi mais acentuada, uma vez que o modelo TF-IDF penaliza palavras muito frequentes como “o” e “no”, realçando termos mais informativos como “flores” ou “carteiro”.

### Análise Comparativa:

<b>Critério</b>	<b>Count Vectorizer</b>	<b>TF-IDF Vectorizer</b>
Simplicidade	Alta	Média
Considera frequência global	Não	Sim
Diferencia termos comuns	Não	Sim

Visualização PCA	Agrupamentos suaves	Agrupamentos mais definidos
------------------	---------------------	-----------------------------

Ambos os modelos permitiram representar os textos de forma vetorial, mas o TF-IDF se mostrou mais eficaz em distinguir conteúdos semânticos, tornando-se mais adequado quando o objetivo é captar as diferenças reais entre os textos. A redução via PCA foi eficaz para visualização, destacando padrões e agrupamentos semânticos nas frases.



```
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import CountVectorizer,
TfidfVectorizer
from sklearn.decomposition import PCA
import numpy as np

Fragmentos de texto
textos = [
 "O gato dorme no sofá.",
 "O cachorro dorme no sofá.",
 "O gato corre no jardim.",
 "O cachorro late para o carteiro.",
 "O carteiro entrega cartas.",
 "O jardim tem flores coloridas."
]

Representação 1: Count Vectorizer
count_vectorizer = CountVectorizer()
vetores_count = count_vectorizer.fit_transform(textos).toarray()

Representação 2: TF-IDF Vectorizer
tfidf_vectorizer = TfidfVectorizer()
```

```
vetores_tfidf = tfidf_vectorizer.fit_transform(textos).toarray()

Redução de dimensionalidade com PCA
pca = PCA(n_components=2)
vetores_count_pca = pca.fit_transform(vetores_count)
vetores_tfidf_pca = pca.fit_transform(vetores_tfidf)

Plot Count Vectorizer
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)

for i, texto in enumerate(textos):
 plt.scatter(*vetores_count_pca[i], label=f"Texto {i+1}")
 plt.text(
 vetores_count_pca[i][0] + 0.1,
 vetores_count_pca[i][1] + 0.1,
 f"T{i+1}",
 fontsize=9
)

plt.title("Count Vectorizer (PCA)")
plt.xlabel("Componente 1")
plt.ylabel("Componente 2")
plt.grid(True)

Plot TF-IDF Vectorizer
plt.subplot(1, 2, 2)

for i, texto in enumerate(textos):
 plt.scatter(*vetores_tfidf_pca[i], label=f"Texto {i+1}")
 plt.text(
 vetores_tfidf_pca[i][0] + 0.1,
 vetores_tfidf_pca[i][1] + 0.1,
 f"T{i+1}",
 fontsize=9
)

plt.title("TF-IDF Vectorizer (PCA)")
plt.xlabel("Componente 1")
plt.ylabel("Componente 2")
plt.grid(True)

plt.tight_layout()
plt.show()
```