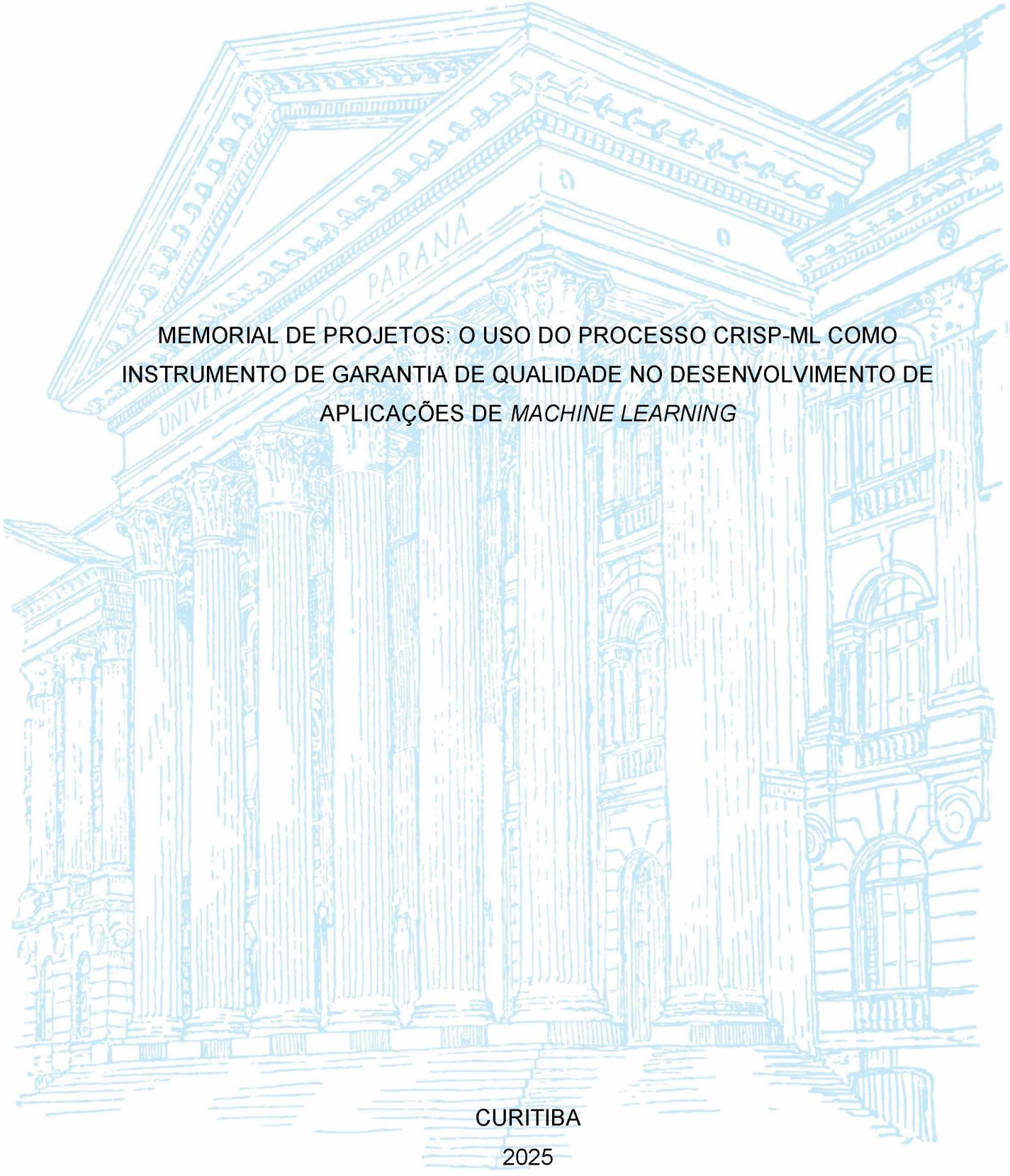
UNIVERSIDADE FEDERAL DO PARANÁ

GLAUCO DAMASCENO SILVA



MEMORIAL DE PROJETOS: O USO DO PROCESSO CRISP-ML COMO INSTRUMENTO DE GARANTIA DE QUALIDADE NO DESENVOLVIMENTO DE APLICAÇÕES DE *MACHINE LEARNING*

CURITIBA

2025

GLAUCO DAMASCENO SILVA

MEMORIAL DE PROJETOS: O USO DO PROCESSO CRISP-ML COMO INSTRUMENTO DE GARANTIA DE QUALIDADE NO DESENVOLVIMENTO DE APLICAÇÕES DE *MACHINE LEARNING*

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

Orientadora: Profa. Dra. Rafaela Mantovani Fontana

CURITIBA

2025

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação Inteligência Artificial Aplicada da Universidade Federal do Paraná foram convocados para realizar a arguição da Monografia de Especialização de **GLAUCO DAMASCENO SILVA**, intitulada: **MEMORIAL DE PROJETOS: O USO DO PROCESSO CRISP-ML COMO INSTRUMENTO DE GARANTIA DE QUALIDADE NO DESENVOLVIMENTO DE APLICAÇÕES DE MACHINE LEARNING**, 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, 21 de Outubro de 2025.

RAFAELA MANTOVANI FONTANA
Presidente da Banca Examinadora

RAZER ANTHOM NIZER ROJAS MONTAÑO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

Neste parecer técnico, faz-se uma discussão acerca da ausência de uma padronização hegemônica de metodologias e modelos para o desenvolvimento de soluções de inteligência artificial e aprendizado de máquina, e da necessidade da sua utilização para garantir a qualidade final destas. A partir do referencial teórico apresentado por livros e artigos científicos relacionados ao tema, faz-se uma análise das metodologias e modelos neles propostos, com foco principal no modelo de desenvolvimento CRISP-ML (*CRoss-Industry Standard Process model for Machine Learning)*, especialmente na sua mais recente versão proposta, CRISP-ML(Q), em que são introduzidas metodologias de garantia de qualidade da solução durante todo o seu ciclo de vida. Observa-se que, devido à natureza não-determinísticas destas aplicações, e à adição da gestão e tratamento dos dados utilizados e do desenvolvimento de modelos ao seu escopo, se faz necessária a utilização de metodologias específicas, que possam direcionar corretamente o processo de desenvolvimento e, assim, garantir a sua qualidade. O modelo é baseado no CRISP-DM (*CRoss-Industry Standard Process model for Data Mining)* que, historicamente, vem sendo utilizado pelo mercado no desenvolvimento deste tipo de aplicação, mas não produzia resultados totalmente satisfatórios por não ter sido desenvolvido para o domínio de inteligência artificial e aprendizado de máquina. Neste contexto, o modelo CRISP-ML(Q) é apresentado como uma opção específica para o domínio em questão e que busca, através do estabelecimento de fases e tarefas bem definidas, garantir que todos os requisitos da solução sejam realmente satisfeitos.

Palavras-chave: metodologias de desenvolvimento de software; CRISP-ML; aprendizado de máquina; inteligência artificial.

ABSTRACT

This technical report discusses the absence of a hegemonic standardization of methodologies and models for the development of artificial intelligence and machine learning solutions, as well as the necessity of their adoption to ensure the final quality of such solutions. Drawing upon the theoretical framework presented in books and scientific articles related to the subject, an analysis is conducted of the methodologies and models proposed therein, with a primary focus on the CRISP-ML (CRoss-Industry Standard Process model for Machine Learning) development model, especially its most recently proposed version, CRISP-ML(Q), which introduces quality assurance methodologies throughout the entire solution lifecycle. It is observed that, due to the non-deterministic nature of these applications and the inclusion of data management and processing and model development within their scope, the use of specific methodologies is required to properly guide the development process and thus guarantee its quality. The model is based on CRISP-DM (CRoss-Industry Standard Process model for Data Mining) which, historically, has been employed by the market for the development of such applications but has not yielded fully satisfactory results, as it was not originally designed for the domain of artificial intelligence and machine learning. In this context, the CRISP-ML(Q) model is presented as a domain-specific option that seeks, through the establishment of well-defined phases and tasks, to ensure that all solution requirements are effectively met.

Keywords: software development methodologies; CRISP-ML; machine learning; artificial intelligence.

SUMÁRIO

1 PARECER TÉCNICO

Desde o advento das chamadas metodologias ágeis para desenvolvimento de software, elas vêm ganhando espaço em relação ao método em cascata devido a sua capacidade de aumentar consideravelmente a produtividade dos times e a qualidade dos produtos desenvolvidos.

Sutherland (2014) argumenta que as equipes que utilizam o Scrum, por exemplo, obtêm o que ele chama de estado de *hiperprodutividade*, onde costuma-se verificar aumentos da produtividade de pelo menos 300% a 400% e, em casos excepcionais, de até 800%, além de reprodutibilidade do sucesso no desenvolvimento, com níveis sempre crescentes de qualidade dos produtos.

Conforme citado por Steidl, Felderer e Ramler (2023), mais recentemente, têm se tornado dominantes os paradigmas de desenvolvimento contínuo de software, tais como DevOps (*Development and Operations*), CI (*Contínuous Integration*) e CD (*Continuous Delivery* ou *Deployment*). Esses paradigmas vêm sendo também utilizados no desenvolvimento de soluções de Inteligência Artificial (IA) e Aprendizado de Máquina (ML) com variado sucesso.

Dentro desse contexto, ainda não há um consenso na indústria acerca de um paradigma de desenvolvimento ágil para sistemas de IA e ML que produzam os mesmos resultados de produtividade e qualidade já introduzidos no desenvolvimento de software tradicional. Steidl, Felderer e Ramler (2023) mencionam que com o advento do poder computacional, cada vez mais soluções de IA necessitam ser integradas em sistemas tradicionais, mas se faz necessário um grau de atenção adicional no seu desenvolvimento, para que se possa assegurar a segurança e a robustez das soluções geradas.

Os mesmos autores ainda sugerem que uma possível solução seria a adoção de *pipelines* automáticas de CI/CD durante todo o desenvolvimento das soluções de IA, como no desenvolvimento de software tradicional. No entanto, pelo fato das soluções de IA e ML serem não-determinísticas e o seu desenvolvimento também abarcar no seu escopo a gestão dos dados utilizados e os próprios modelos em si, o nível de complexidade para a definição de uma possível metodologia ou *framework* aumenta consideravelmente.

Como resultado do seu estudo, que envolveu revisão da literatura acadêmica existente sobre *pipelines* de desenvolvimento de soluções de IA e ML, bem como

entrevistas com acadêmicos que pesquisam o tema e profissionais da indústria que desenvolvem as soluções, eles propõem um *pipeline* para o desenvolvimento de soluções de IA e ML, composto de 4 estágios sequenciais, porém iterativos: (1) Manipulação dos Dados, (2) Aprendizado do Modelo, (3) Desenvolvimento do Software e (4) Operações do Sistema. No entanto, é enfatizado também pelo resultado do estudo, que as etapas do *pipeline*, bem como as tarefas de cada uma, são dependentes do contexto organizacional e de negócios onde a solução está sendo desenvolvida.

Assim, devido a este cenário de processos e *frameworks* relativamente não padronizados para desenvolvimento de soluções de IA e ML, foi proposto o modelo CRISP-ML(Q) (*Cross-Industry Standard Process for Machine Learning applications with Quality assurance*), que visa exatamente guiar os desenvolvedores através do ciclo de vida da solução. Segundo Costa (2023), o modelo CRISP-ML é um *framework* sistemático para a organização e execução de projetos de aprendizado de máquina.

O modelo CRISP-ML(Q) foi derivado do modelo CRISP-DM (*Cross-Industry Standard Process for Data Mining*), que, na falta de um modelo padrão, já vinha sendo utilizado pela indústria para desenvolvimento de aplicações de ML, apresentando, porém, algumas deficiências. Studer *et al.* (2021), citam como dois maiores exemplos que, primeiro, CRISP-DM foca em mineração de dados (*data mining*) e não cobre cenários de aplicações de ML inferindo decisões em tempo real por longos períodos, o que exige adaptabilidade do modelo a mudanças de ambiente e dados ao longo do tempo. Segundo, e mais preocupante, é que CRISP-DM sofre da falta de um direcionamento específico para a garantia da qualidade das soluções, o que aumenta o risco de se ter que corrigir problemas em estágios avançados do desenvolvimento, o que sempre é oneroso para o projeto em todos os aspectos. Studer *et al.* (2021) ainda mencionam que, metodologias de garantia de qualidade e mitigação de risco, são introduzidas em cada uma das tarefas do processo, de forma que se possa mitigar um possível insucesso da solução de ML.

Segundo Costa (2023), o modelo CRISP-ML é dividido em 7 fases: Entendimento do Negócio, Entendimento dos Dados, Modelagem, Avaliação, Implantação e Gerenciamento do Modelo.

No entanto, para Studer *et al.* (2021), no CRISP-ML(Q), as fases de Entendimento do Negócio e Entendimento dos Dados, devem ser fundidas em uma só, já que possíveis restrições na disponibilidade e capacidade de obtenção dos dados

podem ser um fator de impacto na viabilidade do projeto, chegando mesmo a inviabilizá-lo. Ao analisar cada uma das fases do modelo, eles propõem uma série de recomendações que se observadas, podem garantir que o mesmo funcione a contento e garanta o sucesso do projeto.

Na primeira fase Studer *et al.* (2021) recomendam que os critérios de sucesso devem ser analisados sob três diferentes óticas: a de resultados de negócio, a de resultados de ML, inclusive para um possível MVP (*Minimum Viable Product*), e a de impactos econômicos, como por exemplo, redução de custos, aumento de receitas e ROI (*Return Over Investment*). Também deve ser verificada a viabilidade do projeto, sua aplicabilidade, possíveis restrições legais, uso de recursos e atendimento de requisitos, como: escalabilidade, robustez, complexidade e explicabilidade, por exemplo. Além disso devem ser analisados a forma de coleta dos dados, o seu versionamento e a sua qualidade (metadados, inferências estatísticas, *outliers*, dados faltantes etc.).

Para a fase de modelagem, eles propõem que se deva tentar utilizar um modelo específico para o domínio de negócios, caso exista. Além disso, deve-se procurar assegurar a reprodutibilidade do mesmo sob dois aspectos: do método e do resultado, devendo-se guardar o histórico de experimentos efetuados e as mudanças de performance e resultados conseguidos. Também é extremamente importante trabalhar na explicabilidade do modelo para que os *stakeholders* de negócios possam compreender o seu funcionamento, validar a escolha das *features* mais relevantes e assim, assegurar a sua confiança nos resultados; estes devem ser comparados com os critérios de sucesso delimitados.

Já na fase de avaliação do modelo, Studer *et al.* (2021) advogam que se deve testar o modelo com dados reais, no mesmo ambiente de produção, pois pode haver degradação nos resultados do modelo em relação ao que se havia conseguido no desenvolvimento. Além disso, deve-se avaliar se não há queda na usabilidade do modelo nesse ambiente.

Na fase final de monitoramento, eles propõem que se deva verificar, além das saídas do modelo e a sua performance, qualquer modificação nos sinais de entrada, pois mudanças nos mesmos podem comprometer os resultados apresentados e levar a necessidade de retreinamento.

Desta forma, devido a abordagem bastante abrangente que o modelo CRISP-ML apresenta, principalmente na sua versão mais recente CRISP-ML(Q), em que há

um foco muito grande na mitigação de riscos e testes de qualidade em todas as fases do desenvolvimento, o modelo se apresenta como uma opção para padronização do processo de desenvolvimento de soluções de IA e ML. Como concluem Studer *et al.* (2021, p.15, tradução nossa): “Um importante passo futuro, com base em nosso trabalho e em trabalhos relacionados, é a padronização de um modelo de processo. Isso contribuiria para projetos de ML mais bem-sucedidos e, portanto, teria um impacto significativo na comunidade de ML”.

REFERÊNCIAS

COSTA, R. **The CRISP-ML methodology:** A Step-by-Step Approach to Real-World Machine Learning Projects. [*S.l.*]: R. Costa, 2023.

STEIDL, M.; FELDERER, M.; RAMLER, R. The pipeline for the continuous development of artificial intelligence models—Current state of research and practice. **Journal of Systems and Software**, v. 199, 111615, May 2023. DOI. 10.1016/j.jss.2023.111615.

STUDER, S.; BUI, T.B.; DRESCHER, C.; HANUSCHKIN, A.; WINKLER, L.; PETERS, S.; MÜLLER, K.-R. Towards CRISP-ML(Q): A machine learning process model with quality assurance methodology. [preprint]. **arXiv**, 24 Feb 2021. DOI: 10.48550/arXiv.2003.05155v2.

SUTHERLAND, J. **Scrum:** a arte de fazer o dobro do trabalho na metade do tempo. São Paulo: LeYa, 2014.

APÊNDICE A - INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

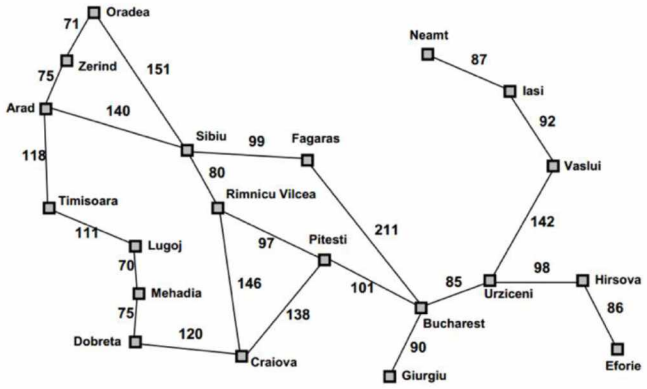
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

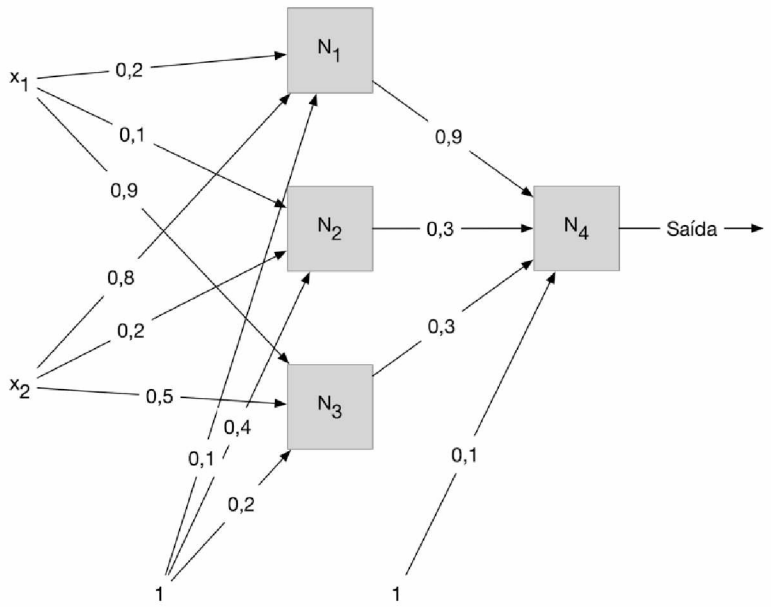
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1 CHATGPT:

a)

Perguntamos ao Chat GPT, na versão 3.5, gratuita, "*O que é Inteligência Artificial*" e obtivemos como resposta o seguinte:

> *"Inteligência Artificial (IA) refere-se à capacidade de um sistema computacional realizar tarefas que normalmente requerem inteligência humana. Isso inclui a capacidade de aprender com experiências passadas, adaptar-se a novas situações, raciocinar, resolver problemas, compreender linguagem natural e até mesmo interagir de forma natural com os humanos."*

Após essa breve descrição, o programa complementou a resposta com mais detalhes, diferenciando as categorias:

> *"Existem duas principais categorias de IA: IA fraca (ou estreita) e IA forte (ou geral).*
>
> ***IA Fraca (Estreita):** Refere-se a sistemas projetados para uma tarefa específica. Eles são especializados em executar uma função específica, como reconhecimento de voz, recomendação de produtos, ou jogar xadrez. Eles não possuem a capacidade de realizar tarefas fora de seu escopo designado.*
>
> ***IA Forte (Geral):** Refere-se a sistemas com a capacidade de realizar qualquer tarefa intelectual que um humano possa fazer. A IA forte seria capaz de generalizar conhecimento e aprender rapidamente em diversos domínios, assim como os seres humanos.*
>
> *As técnicas comuns em IA incluem aprendizado de máquina (machine learning), que envolve treinar modelos com dados para melhorar o desempenho em uma tarefa específica, e processamento de linguagem natural (NLP), que capacita sistemas a entender e gerar linguagem humana.*
>
> *A IA tem aplicações em uma variedade de setores, como saúde, finanças, automotivo, educação, entretenimento e muitos outros, proporcionando soluções inovadoras e otimizando processos. No entanto, também levanta questões éticas e sociais que precisam ser consideradas à medida que a tecnologia avança."*

Por curiosidade, decidimos experimentar a versão 4 do Chat GPT para ver se obtínhamos uma explicação mais detalhada e a resposta foi a seguinte:

"Inteligência Artificial (IA) é um ramo da ciência da computação dedicado a criar sistemas capazes de realizar tarefas que, tradicionalmente, exigiriam inteligência humana. Essas tarefas incluem aprendizado (a capacidade de melhorar o desempenho com base em experiências anteriores), raciocínio (usar informações para chegar a conclusões), percepção (interpretar os estímulos sensoriais), compreensão da linguagem natural e até a criação de arte ou música.

A IA pode ser classificada em dois tipos principais: IA fraca e IA forte. A IA fraca, também conhecida como IA estreita, é projetada e treinada para realizar uma tarefa específica sem a consciência de si mesma. Exemplos incluem assistentes virtuais, como Siri e Alexa, e sistemas de recomendação, como os usados por serviços de streaming de vídeo e música. Já a IA forte, que ainda é teórica e não foi plenamente realizada, teria consciência, senso de si mesma e a capacidade de aprender e aplicar inteligência em uma ampla gama de tarefas, de forma semelhante ou até superior à inteligência humana.

A IA opera usando algoritmos complexos, muitas vezes empregando métodos como aprendizado de máquina (Machine Learning - ML) e aprendizado profundo (Deep Learning - DL), que permitem que o sistema melhore com base na quantidade de dados processados. No aprendizado de máquina, por exemplo, um algoritmo pode aprender a identificar padrões em dados e fazer previsões ou decisões com base nesses padrões, sem ser explicitamente programado para realizar essa tarefa específica.

A aplicação da IA é vasta e permeia diversos setores, incluindo, mas não limitado, à saúde, onde pode ajudar no diagnóstico de doenças; finanças, otimizando operações de trading e gerenciamento de riscos; educação, personalizando o aprendizado; e no comércio varejista, melhorando a experiência do cliente e otimizando a cadeia de suprimentos. No entanto, a IA também levanta questões éticas e sociais importantes, incluindo preocupações com privacidade, segurança e o impacto no mercado de trabalho."

Como previámos, a resposta dada pelo GPT 4 foi um pouco melhor e mais abrangente com a citação de exemplos.

Para efeito de conhecimento, cabe ressaltar que utilizamos ainda, duas outras ferramentas, o ***Copilot*** da Microsoft e o ***Gemini*** da Google. Este último apresentou resultados mais completos, detalhados e com as referências consultadas.

b)

Classificação:

1. Agentes Racionais:

2. Sistemas Baseados em Conhecimento:

O Chat GPT se encaixa na categoria de agentes racionais por sua capacidade de "perceber o ambiente", uma vez que a ferramenta processa informações do mundo real através de linguagem natural, como perguntas na forma de texto, literalmente, e comandos chamados de *prompts*.

A ferramenta também consegue "raciocinar", entenda-se, busca compreender as informações recebidas e utiliza técnicas de IA para analisar essas informações, inferir conclusões e tomar decisões sobre como responder.

Pode ainda, atuar, gerando texto, realizando ações no mundo virtual e interagir com outros agentes.

O Chat GPT também se caracteriza como um sistema baseado em conhecimento:

Por sua Base de conhecimento, a ferramenta possui um extenso conjunto de dados de texto e código que serve como base para suas decisões e ações. Possui ainda, uma capacidade de inferência, já que o Chat GPT utiliza técnicas de IA para inferir novas informações a partir de sua base de conhecimento e das informações recebidas do ambiente.

c)

Segundo Brown et al (2020) o ChatGPT é um modelo transformador pré-treinado generativo, também conhecido como LLM (Large Language Model), que se destaca por sua capacidade de realizar diversas tarefas com alto grau de qualidade.

A geração de texto pelo ChatGPT, que pode criar textos em diferentes estilos e formatos, desde poemas e scripts até e-mails, artigos e código, com alto nível de criatividade e originalidade, ilustra a aplicação dos princípios descritos por Brown et al. (2020).

Esta capacidade é complementada pela tradução automática, onde a ferramenta oferece tradução precisa e fluida entre diversos idiomas, facilitando a comunicação e o acesso à informação em diferentes culturas, um avanço tecnológico que se alinha com as observações de Radford et al. (2019).

Além disso, o ChatGPT responde a perguntas de maneira informativa e abrangente, mesmo que complexas ou abertas, demonstrando sua capacidade de processamento de linguagem natural e compreensão de nuances (DEVLIN 2018). O modelo também é capaz de gerar resumos precisos e concisos de textos longos, preservando as informações mais importantes e facilitando a leitura e a compreensão.

O funcionamento do ChatGPT se baseia em uma arquitetura de rede neural profunda, similar ao funcionamento do cérebro humano. Essa rede neural é treinada em um conjunto de dados massivo, composto por textos e códigos de diversas fontes, como livros, artigos, websites e repositórios de código. Através do processo de aprendizado de máquina, a rede neural identifica padrões nos dados e aprende a utilizá-los para gerar novos textos, uma metodologia que Radford et al. (2019) também discutem.

O ChatGPT utiliza técnicas de atenção para focar em partes específicas do texto durante o processamento, o que permite gerar respostas mais relevantes e contextualmente adequadas, um conceito que está em consonância com as descobertas de Devlin et al. (2018).

REFERÊNCIAS:

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. "Language models are few-shot learners." Advances in Neural Information Processing Systems. 2020.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).: https://arxiv.org/abs/1810.04805

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." OpenAI Blog 1 (8) (2019).: https://openai.com/blog/language-unsupervised/

d)

Conforme a análise a seguir, concluímos que no caso específico do ChatGPT, ele se enquadra nas 4 abordagens estudas. No livro "Inteligência Artificial: Uma abordagem moderna" por Stuart Russell e Peter Norvig, os autores oferecem uma visão abrangente dos princípios e técnicas de inteligência artificial (IA) e exploram ainda, várias abordagens para entender e construir sistemas de IA. Embora o livro não discuta o ChatGPT especificamente, ele apresenta uma estrutura teórica que pode ser aplicada para entender a tecnologia subjacente ao ChatGPT e outros sistemas de IA avançados, vamos a elas:

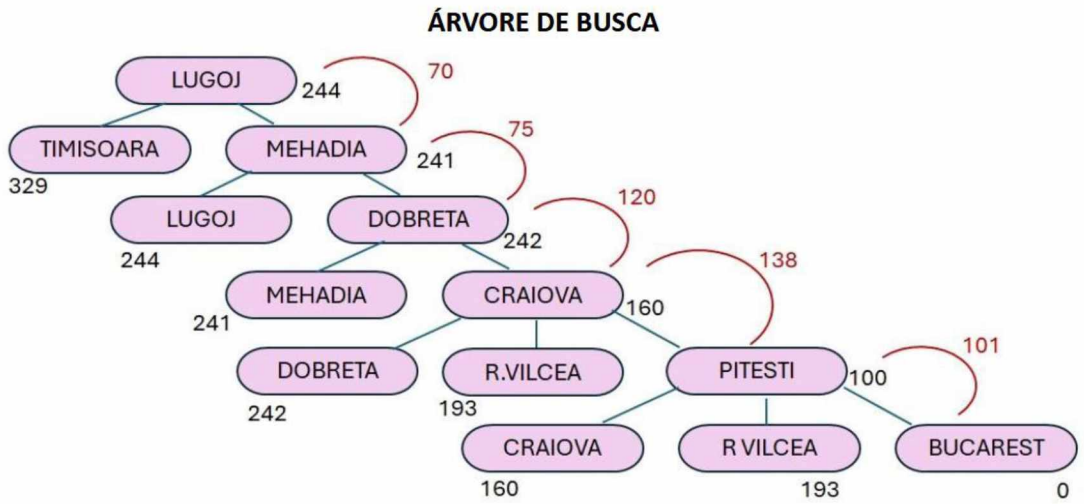
Agentes Racionais: No Capítulo 2, "Agentes", Russell e Norvig introduzem o conceito de agentes racionais, descrevendo-os como entidades que percebem seu ambiente através de sensores e atuam sobre esse ambiente através de atuadores, com base em uma função de “agente” (um algoritmo) que mapeia percepções para ações. O objetivo de um agente racional é realizar ações que maximizem algum critério de sucesso, dada a evidência fornecida pelas percepções e o conhecimento embutido no agente. Esta abordagem é fundamental para o desenvolvimento de sistemas de IA, incluindo modelos de linguagem como o ChatGPT, que processam entrada de texto (percepções) e geram respostas (ações) de maneira a maximizar a relevância e a utilidade da resposta.

Sistemas Baseados em Conhecimento: No Capítulo 7, "Agentes Lógicos", e Capítulo 9, "Interferência em lógica de primeira ordem", os autores discutem sistemas baseados em conhecimento e como o conhecimento pode ser representado e manipulado para realizar inferências. Embora o ChatGPT não opere através de regras lógicas explícitas ou inferência formal, ele é treinado em um vasto conjunto de dados textuais que funcionam como uma base de conhecimento implícita, permitindo que ele gere respostas que refletem o conhecimento adquirido durante o treinamento.

Abordagem Empírica: A aprendizagem é um tema central do livro com destaque no Capítulo 18, "Aprendendo com exemplos", onde os autores abordam o aprendizado de máquina como um meio de melhorar a performance de agentes baseados em experiências passadas. O ChatGPT, sendo um modelo de linguagem baseado em transformers e treinado com técnicas de aprendizado profundo, é um exemplo de aplicação desta abordagem empírica, utilizando grandes quantidades de dados textuais para aprender padrões de linguagem e conhecimento factual.

Abordagem Naturalista: Embora o livro trate de inspirações biológicas para a IA em várias seções, a ideia de imitar processos biológicos, como o funcionamento do cérebro humano, é mais um tema subjacente do que uma seção dedicada. A arquitetura de redes neurais, discutida no contexto de aprendizado de máquina (por exemplo, no Capítulo 18), é inspirada pela estrutura e funcionamento dos neurônios no cérebro humano. O ChatGPT, utilizando uma arquitetura de rede neural profunda (transformer), reflete essa abordagem naturalista ao imitar, em um nível abstrato, os processos de aprendizado do cérebro humano.

2 BUSCA HEURÍSTICA



3 LÓGICA

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

p: as uvas caem

q: a raposa come as uvas r:

as uvas estão maduras

Esperado: $q \leftrightarrow p$

R1: $p \rightarrow q$

R3: $\neg r \vee p$

R4: r → p EI, R3

R5: q → p SI, R2, R4

R6: (q → p) ^ (p →q) CONJ, R5, R1

R7: (q ↔ p) BICOND, R6

4 REDES NEURAIS ARTIFICIAIS

a)

Valor de saída do neurônio N1 – R: 0,3

N1 - = 1 x 0,4 + (-3) x 0,2 + 1 x 0,8

N1 = 0,3

b)

Valor de saída do neurônio N2 – R: 0,3

N2 = 1 x 0,4 + (-3) x 0,1 + 1 x 0,2

N2 = 0,3

c)

Valor de saída do neurônio N3 – R: -2

N3 = 1 x 0,2 + (-3) x 0,9 + 1 x 0,5

N3 = -2

d)

Valor de saída da rede como um todo – R: -0,1391

N4 = 1 x 0,1 + 0,3 x 0,9 + 0,3 x 0,3 + (-2) x 0,3

N4 = -0,14

Tanh (u) = -0,1391

F(a)(u) = -0,13909

APÊNDICE B - LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

2 Visualização dos dados

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

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

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

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

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

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

B - RESOLUÇÃO

1 Análise Exploratória dos dados

A base originalmente possui 267542 linhas e 11 colunas. Dessas linhas, no final do arquivo há 65245 linhas vazias, talvez devido à exportação do arquivo para o formato CSV. Ao eliminar-se essas linhas não resta nenhum valor faltante. Existem duas linhas duplicadas. Das 11 colunas, a base apresenta 3 numéricas e 8 categóricas. Pode-se observar que os dados se distribuem por somente 5 marcas de veículos, porém com uma grande quantidade de modelos.

2 Visualização dos dados

Em quase todas as marcas, há uma grande diferença entre o preço médio dos carros automáticos em relação aos manuais. Exceto a Renault aonde, além dos valores serem bem próximos, o preço médio dos carros manuais é um pouco maior que dos automáticos. A marca com maior valor médio de preço dos modelos automáticos é a Volkswagem, seguida por Fiat e Nissan, respectivamente.

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

As variáveis com maior importância em ambos os tipos de modelos (Random Forest e XGBoost) foram o tamanho do motor e o ano-modelo dos carros. Utilizando-se o modelo XGBoost, houve um aumento razoável na importância das demais variáveis utilizadas na predição (combustível e tipo de engrenagem).

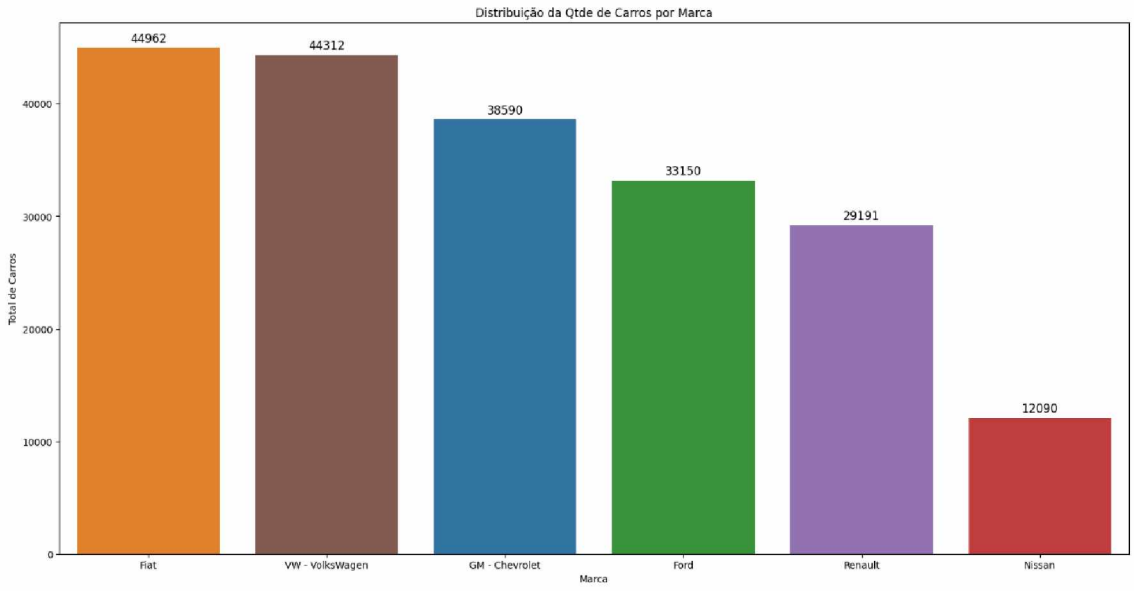
Com as variáveis escolhidas, tanto os erros apresentados quanto o R^2 de ambos os modelos foram quase idênticos. Assim, qualquer um dos dois apresentaria resultados muito parecidos. No entanto, o modelo XGBoost teve erros ligeiramente menores e o R^2 pouca coisa maior, o que nos faria optar por ele.

CÓDIGO

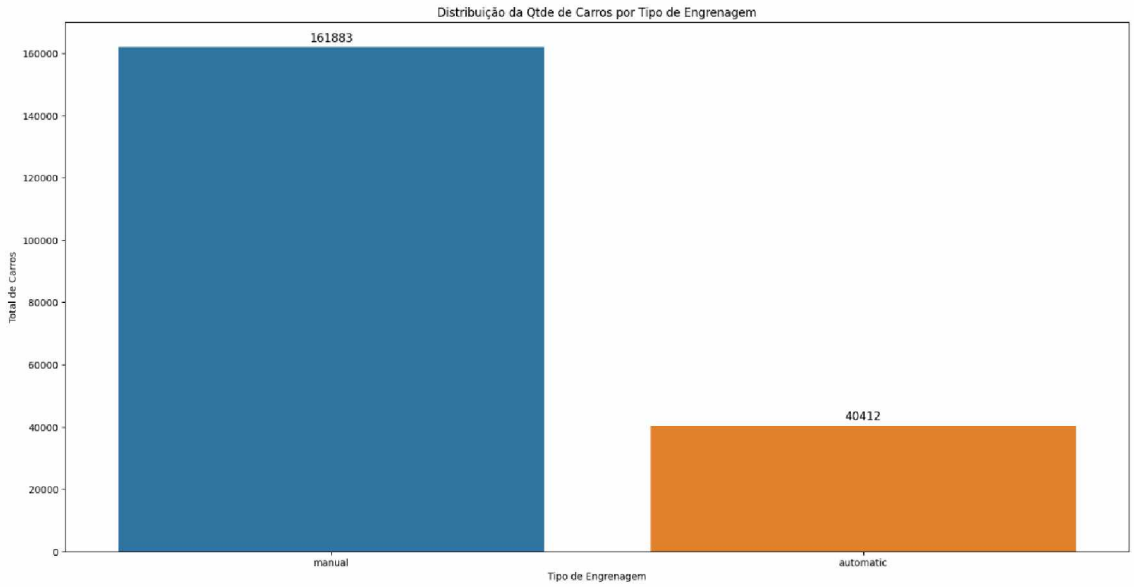
```
# %% [markdown]
# # Trabalho Final IAA002 - Linguagem de Programação Aplicada (turma 2024)

# %% [markdown]
# ## Importando as bibliotecas do Python

# %% [markdown]
# ### Biliotecas para manipulação e exibição dos dados

# %%
# Pandas - Dataframes
import pandas as pd
# Matplotlib.Pyplot - gráficos
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter
# Seaborn - gráficos
import seaborn as sns
# Warnings
import warnings
warnings.filterwarnings('ignore')

# %% [markdown]
# ### Bibliotecas de Machine Learning

# %%
# Scikit Learn
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
# XGBoost
from xgboost import XGBRegressor

# %% [markdown]
# ## 1. Análise Exploratória dos Dados

# %% [markdown]
# ### a. Carregue a base de dados media_precos_carros_brasil.csv

# %%
# Carrega o arquivo com os dados num dataframe e exibe uma amostra
dados_df = pd.read_csv('precos_carros_brasil.csv')
dados_df.head(10)

# %%
# Verifica as dimensões do DF
dados_df.shape

# %%
# Verifica os tipos de dados das colunas
dados_df.dtypes
```

```
# %% [markdown]
# ### b. Verifique se há valores faltantes nos dados. Caso haja, escolha uma
tratativa para resolver o problema de valores faltantes
# %%
# Verificando se há valores faltantes (na)
dados_df.isna().any()

# %%
# Verificando a quantidade de valores faltantes por coluna
dados_df.isna().sum()

# %% [markdown]
# *** Todas as colunas possuem 65245 valores faltantes, ou seja, existem 65245
linhas com todas as colunas em branco. A melhor estratégia é excluir essas linhas.

# %%
# Deleta todas as linhas do dataframe que possuem todas as colunas com dados
faltantes
dados_df.dropna(axis=0, how='all', inplace=True)

# %%
# Verifica as dimensões do DF para comparar após deletar as linhas
dados_df.shape

# %%
# Foram excluídas exatamente 65245 linhas
267542-65245

# %%
# Verificando novamente se há valores faltantes (na)
dados_df.isna().any()

# %%
# Verificando novamente a quantidade de valores faltantes por coluna
dados_df.isna().sum()

# %% [markdown]
# ### c. Verifique se há dados duplicados nos dados

# %%
# Verificando se há dados duplicados
dados_df.duplicated().sum()

# %% [markdown]
# *** Existem dados duplicados, então é melhor excluir a duplicação

# %%
# Exclui as linhas duplicadas do DF
dados_df.drop_duplicates(inplace=True)

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

# %%
# Criando duas categorias para separar colunas numéricas e categóricas
num_cols = [col for col in dados_df.columns if dados_df[col].dtype != 'object']
categ_cols = [col for col in dados_df.columns if dados_df[col].dtype == 'object']
```

```
# %%
# Imprime o resumo descritivo das colunas numéricas
dados_df[num_cols].describe()

# %%
# Imprime o resumo descritivo das colunas categóricas
dados_df[categ_cols].describe()

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

# %%
# Contagem de valores por modelo
dados_df['model'].value_counts()

# %%
# Contagem de valores por marca
dados_df['brand'].value_counts()

# %% [markdown]
# ## 2. Visualização dos dados

# %% [markdown]
# ### a. Gere um gráfico da distribuição da quantidade de carros por marca

# %%
# Gráfico da distribuição da qtde de carros por marca.
plt.figure(figsize=(20,10))
marca_plt = sns.countplot(x="brand", hue="brand", \
                          data=dados_df, \
                          order=dados_df['brand'].value_counts().index)
marca_plt.set_xlabel('Marca')
marca_plt.set_ylabel('Total de Carros')
for c in marca_plt.containers:
    marca_plt.bar_label(c, size=12, padding=3)
marca_plt.set_title('Distribuição da Qtde de Carros por Marca')
```

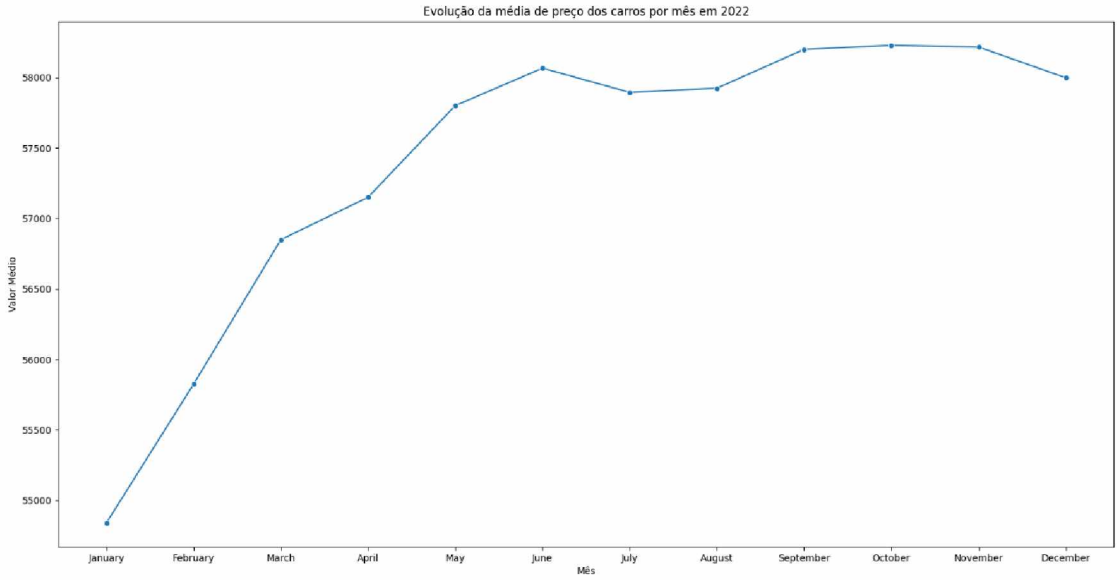


```
# %% [markdown]
# ### b. Gere um gráfico da distribuição da quantidade de carros por tipo de
engrenagem do carro

# %%
# Gráfico da distribuição da qtde de carros por tipo de engrenagem.
plt.figure(figsize=(20,10))
marca_plt = sns.countplot(x="gear", hue="gear", \
                          data=dados_df,
order=dados_df['gear'].value_counts().index)
marca_plt.set_xlabel('Tipo de Engrenagem')
marca_plt.set_ylabel('Total de Carros')
for c in marca_plt.containers:
    marca_plt.bar_label(c, size=12, padding=3)
marca_plt.set_title('Distribuição da Qtde de Carros por Tipo de Engrenagem')
```

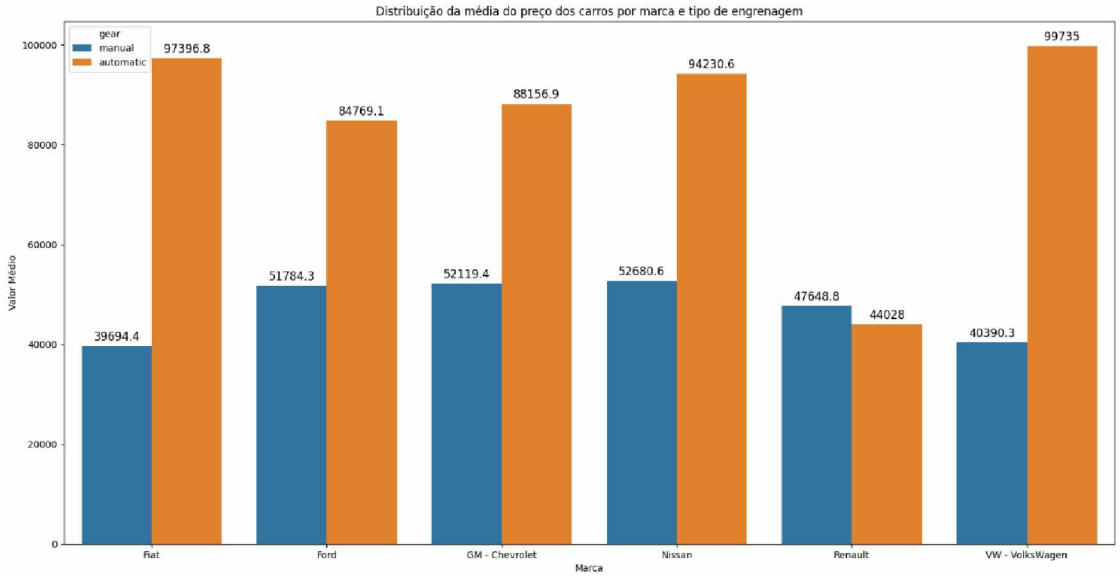



```
# %% [markdown]
# ### c. Gere um gráfico da evolução da média de preço dos carros ao longo dos
meses de 2022 (variável de tempo no eixo X)

# %%
# Cria o dataframe para exibir o gráfico
avg_price_df = dados_df[dados_df['year_of_reference'] == 2022]. \
               groupby('month_of_reference')['avg_price_brl'].mean(). \
               round(2).reset_index()
custom_order = dados_df[dados_df['year_of_reference'] ==
2022]['month_of_reference'].unique()
avg_price_df['month_of_reference'] =
pd.Categorical(avg_price_df['month_of_reference'], \
               categories=custom_order, ordered=True)

# Gráfico da evolução da média de preço do carros por mês em 2022.
plt.figure(figsize=(20,10))
marca_plt = sns.lineplot(x="month_of_reference", y='avg_price_brl', marker='o', \
                         data=avg_price_df)
marca_plt.set_xlabel('Mês')
marca_plt.set_ylabel('Valor Médio')
# marca_plt.bar_label(marca_plt.containers[0], size=12, padding=3)
```

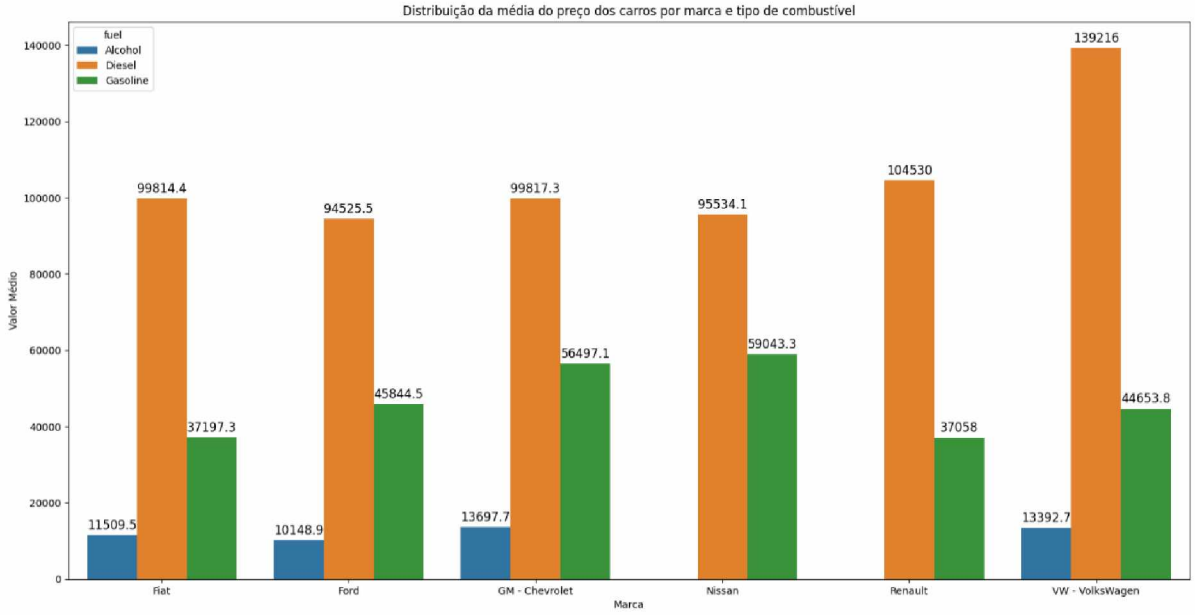
```
marca_plt.set_title('Evolução da média de preço dos carros por mês em 2022')
```



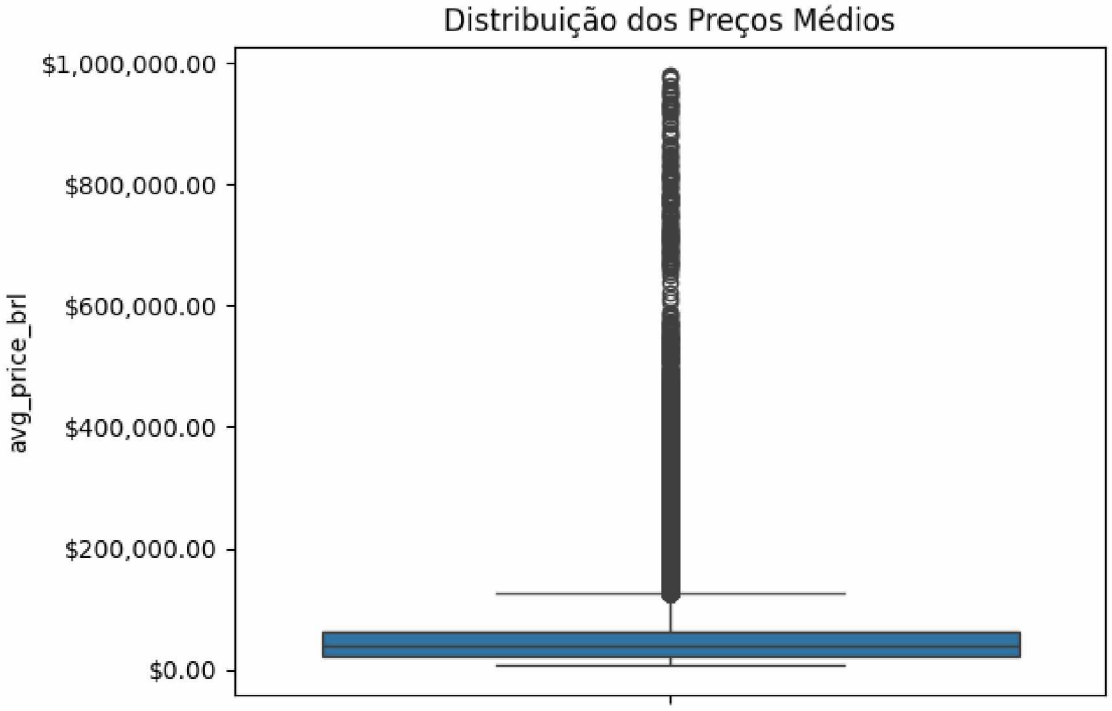

```
# %% [markdown]
# ### d. Gere um gráfico da distribuição da média de preço dos carros por marca e tipo de engrenagem

# %%
# Gráfico da distribuição da média de preço do carros por marca e tipo de engrenagem.
plt.figure(figsize=(20,10))
marca_plt = sns.barplot(x='brand', y='avg_price_brl', hue='gear', \
                        data=dados_df.groupby(['brand', 'gear'])['avg_price_brl'] \
                                                .mean().round(2).reset_index(), \
                                    hue_order=dados_df['gear'].unique())
marca_plt.set_xlabel('Marca')
marca_plt.set_ylabel('Valor Médio')
for container in marca_plt.containers:
    marca_plt.bar_label(container, size=12, padding=3)
marca_plt.set_title('Distribuição da média do preço dos carros por marca e tipo de engrenagem')
```



```
# %% [markdown]
# ### f. Gere um gráfico da distribuição da média de preço dos carros por marca e
tipo de combustível

# %%
# Gráfico da distribuição da média de preço do carros por marca e tipo de
combustível.
plt.figure(figsize=(20,10))
marca_plt = sns.barplot(x='brand', y='avg_price_brl', hue='fuel', \
                        data=dados_df.groupby(['brand', 'fuel'])['avg_price_brl'] \
                                                .mean().round(2).reset_index(), \
                              hue_order=dados_df['fuel'].unique().sort())
marca_plt.set_xlabel('Marca')
marca_plt.set_ylabel('Valor Médio')
for container in marca_plt.containers:
    marca_plt.bar_label(container, size=12, padding=3)

marca_plt.set_title('Distribuição da média do preço dos carros por marca e tipo de
combustível')
```

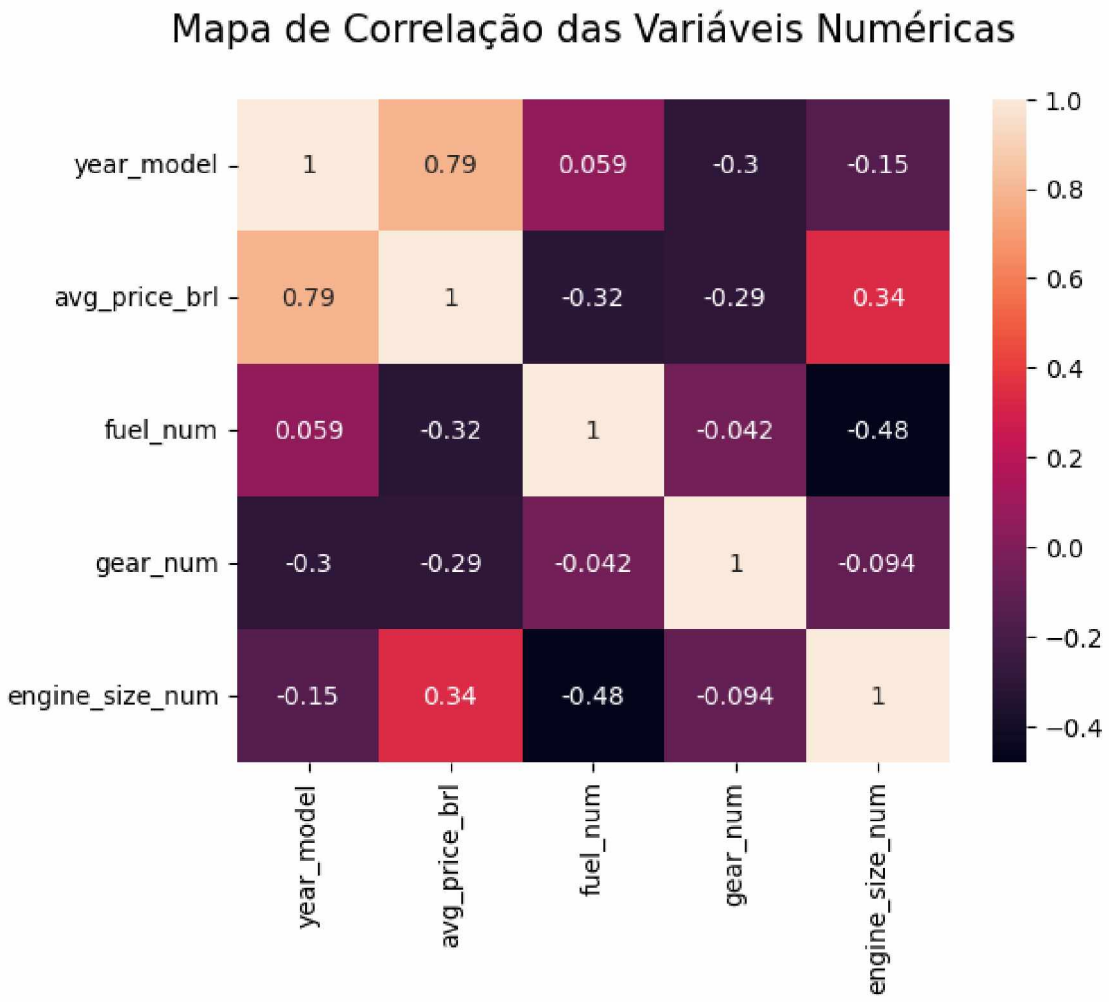



```
# %% [markdown]
# ## 3. Aplicação de modelos de machine learning para prever o preço médio dos
carros

# %% [markdown]
# ### a. Escolha as variáveis numéricas (modelos de Regressão) para serem as
variáveis independentes do modelo. A variável target é avg_price. Observação: caso
julgue necessário, faça a transformação de variáveis categóricas em variáveis
numéricas para inputar no modelo. Indique quais variáveis foram transformadas e
como foram transformadas

# %%
# Verifica os valores da variável target num boxplot
ax = sns.boxplot(y="avg_price_brl", data=dados_df)
ax.set_title("Distribuição dos Preços Médios")
currency_formatter = FuncFormatter(lambda x, _: f'${x:,.2f}')
ax.yaxis.set_major_formatter(currency_formatter)
```




```
# %% [markdown]
# *** Necessário converter as variáveis categóricas **fuel**, **gear** e
**engine_size** em numéricas para poder aplicar a regressão. Converter também a
variáveis **year_model** de **float** para **int**.

# %%
# Converte a coluna year_model de float para int
dados_df['year_model'] = dados_df['year_model'].astype(int)
dados_df.head()

# %%
# Coluna fuel tem três valores: 'Alcohol', 'Diesel, 'Gasoline'.
# Cria uma coluna nova que ficará com valores 0('Alcohol'), 1('Diesel'),
2('Gasoline')
dados_df['fuel_num'] = LabelEncoder().fit_transform(dados_df['fuel'])
dados_df.head()

# %%
# Coluna gear tem dois valores: 'automatic', 'manual'.
# Cria uma coluna que ficará com valores 0('automatic'), 1('manual')
dados_df['gear_num'] = LabelEncoder().fit_transform(dados_df['gear'])
dados_df.head()

# %%
# Cria uma coluna para conter o engine_size em formato numérico
dados_df['engine_size_num'] = dados_df['engine_size'].str.replace(',',
'.').astype(float)
dados_df.head()

# %%
# Cria um novo dataframe contendo somente as colunas numéricas desejadas
dados_num_df = dados_df[[col for col in dados_df.columns if dados_df[col].dtype !=
'object']]
dados_num_df.drop('year_of_reference', axis=1, inplace=True)
```

```
dados_num_df.head()

# %%
# Gera o Mapa de Correlação das variáveis numéricas com a variável Target
sns.heatmap(dados_num_df.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```




```
# %% [markdown]
# ### b. Crie partições contendo 75% dos dados para treino e 25% para teste

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

# %%
# Variável Y contém apenas a variável target - Faixa Salarial
Y = dados_num_df['avg_price_brl']
Y.head()

# %%
# Divisão: 25% dos dados são de teste e 75% de treinamento
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25, \
                                                    random_state = 42)
```

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

# %%
# Observando os dados de teste
print(X_test.shape)
X_test.head(1)

# %% [markdown]
# ### c. Treine modelos RandomForest (biblioteca RandomForestRegressor) e XGBoost
(biblioteca XGBRegressor) para predição dos preços dos carros. Observação: caso
julgue necessário, mude os parâmetros dos modelos e rode novos modelos. Indique
quais parâmetros foram inputados e indique o treinamento de cada modelo

# %%
# Algoritmo Random Forest,
# sem especificar nenhum parâmetro (número de árvores, número de ramificações,
etc)
model_rf = RandomForestRegressor()
# Ajuste do modelo Random Forest, de acordo com as variáveis de treinamento
model_rf.fit(X_train, Y_train)

# %%
# Algoritmo XGBoost, sem especificar nenhum parâmetro
model_xgboost = XGBRegressor()
# Ajuste do modelo XGBoost, de acordo com as variáveis de treinamento
model_xgboost.fit(X_train, Y_train)

# %% [markdown]
# ### d. Grave os valores preditos em variáveis criadas

# %%
# Predição dos preços médios dos carros com base nos dados de teste pelo modelo
Random Forest
valores_preditos_rf = model_rf.predict(X_test)
# Predição dos valores de salário com base nos dados de teste pelo modelo XGBoost
valores_preditos_xgboost = model_xgboost.predict(X_test)

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

# %%
# Analisando a importância das variáveis independentes
# para estimar a variável target no modelo Random Forest
model_rf.feature_importances_
feature_importances_RF = pd.DataFrame(model_rf.feature_importances_, \
                                      index = X_train.columns, \
                                      columns=['importance']) \
                           .sort_values('importance', ascending = False)
feature_importances_RF
```

```
                 importance
engine_size_num    0.498704
year_model         0.424089
gear_num           0.041081
fuel_num           0.036126

# %%
# Analisando a importância das variáveis independentes para estimar a variável
target no modelo XGBoost
model_xgboost.feature_importances_
feature_importances_XB = pd.DataFrame(model_xgboost.feature_importances_, \
                                      index = X_train.columns, \
                                      columns=['importance']) \
                           .sort_values('importance', ascending = False)
feature_importances_XB

                 importance
engine_size_num    0.397114
year_model         0.273311
fuel_num           0.165178
gear_num           0.164398

# %% [markdown]
# ### g. Escolha o melhor modelo com base nas métricas de avaliação MSE, MAE e R²

# %%
# Métricas de avaliação do modelo Random Forest
mse_RF = mean_squared_error(Y_test, valores_preditos_rf)
mae_RF = mean_absolute_error(Y_test, valores_preditos_rf)
r2_RF = r2_score(Y_test, valores_preditos_rf)
print(mse_RF)
print(mae_RF)
print(r2_RF)

191666537.16954657
7820.764541661718
0.9287817870391474

# %%
# Métricas de avaliação do modelo XGBoost
mse_XG = mean_squared_error(Y_test, valores_preditos_xgboost)
mae_XG = mean_absolute_error(Y_test, valores_preditos_xgboost)
r2_XG = r2_score(Y_test, valores_preditos_xgboost)
print(mse_XG)
print(mae_XG)
print(r2_XG)

191424816.71144435
7819.671020396782
0.928871603964503

# %% [markdown]
# *** Melhor modelo é o XGBoost, apesar de terem valores muito próximos
```

APÊNDICE C - LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

onde y_i é o valor observado, $\widehat{y_i}$ 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}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1 Pesquisa com Dados de Satélite (Satellite)

Foram testados os 3 modelos, *Random Forest*, *SVM* e *RNA*, utilizando código R para gerar as tabelas com os resultados e as matrizes de confusão em formato *markdown* (*.md*). Para isso, após o carregamento dos dados e aplicação dos modelos, foi feita uma função para gerar os arquivos utilizando a biblioteca *knitr*. Os resultados apresentados foram os seguintes:

RANDOM FOREST

Para o modelo de Random Forest, foi obtida a seguinte Matriz de Confusão:

	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble	very damp grey soil
red soil	297	0	3	2	6	0
cotton crop	1	126	0	2	4	0
grey soil	3	0	252	28	2	13
damp grey soil	0	1	12	51	1	35
vegetation stubble	5	12	0	1	116	15
very damp grey soil	0	1	4	41	12	238

A Acurácia deste modelo foi de 0.8411215 (84,11%).

SVM

Para o modelo de SVM, foi obtida a seguinte Matriz de Confusão:

	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble	very damp grey soil
red soil	304	0	2	5	9	1
cotton crop	0	126	0	0	1	0
grey soil	1	0	262	27	1	16
damp grey soil	0	0	7	60	0	29
vegetation stubble	1	13	0	0	116	11
very damp grey soil	0	1	0	33	14	244

A Acurácia deste modelo foi de 0.8660436 (86,6%).

RNA

Para o modelo de RNA, foi obtida a seguinte Matriz de Confusão:

	red soil	cotton crop	grey soil	damp grey soil	vegetation stubble	very damp grey soil
red soil	289	3	4	5	21	1
cotton crop	0	123	0	0	47	0
grey soil	3	0	263	39	1	18
damp grey soil	0	0	4	30	1	18
vegetation stubble	14	12	0	0	32	4
very damp grey soil	0	2	0	51	39	260

A Acurácia deste modelo foi de 0.7764798 (77,64%).

CONCLUSÃO

De acordo com os resultados obtidos, o melhor modelo para aplicar nesta base de dados foi o SVM, por apresentar a Acurácia mais elevada.

CÓDIGO

```
# install packages
install.packages("e1071")
install.packages("randomForest")
install.packages("kernlab")
install.packages("mlbench")
install.packages('caret')
install.packages('knitr') # markdown table

# load libraries
library('mlbench')
library('caret')
library('knitr')

# seed for reproduction
seed <- 4
print(paste('Setting seed to ', seed))
set.seed(seed)

# prepare data
print('Preparing dataset...')
data(Satellite)
database <- Satellite
print('Creating data partition...')
indexes <- createDataPartition(database$classes, p=0.80, list=F)
train <- database[indexes,]
test <- database[-indexes,]
```

```
# train models
print('Training models...')
formula <- (classes ~ x.17 + x.18 + x.19 + x.20)
print(' -> Random Forest...')
rf <- train(formula, data=train, method='rf')
print(' -> SVM...')
svm <- train(formula, data=train, method='svmRadial')
print(' -> RNA...')
rna <- train(formula, data=train, method='nnet', trace=F)

# predict results
print('Predicting results...')
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.rna <- predict(rna, test)

# function to generate the markdown files
generatePresentation <- function (predicted, testData, modelName, fileName) {
# generate confusion matrix
cm <- confusionMatrix(predicted, testData$classes)

# covert confusion matrix to a printable format
matrixTable <- paste(knitr::kable(cm$table, 'pipe'), collapse='\n')

# get overall results in printable format
overall <- paste(knitr::kable(as.data.frame(cm$overall), 'pipe',
col.names=c('Propriedade', 'Valor')), collapse='\n')

# generate text
text <- paste(c("# ", toupper(modelName), "\n\n## Matriz de Confusão\n\n",
matrixTable, "\n\n## Resultados\n\n", overall), collapse='')

# create file
write.table(text, fileName, quote=F, row.names=F, col.names=F)
}

# generate for each model
print('Generating presentation...')
generatePresentation(predict.rf, test, 'random forest', 'iaa3-1-rf.md')
generatePresentation(predict.svm, test, 'svm', 'iaa3-1-svm.md')
generatePresentation(predict.rna, test, 'rna', 'iaa3-1-rna.md')
print('Done.')
```

2 Estimativa de Volumes de Árvores

Foram utilizados códigos em R para gerar uma tabela contendo os resultados de todos os modelos, da mesma forma que no exercício anterior, utilizando o pacote *knitr* para salvar os resultados em formato *markdown* (*.md*). Porém, neste caso não há matrizes de confusão, visto que se trata de um problema de **Regressão**, e não Classificação.

Os resultados podem ser visualizados na tabela abaixo:

	Random.Forest	SVM	Redes.Neurais	Modelo.Alométrico.de.SPURR
R^2	0.8098753	0.7099086	0.8525215	0.8147245
Erro Padrão da Estimativa	0.1637666	0.2022896	0.1442349	0.1616646
Percentual de Erro Padrão da Estimativa	12.3738611	15.2845763	10.8980888	12.2150426

CONCLUSÃO

De acordo com esses resultados, o melhor modelo seria a Rede Neural Artificial, visto que possui o Coeficiente de Determinação (R^2) mais próximo de 1, e o Erro Padrão da Estimativa (Syx) e Percentual de Erro Padrão da Estimativa (Syx%) mais próximos de zero, sendo o melhor nos 3 parâmetros.

CÓDIGO

```
# install packages
install.packages("randomForest")
install.packages("kernlab")
install.packages('e1071')
install.packages('caret')
install.packages('knitr')
# load libraries
library(caret)
library(knitr)
# seed for reproduction
seed <- 8
print(paste('Setting seed to ', seed))
set.seed(seed)

# read dataset file
dataset <- read.csv2('http://www.razer.net.br/datasets/Volumes.csv', header=T,
dec=',', sep=';')
```

```
# prepare data
print('Preparing dataset...')
dataset$NR <- NULL
print('Creating data partition...')
indexes <- createDataPartition(y=dataset$VOL, p=0.80, list=F)
test <- dataset[-indexes,]

# train models
print('Training models...')
print(' -> Random Forest...')
rf <- caret::train(VOL ~ DAP * DAP * HT, data=train, method='rf')
print(' -> SVM...')
svm <- caret::train(VOL~ DAP * DAP * HT, data=train, method='svmRadial')
print(' -> RNA...')
rna <- caret::train(VOL~ DAP * DAP * HT, data=train, method='nnet', linout=T)
print(' -> SPURR...')
spurr <- nls(VOL ~ b0 + b1 * DAP * DAP * HT, train, start=list(b0=0.5, b1=0.5))

# predict results
print('Predicting results...')
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.rna <- predict(rna, test)
predict.spurr <- predict(spurr, test)

# generate metrics
r2 <- function(yr, yp) {
return ( 1 - ( sum( (yr - yp) ^ 2 ) / sum( (yr - mean(yr)) ^ 2 ) ) )
}

erroPadraoEstimativa <- function(yr, yp) {
n <- length(yr)
return ( sqrt( sum( (yr - yp) ^ 2 ) / (n - 2) ) )
}

erroPadraoEstimativaPerc <- function(yr, yp) {
return ( (erroPadraoEstimativa(yr, yp) / mean(yr)) * 100 )
}

print('Generating metrics...')
yr <- test$VOL
results <- data.frame(
'Random Forest'=c(r2(yr, predict.rf), erroPadraoEstimativa(yr, predict.rf),
erroPadraoEstimativaPerc(yr, predict.rf)),
'SVM'=c(r2(yr, predict.svm), erroPadraoEstimativa(yr,  predict.svm),
erroPadraoEstimativaPerc(yr, predict.svm)),
'Redes Neurais'=c(r2(yr, predict.rna), erroPadraoEstimativa(yr, predict.rna),
erroPadraoEstimativaPerc(yr, predict.rna)),
'Modelo Alométrico de SPURR'=c(r2(yr, predict.spurr), erroPadraoEstimativa(yr,
predict.spurr), erroPadraoEstimativaPerc(yr, predict.spurr)),
row.names = c('R²', 'Erro Padrão da Estimativa', 'Percentual de Erro Padrão da
Estimativa')
)

print('Generating presentation...')
table <- knitr::kable(results, 'pipe')
write.table(table, 'iaa3-2.md', quote=F, row.names=F, col.names=F)
print('Done.')
```

APÊNDICE D - ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

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

2) Medidas de posição e dispersão

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

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

3) Testes paramétricos ou não paramétricos

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

Obs:

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

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

B – RESOLUÇÃO

1. Gráficos e tabelas

a)

De acordo com os resultados apresentados no Gráfico 1, o boxplot da idade das esposas varia de 18 a 59 anos, enquanto o dos maridos varia de 19 a 86 anos. A linha da mediana indica 39 anos nas esposas e 41 anos nos maridos, sendo a

mediana o segundo quartil, ou seja, o valor que divide os dados ao meio. Observa-se também que, apesar do intervalo interquartil ser bem próximo pelo tamanho das caixas, a dispersão das idades dos maridos é maior que a das esposas, apresentando inclusive outliers na amostra. Com relação aos histogramas (gráficos 2 e 3), a maioria das idades das esposas situa-se entre 25 e 65 anos, enquanto as dos maridos situa-se entre 25 e 60 anos. As curvas, porém, apresentam diferenças de perfil, sendo a dos maridos mais leptocúrtica e assimétrica à esquerda e a das esposas mais platicurtica e assimétrica à direita.

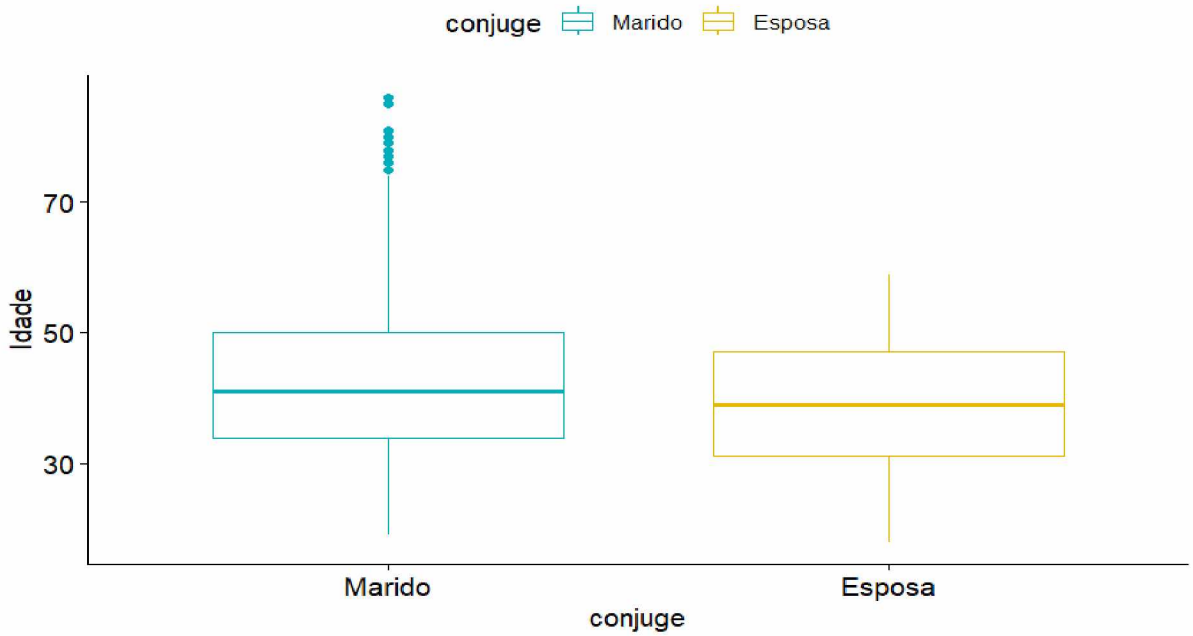


Gráfico 1 – Boxplot das variáveis age e husage

Gráfico 2 – Distribuição Normal x variável husage

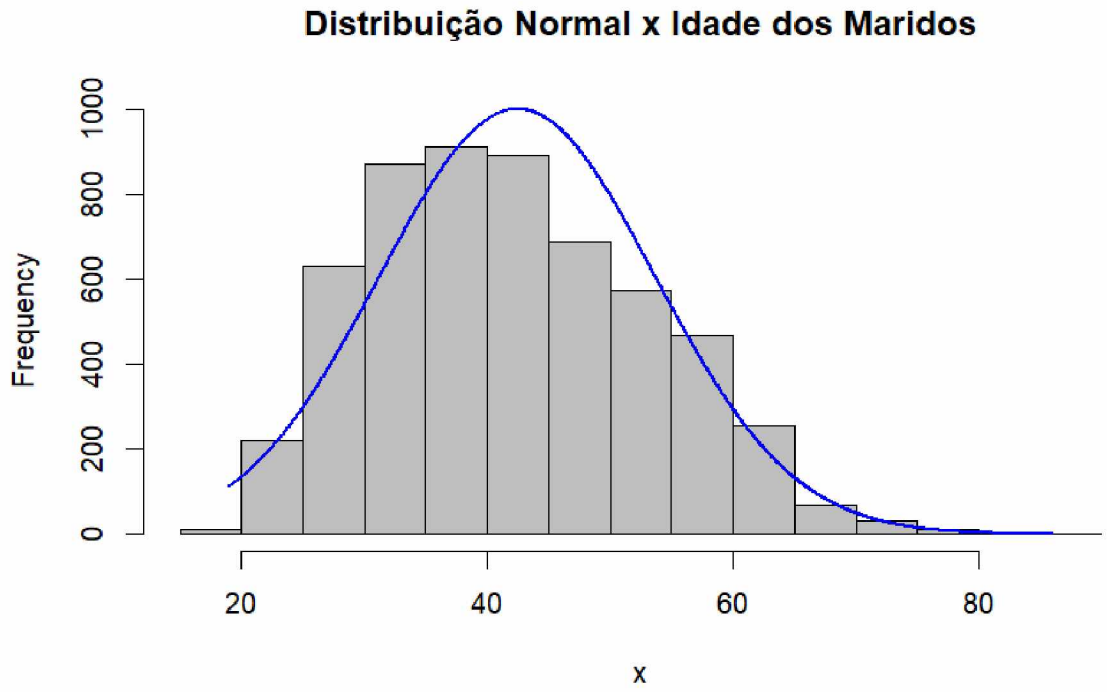
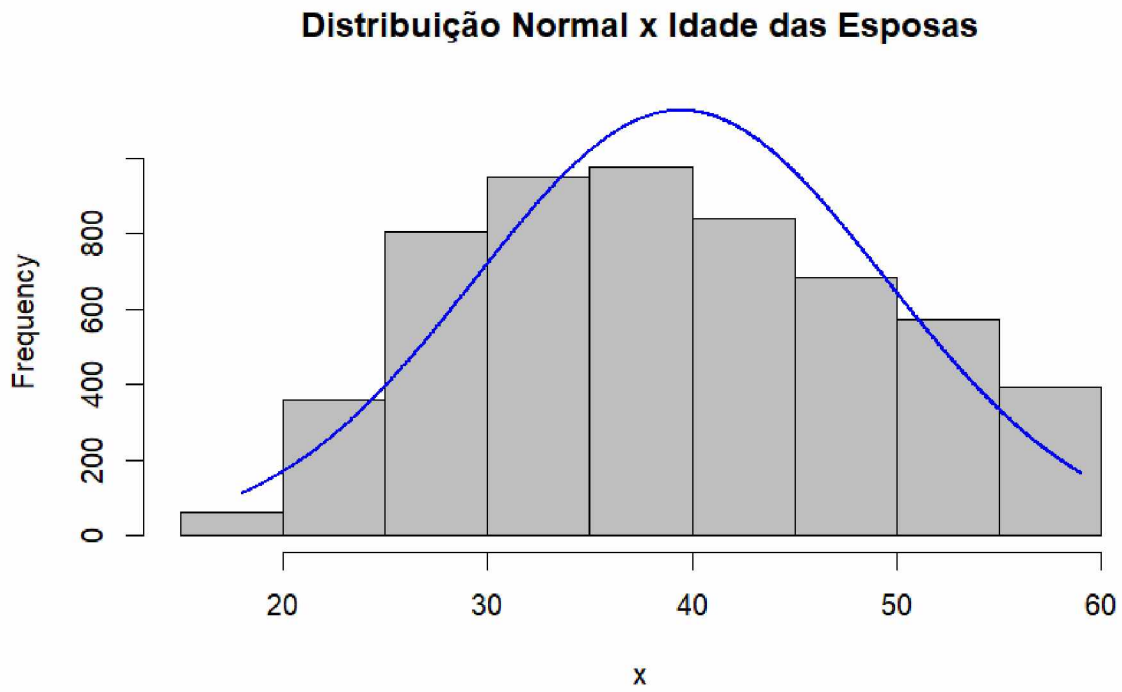


Gráfico 3 – Distribuição Normal x variável age



CÓDIGO

```
# CARREGA AS BIBLIOTECAS NECESSARIAS
library(car)
library(fdth)
library(nortest)
library(DescTools)
library(rcompanion)

options(scipen=999)

# CARREGA A BASE DE DADOS UTILIZADA NO TRABALHO
load('salarios.RData')

# CRIA UM DATAFRAME SOMENTE COM OS DADOS DAS IDADES EM DOIS GRUPOS:
# MARIDOS E ESPOSAS, PARA FACILITAR AS ANÁLISES
idades <- data.frame(
  conjuge = rep(c("Marido", "Esposa"), each = 5634),
  idade = c(salarios$husage,  salarios$age)
)

#Plota os boxplots de ambos os grupos juntos
ggboxplot(idades, x = "conjuge", y = "idade",
          color = "conjuge", palette=c("#00AFBB", "#E7B800"),
          ylab = "Idade", xlab = "conjuge")

# Plota o histograma das idades dos Maridos
plotNormalHistogram(idades$idade[idades$conjuge=="Marido"], prob = FALSE,
                    main = "Distribuição Normal x Idade dos Maridos",
                    length = 1000 )

# Plota o histograma das idades das Esposas
plotNormalHistogram(idades$idade[idades$conjuge=="Esposa"], prob = FALSE,
                    main = "Distribuição Normal x Idade das Esposas",
                    length = 1000 )
```

b)

De acordo com os resultados apresentados nas Tabelas 1 e 2, as idades das esposas situam-se na maioria entre 25 e 55 anos, enquanto as dos maridos estão entre 20 e 60 anos. A frequência acumulada, para as idades das esposas na faixa de 25 a 30 anos é de 18,44% e ao atingir a faixa de 50 a 55 é 91,16% dos dados do grupo. Na idade dos maridos, a frequência acumulada na faixa de 25 a 30 anos é de 12,51% e ao atingir a faixa de 65 a 70 anos é 99,06% dos dados desse grupo. A classe de 35 a 40 anos é a que tem maior frequência relativa 17,43% (982 esposas) para as esposas e nos maridos é a classe de 40 a 45 anos com 16,29% (918 maridos).

Tabela 1 – Tabela de frequências da idade das esposas (*age*), em anos.

Classes	Frequência absoluta	Frequência relativa	Frequência relativa (%)	Frequência acumulada	Frequência acumulada (%)
[15,20)	30	0,005	0,53	30	0,53
[20,25)	276	0,0489	4,90	306	5,43
[25,30)	733	0,1301	13,01	1039	18,44
[30,35)	946	0,1679	16,79	1985	35,23
[35,40)	982	0,1742	17,43	2967	52,66
[40,45)	881	0,1563	15,64	3848	68,30
[45,50)	688	0,1221	12,21	4536	80,51
[50,55)	600	0,1064	10,65	5136	91,16
[55,60)	498	0,0883	8,84	5634	100,00

Tabela 2 – Tabela de frequências da idade dos maridos (husage), em anos

Classes	Frequência absoluta	Frequência relativa	Frequência relativa (%)	Frequência acumulada	Frequência acumulada (%)
[15,20)	5	0,0008	0,09	5	0,09
[20,25)	168	0,0298	2,98	173	3,07
[25,30)	532	0,0944	9,44	705	12,51
[30,35)	868	0,1540	15,40	1573	27,92
[35,40)	899	0,1595	15,96	2472	43,87
[40,45)	918	0,1629	16,29	3390	60,17
[45,50)	710	0,1260	12,60	4100	72,77
[50,55)	573	0,1017	10,17	4673	82,94
[55,60)	512	0,0908	9,09	5185	92,03
[60,65)	307	0,0544	5,45	5492	97,48
[65,70)	89	0,0157	1,58	5581	99,06
[70,75)	32	0,0056	0,568	5613	99,63
[75,80)	17	0,0030	0,308	5630	99,93
[80,85)	2	0,00035	0,0358	5632	99,96
[85,90)	2	0,00035	0,0358	5634	100,00

CÓDIGO

```
# Verificando o range das idades
with(idades,
     range(idade[conjuge=="Marido"])
)
[1] 19 86

with(idades,
     range(idade[conjuge=="Esposa"])
)
[1] 18 59
```

```
# Montando as tabelas
ft_marido <- with(idades,
                  fdt(idade[conjuge=="Marido"],
                      start = 15, end = 90, h = 5
                      )
                  )
ft_marido
Class limits   f   rf rf(%)   cf  cf(%)
      [15,20)   5 0.00  0.09    5   0.09
      [20,25) 168 0.03  2.98  173   3.07
      [25,30) 532 0.09  9.44  705  12.51
      [30,35) 868 0.15 15.41 1573  27.92
      [35,40) 899 0.16 15.96 2472  43.88
      [40,45) 918 0.16 16.29 3390  60.17
      [45,50) 710 0.13 12.60 4100  72.77
      [50,55) 573 0.10 10.17 4673  82.94
      [55,60) 512 0.09  9.09 5185  92.03
      [60,65) 307 0.05  5.45 5492  97.48
      [65,70)  89 0.02  1.58 5581  99.06
      [70,75)  32 0.01  0.57 5613  99.63
      [75,80)  17 0.00  0.30 5630  99.93
      [80,85)   2 0.00  0.04 5632  99.96
      [85,90)   2 0.00  0.04 5634 100.00

ft_esposa <- with(idades,
                  fdt(idade[conjuge=="Esposa"],
                      start = 15, end = 60, h = 5
                  )
)
ft_esposa
Class limits   f   rf rf(%)   cf  cf(%)
      [15,20)  30 0.01  0.53   30   0.53
      [20,25) 276 0.05  4.90  306   5.43
      [25,30) 733 0.13 13.01 1039  18.44
      [30,35) 946 0.17 16.79 1985  35.23
      [35,40) 982 0.17 17.43 2967  52.66
      [40,45) 881 0.16 15.64 3848  68.30
      [45,50) 688 0.12 12.21 4536  80.51
      [50,55) 600 0.11 10.65 5136  91.16
      [55,60) 498 0.09  8.84 5634 100.00
```

2. Medidas de posição e dispersão

a)

Para os grupos analisados da amostra, a média da idade dos maridos (42,5 anos) é 7,87% maior que a das esposas (39,4 anos), enquanto que a mediana da idade dos maridos (41 anos) é 5,13% maior que a das esposas (39 anos) e, finalmente, a moda da idade dos maridos (44 anos, em 201 maridos) é 18,92% maior que a das esposas (37 anos em 217 esposas).

CÓDIGO

```
group_by(idades, conjuge) %>%
  summarise(
    count = n(),
    mean = mean(idade, na.rm = TRUE),
    median = median(idade, na.rm = TRUE),
    var = var(idade, na.rm = TRUE),
    sd = sd(idade, na.rm = TRUE),
    c.var = sd(idade, na.rm = TRUE)/mean(idade, na.rm = TRUE)*100,
    IQR = IQR(idade, na.rm = TRUE)
  )
# A tibble: 2 × 8
  conjuge count  mean median   var    sd c.var   IQR
  <chr>   <int> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
1 Esposa   5634  39.4     39  99.8  9.99  25.3    16
2 Marido   5634  42.5     41 126.  11.2   26.4    16

with(idades,
     subset(table(idade[conjuge=="Esposa"]),
            table(idade[conjuge=="Esposa"])==max(table(idade[conjuge=="Esposa"]))
            )
     )
 37
217

with(idades,
     subset(table(idade[conjuge=="Marido"]),
            table(idade[conjuge=="Marido"])==max(table(idade[conjuge=="Marido"]))
     )
)
 44
201

dif_media <- ((42.5/39.4)-1)*100
dif_media
[1] 7.86802

dif_mediana <- ((41/39)-1)*100
dif_mediana
[1] 5.128205

dif_moda <- ((44/37)-1)*100
dif_moda
[1] 18.91892
```

b)

Para os grupos analisados da amostra, a variância da idade dos maridos (126 anos²) é 26,25% maior que a das esposas (99,8 anos²), enquanto o desvio padrão da idade dos maridos (11,2 anos) foi 12,11 % maior que o das esposas (9,99 anos). No coeficiente de variação, o da idade dos maridos também foi maior que o das esposas com 26,4 % contra 25,3%.

CÓDIGO

```
group_by(idades, conjuge) %>%
  summarise(
    count = n(),
    mean = mean(idade, na.rm = TRUE),
    median = median(idade, na.rm = TRUE),
    var = var(idade, na.rm = TRUE),
    sd = sd(idade, na.rm = TRUE),
    c.var = sd(idade, na.rm = TRUE)/mean(idade, na.rm = TRUE)*100,
    IQR = IQR(idade, na.rm = TRUE)
  )
# A tibble: 2 × 8
  conjuge count  mean median   var    sd c.var   IQR
  <chr>   <int> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
1 Esposa   5634  39.4     39  99.8  9.99  25.3    16
2 Marido   5634  42.5     41 126.  11.2   26.4    16

dif_VAR <- ((126/99.8)-1)*100
dif_VAR
[1] 26.25251

dif_DP <- ((11.2/9.99)-1)*100
dif_DP
[1] 12.11211
```

3. Testes paramétricos ou não paramétricos

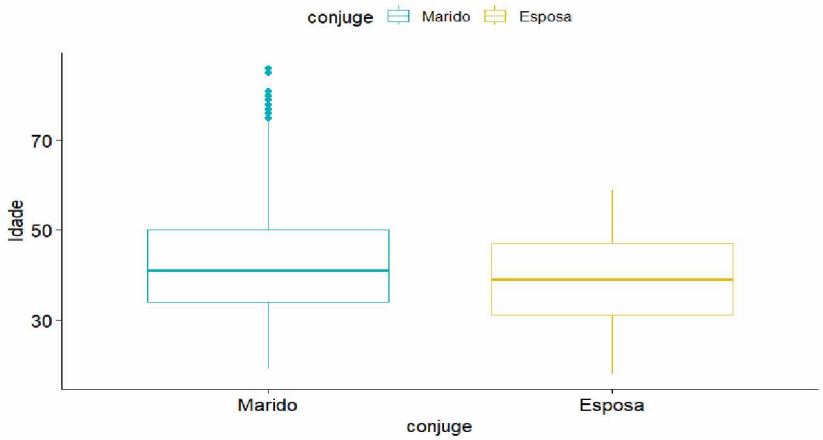
a)

Para esse caso, devem ser efetuados os testes de premissa para saber se deve-se utilizar um teste paramétrico ou não paramétrico. Após a constatação da independência das amostras, foi efetuado o teste de normalidade dos dados dos grupos, usando-se o teste de Jarque Berra, que mostrou que os dados dos mesmos não seguiam uma distribuição normal e, portanto, dever-se-ia utilizar um teste não-paramétrico; sendo o indicado, o Mann-Whitney "U" test, já que os dados dos grupos são independentes e não pareados.

O resultado do teste comprovou que realmente, a mediana das idades das esposas é estatisticamente diferente e menor do que a mediana das idades dos maridos.

CÓDIGO

```
# Efetuando os testes das premissas para identificar se utiliza o teste
# paramétrico ou o não-paramétrico

# Premissa 1: Independência das amostras.
# R: As amostras são independentes pois os grupos não se relacionam, ou seja,
#    não são amostras emparelhadas.

# Premissa 2: Os dados das amostras/grupos possuem distribuição normal.
# Utilizar o teste de normalidade de Jarque Bera pois é uma a amostra grande.
with(idades,
     JarqueBeraTest(idade[conjuge=="Esposa"],
                    robust = TRUE
                    )
)
       Robust Jarque Bera Test

data:  idade[conjuge == "Esposa"]
X-squared = 158.49, df = 2, p-value < 0.00000000000000022

with(idades,
     JarqueBeraTest(idade[conjuge=="Marido"],
                    robust = TRUE
     )
)
       Robust Jarque Bera Test

data:  idade[conjuge == "Marido"]
X-squared = 153.12, df = 2, p-value < 0.00000000000000022

# O p-value de ambos os grupos é menor que 0.05. Ambas as amostras não seguem
# uma distribuição normal.
# PORTANTO, DEVERÁ SER UTILIZADO UM TESTE NÃO PARAMÉTRICO: Mann-Whitney "U" test

# Testando se a mediana da idade dos maridos e esposas são estatisticamente iguais
# Vendo novamente o resumo e os boxplots
group_by(idades, conjuge) %>%
  summarise(
    count = n(),
    mean = mean(idade, na.rm = TRUE),
    median = median(idade, na.rm = TRUE),
    var = var(idade, na.rm = TRUE),
    sd = sd(idade, na.rm = TRUE),
    c.var = sd(idade, na.rm = TRUE)/mean(idade, na.rm = TRUE)*100,
    IQR = IQR(idade, na.rm = TRUE)
  )
# A tibble: 2 × 8
  conjuge count  mean median   var    sd c.var   IQR
  <chr>   <int> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
1 Esposa   5634  39.4     39  99.8  9.99  25.3    16
2 Marido   5634  42.5     41 126.  11.2   26.4    16

ggboxplot(idades, x = "conjuge", y = "idade",
          color = "conjuge", palette=c("#00AFBB", "#E7B800"),
          ylab = "Idade", xlab = "conjuge")
```




```
# Hipoteses do teste:
#  -H0: A idade mediana dos maridos é estatisticamente igual a das esposas;
#  -Ha: A idade mediana dos maridos não é estatisticamente igual a das esposas;
wilcox.test(idade ~ conjuge, data = idades,
            exact = FALSE,
            conf.int=TRUE)
```

```
       Wilcoxon rank sum test with continuity correction

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

```
# O p-value < 0.05, portanto as medianas são diferentes.
# O intervalo de confianca da diferenca entre as medianas esta
# entre aprox. -3 e -2, com uma mediana de aprox. -3.

# Testando se a mediana da idade das esposas é menor que a dos maridos
# Hipoteses do teste:
#  -H0: A idade mediana das esposas não é menor que a dos maridos;
#  -Ha: A idade mediana das esposas é menor que a dos maridos;
wilcox.test(idade ~ conjuge, data = idades,
            exact = FALSE,
            alternative='less',
            conf.int=TRUE)
```

```
       Wilcoxon rank sum test with continuity correction

data:  idade by conjuge
W = 13619912, p-value < 0.00000000000000022
alternative hypothesis: true location shift is less than 0
95 percent confidence interval:
      -Inf -2.000046
sample estimates:
difference in location
             -2.999966
```

```
# O p-value < 0.05, portanto H0 é rejeitada.
# A idade mediana das esposas é menor que a dos maridos.
# O intervalo de confianca da diferenca entre as medianas é um valor menor
# que aprox. -2, com uma mediana de aprox. -3.
# Testando se a mediana da idade das esposas é maior que a dos maridos
# Hipoteses do teste:
#  -H0: A idade mediana das esposas não é maior que a dos maridos;
#  -Ha: A idade mediana das esposas é maior que a dos maridos;
wilcox.test(idade ~ conjuge, data = idades,
            exact = FALSE,
            alternative='greater',
            conf.int=TRUE)

       Wilcoxon rank sum test with continuity correction

data:  idade by conjuge
W = 13619912, p-value = 1
alternative hypothesis: true location shift is greater than 0
95 percent confidence interval:
 -3.000034       Inf
sample estimates:
difference in location
             -2.999966

# O p-value > 0.05, portanto H0 não é rejeitada.
# A idade mediana das esposas não é maior que a dos maridos.
# O intervalo de confianca da diferenca entre as medianas é um valor maior
# que aprox. -3, com uma mediana de aprox. -3.
```

APÊNDICE E - ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

Regressão Ridge, Lasso e Elastic Net

Na primeira parte do código, são carregados os pacotes necessários para a análise dos modelos de regressão Ridge, Lasso e Elastic-Net.

Na segunda parte, busca-se os dados no R, faz-se a leitura das primeiras linhas dos dados, a separação em dados de treino (80%) e teste (20%) e a definição da variável resposta (dependente) e as variáveis independentes.

Na terceira seção de código, são analisados os modelos de regressão Ridge, Lasso e Elastic-Net. Em média, o Modelo de Regressão Ridge apresentou um R2 de 69,71% sendo o total da variabilidade do salário-hora da esposa em logaritmo com base nas demais variáveis de entrada, explicado pelo modelo e o seu RMSE foi de 0,2903796. Com relação ao modelo de regressão Lasso, ele apresentou um R2 de 69,90% sendo o total da variabilidade do salário-hora da esposa em logaritmo com base nas demais variáveis de entrada, explicado pelo modelo e o seu RMSE foi de 0,2887303. Finalmente, o Modelo de Regressão Elastic-Net apresentou um R2 de 69,85% sendo o total da variabilidade do salário-hora da esposa em logaritmo com base nas demais variáveis de entrada, explicado pelo modelo e o seu RMSE foi de 0,2889146. Dos três modelos avaliados, o Modelo de Regressão Lasso apresenta o menor valor de RMSE e maior valor R2 (coeficiente de determinação).

Na quarta seção do código, são gerados os coeficientes de regressão das demais variáveis de entrada para o Modelo de Regressão Lasso e indicadas as mais significativas.

Finalmente, na quinta seção do código, é gerada a predição para o valor do salário-hora da esposa (lwage) baseado nos valores solicitados pelo problema para as demais variáveis, apresentando um resultado de 1.586443 para o valor ainda em logaritmo e de 4.886336 após o antilog. Também são calculados os valores inferior e superior (já sem logaritmo) para um intervalo de confiança da predição de 95%, que são respectivamente 4.285364 e 5.487307.

Conclusão:

Tabela 1 – Valores de RMSE e R2 para cada método de regressão

Método	**RMSE**	**R2**
Ridge	0,2903796	0,6971443
Lasso	0,2887303	0,6990573
Elastic Net	0,2889146	0,6985350

Foram ajustados três métodos de Regressão Ridge, Lasso e Elastic-Net. Conforme a Tabela 1 acima, o melhor modelo de regressão ajustado para a variável dependente “lwage”, que apresentava o valor em logaritmo, foi o Lasso. Aplicando-se

a exponencial no valor ajustado, obteve-se uma estimativa de 4,89 com um intervalo de confiança de 95%variando de 4,29 a 5,49. As rotinas dos programas em R estão descritas abaixo passo a passo.

CÓDIGO

```
# Carrega os pacotes para Regressão Ridge/Lasso/Elastic-Net
library(glmnet)
require(dplyr)
library(tidyverse)
library(caret)
library(car)
library(lmtest)
library(olsrr)

# Buscando o conjunto de dados
load("trabalhosalarios.RData")
attach(trabalhosalarios)

# Lendo as primeiras linhas dos dados
head(trabalhosalarios)

   husage husunion husearns huseduc husblck hushisp hushrs kidge6 earns age black educ
3      56        0     1500      14       0       0     40      1   100  49     0   12
13     31        0      800      17       0       0     40      0   480  29     0   14
20     33        0      950      13       0       0     60      0   455  30     0   12
21     34        0     1000      14       0       0     50      1   102  31     0   12
22     42        0      730      14       0       0     40      1   300  41     0   12
25     45        0     1154      16       0       0     38      1   425  45     0   18
   hispanic union exper kidlt6    lwage
3         0     0    31      0 1.897120
13        0     0     9      0 2.484907
20        0     0    12      1 2.431418
21        0     0    13      0 1.629241
22        0     0    23      0 2.302585
25        0     0    21      0 2.496741

# Estrutura dos dados
str(trabalhosalarios)
'data.frame':    2574 obs. of  17 variables:
 $ husage  : num  56 31 33 34 42 45 33 31 31 44 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  1500 800 950 1000 730 ...
 $ huseduc : num  14 17 13 14 14 16 16 18 12 12 ...
 $ husblck : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushisp : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushrs  : num  40 40 60 50 40 38 40 55 40 40 ...
 $ kidge6  : num  1 0 0 1 1 1 0 0 0 1 ...
 $ earns   : num  100 480 455 102 300 425 770 125 245 539 ...
 $ age     : num  49 29 30 31 41 45 32 27 30 42 ...
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ educ    : num  12 14 12 12 12 18 12 14 15 12 ...
 $ hispanic: num  0 0 0 0 0 0 0 0 0 0 ...
 $ union   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ exper   : num  31 9 12 13 23 21 14 7 9 24 ...
 $ kidlt6  : num  0 0 1 0 0 0 0 1 1 0 ...
 $ lwage   : num  1.9 2.48 2.43 1.63 2.3 ...
 - attr(*, "na.action")= 'omit' Named int [1:3060] 1 2 4 5 6 7 8 9 10 11 ...
  ..- attr(*, "names")= chr [1:3060] "1" "2" "4" "5" ...
```

```
set.seed(7)
##Selecionando os dados de treino e teste
amostra<- sample(c(TRUE,FALSE), nrow(trabalhosalarios),
                 replace=TRUE, prob=c(0.8,0.2))
dados.treino<-trabalhosalarios[amostra, ]
dados.teste<-trabalhosalarios[!amostra, ]

# Estrutura dos dados
str(dados.treino)
```

```
'data.frame':    2050 obs. of  17 variables:
 $ husage  : num  31 33 34 42 45 33 31 44 45 22 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  800 950 1000 730 1154 ...
 $ huseduc : num  17 13 14 14 16 16 12 12 12 12 ...
 $ husblck : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushisp : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushrs  : num  40 60 50 40 38 40 40 40 50 40 ...
 $ kidge6  : num  0 0 1 1 1 0 0 1 0 0 ...
 $ earns   : num  480 455 102 300 425 770 245 539 300 299 ...
 $ age     : num  29 30 31 41 45 32 30 42 42 23 ...
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ educ    : num  14 12 12 12 18 12 15 12 12 13 ...
 $ hispanic: num  0 0 0 0 0 0 0 0 0 0 ...
 $ union   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ exper   : num  9 12 13 23 21 14 9 24 24 4 ...
 $ kidlt6  : num  0 1 0 0 0 0 1 0 0 0 ...
 $ lwage   : num  2.48 2.43 1.63 2.3 2.5 ...
 - attr(*, "na.action")= 'omit' Named int [1:3060] 1 2 4 5 6 7 8 9 10 11 ...
  ..- attr(*, "names")= chr [1:3060] "1" "2" "4" "5" ...
```

```
str(dados.teste)
```

```
'data.frame':    524 obs. of  17 variables:
 $ husage  : num  56 31 37 38 44 31 53 35 31 37 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  1500 769 465 240 1450 675 385 300 266 200 ...
 $ huseduc : num  14 18 12 18 18 16 12 9 13 12 ...
 $ husblck : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushisp : num  0 0 0 0 0 0 0 0 1 0 ...
 $ hushrs  : num  40 55 50 48 40 40 0 40 38 40 ...
 $ kidge6  : num  1 0 0 1 1 0 0 1 0 1 ...
 $ earns   : num  100 125 687 400 1100 434 576 88 200 286 ...
 $ age     : num  49 27 32 36 45 32 45 37 27 47 ...
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ educ    : num  12 14 17 16 18 14 12 12 12 12 ...
 $ hispanic: num  0 0 0 0 0 0 0 0 1 0 ...
 $ union   : num  0 0 1 0 0 0 0 0 0 0 ...
 $ exper   : num  31 7 9 14 21 12 27 19 9 29 ...
 $ kidlt6  : num  0 1 1 0 0 1 0 0 1 0 ...
 $ lwage   : num  1.9 1.78 2.84 2.3 3.31 ...
 - attr(*, "na.action")= 'omit' Named int [1:3060] 1 2 4 5 6 7 8 9 10 11 ...
  ..- attr(*, "names")= chr [1:3060] "1" "2" "4" "5" ...
```

```
set.seed(7)
# Buscando preditores (x e y - Iwage) no conjunto de treino e teste
x_treino <- dados.treino %>% select(-lwage) %>% as.matrix()
y_treino <- dados.treino %>% select(lwage) %>% as.matrix()
x_teste <- dados.teste %>% select(-lwage) %>% as.matrix()
```

```
y_teste <- dados.teste %>% select(lwage) %>% as.matrix()
# Validação cruzada para obter melhor valor de lambda para Regressão Ridg e (alpha
= 0)
cv_best_lambda <- cv.glmnet(x_treino, y_treino,
                            family = "gaussian", alpha = 0,
                            type.measure = "mse")
print(cv_best_lambda)

Call:  cv.glmnet(x = x_treino, y = y_treino, type.measure = "mse", family =
"gaussian",      alpha = 0)

Measure: Mean-Squared Error

     Lambda Index Measure       SE Nonzero
min 0.04177   100 0.07816 0.005153      16
1se 0.12755    88 0.08267 0.004807      16

best_lambda <-cv_best_lambda$lambda.min

# Modelo Regressão Ridge
ridge = glmnet(x_treino, y_treino, family = "gaussian",
               alpha = 0, lambda = best_lambda)


# Desempenho dos dados de treino na Regressão Ridge
treino_preditos <- ridge %>% predict(x_treino)
data.frame( R2 = R2(treino_preditos, y_treino),
            RMSE = RMSE(treino_preditos, y_treino))

              s0      RMSE
lwage 0.7063174 0.2760659

# Desempenho dos dados de teste na Regressão Ridge
teste_preditos <- ridge %>% predict(x_teste)
data.frame( R2 = R2(teste_preditos, y_teste),
            RMSE = RMSE(teste_preditos, y_teste))

              s0     RMSE
lwage 0.6411168 0.350727

# Validação cruzada Regressão Ridge
set.seed(7)
train.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)
# Modelo final Regressão Ridge
Ridge_modelo_cv <- train(lwage ~ ., data = trabalhosalarios,
                         method="glmnet",
                         trControl = train.control,
                         tuneGrid = expand.grid(alpha = 0,
                                                lambda = best_lambda))

# Resultados Modelo final Regressão Ridge
print(Ridge_modelo_cv)
```

```
glmnet

2574 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2317, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2903796  0.6971443  0.1886911

Tuning parameter 'alpha' was held constant at a value of 0
Tuning parameter
 'lambda' was held constant at a value of 0.04176709
```

```
# Validação cruzada para obter melhor valor de lambda para Regressão Lasso (alpha
= 1)
set.seed(7)
cv_best_lambda <- cv.glmnet(x_treino, y_treino,
                            family = "gaussian", alpha = 1,
                            type.measure = "mse")
print(cv_best_lambda)

Call:  cv.glmnet(x = x_treino, y = y_treino, type.measure = "mse", family =
"gaussian",      alpha = 1)

Measure: Mean-Squared Error

     Lambda Index Measure       SE Nonzero
min 0.00331    53 0.07738 0.005704      13
1se 0.05394    23 0.08302 0.004578       3

best_lambda <-cv_best_lambda$lambda.min

# Modelo Regressão Lasso
lasso = glmnet(x_treino, y_treino, family = "gaussian",
               alpha = 1, lambda = best_lambda)

# Desempenho dos dados de treino na Regressão Lasso
treino_preditos <- lasso %>% predict(x_treino)
data.frame( R2 = R2(treino_preditos, y_treino),
            RMSE = RMSE(treino_preditos, y_treino))

            s0      RMSE
lwage 0.707044 0.2745976

# Desempenho dos dados de teste na Regressão Lasso
teste_preditos <- lasso %>% predict(x_teste)
data.frame( R2 = R2(teste_preditos, y_teste),
            RMSE = RMSE(teste_preditos, y_teste))

             s0      RMSE
lwage 0.6439931 0.3461028
```

```
# Validação cruzada Regressão Lasso
set.seed(7)
train.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)
# Modelo final Regressão Lasso
Lasso_model_cv <- train(lwage ~ ., data = trabalhosalarios,
                        method="glmnet",
                        trControl = train.control,
                        tuneGrid = expand.grid(alpha = 1,
                                               lambda = best_lambda))

# Resultados Modelo final Regressão Lasso
print(Lasso_model_cv)


glmnet

2574 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2317, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2887303  0.6990573  0.1830369

Tuning parameter 'alpha' was held constant at a value of 1
Tuning parameter
 'lambda' was held constant at a value of 0.003309971

# Validação cruzada para Regressão Elastic Net
set.seed(7)
train.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3,
                              search = "random")
# Modelo de Regressão Elastic Net
cv_for_best_value <- train(lwage ~ ., data = dados.treino,
                           method="glmnet",
                           trControl = train.control)

# Obtendo melhor valor de alpha e lambda
cv_for_best_value$bestTune

       alpha     lambda
1 0.06974868 0.02091687

# Modelo Elastic Net
enet <- glmnet(x_treino, y_treino,
               alpha = 0.4089769,
               lambda = 0.001472246,
               family = "gaussian")

# Desempenho dos dados de treino na Regressão Elastic Net
treino_preditos <- enet %>% predict(x_treino)
data.frame( R2 = R2(treino_preditos, y_treino),
            RMSE = RMSE(treino_preditos, y_treino))
```

```
             s0      RMSE
lwage 0.7074028 0.2743984

# Desempenho dos dados de teste na Regressão Elastic Net
teste_preditos <- enet %>% predict(x_teste)
data.frame( R2 = R2(teste_preditos, y_teste),
                 RMSE = RMSE(teste_preditos, y_teste))

             s0     RMSE
lwage 0.6438279 0.345857

# Validação cruzada para Regressão Elastic Net
set.seed(7)
train.control <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 3)
# Modelo final Regressão Elastic Net
Enet_modelo_cv <- train(lwage ~ ., data = trabalhosalarios,
                        method="glmnet",
                        trControl = train.control,
                        tuneGrid = expand.grid(alpha = 0.4089769,
                                               lambda = 0.001472246))

# Resultados Modelo final Regressão Elastic Net
print(Enet_modelo_cv)

glmnet

2574 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2317, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.2889146  0.698535  0.1832527

Tuning parameter 'alpha' was held constant at a value of 0.4089769
Tuning
 parameter 'lambda' was held constant at a value of 0.001472246

#Resultados dos modelos
resultados <- data.frame(Metodo = c("Ridge", "Lasso", "Elastic Net"),
                         RMSE = c(0.2903796, 0.2887303, 0.2889146),
                         R2 = c(0.6971443, 0.6990573, 0.698535))
knitr::kable(resultados, align = "c", caption = "Métricas dos modelos")

Table: Métricas dos modelos

|   Metodo    |   RMSE    |    R2     |
|:-----------:|:---------:|:---------:|
|    Ridge    | 0.2903796 | 0.6971443 |
|    Lasso    | 0.2887303 | 0.6990573 |
| Elastic Net | 0.2889146 | 0.6985350 |
```

```
# Coeficientes do modelo de Regressão Lasso
lasso_coefficients<-lasso$beta
lasso_coefficients
```

```
16 x 1 sparse Matrix of class "dgCMatrix"
                    s0
husage    0.0012063446
husunion  .
husearns  0.0001123423
huseduc   0.0028789542
husblck  -0.0022099861
hushisp  -0.0026622297
hushrs   -0.0004161924
kidge6    .
earns     0.0015639836
age       0.0005813280
black    -0.0275079884
educ      0.0222733536
hispanic -0.0509049385
union     0.0539319167
exper     .
kidlt6    0.0509813432
```

```
# Identificando coeficientes significativos
significant_indices <- which(lasso_coefficients != 0)
significant_predictors <- rownames(lasso_coefficients)[significant_indices]
print(significant_predictors)
```

```
[1] "husage"   "husearns" "huseduc"  "husblck"  "hushisp"  "hushrs"   "earns"
[8] "age"      "black"    "educ"     "hispanic" "union"    "kidlt6"
```

```
#Valores para predição
dadospredicao = matrix(c(40,0,600,13,1,0,40,1,0,38,0,13,1,0,18,1),
                       nrow=1, ncol=16)
```

```
#Valor predito pelo modelo
valorpreditoIwage<-predict(lasso, s = best_lambda, newx = dadospredicao)
valorpreditoIwage
```

```
           s1
[1,] 1.586443
```

```
exp(valorpreditoIwage)
```

```
           s1
[1,] 4.886336
```

```
# Erro padrão
se <- sqrt(cv_best_lambda$cvm[cv_best_lambda$lambda == cv_best_lambda$lambda.min])
se
```

```
[1] 0.2781802
```

```
#Intervalo de confiança
alpha <- 0.05 # 95% confidence interval
```

```
# Construindo intervalo de confiança
lower_bound <- exp(valorpreditoIwage) - qt(1 - alpha / 2, lasso$df) * se
upper_bound <- exp(valorpreditoIwage) + qt(1 - alpha / 2,lasso$df) * se

# Resultado intervalo de confiança
cat("IC inferior:", lower_bound, "\n")
```

IC inferior: 4.285364

```
cat("IC superior:", upper_bound, "\n")
```

IC superior: 5.487307

APÊNDICE F - ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

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

Nessa atividade é para "construir características".

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

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

2 Melhore uma base de dados ruim

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

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

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

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

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

B – RESOLUÇÃO

1. Construção de Características: Identificador automático de idioma

CÓDIGO

```
# %% [markdown]
# <h1>Identificador automático de idioma</h1>
# <p><b>Problema</b>: Dados um texto de entrada, é possível identificar em qual
língua o texto está escrito?</p>
# <p>
# Entrada: "texto qualquer" <br />
# Saída: português ou inglês ou francês ou italiano ou...
# </p>
# <p> </p>
#
# <h2>O processo de Reconhecimento de Padrões</h2>
# <p>O objetivo desse trabalho é demonstrar o processo de "construção de
atributos" e como ele é fundamental para o <b>Reconhecimento de Padrões
(RP)</b>.</p>
#
# <p>Primeiro um conjunto de "amostras" previamente conhecido (classificado)</p>

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

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

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
"Preciso comprar leite e pão.",
"Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.",
"O trânsito está terrível hoje.",
"A comida estava deliciosa!",
"Você já visitou o Rio de Janeiro?",
"Tenho uma reunião importante amanhã.",
"A festa começa às 20h.",
"Estou cansado depois de um longo dia de trabalho.",
"Vamos fazer um churrasco no final de semana.",
"O livro que estou lendo é muito interessante.",
"Estou aprendendo a cozinhar pratos novos.",
"Preciso fazer exercícios físicos regularmente.",
"Vou viajar para o exterior nas férias.",
"Você gosta de dançar?",
"Hoje é meu aniversário!",
"Gosto de ouvir música clássica.",
"Estou estudando para o vestibular.",
"Meu time de futebol favorito ganhou o jogo.",
"Quero aprender a tocar violão.",
"Vamos fazer uma viagem de carro.",
"O parque fica cheio aos finais de semana.",
```

```
"O filme que assisti ontem foi ótimo.",
"Preciso resolver esse problema o mais rápido possível.",
"Adoro explorar novos lugares.",
"Vou visitar meus avós no domingo.",
"Estou ansioso para as férias de verão.",
"Gosto de fazer caminhadas na natureza.",
"O restaurante tem uma vista incrível.",
"Vamos sair para jantar no sábado.",
]

# %% [markdown]
# A "amostras" de texto precisa ser "transformada" em <b>padrões</b>
#
# <p>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 <b>classe</b></p>

# %%
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)

# %% [markdown]
# O DataFrame do pandas facilita a visualização.

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

# %% [markdown]
# <h1>Construção dos atributos</h1>
# <p>Esse é o coração desse trabalho e que deverá ser desenvolvido por vocês.
Pensem em como podemos "medir" cadas frase/sentença e extrair características que
melhorem o resultado do processo de identificação.<p>
# <p>Após a criação de cada novo atributo, execute as etapas seguintes e registre
as métricas da matriz de confusão. Principalmente acurácia e a precisão.</p>
```

```
# %%
# a entrada é o vetor pre_padroes e a saída desse passo deverá ser "padrões"
import re

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

def contarCaracteresEspecificos(texto):
    contar_ñ = texto.lower().count('ñ')
    contar_caracteres_esp = sum(texto.lower().count(char) for char in 'çàáéíóú')
    return contar_ñ,  contar_caracteres_esp

def contarPontuacao(texto):
    contar_pont = re.findall(r'[.,;!?]', texto)
    return len(contar_pont)

def contarDigitos(texto):
    digitos = sum(char.isdigit() for char in texto)
    return digitos

def contarPalavrasComuns(texto):
    palavras_comum_es = ['estoy', 'gusta', 'soy', 'quiero', 'hola', \
                         'tengo', 'hoy', 'dónde', 'playa','las', 'la', \
                         'alguna', 'soy', 'pero', 'mucho']
    palavras_comum_en = ['the', 'and', 'is', 'to', 'it', 'in', 'of', \
                         'that', 'as', 'was', 'love', 'you', 'where', \
                         'what', 'have', 'my', 'are']
    palavras_comum_pt = ['hoje', 'você', 'estou', 'vou',  'praia', \
                         'tenho', 'gosto', 'agora', 'ontem', 'novos', \
                         'muito' , 'sou', 'quero', 'mas']
    contar_en = sum(texto.lower().count(word) for word in palavras_comum_en)
    contar_es = sum(texto.lower().count(word) for word in palavras_comum_es)
    contar_pt = sum(texto.lower().count(word) for word in palavras_comum_pt)
    return contar_en, contar_es, contar_pt

def extraiCaracteristicas(frase):
    texto = frase[0]
    pattern_regex = re.compile('[^\w+]', re.UNICODE)
    texto = re.sub(pattern_regex, ' ', texto)
    caracteristica1 = tamanhoMedioFrases(texto)
    caracteristica2, caracteristica3 = contarCaracteresEspecificos(texto)
    caracteristica4 = contarPontuacao(texto)
    caracteristica5 = contarDigitos(texto)
    caracteristica6, caracteristica7, caracteristica8 =
contarPalavrasComuns(texto)
    padrao = [caracteristica1, caracteristica2, caracteristica3, \
              caracteristica4, caracteristica5, caracteristica6, \
              caracteristica7, caracteristica8, frase[1]]
    return padrao


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

```
  return padroes

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

#
# apenas para visualizacao
print(padroes)

dados = pd.DataFrame(padroes)
dados

# %% [markdown]
# <h2>Treinando o modelo com SVM</h1>
# <p>Separando o conjunto de treinamento do conjunto de testes</p>

# %%
from sklearn.model_selection import train_test_split
import numpy as np


#from sklearn.metrics import confusion_matrix

vet = np.array(padroes)
classes = vet[:,-1]         # classes = [p[-1] for p in padroes]
#print(classes)
padroes_sem_classe = vet[:,0:-1]
#print(padroes_sem_classe)
X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe, classes,
test_size=0.25, stratify=classes)


# %% [markdown]
# Com os conjuntos separados, podemos "treinar" o modelo usando a SVM.

# %%
from sklearn import svm
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report

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

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

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

#
# com dados de teste que não foram usados no treinamento
print('métricas mais confiáveis')
```

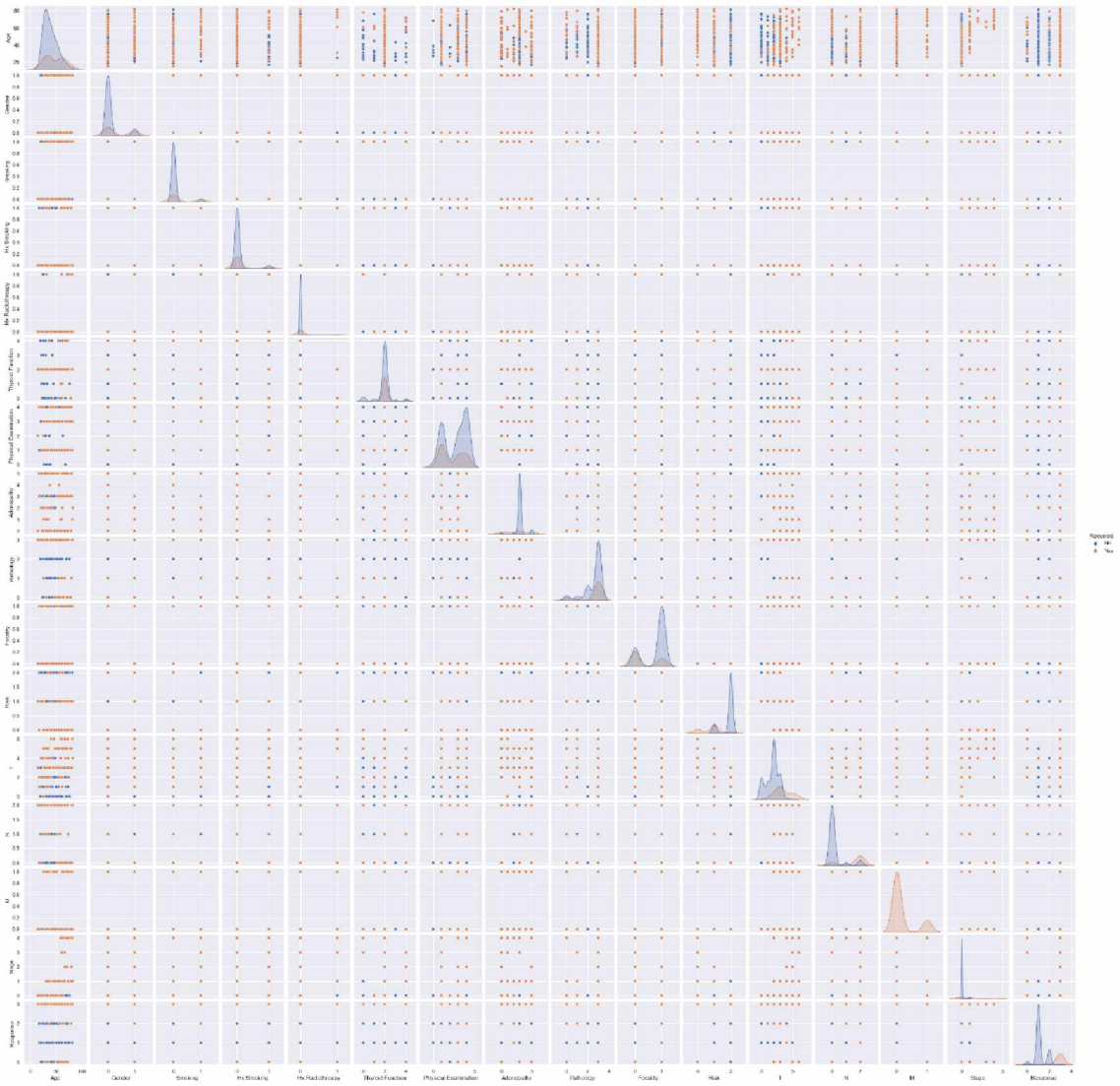
```
y_pred2 = modelo.predict(X_test)
cm = confusion_matrix(y_test, y_pred2)
print(cm)
print(classification_report(y_test, y_pred2))

Acurácia nos dados de treinamento: 76.81%
[[14  3  5]
 [ 2 19  2]
 [ 1  3 20]]
              precision    recall  f1-score   support

    espanhol       0.82      0.64      0.72        22
      inglês       0.76      0.83      0.79        23
   português       0.74      0.83      0.78        24

    accuracy                           0.77        69
   macro avg       0.77      0.77      0.76        69
weighted avg       0.77      0.77      0.77        69

métricas mais confiáveis
[[7 1 0]
 [0 7 0]
 [2 1 5]]
              precision    recall  f1-score   support

    espanhol       0.78      0.88      0.82         8
      inglês       0.78      1.00      0.88         7
   português       1.00      0.62      0.77         8

    accuracy                           0.83        23
   macro avg       0.85      0.83      0.82        23
weighted avg       0.86      0.83      0.82        23
```

2 Melhore uma base de dados ruim

CÓDIGO

```
# %% [markdown]
# <h1>Atividade 02 - melhorar o desempenho de RP em conjunto de dados
existentes</h1>
# <p>A atividade 02 visa trabalhar com um conjunto de dados pré-construído, onde
as opções que o desenvolvedor tem, são de aplicar as técnicas de pré-processamento
abaixo relacionadas:</p>
# <ul><li>Seleção</li>
# <li>Limpeza</li>
# <li>Codificação</li>
# <li>Enriquecimento</li>
# <li>Normalização</li>
# <li>Construção de Atributos</li>
# <li>Correção de Prevalência</li>
# <li>Partição do Conjunto de Dados</li>
# </ul>
# <p>Busque uma base de dados na UCI Machine Learning que seja indicada para
problemas de classificação. (<a target="_blank"
href="https://archive.ics.uci.edu/datasets">https://archive.ics.uci.edu/datasets</
a>)</p>
#
```

```
# %% [markdown]
# <p>Opção 01 - Carregada a base de dados para o computador local.</p>
# <p>OBS.: Para o desenvlvimento, foi utilizado o Visual Studio Code com a
extensão para Jupyter Notebooks em máquina local.</p>
# <p>A base utilizada foi a "Differentiated Thyroid Cancer Recurrence"</P>

# %%
import numpy as np
import pandas as pd
#
# base de dados disponível na UCI Machine Learning
# https://archive.ics.uci.edu/dataset/915/differentiated+thyroid+cancer+recurrence
thyro_colunas = ['Age', 'Gender', 'Smoking', 'Hx Smoking', 'Hx Radiothreapy', \
                 'Thyroid Function', 'Physical Examination', 'Adenopathy', \
                 'Pathology', 'Focality', 'Risk', 'T', 'N', 'M', 'Stage', \
                 'Response', 'Recurred']
thyro = pd.read_csv('Thyroid_Diff.csv',header=0,
                     names=thyro_colunas, lineterminator='\n', na_values='?')

# visualizar parte dos dados
print(thyro.head())
# Verifica se as classes estão balanceadas
thyro['Recurred'].value_counts(dropna=False) # As classes não estão balanceadas!

# %% [markdown]
# <h2>Necessário ajustar as features categóricas para valores numéricos
inteiros</h2>

# %%
from sklearn.preprocessing import LabelEncoder
num_cols = ['Age']
cat_cols = ['Gender', 'Smoking', 'Hx Smoking', 'Hx Radiothreapy', \
            'Thyroid Function', 'Physical Examination', 'Adenopathy', \
            'Pathology', 'Focality', 'Risk', 'T', 'N', 'M', 'Stage', 'Response']
tgt_cols = ['Recurred']

#Separa as features numéricas das categóricas e da feature target
num_cols_data = thyro[num_cols]
cat_cols_data = thyro[cat_cols]
tgt_cols_data = thyro[tgt_cols]

# Codifica as colunas categóricas usando o LabelEncder
cat_cols_num_data = cat_cols_data.apply(LabelEncoder().fit_transform)

#Remonta o Dataframe com os novos valores
ori_num_data = pd.concat([num_cols_data, cat_cols_num_data, tgt_cols_data],
                         axis=1)
print(ori_num_data.head())

# %% [markdown]
# <h2>Plots para ver a correlação 2 a 2 das features e se há separação clara de
classe em alguma</h2>

# %%
import seaborn as sns
sns.set()
sns.pairplot(ori_num_data, hue='Recurred', height=2)
# APARENTEMENTE NÃO HÁ NENHUMA CORRELAÇÃO LINEAR CLARA ENTRE AS FEATURES
```



```
# %% [markdown]
# <h2>Separar a classe dos atributos e efetuar os tratamentos para melhorar a performance do modelo</h2>

# %% [markdown]
# <p>Separa a classe do restante das features</p>

# %%
X = ori_num_data.iloc[:,:16]
cols = ori_num_data[:16]
print(X.head())
Y = ori_num_data['Recurred']
Y_orig = ori_num_data['Recurred']
print(Y.unique())
```

```
# %% [markdown]
# Normaliza a feature Age (Idade) e remove a feature M que aparenta estar
qualificando as classes, podendo gerar overfitting

# %%
from sklearn.preprocessing import scale
from sklearn.preprocessing import minmax_scale
import pandas as pd

X_orig =  X.copy()
print(Y_orig.unique() )
print(X_orig['M'].unique())

# normalização min-max na feature da Idade
X['Age'] = minmax_scale(X['Age'])
# remoção da feature M que praticamente qualifica a classe target de acordo com os
gráficos
X.drop(columns=['M'], axis=1, inplace=True)

print(X_orig.head())
print(X.head())

# %% [markdown]
# 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_orig_train, y_orig_test = train_test_split(X_orig,
                      Y_orig, test_size=0.25, stratify=Y_orig,random_state=10)

# com os dados tratados
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                                                    stratify=Y,random_state=10)

# %% [markdown]
# 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_orig_train)

# predição com os mesmos dados usados para treinar
y_orig_pred = modelo_orig.predict(X_oring_train)
cm_orig_train = confusion_matrix(y_orig_train, y_orig_pred)
print('Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO')
print(cm_orig_train)
print(classification_report(y_orig_train, y_orig_pred))
```

```
# predição com os mesmos dados usados para testar
print('Matriz de confusão - com os dados ORIGINAIS usados para TESTES')
y2_orig_pred = modelo_orig.predict(X_orig_test)
cm_orig_test = confusion_matrix(y_orig_test, y2_orig_pred)
print(cm_orig_test)
print(classification_report(y_orig_test, y2_orig_pred))
```

```
Matriz de confusão - com os dados ORIGINAIS usados no TREINAMENTO
[[204   2]
 [ 61  20]]
              precision    recall  f1-score   support

          No       0.77      0.99      0.87       206
         Yes       0.91      0.25      0.39        81

    accuracy                           0.78       287
   macro avg       0.84      0.62      0.63       287
weighted avg       0.81      0.78      0.73       287

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[67  2]
 [18  9]]
              precision    recall  f1-score   support

          No       0.79      0.97      0.87        69
         Yes       0.82      0.33      0.47        27

    accuracy                           0.79        96
   macro avg       0.80      0.65      0.67        96
weighted avg       0.80      0.79      0.76        96
```

```
# %% [markdown]
# Como os dados ficam 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
[[204   2]
 [ 13  68]]
              precision    recall  f1-score   support

          No       0.94      0.99      0.96       206
         Yes       0.97      0.84      0.90        81

    accuracy                           0.95       287
   macro avg       0.96      0.91      0.93       287
weighted avg       0.95      0.95      0.95       287

Matriz de confusão - com os dados ORIGINAIS usados para TESTES
[[67  2]
 [ 5 22]]
              precision    recall  f1-score   support

          No       0.93      0.97      0.95        69
         Yes       0.92      0.81      0.86        27

    accuracy                           0.93        96
   macro avg       0.92      0.89      0.91        96
weighted avg       0.93      0.93      0.93        96
```

APÊNDICE G - APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Seed: 202485

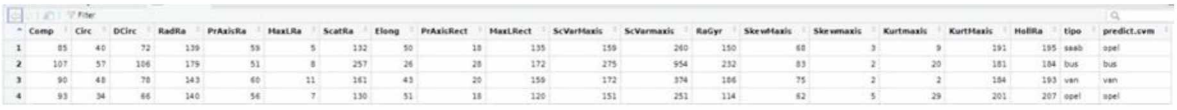
CLASSIFICAÇÃO – VEÍCULO

TABELA 1 – Comparativo do resultado das técnicas para a base de Veículos

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV com grid	C=100, Sigma= 0.015	0.8482896	0.8323
RNA – CV com grid	size=21, decay=0.1	0.8395732	0.7904
SVM – CV sem grid	C=1, Sigma= 0.06437798	0.7703150	0.7425
RF – CV com grid	mtry=18	0.7615115	0.7425
SVM – Hold-out	C=1, Sigma= 0.06437798	0.7586751	0.7425
RF – CV sem grid	mtry=10	0.7556298	0.7365
RF – Hold-out	mtry=10	0.7379643	0.7725
KNN	k=1	0.6390135	0.6176
RNA – CV sem grid	size=3, decay=0.1	0.6227787	0.4671
RNA – Hold-out	size=5, decay=0.1	0.5659535	0.6527

Técnica com melhor desempenho: SVM – CV, acurácia: 0,8482896

Predição de novos casos:



	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo	predict.cvm
1	85	40	72	139	59	5	132	50	18	135	159	260	150	68	3	9	191	195	saab	opel
2	107	57	106	179	51	8	257	26	28	172	275	954	232	83	2	20	181	184	bus	bus
3	90	48	78	143	60	11	161	43	20	159	172	374	186	75	2	2	184	193	van	van
4	93	34	66	140	56	7	130	51	18	120	151	251	114	62	5	29	201	207	opel	opel

CÓDIGO

```
## Configura o ambiente de execução e as bibliotecas necessárias
library("caret")
setwd("C:/Users/glauc/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/06 - Veículos")
### Carrega os arquivos CSV
dados <- read.csv("6 - Veiculos - Dados.csv")
dados_novos <- read.csv("6 - Veiculos - Dados - Novos.csv")
### Remove as colunas de identificação das linhas
dados$a <- NULL
dados_novos$a <- NULL
### Visualiza os dataframes
View(dados)
View(dados_novos)

## KNN
### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma
randomizada
set.seed(202485)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

### Faz um grid com valores para K
tuneGrid <- expand.grid(k = c(1,3,5,7,9))
### Executa o treinamento do KNN
set.seed(202485)
knn <- train(tipo~., data = treino, method = "knn", tuneGrid=tuneGrid)
knn

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

## RNA
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinamento do modelo RNA com Hold-out
set.seed(202485)
rna <- train(tipo~., data=treino, method="nnet", trace=FALSE)
rna
### Predições dos valores do conjunto de teste (HO)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))
```

```
### Treinamento do modelo RNA com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna <- train(tipo~., data=treino, method="nnet", trace=FALSE, trControl=ctrl)
rna
## Predições dos valores do conjunto de teste (CV)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))
### Treinamento do modelo RNA com Cross-validation e Tunegrid
grid <- expand.grid(size = seq(from = 1, to = 35, by = 10),
                    decay = seq(from = 0.1, to = 0.6, by = 0.3))
set.seed(202485)
rna <- train(form = tipo~., data = treino, method = "nnet", tuneGrid = grid,
trControl = ctrl, maxit = 2000, trace=FALSE)
rna
## Predições dos valores do conjunto de teste (CV+TG)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$tipo))

## SVM
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinamento do modelo SVM com Hold-out
set.seed(202485)
svm <- train(tipo~., data=treino, method="svmRadial")
svm
### Predições dos valores do conjunto de teste (HO)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

### Treinamento do modelo SVM com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl)
svm
## Predições dos valores do conjunto de teste (CV)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

### Treinamento do modelo SVM com Cross-validation e Tunegrid
grid <- expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(0.01, 0.015, 0.2))
set.seed(202485)
svm <- train(form = tipo~., data = treino, method = "svmRadial",
             trControl = ctrl, tuneGrid = grid)
svm
## Predições dos valores do conjunto de teste (CV+TG)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$tipo))

## Random Forest
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]
```

```
### Treinamento do modelo RF com Hold-out
set.seed(202485)
rf <- train(tipo~., data=treino, method="rf")
rf
### Predições dos valores do conjunto de teste (HO)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$tipo))
### Treinamento do modelo RF com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rf <- train(tipo~., data=treino, method="rf", trControl=ctrl)
rf
## Predições dos valores do conjunto de teste (CV)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$tipo))

### Treinamento do modelo RF com Cross-validation e Tunegrid
grid <- expand.grid(mtry = c(2, 3, 6, 9, 12, 15, 18))
set.seed(202485)
rf <- train(form = tipo~., data = treino, method = "rf",
            trControl = ctrl, tuneGrid = grid)
rf
## Predições dos valores do conjunto de teste (CV+TG)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$tipo))

### Predição de Novos Casos usando SVM (CV+TG)
predict.svm <- predict(svm, dados_novos)
resultado <- cbind(dados_novos, predict.svm)
View(resultado)
```

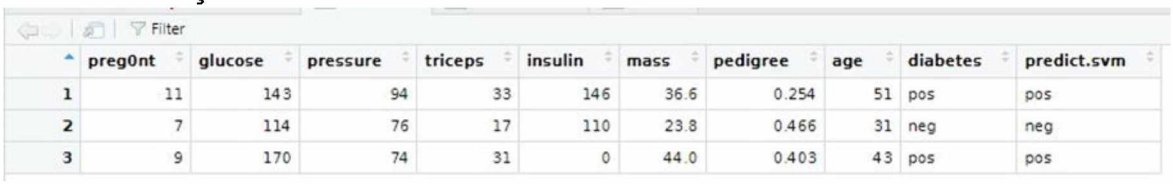
CLASSIFICAÇÃO – DIABETES

TABELA 2 – Comparativo do resultado das técnicas para a base de Diabetes

Técnica	Parâmetro	Acurácia	Matriz de Confusão
SVM – CV com grid	C=1, Sigma=0.01	0.7934162	0.7255
SVM – CV sem grid	C=0.25, Sigma=0.1129486	0.7787150	0.7255
RF – CV com grid	mtry=2	0.7770756	0.7059
SVM – Hold-out	C=0.25, Sigma=0.1129486	0.7696365	0.7255
RF – CV sem grid	mtry=2	0.7689847	0.7255
RF – Hold-out	mtry=2	0.7637726	0.7059
RNA – CV com grid	size=21, decay=0.1	0.7609201	0.7386
RNA – CV sem grid	size=3, decay=0.1	0.7479905	0.6863
KNN	k=9	0.7125222	0.7208
RNA – Hold-out	size=3, decay=0.1	0.6794497	0.634

Técnica com melhor desempenho: SVM – CV, acurácia: 0,8482896

Predição de novos casos:



Filter

	preg0nt	glucose	pressure	triceps	insulin	mass	pedigree	age	diabetes	predict.svm
1	11	143	94	33	146	36.6	0.254	51	pos	pos
2	7	114	76	17	110	23.8	0.466	31	neg	neg
3	9	170	74	31	0	44.0	0.403	43	pos	pos

CÓDIGO

```
## Configura o ambiente de execução e as bibliotecas necessárias
library("caret")
setwd("C:/Users/glauc/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/10 - Diabetes")
### Carrega os arquivos CSV
dados <- read.csv("10 - Diabetes - Dados.csv")
dados_novos <- read.csv("10 - Diabetes - Dados - Novos.csv")
### Remove as colunas de identificação das linhas
dados$num <- NULL
dados_novos$num <- NULL
### Visualiza os dataframes
View(dados)
View(dados_novos)

## KNN
### Cria um arquivo com treino com 80% e teste com 20% das linhas de forma
randomizada
set.seed(202485)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

### Faz um grid com valores para K
tuneGrid <- expand.grid(k = c(1,3,5,7,9))

### Executa o treinamento do KNN
set.seed(202485)
knn <- train(diabetes~., data = treino, method = "knn", tuneGrid=tuneGrid)
knn

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

## RNA
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinamento do modelo RNA com Hold-out
set.seed(202485)
rna <- train(diabetes~., data=treino, method="nnet", trace=FALSE)
rna
### Predições dos valores do conjunto de teste (HO)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))
```

```
### Treinamento do modelo RNA com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna <- train(diabetes~., data=treino, method="nnet",
             trace=FALSE, trControl=ctrl)
rna
## Predições dos valores do conjunto de teste (CV)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

### Treinamento do modelo RNA com Cross-validation e Tunegrid
grid <- expand.grid(size = seq(from = 1, to = 35, by = 10),
                    decay = seq(from = 0.1, to = 0.6, by = 0.3))
set.seed(202485)
rna <- train(form = diabetes~., data = treino, method = "nnet",
             tuneGrid = grid, trControl = ctrl, maxit = 2000, trace=FALSE)
rna
## Predições dos valores do conjunto de teste (CV+TG)
predict.rna <- predict(rna, teste)
confusionMatrix(predict.rna, as.factor(teste$diabetes))

## SVM
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinamento do modelo SVM com Hold-out
set.seed(202485)
svm <- train(diabetes~., data=treino, method="svmRadial")
svm
### Predições dos valores do conjunto de teste (HO)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$diabetes))

### Treinamento do modelo SVM com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
svm <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl)
svm
## Predições dos valores do conjunto de teste (CV)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$diabetes))

### Treinamento do modelo SVM com Cross-validation e Tunegrid
grid <- expand.grid(C = c(1, 2, 10, 50, 100), sigma = c(0.01, 0.015, 0.2))
set.seed(202485)
svm <- train(form = diabetes~. , data = treino, method = "svmRadial", trControl =
ctrl, tuneGrid = grid)
svm
```

```
## Predições dos valores do conjunto de teste (CV+TG)
predict.svm <- predict(svm, teste)
confusionMatrix(predict.svm, as.factor(teste$diabetes))

## Random Forest
### Particionar a bases em treino (80%) e teste (20%)
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

### Treinamento do modelo RF com Hold-out
set.seed(202485)
rf <- train(diabetes~., data=treino, method="rf")
rf
### Predições dos valores do conjunto de teste (HO)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$diabetes))
### Treinamento do modelo RF com Cross-validation
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rf <- train(diabetes~., data=treino, method="rf", trControl=ctrl)
rf
## Predições dos valores do conjunto de teste (CV)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$diabetes))

### Treinamento do modelo RF com Cross-validation e Tunegrid
grid <- expand.grid(mtry = c(2, 4, 6, 8))
set.seed(202485)
rf <- train(form = diabetes~. , data = treino, method = "rf",
            trControl = ctrl, tuneGrid = grid)
rf
## Predições dos valores do conjunto de teste (CV+TG)
predict.rf <- predict(rf, teste)
confusionMatrix(predict.rf, as.factor(teste$diabetes))

### Predição de Novos Casos usando SVM (CV+TG)
predict.svm <- predict(svm, dados_novos)
resultado <- cbind(dados_novos, predict.svm)
View(resultado)
```

REGRESSÃO – ADMISSÃO

TABELA 3 – Comparativo do resultado das técnicas para a base de Admissões

Técnica	Parâmetro	R^2	Syx	Pearson	RMSE	MAE
SVM – CV Com grid	C = 10 sigma = 0.01	0.789376	0.059158	0.923379	0.058551	0.038229
SVM – Hold-out Com grid	C = 2 sigma = 0.01	0.781435	0.059204	0.924785	0.058597	0.038832
RF – CV Com grid	mtry = 2	0.758387	0.062267	0.911964	0.061629	0.041319
RF – Hold-out Sem grid	mtry = 2	0.756269	0.062346	0.911978	0.061707	0.041387
SVM – CV Sem grid	C = 1 sigma = 0.2074478	0.755719	0.061922	0.916669	0.061287	0.041750
RF – CV Sem grid	mtry = 2	0.755678	0.062506	0.911280	0.061865	0.041780
RF – Hold-out Com grid	mtry = 2	0.754565	0.062347	0.912288	0.061708	0.041125
RNA – Hold-out Com grid	size = 9 decay = 0.1	0.743898	0.062546	0.912788	0.061905	0.045973
RNA – CV Sem grid	size = 5 decay = 0.1	0.734980	0.063313	0.910600	0.062664	0.046591
SVM – Hold-out Sem grid	C = 0.5 sigma = 0.2074478	0.734136	0.062447	0.918901	0.061807	0.042691
RNA – Hold-out Sem grid	size = 5 decay = 0.1	0.709483	0.065351	0.904786	0.064681	0.048558
RNA – CV Com grid	size = 9 decay = 0.1	0.684974	0.066554	0.902630	0.065871	0.050512
KNN	K = 9	0.633991	0.074328	0.869567	0.073566	0.054662

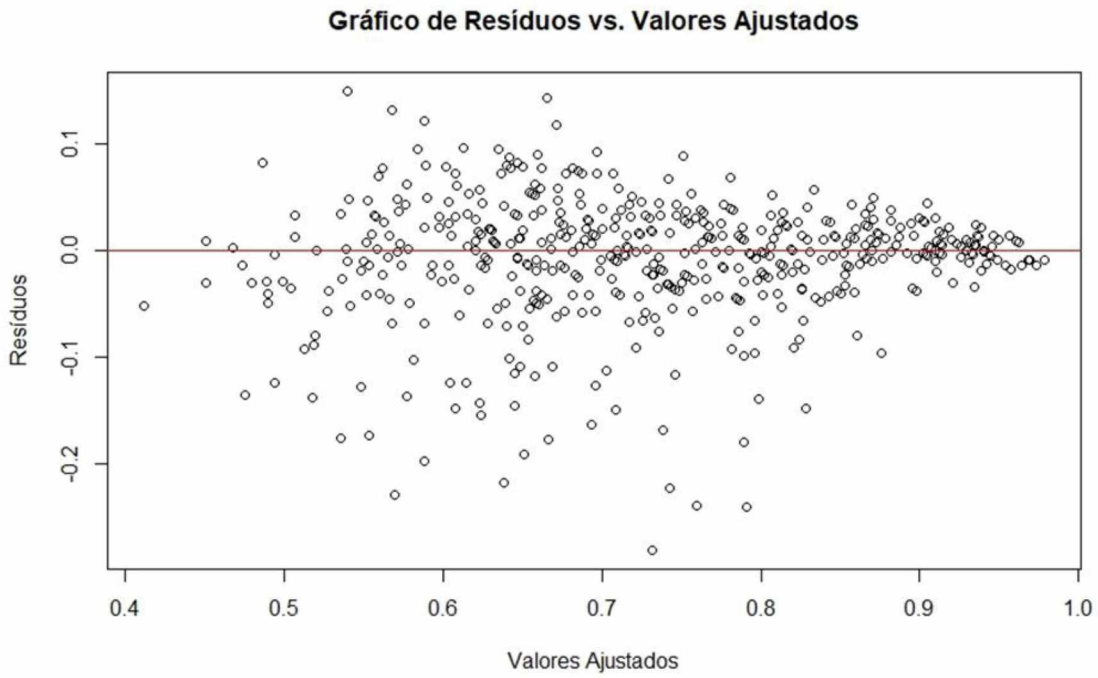
Técnica com melhor desempenho: SVM – CV com grid, R^2: 0,789376

Predição de novo casos:

Filter

	GRE.Score	TOEFL.Score	University.Rating	SOP	LOR	CGPA	Research	pred_novos
1	340	102	3	2.5	3.5	7.46	1	0.6421862
2	296	98	2	4.0	3.0	8.00	0	0.5609507
3	329	114	4	3.0	4.0	9.15	1	0.8564272

Gráfico de Resíduos:



CÓDIGO

```
### Declara uma função para o cálculo do índice R2 da regressão
r2 <- function(observados, estimados) {
  ret <- (1 - (sum((observados-estimados)^2)
               /sum((observados-mean(observados))^2)))
  return(ret)
}
#############################################################################
## Função para executar todas as predições e cálculos
#############################################################################
## seed: Seta o seed para reproduzir o experimento.
## filepath: Caminho da pasta aonde está o arquivo de dados.
## filename: Nome do arquivo de dados.
## filenewcases: Nome do arquivo com os novos dados para predição.
## fileheader: Se o arquivo dos dados tem header ou não. -> TRUE ou FALSE
## colX: Critério da coluna de predição usado no treinamento do modelo
## metodo: Método de predição. -> "knn", "nnet", "svmRadial", "RF"
## separacao: Critério de separação da base de teste. -> "ho" ou "cv"
## usaGrid: Se usa um grid de configuração de parâmetros. -> TRUE ou FALSE
#############################################################################
trabRegressao <- function(seed, filepath, filename, filenewcases,
                          fileheader=T, colX, metodo,
                          separacao="ho", usaGrid=F){
```

```
### Carrega os pacotes de acordo com o método
if(metodo == "knn"){
  library(caret)
  library(Metrics)
  library(Fgmutils) #para usar a função do cálculo do Syx
} else if(metodo == "nnet") {
  library(mlbench)
  library(caret)
  library(mice)
  library(Metrics)
  library(Fgmutils) #para usar a função do cálculo do Syx
} else if( (metodo == "svmRadial") || (metodo == "rf") ) {
  library(e1071)
  library(kernlab)
  library(caret)
  library(Metrics)
  library(Fgmutils) #para usar a função do cálculo do Syx
}

### Leitura dos dados do arquivo
setwd(filepath)
dados <- read.csv(filename, header=fileheader)
dados$num <- NULL

### Divide os dados em DFs de treino e teste
set.seed(seed)
ind <- createDataPartition(dados[,ncol(dados)], p=0.80, list=F)
treino_df <- dados[ind,]
teste_df <- dados[-ind,]

### Treina o modelo de acordo com o método selecionado e exibe
if(metodo == "knn") { # Se selecionou KNN
  # valores de k-neigbors que serão testados
  tng <- expand.grid(k = c(1,3,5,7,9))
  set.seed(seed)
  modelo <- train(colX, data = treino_df, method = metodo, tuneGrid = tng)
} else { # Se selecionou qualquer dos outros modelos: RNA, SVM ou RF
  if(separacao == "cv"){ # separação por Cross Validation
    # controlador para Cross Validation usando 10 divisões
    ctrl <- trainControl(method = "cv", number = 10)
    if(usaGrid){ # se quer configurar um grid de parâmetros
      if(metodo == "nnet"){
        tng <- expand.grid(size = seq(from = 1, to = 10, by = 1),
                           decay = seq(from = 0.1, to = 0.9, by = 0.3))
      } else if(metodo == "svmRadial") {
        tng <- expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
      } else if(metodo == "rf") {
        tng <- expand.grid(mtry=c(2, 4, 6, 8))
      }
      if(metodo == "nnet"){
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        trControl = ctrl, tuneGrid = tng,
                        linout = T, trace = F)
      } else {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        trControl = ctrl, tuneGrid = tng)
      }
```

```
    } else { # sem grid de parâmetros
      if(metodo == "nnet") {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        trControl = ctrl, linout = T, trace = F)
      } else {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        trControl = ctrl)
      }
    }
  } else { # Se não for Cross Validation, usa HoldOut
    if(usaGrid){ # se quer configurar um grid de parâmetros
      if(metodo == "nnet"){
        tng <- expand.grid(size = seq(from = 1, to = 10, by = 1),
                           decay = seq(from = 0.1, to = 0.9, by = 0.3))
      } else if(metodo == "svmRadial") {
        tng <- expand.grid(C=c(1, 2, 10, 50, 100), sigma=c(.01, .015, 0.2))
      } else if(metodo == "rf") {
        tng <- expand.grid(mtry=c(2, 4, 6, 8))
      }
      if(metodo == "nnet"){
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        tuneGrid = tng, linout = T, trace = F)
      } else {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        tuneGrid = tng)
      }
    } else { # sem grid de parâmetros
      if(metodo == "nnet") {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo,
                        linout = T, trace = F)
      } else {
        set.seed(seed)
        modelo <- train(colX, data = treino_df, method = metodo)
      }
    }
  }
}
```

```
  ### Predição com o modelo atual no dataframe de teste de acordo com o método
  pred <- predict(modelo, teste_df)

  ### Calcula os indicadores para a previsão
  r2 <- r2(pred, teste_df[,ncol(dados)])
  syx <- syx(teste_df[,ncol(dados)], pred, n=nrow(teste_df), p=1)
  pearson <- cor(teste_df[,ncol(dados)], pred, method = "pearson")
  rmse <- rmse(teste_df[,ncol(dados)], pred)
  mae <- mae(teste_df[,ncol(dados)], pred)

  ### Carrega os dados dos novos casos para predição
  novos_casos <- read.csv(filenewcases, header=fileheader)
  novos_casos$num <- NULL

  ### Predição de novos casos
  pred_novos <- predict(modelo, novos_casos)
  novos_casos[,ncol(dados)] <- NULL
  result_pred <- cbind(novos_casos, pred_novos)

  ### Retorna um vetor com o modelo, indicadores e um dataframe com a predição
  return(list(model=modelo, R2=r2, Syx=syx, Pearson=pearson, RMSE=rmse, MAE=mae,
              dfPredicao=result_pred))
}

## declara as variáveis para execução
filepath <- "C:/Users/glauc/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/09 -
Admissão"
filename <- "9 - Admissao - Dados.csv"
filenewcases <- "9 - Admissao - NovosCasos.csv"
fileheader <- TRUE
colX <- eval(ChanceOfAdmit~.) # Critério da coluna de predição. Tem que usar eval!
seed <- 202485 #para setar o seed do experimento
### Carrega as funções
source("Funcoes_Trabalho.R")
setwd(filepath)

### KNN ###
metodo <- "knn"
## HoldOut - Sem Grid ##
separacao <- "ho" #Separação das bases de validação por HoldOut
usaGrid <- FALSE #Se usa grid para tentar calcular o size e o decay melhor
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### KNN ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)

# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
```

```
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### KNN ###")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)

# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### RNA ###
metodo <- "nnet"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### RNA ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
```

```
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
```

```
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### SVM ###
metodo <- "svmRadial"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### SVM ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
```

```
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### RANDOM FOREST ###
metodo <- "rf"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### RANDOM FOREST ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
```

```
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
```

```
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
```

REGRESSÃO – BIOMASSA

TABELA 4 – Comparativo do resultado das técnicas para a base de Biomassa

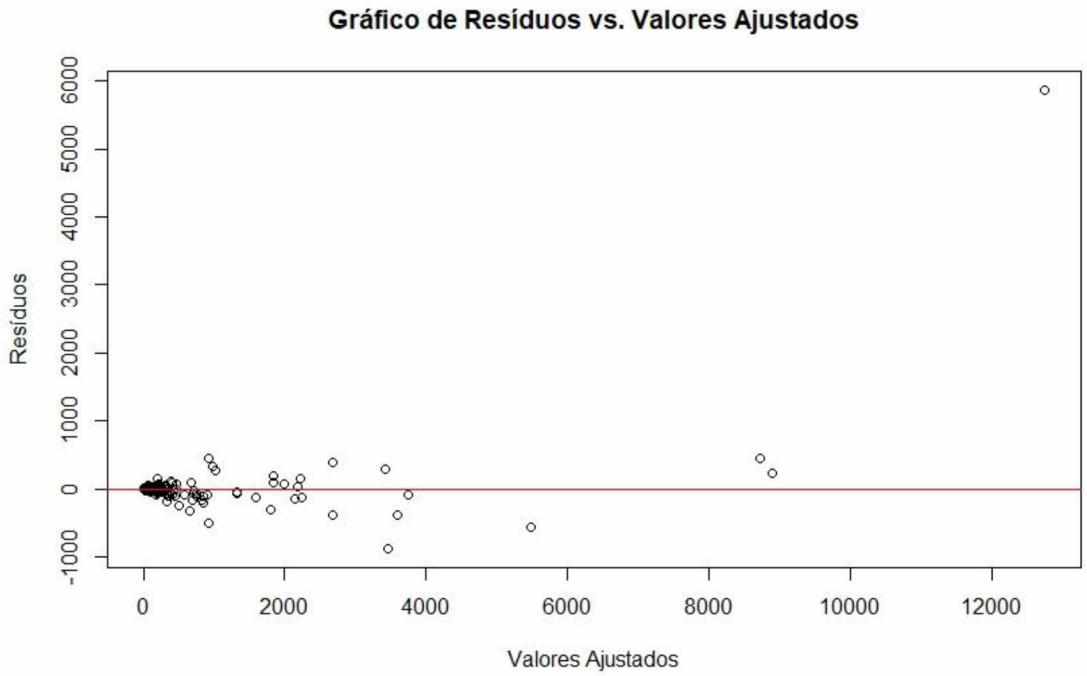
Técnica	Parâmetro	R²	Syx	Pearson	Rmse	MAE
RF – Hold out Sem grid	mtry=2	0.984068	157.786610	0.992201	155.134545	63.856373
RF – CV Com grid	mtry=2	0.976586	181.697563	0.990147	178.643604	72.972522
RF – CV Sem grid	mtry=2	0.973855	190.189821	0.989462	186.993126	74.669903
RF – Hold out Com grid	mtry=2	0.965333	212.818902	0.987972	209.241859	79.451578
RNA – Hold out Com grid	Size=2 Decay =0.7	0.963713	255.742350	0.983519	251.443853	148.422506
SVM – CV Com grid	C=100 Sigma = 0.01	0.916314	303.006088	0.980208	297.913186	115.408387
SVM – Hold out Com grid	C=100 Sigma=0.01	0.916314	303.006088	0.980208	297.913186	115.408387
KNN	K =3	0.713939	484.145732	0.948596	476.008249	135.896425
RNA – CV Com grid	Size=6 Decay =0.4	0.217431	707.588618	0.849904	695.695526	237.414720
RNA – CV Sem grid	Size=5 Decay =0.1	0.197787	689.456434	0.875019	677.868106	173.054202
SVM – Hold out Sem grid	C=1 Sigma=1.027848	-3.391228	1054.157347	0.559121	1036.439155	238.325013
SVM – CV Sem grid	C=1 Sigma =1.027848	-3.391228	1054.157347	0.559121	1036.439155	238.325013
RNA – Hold out Sem grid	Size=5 Decay=0.1	-7147890531188040.0	1247.907676	0.054192	1226.932944	497.813797

Técnica com melhor desempenho: RF – Hold out sem grid, R²: 0,984068

Predição de novos casos:

	dap	h	Me	pred_novos
1	12.7	8.7	0.83	30.55008
2	41.0	32.0	0.64	2238.01339
3	8.2	9.1	0.39	20.62286

Gráfico de Resíduos:



CÓDIGO

```
### Declara uma função para o cálculo do índice R2 da regressão
r2 <- function(observados, estimados) {
  ret <- (1 - (sum((observados-estimados)^2)
               /sum((observados-mean(observados))^2)))
  return(ret)
}
```

```
#########################################################################
## Função para executar todas as predições e cálculos
#########################################################################
## seed: Seta o seed para reproduzir o experimento.
## filepath: Caminho da pasta aonde está o arquivo de dados.
## filename: Nome do arquivo de dados.
## filenewcases: Nome do arquivo com os novos dados para predição.
## fileheader: Se o arquivo dos dados tem header ou não. -> TRUE ou FALSE
## colX: Critério da coluna de predição usado no treinamento do modelo
## metodo: Método de predição. -> "knn", "nnet", "svmRadial", "RF"
## separacao: Critério de separação da base de teste. -> "ho" ou "cv"
## usaGrid: Se usa um grid de configuração de parâmetros. -> TRUE ou FALSE
#########################################################################
trabRegressao <- function(seed, filepath, filename, filenewcases,
                          fileheader=T, colX, metodo,
                          separacao="ho", usaGrid=F){

  ### Carrega os pacotes de acordo com o método
  if(metodo == "knn"){
    library(caret)
    library(Metrics)
    library(Fgmutils) #para usar a função do cálculo do Syx
  } else if(metodo == "nnet") {
    library(mlbench)
    library(caret)
    library(mice)
    library(Metrics)
    library(Fgmutils) #para usar a função do cálculo do Syx
  } else if( (metodo == "svmRadial") || (metodo == "rf") ) {
    library(e1071)
    library(kernlab)
    library(caret)
    library(Metrics)
    library(Fgmutils) #para usar a função do cálculo do Syx
  }

  ### Leitura dos dados do arquivo
  setwd(filepath)
  dados <- read.csv(filename, header=fileheader)
  dados$num <- NULL

  ### Divide os dados em DFs de treino e teste
  set.seed(seed)
  ind <- createDataPartition(dados[,ncol(dados)], p=0.80, list=F)
  treino_df <- dados[ind,]
  teste_df <- dados[-ind,]

  ### Treina o modelo de acordo com o método selecionado e exibe
  if(metodo == "knn") { # Se selecionou KNN
    # valores de k-neigbors que serão testados
    tng <- expand.grid(k = c(1,3,5,7,9))
    set.seed(seed)
    modelo <- train(colX, data = treino_df,
                    method = metodo, tuneGrid = tng)
  } else { # Se selecionou qualquer dos outros modelos: RNA, SVM ou RF
    if(separacao == "cv"){ # separação por Cross Validation
      # controlador para Cross Validation usando 10 divisões
      ctrl <- trainControl(method = "cv", number = 10)
      if(usaGrid){ # se quer configurar um grid de parâmetros
```

```
    if(metodo == "nnet"){
      tng <- expand.grid(size = seq(from = 1, to = 10, by = 1),
                         decay = seq(from = 0.1, to = 0.9, by = 0.3))
    } else if(metodo == "svmRadial") {
      tng <- expand.grid(C=c(1, 2, 10, 50, 100),
                         sigma=c(.01, .015, 0.2))
    } else if(metodo == "rf") {
      tng <- expand.grid(mtry=c(2, 4, 6, 8))
    }
    if(metodo == "nnet"){
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      trControl = ctrl, tuneGrid = tng,
                      linout = T, trace = F)
    } else {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      trControl = ctrl, tuneGrid = tng)
    }
  } else { # sem grid de parâmetros
    if(metodo == "nnet") {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      trControl = ctrl, linout = T, trace = F)
    } else {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      trControl = ctrl)
    }
  }
} else { # Se não for Cross Validation, usa HoldOut
  if(usaGrid){ # se quer configurar um grid de parâmetros
    if(metodo == "nnet"){
      tng <- expand.grid(size = seq(from = 1, to = 10, by = 1),
                         decay = seq(from = 0.1, to = 0.9, by = 0.3))
    } else if(metodo == "svmRadial") {
      tng <- expand.grid(C=c(1, 2, 10, 50, 100),
                         sigma=c(.01, .015, 0.2))
    } else if(metodo == "rf") {
      tng <- expand.grid(mtry=c(2, 4, 6, 8))
    }
    if(metodo == "nnet"){
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      tuneGrid = tng, linout = T, trace = F)
    } else {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      tuneGrid = tng)
    }
  } else { # sem grid de parâmetros
    if(metodo == "nnet") {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo,
                      linout = T, trace = F)
    } else {
      set.seed(seed)
      modelo <- train(colX, data = treino_df, method = metodo)
    }
```

```
      }
    }
  }

  ### Predição com o modelo atual no dataframe de teste de acordo com o método
  pred <- predict(modelo, teste_df)

  ### Calcula os indicadores para a previsão
  r2 <- r2(pred, teste_df[,ncol(dados)])
  syx <- syx(teste_df[,ncol(dados)], pred, n=nrow(teste_df), p=1)
  pearson <- cor(teste_df[,ncol(dados)], pred, method = "pearson")
  rmse <- rmse(teste_df[,ncol(dados)], pred)
  mae <- mae(teste_df[,ncol(dados)], pred)

  ### Carrega os dados dos novos casos para predição
  novos_casos <- read.csv(filenewcases, header=fileheader)
  novos_casos$num <- NULL

  ### Predição de novos casos
  pred_novos <- predict(modelo, novos_casos)
  novos_casos[,ncol(dados)] <- NULL
  result_pred <- cbind(novos_casos, pred_novos)

  ### Retorna um vetor com o modelo, indicadores e um dataframe com a predição
  return(list(model=modelo, R2=r2, Syx=syx, Pearson=pearson,
              RMSE=rmse, MAE=mae, dfPredicao=result_pred))
}

## declara as variáveis para execução
filepath <- "C:/Users/glauc/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/05 -
Biomassa"
filename <- "5 - Biomassa - Dados.csv"
filenewcases <- "5 - Biomassa - Dados - Novos.csv"
fileheader <- TRUE
colX <- eval(biomassa~.) # Critério da coluna de predição. Tem que usar eval!
seed <- 202485 #para setar o seed do experimento
### Carrega as funções
source("Funcoes_Trabalho.R")
setwd(filepath)
```

```
### KNN ###
metodo <- "knn"
## HoldOut - Sem Grid ##
separacao <- "ho" #Separação das bases de validação por HoldOut
usaGrid <- FALSE #Se usa grid para tentar calcular o size e o decay melhor
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### KNN ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)

# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### KNN ###")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)

# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### RNA ###
metodo <- "nnet"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### RNA ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
```

```
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
```

```
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### SVM ###
metodo <- "svmRadial"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### SVM ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
```

```
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)

### RANDOM FOREST ###
metodo <- "rf"
## HoldOut - Sem Grid ##
separacao <- "ho"
usaGrid <- FALSE
```

```
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("### RANDOM FOREST ###")
print("..HOLDOUT - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## HoldOut - Com Grid ##
separacao <- "ho"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..HOLDOUT - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Sem Grid ##
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
```

```
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
## Cross Validation - Com Grid ##
separacao <- "cv"
usaGrid <- TRUE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - COM GRID")
# Exibe o modelo
print(vRet$model)
# Exibe os indicadores
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
# Exibe o dataframe com os valores preditos
View(vRet$dfPredicao)
```

AGRUPAMENTO – VEÍCULO

K-modes clustering with 10 clusters of sizes 130, 77, 62, 57, 62, 71, 60, 126, 99, 102

Cluster modes:

	Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis
1	104	54	104	201	62	10	213	31	24	162	226	635
2	90	52	105	197	64	11	198	33	23	142	214	268
3	89	41	66	155	64	7	149	45	19	143	169	327
4	100	49	90	178	61	6	177	37	21	158	189	457
5	85	45	66	125	55	6	149	46	19	146	170	321
6	86	43	72	123	56	7	150	46	19	144	169	319
7	85	45	68	120	53	7	151	45	19	145	170	333
8	89	43	85	136	58	10	155	44	19	160	173	351
9	92	38	66	169	59	7	169	39	20	131	184	259
10	101	55	103	230	57	11	219	30	25	172	228	706

	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo
1	214	72	0	21	189	197	saab
2	186	65	4	20	192	201	saab
3	152	72	6	0	188	195	bus
4	198	72	1	11	196	199	bus
5	172	71	10	7	187	191	bus
6	171	85	6	14	180	183	bus
7	173	81	1	4	180	184	bus
8	176	75	1	9	183	196	van
9	151	67	1	10	193	202	opel
10	218	71	5	11	186	198	opel

Clustering vector:

```
  [1]  1  3  1  3  6  6  5  9  4  2  9  2  3  8  4 10  8  4  1 10  3  8  6  4  2  3  6  1  5  6  8  2
 [33]  9  9  9  9  6  8  1  8 10  8  8  2  1  8  5  6  8  9  1  1  1  3  7  4  6  8 10  8  2  9  8  7
 [65]  1  5  5 10  6  4  1 10  9  7  7 10  3  7  1  6  7  1  8  5  8  4  3 10  4  5  1  9  2  2  5  1
 [97]  1  6  2  5  7  8  3  1  5  1  2  7  7  3  8  8  9  9  8 10  1  1  6  4  2  8  8 10  5  2  3  3
[129]  8  4  1  7  9  2  1  8  8  2  2  8  1  5 10  7  9 10  8  3  6  9  1  6  3  4  5  6  5  1  8  5
[161]  9  2  9  2  1 10  3  1  3  8 10 10  6  1  8  9  4  6  7  5  1  9  3  5  2  4  9  9 10  2 10  8
[193]  8 10  9  6 10  6  6  6  8  1 10  1  8  3  9  5  5 10  9  8  6  4  3  9  8 10  3  8 10  8  4  5
[225]  1  5  1  1  1  7  7  1  7  1  2  6  4  7 10  6  8  9  5  1  8  8  5  9 10  2  9 10  8  5  8  1
[257]  2  8  1 10  7  4  8  8  8  7  7  4  8  8  7  2  8  5  7  3  5  5  1  5  9  8  7  1  8  9  3  8
[289]  2  1  5  8  5  9  5  7  2  1  4  4  1  8  6  8  7  5  1 10  9  1  4  1  3  2  7  7  3  2  8 10
[321]  1  6  6  1  9  7  4  3  8  9  6  9 10  6  5 10  8  4  7  3  3  1  6 10 10 10  1  7  5  9 10  6
[353]  7  7  4  1  6  1  6  4  1  2  9  8  7  2  8  1  5  9  4  7  8  3  6  5  1 10  7  1  9  4  6  1
[385]  3  8  2  8  8 10  6  9  8  3  4  9  4  3  9  8  2  1  1  8  2  6  9  5  3  9  2  7  7  9  5 10
[417]  4  5  8  2  8 10  1  7  9 10 10  9  2  9  1  2  3  6  1  2  6  6  4  2  2  8 10  2  7  8  1  1
[449]  6  8  3  9  9  3  2  6  8  5 10  4  3  2  9  6  4  3  6  1  1  9  3 10 10 10  8  1 10  8  5  6
[481]  1  1  9  9  2  7 10  2  3  1  1  9  7 10  6  6  1  8  6  1 10 10 10  5  1 10  1 10  1 10  6  9
[513] 10  4  9  9  9  1  8  1  4  8  9  8  2  9  1  2  9  6  4 10  1  8  8  8 10  4 10 10  5  9  9  1
```

```
[545]  8  8  8  6  5  1  3  7  1  5  2  4  3  9  6  1 10  9  1  5  5 10  1  1  6  2  1  9  8  8 10  1
[577] 10  1 10  9  7  8  1  1  4  4 10 10  5 10  6 10  6  5  6 10  4 10  8  2  9  8 10  2  1 10  4  8
[609]  9  6  1 10 10  6  1 10  9  7  8  1  1  8  4  2  5  1  6  5  4  9  5  9  7  2  9  3  1  6  4 10
[641]  9  7  1  4  8  3  9  8 10  3  1  8  9  4  1  6  3  8  8  2  9  2  3  4  8  8  2  1  1  7 10  2
[673]  2  9  3 10  7  2  7  3 10  9  2  6  9  5  9  7 10  2  9  5  2  8  1  3  8  9  3  9 10  1  9  8
[705]  1  1  8  8  8  9  3  2  2 10  6 10  4  9 10  1 10  1  8  9  8  8  2  7 10  8 10  8  5  7  8 10
[737]  1  3  4  8  1  3  9  1  8  7  2  6  8  1  1  7  4  8  2  5  8  3  7  8  2  1  5 10  1  7  9  8
[769] 10  1  7  2  3  3  2  4  1  7  1  8  2  1  1  1  8  6 10  5  1  9  7  2  1  6  3  8  9  2 10  3
[801]  3  9  4  4  2  6  3  2  5  6 10  9  9  8  6  7  2  1 10  1  9 10  3  2 10 10  4 10  5  9  7  4
[833]  4  5  2  6  8  9  8  6 10  3  8 10  1  5

Within cluster simple-matching distance by cluster:
 [1] 2002 1219  955  906  970 1100  909 2058 1608 1546

Available components:
[1] "cluster"    "size"       "modes"      "withindiff" "iterations" "weighted"
```

Comp	Circ	DCirc	RadRa	PrAxisRa	MaxLRa	ScatRa	Elong	PrAxisRect	MaxLRect	ScVarMaxis	ScVarmaxis
1	95	48	83	178	72	10	162	42	20	159	176	379
2	91	41	84	141	57	9	149	45	19	143	170	330
3	104	50	106	209	66	10	207	32	23	158	223	635
4	93	41	82	159	63	9	144	46	19	143	160	309
5	85	44	70	205	103	52	149	45	19	144	241	325
6	107	57	106	172	50	6	255	26	28	169	280	957
7	97	43	73	173	65	6	153	42	19	143	176	361
8	90	43	66	157	65	9	137	48	18	146	162	281
9	86	34	62	140	61	7	122	54	17	127	141	223
10	93	44	98	197	62	11	183	36	22	146	202	505
11	86	36	70	143	61	9	133	50	18	130	153	266
12	90	34	66	136	55	6	123	54	17	118	148	224
13	88	46	74	171	68	6	152	43	19	148	180	349
14	89	42	85	144	58	10	152	44	19	144	173	345
15	94	49	79	203	71	5	174	37	21	154	196	465
16	96	55	103	201	65	9	204	32	23	166	227	624
17	89	36	51	109	52	6	118	57	17	129	137	206
18	99	41	77	197	69	6	177	36	21	139	202	485
19	104	54	100	186	61	10	216	31	24	173	225	686
20	101	56	100	215	69	10	208	32	24	169	227	651
21	84	47	75	153	64	6	154	43	19	145	175	354
22	84	37	53	121	59	5	123	55	17	125	141	221
23	94	43	64	173	69	7	150	43	19	142	169	344
24	87	39	70	148	61	7	143	46	18	136	164	307
25	99	53	105	219	66	11	204	32	23	165	221	623
26	85	45	80	154	64	9	147	45	19	148	169	324
27	83	36	54	119	57	6	128	53	18	125	143	238
28	107	54	98	203	65	11	218	31	25	167	229	696
29	102	45	85	193	64	6	192	33	22	146	217	570
30	80	38	63	129	55	7	146	46	19	130	168	314
31	89	43	85	160	64	11	155	43	19	151	173	356
32	88	42	77	151	58	8	140	47	18	142	165	293
33	93	35	66	154	59	6	142	46	18	128	162	304
34	101	48	107	222	68	10	208	32	24	154	232	641
35	87	38	85	177	61	8	164	40	20	129	186	402
36	100	46	90	172	67	9	157	43	20	150	170	363
37	82	44	72	118	52	7	152	44	19	147	174	340
38	90	48	86	306	126	49	153	44	19	156	272	346
39	106	53	98	176	54	10	216	31	24	171	235	691
40	81	45	68	169	73	6	151	44	19	146	173	336
41	95	48	104	214	67	9	205	32	23	151	227	628
42	88	37	51	105	52	5	119	57	17	128	135	207
43	94	49	87	137	54	11	158	43	20	162	178	366
44	93	37	76	183	63	8	164	40	20	134	191	405
45	119	54	106	220	65	12	213	31	24	167	223	675
46	93	46	82	145	58	11	159	43	20	160	180	371
47	91	43	70	133	55	8	130	51	18	146	159	253
48	85	42	66	122	54	6	148	46	19	141	172	317
49	89	47	81	147	64	11	156	44	20	163	170	352
50	91	45	79	176	59	9	163	40	20	148	184	404

	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo	cluster.res$cluster
1	184	70	6	16	187	197	van	1
2	158	72	9	14	189	199	van	3
3	220	73	14	9	188	196	saab	1
4	127	63	6	10	199	207	van	3
5	188	127	9	11	180	183	bus	6
6	264	85	5	9	181	183	bus	6
7	172	66	13	1	200	204	bus	5
8	164	67	3	3	193	202	van	9
9	112	64	2	14	200	208	van	4
10	152	64	4	14	195	204	saab	2
11	127	66	2	10	194	202	van	9
12	118	65	5	26	196	202	saab	2
13	192	71	5	11	189	195	bus	3

```
14   161        72         8        13       187    197  van                 8
15   206        71         6         2       197    199  bus                 4
16   246        74         6         2       186    194 opel                10
17   125        80         2        14       181    185  van                 8
18   151        72         4        10       198    199  bus                 4
19   220        74         5        11       185    195 saab                 1
20   223        74         6         5       186    193 opel                10
21   184        75         0         3       185    192  bus                 3
22   133        82         7         1       179    183  van                 8
23   177        68         9         1       199    206  bus                 6
24   141        69         1         2       192    199  bus                 4
25   224        68         0         6       191    201 saab                 2
26   174        71         1         4       188    199  van                 3
27   139        82         6         3       179    183 saab                 6
28   216        72         1        28       187    199 saab                 1
29   163        76         6         7       195    193  bus                 5
30   158        83         9        20       180    185 saab                 6
31   174        72         5         9       185    196  van                 8
32   158        64        10        11       198    205 saab                 2
33   120        64         5        13       197    202 opel                 9
34   204        70         5        38       190    202 opel                 9
35   130        63         1        25       198    205 opel                 9
36   184        67        17         7       192    200  van                 9
37   177        82         2         2       180    185  bus                 6
38   200       118         0        15       185    194  van                 8
39   218        74         1         9       187    197 saab                 1
40   186        75         7         0       183    189  bus                 8
41   202        74         5         9       186    193 opel                10
42   125        86         8        16       179    183  van                 8
43   186        75         5         5       183    194  van                 8
44   139        67         4         7       192    197 saab                 2
45   232        66        20         1       192    202 saab                 1
46   189        77         2         4       183    194  van                 8
47   156        70         1         8       190    194  van                 5
48   174        88         6        14       180    182  bus                 6
49   188        76         6        13       184    193  van                 8
50   179        62         0        10       199    208 saab                 9
 [ reached 'max' / getOption("max.print") -- omitted 796 rows ]
```

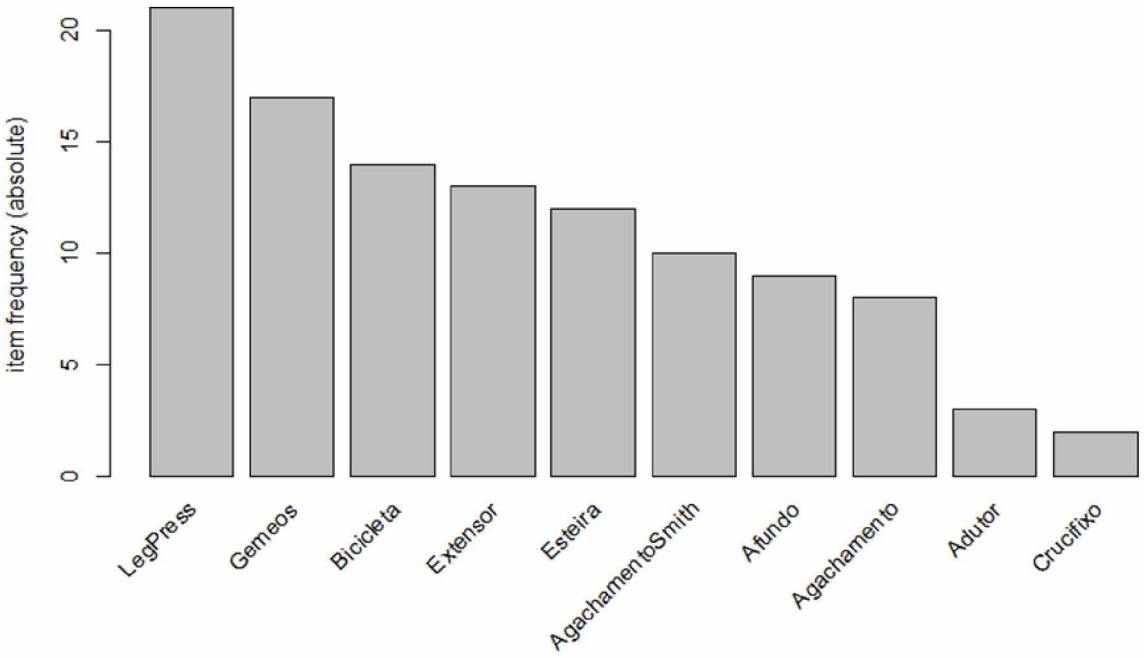
CÓDIGO

```
library(readr)
library(klaR)
dados <- read_csv("6 - Veiculos - Dados.csv")
View(dados)

## exclusão coluna de ID
dados$a <- NULL

## execução do cluster
set.seed(202485)
K = 10
cluster.res <- kmodes(dados, K, iter.max = 10, weighted = FALSE )
cluster.res

## visualiza as linhas dos dados com o seu cluster corespondente
resultado <- cbind(dados, cluster.res$cluster)
resultado
```

REGRAS DE ASSOCIAÇÃO – MUSCULAÇÃO

```
    items
[1] {Afundo, Crucifixo, Gemeos, LegPress}
[2] {Agachamento, Gemeos, LegPress}
[3] {Afundo, Agachamento, Gemeos, LegPress}
[4] {Adutor, Agachamento, LegPress}
```

```
1
```




```
Apriori

Parameter specification:
 confidence minval smax arem  aval originalSupport maxtime support minlen maxlen target  ext
        0.8    0.1    1 none FALSE            TRUE       5     0.3      1     10  rules TRUE

Algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

Absolute minimum support count: 7

set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[11 item(s), 26 transaction(s)] done [0.00s].
sorting and recoding items ... [8 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 done [0.00s].
writing ... [16 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

```
set of 16 rules

rule length distribution (lhs + rhs):sizes
 1  2  3
 1 10  5

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    2.00    2.00    2.25    3.00    3.00

summary of quality measures:
    support         confidence        coverage           lift           count
 Min.   :0.3077   Min.   :0.8000   Min.   :0.3077   Min.   :1.000   Min.   : 8.00
 1st Qu.:0.3077   1st Qu.:0.8333   1st Qu.:0.3750   1st Qu.:1.543   1st Qu.: 8.00
 Median :0.3846   Median :0.8944   Median :0.4038   Median :1.714   Median :10.00
 Mean   :0.3966   Mean   :0.8947   Mean   :0.4495   Mean   :1.661   Mean   :10.31
 3rd Qu.:0.4231   3rd Qu.:0.9423   3rd Qu.:0.4712   3rd Qu.:1.812   3rd Qu.:11.00
 Max.   :0.8077   Max.   :1.0000   Max.   :1.0000   Max.   :2.000   Max.   :21.00

mining info:
  data ntransactions support confidence
 dados            26     0.3        0.8
                                                                          call
 apriori(data = dados, parameter = list(supp = 0.3, conf = 0.8, target = "rules"))


     lhs                               rhs         support   confidence coverage  lift     count
[1]  {}                             => {LegPress}  0.8076923 0.8076923  1.0000000 1.000000 21
[2]  {Agachamento}                  => {LegPress}  0.3076923 1.0000000  0.3076923 1.238095  8
[3]  {Afundo}                       => {Gemeos}    0.3461538 1.0000000  0.3461538 1.529412  9
[4]  {AgachamentoSmith}             => {Esteira}   0.3076923 0.8000000  0.3846154 1.733333  8
[5]  {AgachamentoSmith}             => {Extensor}  0.3461538 0.9000000  0.3846154 1.800000  9
[6]  {AgachamentoSmith}             => {Bicicleta} 0.3076923 0.8000000  0.3846154 1.485714  8
[7]  {Esteira}                      => {Extensor}  0.4230769 0.9166667  0.4615385 1.833333 11
[8]  {Extensor}                     => {Esteira}   0.4230769 0.8461538  0.5000000 1.833333 11
[9]  {Esteira}                      => {Bicicleta} 0.3846154 0.8333333  0.4615385 1.547619 10
[10] {Extensor}                     => {Bicicleta} 0.4615385 0.9230769  0.5000000 1.714286 12
[11] {Bicicleta}                    => {Extensor}  0.4615385 0.8571429  0.5384615 1.714286 12
[12] {AgachamentoSmith, Extensor}   => {Bicicleta} 0.3076923 0.8888889  0.3461538 1.650794  8
[13] {AgachamentoSmith, Bicicleta}  => {Extensor}  0.3076923 1.0000000  0.3076923 2.000000  8
[14] {Esteira, Extensor}            => {Bicicleta} 0.3846154 0.9090909  0.4230769 1.688312 10
[15] {Bicicleta, Esteira}           => {Extensor}  0.3846154 1.0000000  0.3846154 2.000000 10
[16] {Bicicleta, Extensor}          => {Esteira}   0.3846154 0.8333333  0.4615385 1.805556 10
```

CÓDIGO

```
library(arules)
dados <- read.transactions("2 - Musculacao - Dados.csv", format="basket")
inspect(dados[1:4])
set.seed(202485)
itemFrequencyPlot(dados, topN=10, type="absolute")
rules <- apriori(dados, parameter = list(supp = 0.3,
                                         conf = 0.8,
                                         target = "rules"))
summary(rules)
inspect(rules)
```

APÊNDICE H - DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1 Classificação de Imagens (CNN)

CÓDIGO

```
# %% [markdown]
# Importação das bibliotecas

# %%
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
```

```
# %% [markdown]
# Carrega a Base de dados

# %%
# 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))
print("Número de classes: ", K)

#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 começ.a 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)
```

```
Número de classes:  10
```

```
# %%
model.summary()
```

Model: "functional"

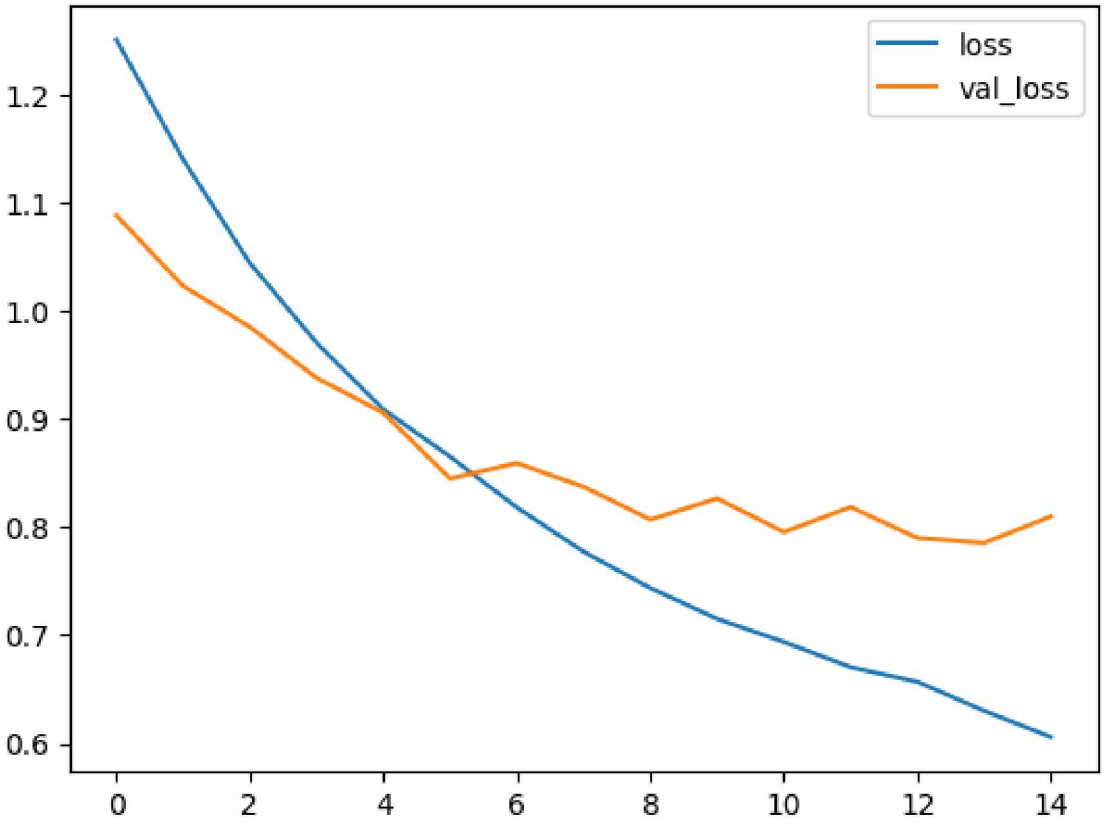
Layer (type)	*Output Shape*	*Param #*
input_layer (InputLayer)	*(None, 32, 32, 3)*	*0*
conv2d (Conv2D)	*(None, 15, 15, 32)*	*896*
conv2d_1 (Conv2D)	*(None, 7, 7, 64)*	*18,496*
conv2d_2 (Conv2D)	*(None, 3, 3, 128)*	*73,856*
flatten (Flatten)	*(None, 1152)*	*0*
dropout (Dropout)	*(None, 1152)*	*0*
dense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

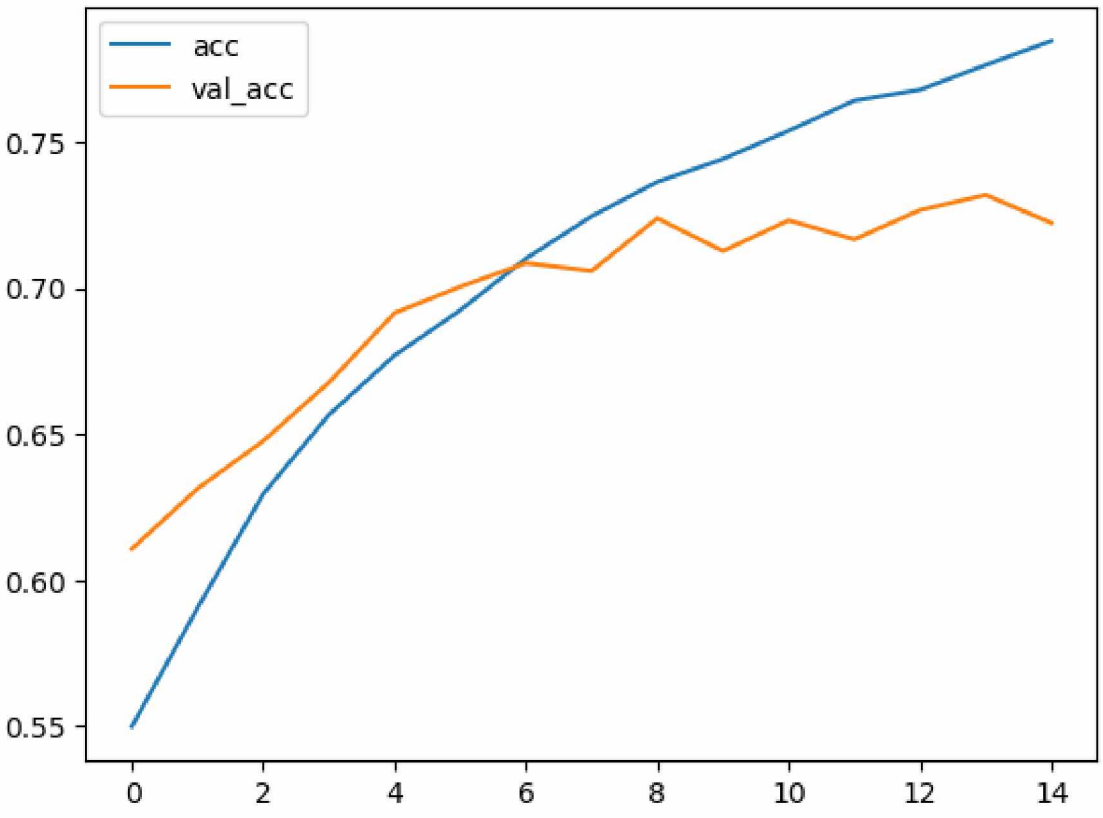
Total params: 1,284,170 (4.90 MB)
Trainable params: 1,284,170 (4.90 MB)
Non-trainable params: 0 (0.00 B)

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

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

Epoch 1/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 13s 6ms/step - accuracy: 0.5373 - loss: 1.2847 - val_accuracy: 0.6107 - val_loss: 1.0887
...
Epoch 15/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━ 6s 4ms/step - accuracy: 0.7896 - loss: 0.5868 - val_accuracy: 0.7222 - val_loss: 0.8100

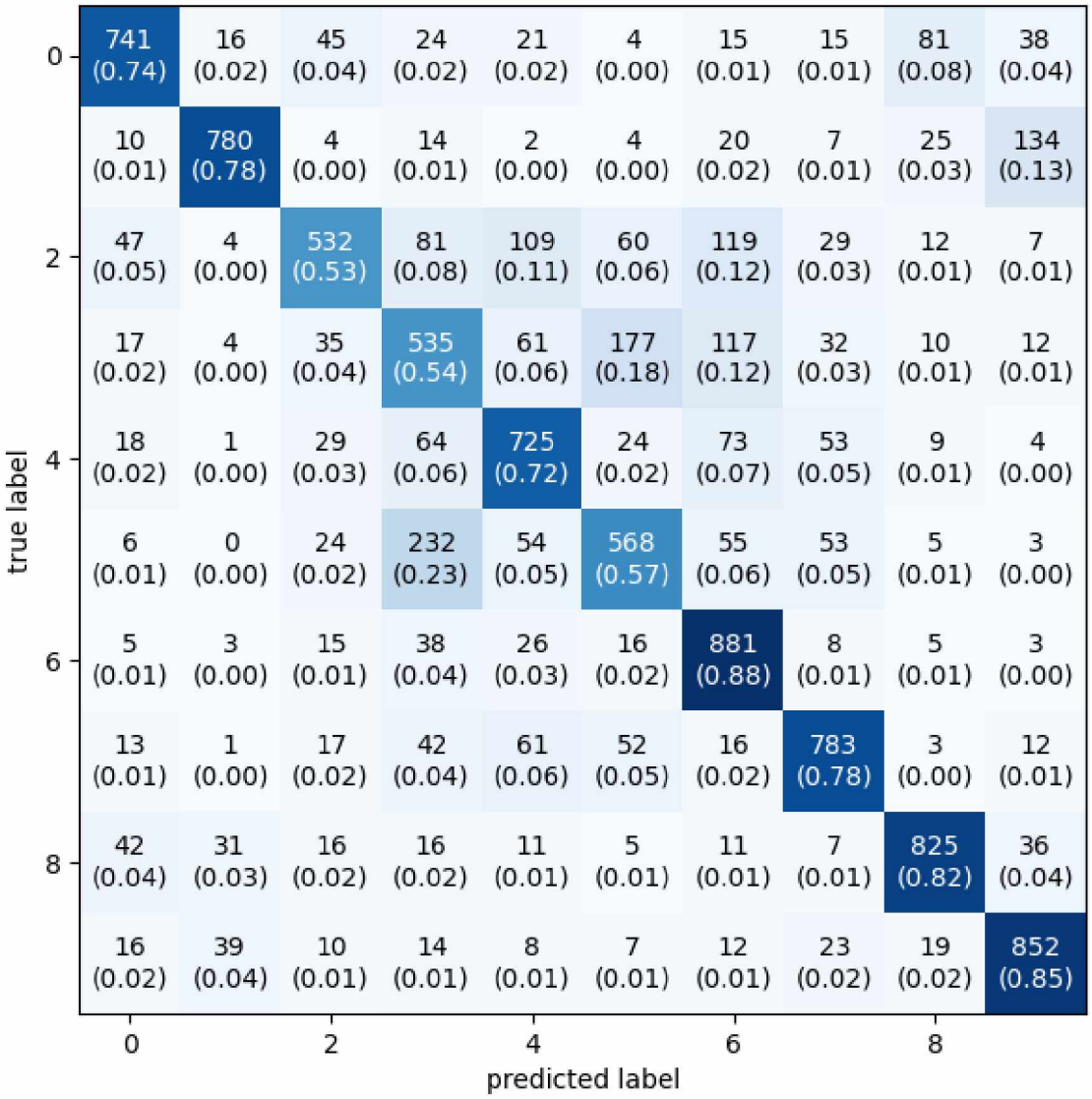
```
# %%
# Plotar a função de perda, treino e validação
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

# 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()
```


loss
val_loss
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0
2
4
6
8
10
12
14


acc
val_acc
0.75
0.70
0.65
0.60
0.55
0
2
4
6
8
10
12
14

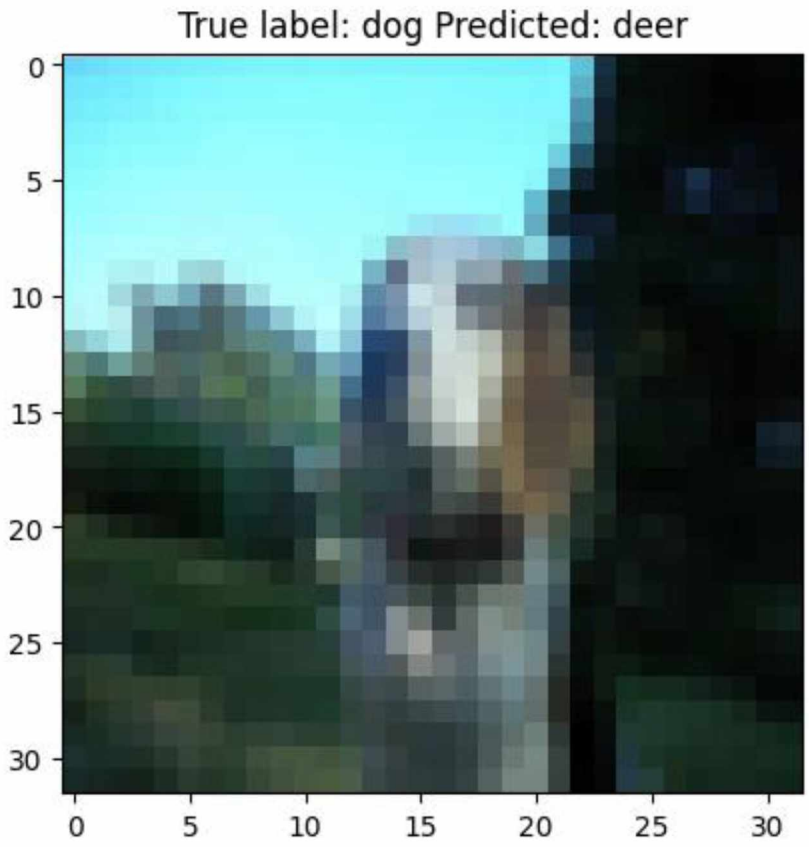
```
# %%
# Efetuar predições na base de teste
# argmax é usado pois a função de ativação da saída é softmax
# argmax pega o neurônio que deu o maior resultado, isto é,
# a maior probabilidade de saída
y_pred = model.predict(x_test).argmax(axis=1)

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

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step
(<Figure size 700x700 with 1 Axes>,
 <Axes: xlabel='predicted label', ylabel='true label'>)
```



```
# %%
# 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]]))
```

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



2 Detector de SPAM (RNN)

CÓDIGO

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

# %%
!wget http://www.razer.net.br/datasets/spam.csv
df = pd.read_csv("spam.csv", encoding="ISO-8859-1")
df.head()

df = df.drop(["Unnamed: 2", "Unnamed: 3", "Unnamed: 4"], axis=1)
df.columns = ["labels", "data"]
df["b_labels"] = df["labels"].map({ "ham": 0, "spam": 1})
y = df["b_labels"].values

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

# %%
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)

7230 tokens

# %%
data_train = pad_sequences(sequences_train)
T = data_train.shape[1]

data_test = pad_sequences(sequences_test, maxlen=T)

print("data_train.shape: ", data_train.shape)
print("data_test.shape: ", data_test.shape)

data_train.shape:  (3733, 189)
data_test.shape:  (1839, 189)
```

```
# %%
D = 20
M = 5

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

model = Model(i, x)

# %%
model.summary()
```

Model: "functional"

Layer (type)	*Output Shape*	*Param #*
input_layer (InputLayer)	*(None, 189)*	*0*
embedding (Embedding)	*(None, 189, 20)*	*144,620*
lstm (LSTM)	*(None, 5)*	*520*
dense (Dense)	*(None, 1)*	*6*

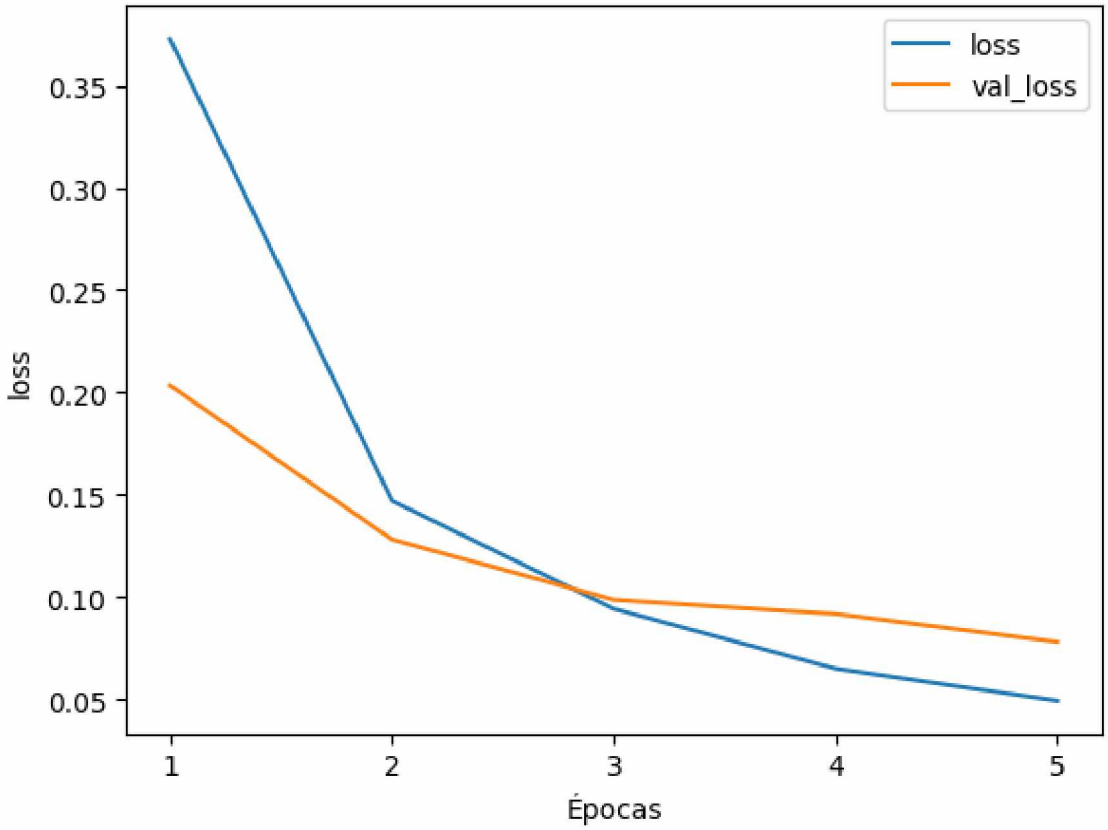
Total params: 145,146 (566.98 KB)
Trainable params: 145,146 (566.98 KB)
Non-trainable params: 0 (0.00 B)

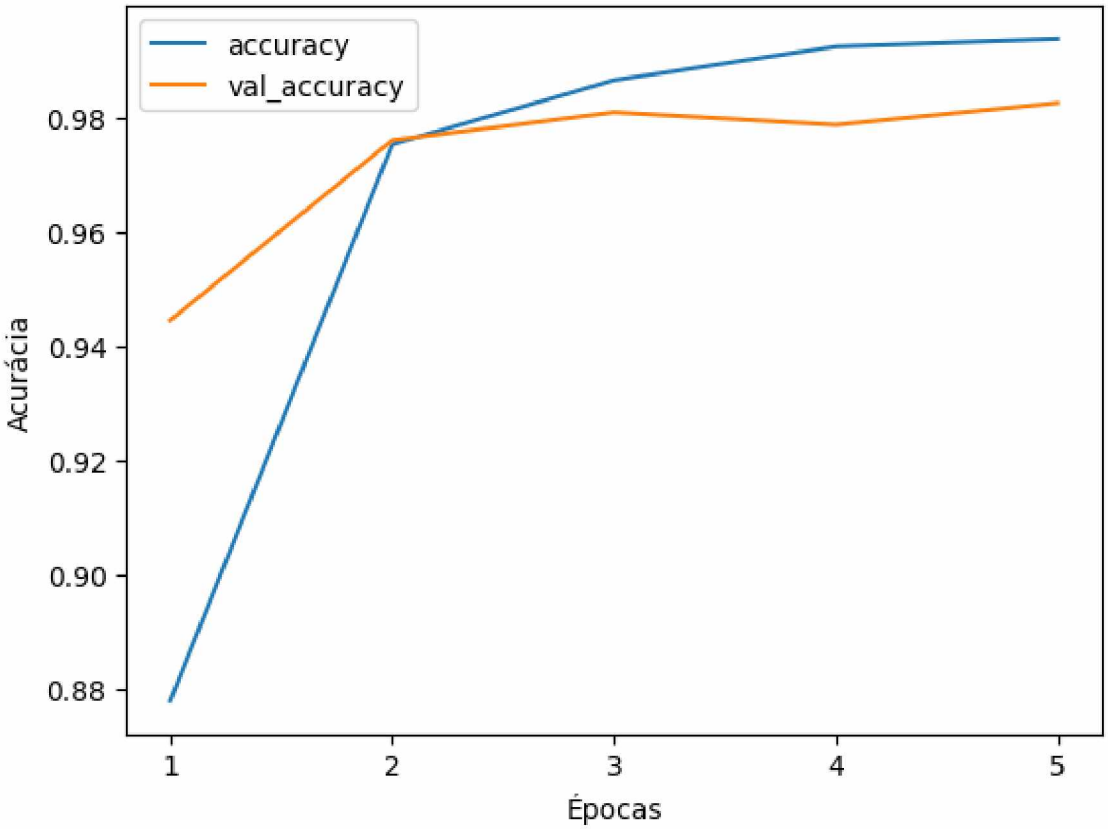
```
# %%
model.compile(loss="binary_crossentropy", optimizer="adam",metrics=["accuracy"])

epochs = 5

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

```
Epoch 1/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 6s 23ms/step - accuracy: 0.8192 - loss: 0.5039 -
val_accuracy: 0.9445 - val_loss: 0.2035
...
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step - accuracy: 0.9904 - loss: 0.0553 -
val_accuracy: 0.9826 - val_loss: 0.0781
```

```
# %%
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()
```





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

pred = model.predict(data_texto)
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
[[0.562104]]
SPAM
```

3 Gerador de Dígitos Fake (GAN)

CÓDIGO

```
# %% [markdown]
# Gerador de Dígitos Fake (GAN)

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

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

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

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

BUFFER_SIZE = 60000
BATCH_SIZE = 256

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

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

```
    assert model.output_shape == (None, 7, 7, 256)

    model.add(layers.Conv2DTranspose(128, (5, 5), strides=(1, 1),
                                     padding='same', use_bias=False))
    assert model.output_shape == (None, 7, 7, 128)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())
    model.add(layers.Conv2DTranspose(64, (5, 5), strides=(2, 2),
                                     padding='same', use_bias=False))
    assert model.output_shape == (None, 14, 14, 64)
    model.add(layers.BatchNormalization())
    model.add(layers.LeakyReLU())

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

    return model

# %%
generator = make_generator_model()

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

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

# %% [markdown]
# Criação do modelo do 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)) #camada densa com somente 1 neurônio

    return model

# %% [markdown]
# Testando o modelo do discriminador

# %%
discriminator = make_discriminator_model()
decision = discriminator(generated_image)
print(decision)

# %% [markdown]
# Funções de perda

# %%
```

```
cross_entropy = tf.keras.losses.BinaryCrossentropy(from_logits=True)

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

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

# %% [markdown]
# Criação dos otimizadores

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

# %% [markdown]
# Criação dos checkpoints

# %%
checkpoint_dir = './training_checkpoints'
checkpoint_prefix = os.path.join(checkpoint_dir, "ckpt")
checkpoint = tf.train.Checkpoint(generator_optimizer=generator_optimizer,
                                 discriminator_optimizer=discriminator_optimizer,
                                 generator=generator,
                                 discriminator=discriminator)

# %% [markdown]
# Início do loop de treinamento, criação das épocas

# %%
EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

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

# %% [markdown]
# ==> Passo de treinamento

# %%
@tf.function #para deixar a execução mais rápida
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) #função de perda
        disc_loss = discriminator_loss(real_output, fake_output) #perda do
discriminador

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

discriminator.trainable_variables)
```

```
    #aplicação dos gradientes no otimizador
    generator_optimizer.apply_gradients(zip(gradients_of_generator,
                                            generator.trainable_variables))
    #aplicação dos gradientes no discriminador
    discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,

discriminator.trainable_variables))

# %% [markdown]
# Chamado do passo de treinamento na chamada geral de treinamento

# %%
def generate_and_save_images(model, epoch, test_input):
    predictions = model(test_input, training=False)
    fig = plt.figure(figsize=[4,4])

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

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

# %%
def train(dataset, epochs):
    for epoch in range(epochs):
        start = time.time()
        for image_batch in dataset:
            train_step(image_batch)
        display.clear_output(wait=True)
        if (epoch + 1) % 15 == 0:
            checkpoint.save(file_prefix = checkpoint_prefix)
            print ('Time for epoch {} is {} sec'. \
                   format(epoch + 1, time.time()-start))
            generate_and_save_images(generator, epoch + 1, seed)

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

# %% [markdown]
# Criação da função generate_and_save_images

# %% [markdown]
# Enfim o treinamento do modelo

# %%
train(train_dataset, EPOCHS)
#restaura o último checkpoint
checkpoint.restore(tf.train.latest_checkpoint(checkpoint_dir))

# %% [markdown]
# criação do gif com a imagem gerada

# %%
def display_image(epoch_no):
    return PIL.Image.open('image_at_epoch_{:04d}.png'.format(epoch_no))

display_image(EPOCHS)
```

```
anim_file = 'dcgan.gif'
with imageio.get_writer(anim_file, mode='I') as writer:
    filenames = glob.glob('image*.png')
    filenames = sorted(filenames)
    last = -1
    for i,filename in enumerate(filenames):
        frame = 2*(i**0.5)
        if round(frame) > round(last):
            last = frame
        else:
            continue
            image = imageio.imread(filename)
            writer.append_data(image)
    image = imageio.imread(filename)
    writer.append_data(image)

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

4 Tradutor de Textos (Transformer)

CÓDIGO

```
# %%
!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

# %%
logging.getLogger('tensorflow').setLevel(logging.ERROR)

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

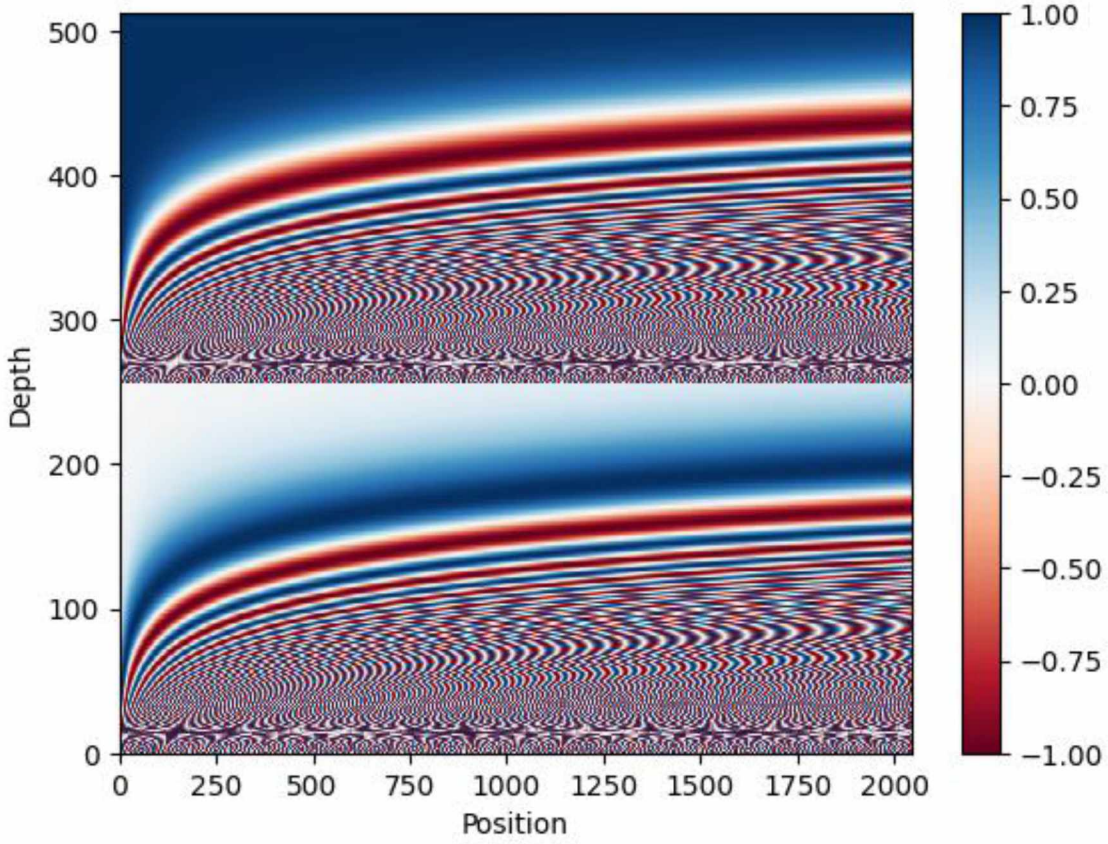
```
# %%
for pt_examples, en_examples in train_examples.batch(3).take(1):
  for pt in pt_examples.numpy():
    print(pt.decode( 'utf-8'))
  print()
  for en in en_examples.numpy():
    print(en.decode('utf-8'))
```

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 .

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

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

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

# %%
BUFFER_SIZE = 20000
BATCH_SIZE = 64

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

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

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

def positional_encoding(position, d_model):
  angle_rads = get_angles(np.arange(position)[: , np.newaxis], \
                          np.arange(d_model)[np.newaxis, :], d_model)
  angle_rads[:, 0::2] = np.sin(angle_rads[:, 0::2])
  angle_rads[:, 1::2] = np.cos(angle_rads[:, 1::2])
  pos_encoding = angle_rads[np.newaxis, ...]
  return tf.cast(pos_encoding, dtype=tf.float32)
```

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

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

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




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

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

```
# %%
def scaled_dot_product_attention(q, k, v, mask):

  matmul_qk = tf.matmul(q, k, transpose_b=True)
  dk = tf.cast(tf.shape(k)[-1], tf.float32)

  scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)
  if mask is not None:
    scaled_attention_logits += (mask * -1e9)

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

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

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

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

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

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

    scaled_attention, attention_weights = scaled_dot_product_attention(q, k, v,
mask)

    scaled_attention = tf.transpose(scaled_attention, perm=[0, 2, 1, 3])
    concat_attention = tf.reshape(scaled_attention, (batch_size, -1,
self.d_model))
    output = self.dense(concat_attention)
    return output, attention_weights

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

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

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

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

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

    return out2

# %%
class DecoderLayer(tf.keras.layers.Layer):
  def __init__(self, d_model, num_heads, dff, rate=0.1):
    super(DecoderLayer, self).__init__()
    self.mha1 = MultiHeadAttention(d_model, num_heads)
    self.mha2 = MultiHeadAttention(d_model, num_heads)
    self.ffn = point_wise_feed_forward_network(d_model, dff)
    self.layernorm1 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm2 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.layernorm3 = tf.keras.layers.LayerNormalization(epsilon=1e-6)
    self.dropout1 = tf.keras.layers.Dropout(rate)
    self.dropout2 = tf.keras.layers.Dropout(rate)
    self.dropout3 = tf.keras.layers.Dropout(rate)
  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):

    attn1, attn_weights_block1 = self.mha1(x, x, x, look_ahead_mask)
    attn1 = self.dropout1(attn1, training=training)
    out1 = self.layernorm1(attn1 + x)

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

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

    return out3, attn_weights_block1, attn_weights_block2
```

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

# %%
class Decoder(tf.keras.layers.Layer):
  def __init__(self, num_layers, d_model, num_heads, dff, target_vocab_size,
maximum_position_encoding, rate=0.1):
    super(Decoder, self).__init__()
    self.d_model = d_model
    self.num_layers = num_layers
    self.embedding = tf.keras.layers.Embedding(target_vocab_size, d_model)
    self.pos_encoding = positional_encoding(maximum_position_encoding, d_model)
    self.dec_layers = [DecoderLayer(d_model, num_heads, dff, rate) for _ in
range(num_layers)]
    self.dropout = tf.keras.layers.Dropout(rate)
  def call(self, x, enc_output, training, look_ahead_mask, padding_mask):
    seq_len = tf.shape(x)[1]
    attention_weights = {}
    x = self.embedding(x)
    x *= tf.math.sqrt(tf.cast(self.d_model, tf.float32))
    x += self.pos_encoding[:, :seq_len, :]
    x = self.dropout(x, training=training)
    for i in range(self.num_layers):
      x, block1, block2 = self.dec_layers[i](x, enc_output, training,
look_ahead_mask, padding_mask)
      attention_weights[f'decoder_layer{i+1}_block1'] = block1
      attention_weights[f'decoder_layer{i+1}_block2'] = block2
    return x, attention_weights
```

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

  def call(self, inputs, training):
    inp, tar = inputs

    enc_padding_mask, look_ahead_mask, dec_padding_mask = self.create_masks(inp,
tar)
    enc_output = self.encoder(inp, training, enc_padding_mask)
    dec_output, attention_weights = self.decoder(tar, enc_output, training,
look_ahead_mask, dec_padding_mask)
    final_output = self.final_layer(dec_output)
    return final_output, attention_weights

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

# %%
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)
    arg1 = tf.math.rsqrt(step)
    arg2 = step * (self.warmup_steps ** -1.5)
    return tf.math.rsqrt(self.d_model) * tf.math.minimum(arg1, arg2)

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

```
# %%
loss_object = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True,
reduction='none')
def loss_function(real, pred):
  mask = tf.math.logical_not(tf.math.equal(real, 0))
  loss_ = loss_object(real, pred)
  mask = tf.cast(mask, dtype=loss_.dtype)
  loss_ *= mask
  return tf.reduce_sum(loss_)/tf.reduce_sum(mask)

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

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

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

# %%
checkpoint_path = "./checkpoints/train"

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

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

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

# %%
EPOCHS = 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()
  for (batch, (inp, tar)) in enumerate(train_batches):
    train_step(inp, tar)
    if batch % 50 == 0:
      print(f'Epoch {epoch + 1} Batch {batch} Loss {train_loss.result():.4f}
Accuracy {train_accuracy.result():.4f}')
  if (epoch + 1) % 5 == 0:
    ckpt_save_path = ckpt_manager.save()
    print(f'Saving checkpoint for epoch {epoch+1} at {ckpt_save_path}')

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

```
Epoch 1 Batch 0 Loss 8.8531 Accuracy 0.0000
...
Epoch 1 Loss 6.6723 Accuracy 0.1179
Time taken for 1 epoch: 170.27 secs
...
Saving checkpoint for epoch 20 at ./checkpoints/train/ckpt-4
Epoch 20 Loss 1.8344 Accuracy 0.6138
Time taken for 1 epoch: 94.59 secs
```

```
# %%
class Translator(tf.Module):
  def __init__(self, tokenizers, transformer):
    self.tokenizers = tokenizers
    self.transformer = transformer
  def __call__(self, sentence, max_length=20):
    assert isinstance(sentence, tf.Tensor)
    if len(sentence.shape) == 0:
      sentence = sentence[tf.newaxis]
    sentence = self.tokenizers.pt.tokenize(sentence).to_tensor()
    encoder_input = sentence
    start_end = self.tokenizers.en.tokenize([''])[0]
    start = start_end[0][tf.newaxis]
    end = start_end[1][tf.newaxis]
    output_array = tf.TensorArray(dtype=tf.int64, size=0, dynamic_size=True)
    output_array = output_array.write(0, start)
    for i in tf.range(max_length):
      output = tf.transpose(output_array.stack())
      predictions, _ = self.transformer([encoder_input, output], training=False)
      predictions = predictions[:, -1:, :]
      predicted_id = tf.argmax(predictions, axis=-1)
      output_array = output_array.write(i+1, predicted_id[0])
      if predicted_id == end:
        break
    output = tf.transpose(output_array.stack())
    text = tokenizers.en.detokenize(output)[0]
```

```
    tokens = tokenizers.en.lookup(output)[0]
    _, attention_weights = self.transformer([encoder_input, output[:,:-1]],
training=False)
    return text, tokens, attention_weights

# %%
translator = Translator(tokenizers, transformer)

sentence = "Eu li sobre triceratops na enciclopédia."

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

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

Prediction     : b'i read about tridad and in the counsele .'
```

APÊNDICE I - BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

No ambiente digital interconectado, a área de entretenimento, especialmente a Netflix, destaca-se por oferecer uma experiência de streaming abrangente. Lançado em 2010 e disponível em mais de 190 países e 30 idiomas, a empresa é uma líder nesse segmento, oferecendo filmes, documentários, séries e jogos através de uma plataforma acessível em diversos dispositivos, desde streaming sticks até smart TVs. A inovação tecnológica da Netflix vai além do seu vasto catálogo, empregando soluções avançadas para garantir uma experiência de usuário sem interrupções.

Em 2019, a Netflix apresentou um estudo de caso no Kafka Summit sobre o "Netflix Trivia", um sistema interativo de perguntas e respostas para o processamento de dados em tempo real. Essa apresentação evidenciou a capacidade da empresa em integrar tecnologias avançadas e forneceu insights valiosos sobre como gerenciar grandes volumes de dados e proporcionar uma experiência de usuário fluida e responsiva. A base do Netflix Trivia é composta pelo Apache Kafka e pelo Apache Flink, plataformas de streaming dirigidas a eventos que permitem a ingestão e o processamento de dados em tempo real. A escolha do Kafka foi estratégica devido à sua capacidade de lidar com grandes volumes de dados e sua arquitetura distribuída, que oferece alta disponibilidade e resiliência. Isso também possibilitou à Netflix um melhor gerenciamento dos custos de criação e investimento em novos projetos, equilibrando a relação entre produção e custos.

A comunicação dirigida por requisições, como o modelo síncrono, pode levar ao caos e atrasos devido à complexidade dos fluxos de trabalho, necessidade de rastreabilidade e inconsistência em todo o sistema, além de gerar retrabalho. Em contraste, uma comunicação centrada em eventos, como a implementada com Kafka

e Flink, é mais eficiente. Esse modelo proporciona um fluxo canônico de fatos, desacoplamento dos componentes, e melhoria na gestão de dados e triggers. Além disso, oferece uma rastreabilidade eficiente através de logs.

No ecossistema Kafka da Netflix, a transição de uma comunicação complexa baseada em requisições síncronas para uma abordagem centrada em eventos é facilitada pela arquitetura de processamento de dados em tempo real oferecida por Kafka e Flink. Essas ferramentas permitem um processamento de streams eficiente, com alta tolerância a falhas, observabilidade nativa e facilidade na inicialização de listeners de eventos.

No contexto de uma empresa que utiliza a suíte Kafka, o tratamento de dados é dividido em três etapas principais: input (entrada), process (processamento) e output (saída). Na etapa de entrada, ferramentas como o Kafka são usadas para coletar e armazenar dados de várias fontes de forma desordenada, garantindo alta disponibilidade e escalabilidade. Durante a fase de processamento, o Apache Flink é frequentemente empregado. O Flink, com sua arquitetura robusta, realiza computações complexas sobre fluxos de dados contínuos com baixa latência, garantindo consistência de estado e possibilitando operações como junções e agregações. Finalmente, na etapa de saída, um stream Kafka ordenado e chaveado é utilizado para garantir que os dados processados sejam entregues de forma organizada e associados a chaves específicas. Um índice de busca pode ser empregado para permitir consultas rápidas e eficientes aos dados processados. Esse pipeline de dados robusto é essencial para empresas que precisam processar grandes volumes de dados em tempo real, mantendo a competitividade no mercado. A integração dessas ferramentas proporciona uma solução poderosa para a gestão de dados em ambientes corporativos, onde a velocidade e a precisão na entrega de informações são cruciais.

No controle do processo de produção de conteúdo da Netflix, as aplicações nas áreas de produção, cronograma, contas a pagar, tesouraria e custos são desenvolvidos como microserviços. Esses microserviços comunicam-se e recebem eventos através de tópicos do Kafka. No Flink, os eventos são recebidos na ordem em que chegam, vinculados ao ID do produto e passam por um processamento detalhado que inclui materialização com atraso para correção de dados, filtragem, agrupamento por janelas de tempo, ordenação cronológica, particionamento através de Partition Key, enriquecimento com dados de outros microserviços, transformação

e disponibilização final em tópicos do Kafka. Esse nível de desacoplamento e o uso do Flink garantem a manutenção do estado correto dos eventos, possibilitando a recuperação e a atualização de dados sem afetar as aplicações envolvidas.

A conclusão do estudo de caso destaca como o Big Data pode transformar uma empresa e aprimorar a experiência do usuário. A Netflix aplica os "5 V's" do Big Data – Volume, Velocidade, Variedade, Veracidade e Valor – para otimizar seus serviços e oferecer uma experiência personalizada aos seus assinantes.

Volume: A Netflix gerencia um imenso volume de dados diariamente. Cada clique, visualização e interação dos usuários geram uma quantidade significativa de informações. Esse volume colossal de dados é essencial para entender melhor os hábitos e preferências dos assinantes.

Velocidade: A agilidade no processamento dos dados é fundamental para a Netflix. A plataforma realiza análises em tempo real para oferecer recomendações instantâneas e ajustar a qualidade do streaming conforme necessário. Esse processamento rápido garante uma experiência de visualização contínua e sem interrupções.

Variedade: A Netflix coleta uma ampla gama de dados, desde informações estruturadas, como classificações e histórico de visualização, até dados não estruturados, como comentários e interações nas redes sociais. Essa diversidade permite uma análise mais abrangente e detalhada do comportamento dos usuários.

Veracidade: A qualidade e a confiabilidade dos dados são vitais para a Netflix. A empresa implementa rigorosos processos de verificação para garantir a precisão e a utilidade dos dados. Esse compromisso com a veracidade é essencial para fornecer recomendações relevantes e tomar decisões estratégicas bem-informadas.

Valor: O verdadeiro valor do Big Data para a Netflix está na capacidade de transformar essas informações em insights acionáveis. Através da análise de dados, a Netflix personaliza a experiência de cada usuário, sugerindo filmes e séries que correspondem aos seus interesses individuais. Isso não só aumenta a satisfação dos clientes, mas também contribui para a retenção e lealdade dos assinantes.

APÊNDICE J - VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

2) Redes Neurais

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

Tarefas:

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

B – RESOLUÇÃO

QUESTÃO 1

METODOLOGIA UTILIZADA

A base de imagens de treino original, contida no arquivo Train_Warwick.zip, foi dividida manualmente em bases de treino e validação, conforme orientação do enunciado do exercício, ou seja, 80% das imagens para treino e 20% para validação, segregando os pacientes, fazendo com que não houvesse imagens de um mesmo paciente simultaneamente em ambas. Desta forma, a divisão apresentou imagens de 16 (dezesseis) pacientes na base de treino (pacientes 1, 4, 6, 9, 11, 15, 16, 24, 25, 29, 32, 46 e 57) e de 4 (quatro) pacientes na base de validação (pacientes 14, 18, 22 e 36).

Para controlar o processo de leitura dos arquivos de imagem de forma correta para cada tipo de base (treinamento, validação e teste), foram criados 3 (três arquivos) texto: Test.txt, Valid.txt e Test.txt, contendo respectivamente os caminhos relativos dos arquivos de imagem das bases de treino, validação e teste. Esses arquivos são compactados no arquivo FilePaths.tar, de forma que possam ser descompactados no local correto, através de código no notebook, quando da sua execução. Para que o código execute corretamente, previamente a sua execução, as imagens devem estar divididas em pastas de forma idêntica ao que está apontado nesses arquivos texto. Para o desenvolvimento e treinamento dos modelos de classificação propostos, foi utilizada a biblioteca Python scikit-learn. Para o desenvolvimento e treinamento dos modelos de redes neurais convolucionais profundas, foi utilizada a biblioteca Tensorflow.

COMPARAÇÃO DOS MODELOS E MÉTRICAS UTILIZADAS

A análise comparativa entre os modelos Random Forest, SVM e RNA para predição dos dados extraídos, revelou diferenças significativas no desempenho e na influência de cada tipo de técnica sobre os resultados obtidos. Além disso, pode-se observar diferenças significativas nos resultados de cada uma das técnicas, dependendo do tipo de features submetidas a elas como insumo para o treinamento: features geradas pelo processo de LBP ou features geradas pela rede neural convolucional VGG16 carregada com os pesos pré-treinados com a base imagenet.

Features LBP	SVM	Random Forest	RNA
Acurácia	66,85%	69,81%	56,87%
Sensibilidade	67,26%	70,55%	56,52%
Especificidade	88,96%	89,96%	85,55%
F1-Score	67,19%	69,40%	52,88%

Quadro 1

Utilizando as features LBP, a especificidade das classificações está relativamente equilibrada entre os algoritmos, com o SVM e o Random Forest apresentando desempenho próximo em todas as métricas e a RNA pior em relação a ambas.

Features VGG16	SVM	Random Forest	RNA
Acurácia	76,82%	75,74%	76,01%
Sensibilidade	76,14%	75,24%	76,22%
Especificidade	92,27%	91,90%	92,07%
F1-Score	72,29%	72,24%	75,64%

Quadro 2

Com as features VGG16, as métricas produzidas pelos três modelos distintos de classificação ficam muito próximas umas das outras, com vantagem para um modelo ou outro dependendo da métrica observada.

Observa-se também que houve uma melhora significativa nas métricas em relação aos modelos treinados com as features geradas pelo processo LBP.

Como o problema em questão envolve a detecção de câncer de mama, deve-se dar especial ênfase às predições falso negativas, já que o risco para o paciente é menor de receber um diagnóstico falso positivo do que ser diagnosticado como negativo, quando na verdade existe a doença. Desta forma, a principal métrica

utilizada na avaliação deve ser a Sensibilidade, o que leva a classificar como o melhor modelo, o RNA treinado com as features VGG16, já que ele apresentou a maior sensibilidade: 76,22%, mesmo tendo o segundo melhor índice de acurácia.

CÓDIGO

```
# %% [markdown]
# # **Preparação do Ambiente**
#
#

# %% [markdown]
# ## Define as constantes globais que serão utilizadas

# %%
BATCH_SIZE = 32 # quantidade de imagens criadas em cada ciclo

WORK_FOLDER = '.' # Pasta base onde estão os dados
BASES_FOLDER = WORK_FOLDER + '/Bases' # Pasta onde estão as bases de dados
FEATURES_FOLDER = WORK_FOLDER + '/Features' # Pasta onde estão as features

TRAINING_TXTFILE_PATH = WORK_FOLDER + '/Train.txt' # Caminho para o arquivo de
paths de treinamento
VALIDATION_TXTFILE_PATH = WORK_FOLDER + '/Valid.txt' # Caminho para o arquivo de
paths de validação
TEST_TXTFILE_PATH = WORK_FOLDER + '/Test.txt' # Caminho para o arquivo de paths de
teste

TRAINING_BASE_PATH = BASES_FOLDER + '/Train' # Caminho para a base de treinamento
VALIDATION_BASE_PATH = BASES_FOLDER + '/Valid' # Caminho para a base de validação
TEST_BASE_PATH = BASES_FOLDER + '/Test' # Caminho para a base de teste

# define os parametros para geração do LBP
LBP_RADIUS = 1
LBP_NUM_POINTS = 8

# %% [markdown]
# ## Importação das bibliotecas utilizadas

# %%
import os
import numpy as np
import cv2
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from skimage.feature import local_binary_pattern
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Flatten, Dense
from keras.models import Model
```

```
# %% [markdown]
# Verifica se tem uma GPU disponível

# %%
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

# %% [markdown]
# ## Carga dos dados
# Os arquivo Bases.tar e FilePaths.tar devem ser colocados na pasta definida na
variável BASE_FOLDER. Os arquivos de validação já estão separados no diretório
Valid, respeitando a proporção de 20% da imagens e sem pacientes que estão na base
de treino.

# %%
!tar -xvf ./Bases.tar -C ./
!tar -xvf ./FilePaths.tar -C ./

# %%
# Cria os diretórios para armazenar as features
os.makedirs(FEATURES_FOLDER, exist_ok=True)

# %% [markdown]
# # **Extração de Características**

# %% [markdown]
# ## Extração de features para a RIU LBP
#
# LBP uniforme e invariante a rotação. A quantidade de características finais são
**P+2**, onde **P** é a quantidade de vizinhos utilizados.

# %% [markdown]
# ### Cálculo do LBP

# %%
def lbp_riu(img, n_points:int, radius:int):
    """
    Função LBP para extração das features.
    Gera as LBP features para cada imagem e persiste num arquivo.
    @param img: Object da Imagem
    @param n_points: Número de pontos da vizinhança ( P )
    @param radius: Raio da vizinhança do ponto central ( R )
    @return: feature_array: o array com as features LBP, com P+2 posições
        lbp: a imagem LBP com o mesmo tamanho de img
    """
    n_points *= radius # adjust # of neighbours according to radius
    # Define a função para cálculo do LBP
    lbp = local_binary_pattern(img, n_points, radius, method='uniform')
    #Array de zeros com P+2 posições +1 para o lable
    feature_array = np.zeros(n_points+2, dtype=int)
    rows = img.shape[0]
    cols = img.shape[1]
    for r in range (0, rows):
        for c in range (0, cols):
            feature_array[int(lbp[r][c])] += 1

    return feature_array, lbp
```

```
# %% [markdown]
# ### Leitura das Imagens, geração das features LBP e gravação do arquivo CSV
#

# %%
def calc_features(arq):
    """
    Função que recebe o caminho do arquivo, gera as LBP features
    e persiste num arquivo, com a classe na primeira posição da linha do CSV.
    @param arq: Caminho do arquivo da imagem.
    @return: None
    """
    # abre o arquivo com os caminhos dos arquivos e processa um por um
    img_path = open(arq, 'r')
    for line in img_path:
        label = line.rstrip('\n').split('/')[-2]
        ftype = line.rstrip('\n').split('/')[-3]
        nome_arq =
f'{FEATURES_FOLDER}/lbp_riu_{LBP_NUM_POINTS}_{LBP_RADIUS}_{ftype}.csv'
        # le imagem original
        img = cv2.imread(line.strip(),0)
        # Verifica se a imagem foi carregada corretamente
        # caso contrário exibe uma msg de erro e continua
        if img is None:
            print(f"Erro ao carregar a imagem: {line.strip()}")
            continue
        # Garante que a imagem seja bidimensional (escala de cinza)
        if len(img.shape) > 2:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # chama a funcão que calcula as features LBP
        features, lbp = lbp_riu(img, LBP_NUM_POINTS, LBP_RADIUS)
        # formata uma string com as features e a classe
        # cria a linha no formato texto svmlight para salvar
        features_str = '%s' % (label)
        for P in features:
            features_str += ",%d" % (P)
        features_str += '\n'
        # persiste a linha das features extraídas no arquivo texto
        arquivo = open(nome_arq, 'a')
        arquivo.write(features_str)
        arquivo.close()
    # fecha o arquivo com os caminhos
    img_path.close()

# %%
print('Gerando Features da base de Treino...')
calc_features(TRAINING_TXTFILE_PATH)
print('Features de Treino geradas!')

print('Gerando Features da base de Validação...')
calc_features(VALIDATION_TXTFILE_PATH)
print('Features de Validação geradas!')

print('Gerando Features da base de Teste...')
calc_features(TEST_TXTFILE_PATH)
print('Features de Teste geradas!')
```

```
# %% [markdown]
# ## Extração de features para VGG16

# %% [markdown]
# ### Monta rede VGG16 já com os pesos imagenet carregados e remove a camada de
classificação

# %%
vgg_0 = VGG16(input_shape=(224,224,3),
              weights='imagenet',
              include_top=False)

flatten = Flatten()(vgg_0.output)
vgg = Model(inputs=vgg_0.input, outputs=flatten)

for l in vgg_0.layers:
    l.trainable = False

vgg.summary() # Impressão das arquitetura da rede
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808
```

```
 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

=================================================================
Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

# %% [markdown]
# ### Configura os objetos e bases de Treino e Teste

# %% [markdown]
# Cria os objetos geradores da imagens

# %%
train_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
valid_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)

# %% [markdown]
# Cria os objetos de treinamento e teste

# %%
traingen = train_generator.flow_from_directory(TRAINING_BASE_PATH,
                                               target_size=(224, 224),
                                               batch_size=BATCH_SIZE,
                                               class_mode='categorical',
                                               classes=['0','1','2','3'],
                                               shuffle=False,
                                               seed=42)

validgen = valid_generator.flow_from_directory(VALIDATION_BASE_PATH,
                                               target_size=(224, 224),
                                               batch_size=BATCH_SIZE,
                                               class_mode=None,
                                               classes=['0','1','2','3'],
                                               shuffle=False,
                                               seed=42)

testgen = test_generator.flow_from_directory(TEST_BASE_PATH,
                                             target_size=(224, 224),
                                             batch_size=BATCH_SIZE,
                                             class_mode=None,
                                             classes=['0','1','2','3'],
                                             shuffle=False,
                                             seed=42)

Found 478 images belonging to 4 classes.
Found 115 images belonging to 4 classes.
Found 371 images belonging to 4 classes.

# %% [markdown]
# ### Extrai as features para a rede VGG16 e grava os arquivos CSV

# %% [markdown]
# Coleta as features executando o modelo contra as imagens de treino carregadas
```

```
# %%
# gera as features para a base de treino
print('Gerando Features da base de Treino...')
features_vgg_train = vgg.predict(traingen)
print('Features de Treino Geradas!')

# gera as features para a base de validação
print('---------------------------------------')
print('Gerando Features da base de Validação...')
features_vgg_valid = vgg.predict(validgen)
print('Features de Validação Geradas!')

# gera as features para a base de teste
print('---------------------------------------')
print('Gerando Features da base de Teste...')
features_vgg_test = vgg.predict(testgen)
print('Features de Teste Geradas!')
```

```
Gerando Features da base de Treino...
15/15 [==============================] - 48s 2s/step
Features de Treino Geradas!
---------------------------------------
Gerando Features da base de Validação...
4/4 [==============================] - 17s 5s/step
Features de Validação Geradas!
---------------------------------------
Gerando Features da base de Teste...
12/12 [==============================] - 6s 540ms/step
Features de Teste Geradas!
```

```
# %% [markdown]
# Salva os arquivos CSV com as features das bases

# %%
def extrai_features_vgg(tipo_base:str):
    if tipo_base == 'Train':
        generator = traingen
        features = features_vgg_train
    elif tipo_base == 'Valid':
        generator = validgen
        features = features_vgg_valid
    elif tipo_base == 'Test':
        generator = testgen
        features = features_vgg_test
    else:
        raise ValueError('Valor de parâmetro inválido para tipo-base')

    nome_arq = f'{FEATURES_FOLDER}/vgg16_{tipo_base}.csv'
    for i in range(0, len(generator.labels)):
        # formata uma string com as features e a classe
        # cria a linha no formato texto svmlight para salvar
        features_str = '%d' % (generator.labels[i])
        for ft in features[i]:
            # para poupar espaço no arquivo texto
            if ft == 0:
                features_str += ",%d" % (ft)
            else:
                features_str += ",%f" % (ft)
```

```
        features_str += '\n'
        # persiste a linha das features extraídas no arquivo texto
        arquivo = open(nome_arq, 'a')
        arquivo.write(features_str)
        arquivo.close()

# %%
print("Salvando features da base de Treino...")
extrai_features_vgg('Train')
print("Salvando features da base de Validação...")
extrai_features_vgg('Valid')
print("Salvando features da base de Teste...")
extrai_features_vgg('Test')
print("fim")

# %% [markdown]
# # **Treinamento dos modelos**

# %% [markdown]
# ## Treinamento com as features LBP e VGG16

# %% [markdown]
# ### Carga das Features de Treino e Validação LBP e VGG16

# %%
# loads data
# x sempre e a matriz de caracteristicas
# y sempre e o vetor de classes

print ("Carregando features LBP...")
dados_train =
pd.read_csv(f"{FEATURES_FOLDER}/lbp_riu_{LBP_NUM_POINTS}_{LBP_RADIUS}_Train.csv",
header=None)
dados_valid =
pd.read_csv(f"{FEATURES_FOLDER}/lbp_riu_{LBP_NUM_POINTS}_{LBP_RADIUS}_Valid.csv",
header=None)

X_train_lbp = dados_train.iloc[:, 1:]
y_train_lbp = dados_train.iloc[:, 0]

X_valid_lbp = dados_valid.iloc[:, 1:]
y_valid_lbp = dados_valid.iloc[:, 0]
print("Features LBP Carregadas")

print("Carregando features VGG16...")
dados_train_vgg = pd.read_csv(f"{FEATURES_FOLDER}/vgg16_Train.csv", header=None)
dados_valid_vgg = pd.read_csv(f"{FEATURES_FOLDER}/vgg16_Valid.csv", header=None)

X_train_vgg = dados_train_vgg.iloc[:, 1:]
y_train_vgg = dados_train_vgg.iloc[:, 0]

X_valid_vgg = dados_valid_vgg.iloc[:, 1:]
y_valid_vgg = dados_valid_vgg.iloc[:, 0]
print("Features VGG16 Carregadas")

# %% [markdown]
# ### Criação e Treinamento do modelo SVM com features LBP
```

```
# %%
# Cria uma instância do SVM
model_svm_lbp = svm.SVC(kernel='linear')

print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_svm_lbp.fit(X_train_lbp, y_train_lbp)
print (f"Treinamento finalizado - kernel:{model_svm_lbp.kernel}")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_svm = model_svm_lbp.predict(X_valid_lbp)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_svm_lbp}:\n"
      f"{classification_report(y_valid_lbp, predicted_svm)}")
```

```
Treinando Modelo...
Treinamento finalizado - kernel:linear
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador
SVC(kernel='linear'):
              precision    recall  f1-score   support

           0       0.78      0.75      0.76        28
           1       0.50      0.07      0.13        27
           2       0.52      0.90      0.66        30
           3       0.88      0.93      0.90        30

    accuracy                           0.68       115
   macro avg       0.67      0.66      0.61       115
weighted avg       0.67      0.68      0.62       115
```

```
# %% [markdown]
# ### Criação e Treinamento do modelo SVM com features VGG16

# %%
# Cria uma instância do SVM
model_svm_vgg = svm.SVC(kernel='rbf')

print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_svm_vgg.fit(X_train_vgg, y_train_vgg)
print (f"Treinamento finalizado - kernel:{model_svm_vgg.kernel}")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_svm = model_svm_vgg.predict(X_valid_vgg)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_svm_vgg}:\n"
      f"{classification_report(y_valid_vgg, predicted_svm)}")
```

```
Treinando Modelo...
Treinamento finalizado - kernel:rbf
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador SVC():
              precision    recall  f1-score   support

           0       0.62      1.00      0.77        28
           1       1.00      0.30      0.46        27
           2       0.92      0.77      0.84        30
           3       0.81      1.00      0.90        30

    accuracy                           0.77       115
   macro avg       0.84      0.77      0.74       115
weighted avg       0.84      0.77      0.75       115
```

```
# %% [markdown]
# ### Criação de Treinamento do modelo Random Forest com features LBP
```

```
# %%
# Cria uma instância do RandomForest
model_rf_lbp = RandomForestClassifier()

print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_rf_lbp.fit(X_train_lbp, y_train_lbp)
print (f"Treinamento finalizado.")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_rf = model_rf_lbp.predict(X_valid_lbp)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_rf_lbp}:\n"
      f"{classification_report(y_valid_lbp, predicted_rf)}")
```

```
Treinando Modelo...
Treinamento finalizado.
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador
RandomForestClassifier():
              precision    recall  f1-score   support

           0       0.72      0.75      0.74        28
           1       0.71      0.37      0.49        27
           2       0.65      0.87      0.74        30
           3       0.91      0.97      0.94        30

    accuracy                           0.75       115
   macro avg       0.75      0.74      0.73       115
weighted avg       0.75      0.75      0.73       115
```

```
# %% [markdown]
# ### Criação de Treinamento do modelo Random Forest com features VGG16
```

```
# %%
# Cria uma instância do RandomForest
model_rf_vgg = RandomForestClassifier()
print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_rf_vgg.fit(X_train_vgg, y_train_vgg)
print (f"Treinamento finalizado.")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_rf = model_rf_vgg.predict(X_valid_vgg)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_rf_vgg}:\n"
      f"{classification_report(y_valid_vgg, predicted_rf)}")
```

```
Treinando Modelo...
Treinamento finalizado.
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador
RandomForestClassifier():
              precision    recall  f1-score   support

           0       0.59      0.96      0.73        28
           1       0.67      0.15      0.24        27
           2       0.85      0.73      0.79        30
           3       0.81      1.00      0.90        30

    accuracy                           0.72       115
   macro avg       0.73      0.71      0.66       115
weighted avg       0.73      0.72      0.67       115
```

```
# %% [markdown]
# ### Criação e Treinamento do modelo RNA com features LBP

# %%
# Cria uma instância do MLP
model_mlp_lbp = MLPClassifier(random_state=1, max_iter=300,
learning_rate='adaptive')

print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_mlp_lbp.fit(X_train_lbp, y_train_lbp)
print (f"Treinamento finalizado.")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_mlp = model_mlp_lbp.predict(X_valid_lbp)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_mlp_lbp}:\n"
      f"{classification_report(y_valid_lbp, predicted_mlp)}")
```

```
Treinando Modelo...
Treinamento finalizado.
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador
MLPClassifier(learning_rate='adaptive', max_iter=300, random_state=1):
              precision    recall  f1-score   support

           0       0.50      0.57      0.53        28
           1       0.00      0.00      0.00        27
           2       0.38      0.63      0.47        30
           3       0.61      0.67      0.63        30

    accuracy                           0.48       115
   macro avg       0.37      0.47      0.41       115
weighted avg       0.38      0.48      0.42       115
```

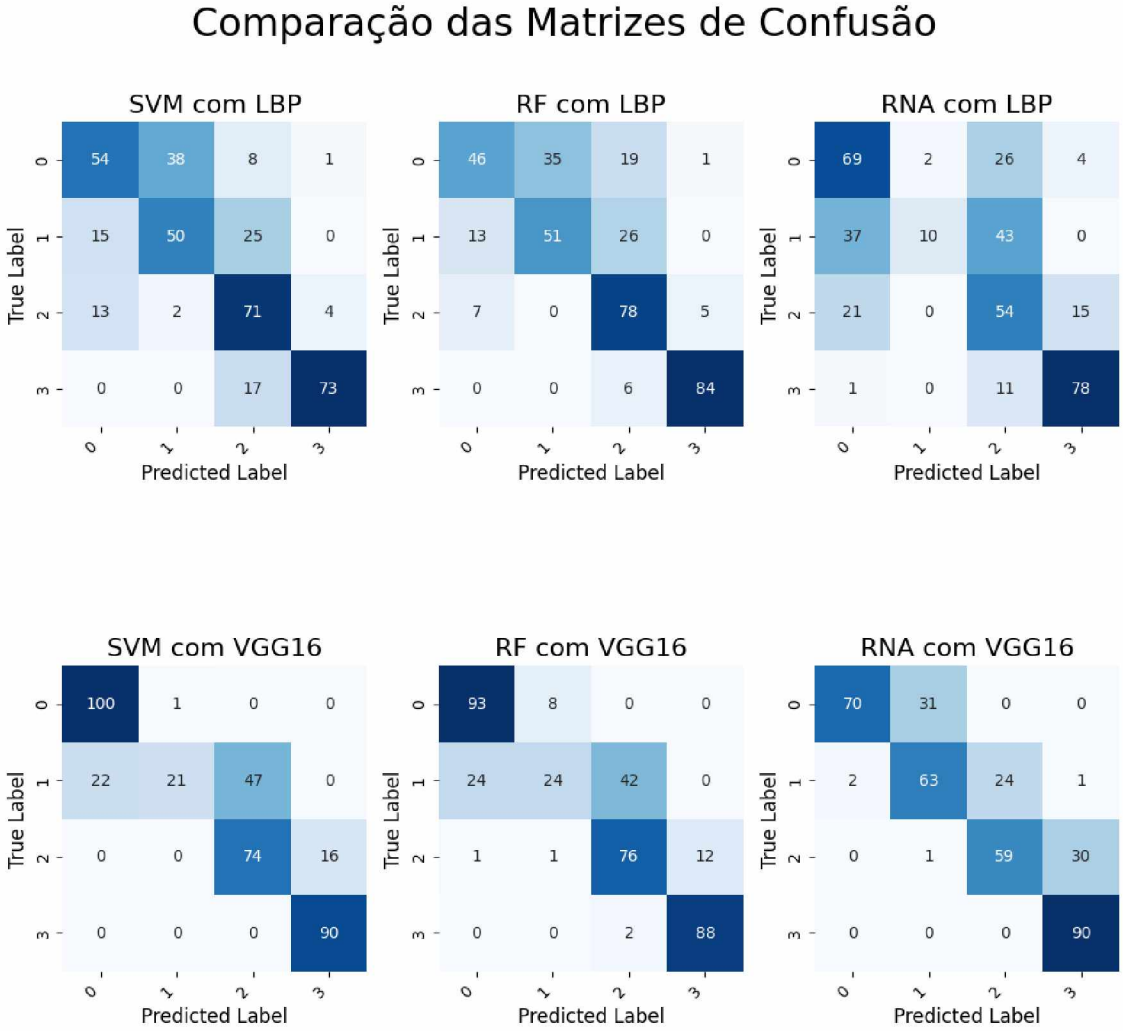
```
# %% [markdown]
# ### Criação e Treinamento do modelo RNA com features VGG16

# %%
# Cria uma instância do MLP
model_mlp_vgg = MLPClassifier(random_state=1, max_iter=300,
learning_rate='adaptive')

print ("Treinando Modelo...")
# Treina o modelo SVM com os dados carregados
model_mlp_vgg.fit(X_train_vgg, y_train_vgg)
print (f"Treinamento finalizado.")
print ("Efetuando Predições com a base de validação para validar o modelo...")
# Fazendo previsões nos dados de validação
predicted_mlp = model_mlp_vgg.predict(X_valid_vgg)
print ("Predições Efetuadas!")
print(f"Classification report para predições com a base de validação.
Classificador {model_mlp_vgg}:\n"
      f"{classification_report(y_valid_vgg, predicted_mlp)}")
```

```
Treinando Modelo...
Treinamento finalizado.
Efetuando Predições com a base de validação para validar o modelo...
Predições Efetuadas!
Classification report para predições com a base de validação. Classificador
MLPClassifier(learning_rate='adaptive', max_iter=300, random_state=1):
              precision    recall  f1-score   support

           0       0.85      0.82      0.84        28
           1       0.72      0.48      0.58        27
           2       0.79      0.50      0.61        30
           3       0.59      1.00      0.74        30

    accuracy                           0.70       115
   macro avg       0.74      0.70      0.69       115
weighted avg       0.74      0.70      0.69       115
```

```
# %% [markdown]
# # **Predições e Métricas**

# %% [markdown]
# ### Carga das Features de Teste LBP e VGG16

# %%
# loads data
# x sempre e a matriz de caracteristicas
# y sempre e o vetor de classes

print ("Carregando features LBP...")
dados_test_lbp =
pd.read_csv(f"{FEATURES_FOLDER}/lbp_riu_{LBP_NUM_POINTS}_{LBP_RADIUS}_Test.csv",
header=None)
X_test_lbp = dados_test_lbp.iloc[:, 1:]
y_test_lbp = dados_test_lbp.iloc[:, 0]
print("Features LBP Carregadas")

print("Carregando features VGG16...")
dados_test_vgg = pd.read_csv(f"{FEATURES_FOLDER}/vgg16_Test.csv", header=None)
X_test_vgg = dados_test_vgg.iloc[:, 1:]
y_test_vgg = dados_test_vgg.iloc[:, 0]
print("Features VGG16 Carregadas")

# %% [markdown]
# ### Predições dos modelos SVM na base de Teste com features LBP e VGG16

# %%
print ("Efetuando Predições...")

final_predict_svm_lbp = model_svm_lbp.predict(X_test_lbp)
final_predict_svm_vgg = model_svm_vgg.predict(X_test_vgg)
final_predict_rf_lbp = model_rf_lbp.predict(X_test_lbp)
final_predict_rf_vgg = model_rf_vgg.predict(X_test_vgg)
final_predict_mlp_lbp = model_mlp_lbp.predict(X_test_lbp)
final_predict_mlp_vgg = model_mlp_vgg.predict(X_test_vgg)

print ("Predições Efetuadas!")

# %% [markdown]
# # **Comparação de resultados dos modelos**

# %% [markdown]
# ## Compara o desempenho dos modelos com features LBP e VGG16

# %% [markdown]
# ### Comparação de Métricas dos modelos

# %%
def imprime_metricas_modelo(true_classes, pred_classes, nome_modelo):

    def especificidade(true_classes, pred_classes): # Define uma função para
calcular a especificidade que não existe no sklearn
        cm = confusion_matrix(true_classes, pred_classes)
        specificities = []
        for i in range(len(cm)):
            # Exclude row i and column i to get TN
```

```
            tn = np.sum(np.delete(np.delete(cm, i, axis=0), i, axis=1))
            fp = np.sum(np.delete(cm, i, axis=0)[:, i])
            specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
            specificities.append(specificity)
        return np.mean(specificities)

    # Calcula as métricas
    acuracia = accuracy_score(true_classes, pred_classes)
    sensibilidade = recall_score(true_classes, pred_classes, average='macro')
    especificidade = especificidade(true_classes, pred_classes)
    f1 = f1_score(true_classes, pred_classes, average='macro')

    print("-----------------------------------------------------")
    print(f"Acurácia Modelo {nome_modelo}: {(acuracia * 100):.2f}%")
    print(f"Sensibilidade Modelo {nome_modelo}: {(sensibilidade * 100):.2f}%")
    print(f"Especificidade Modelo {nome_modelo}: {(especificidade * 100):.2f}%")
    print(f"F1 Modelo {nome_modelo}: {(f1 * 100):.2f}%")

# %%
# Get the names of the ten classes
class_names = ['0','1','2','3']

def plot_heatmap(y_true, y_pred, class_names, ax, title):
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(
        cm,
        annot=True,
        square=True,
        xticklabels=class_names,
        yticklabels=class_names,
        fmt='d',
        cmap=plt.cm.Blues,
        cbar=False,
        ax=ax
    )
    ax.set_title(title, fontsize=16)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right")
    ax.set_ylabel('True Label', fontsize=12)
    ax.set_xlabel('Predicted Label', fontsize=12)

fig, ((ax1, ax2, ax3), (ax4, ax5, ax6)) = plt.subplots(2, 3, figsize=(10, 10))

plot_heatmap(y_test_lbp, final_predict_svm_lbp, class_names, ax1, \
             title="SVM com LBP")
plot_heatmap(y_test_lbp, final_predict_rf_lbp, class_names, ax2, \
             title="RF com LBP")
plot_heatmap(y_test_lbp, final_predict_mlp_lbp, class_names, ax3, \
             title="RNA com LBP")
plot_heatmap(y_test_vgg, final_predict_svm_vgg, class_names, ax4, \
             title="SVM com VGG16")
plot_heatmap(y_test_vgg, final_predict_rf_vgg, class_names, ax5, \
             title="RF com VGG16")
plot_heatmap(y_test_vgg, final_predict_mlp_vgg, class_names, ax6, \
             title="RNA com VGG16")

fig.suptitle("Comparação das Matrizes de Confusão", fontsize=24)
fig.tight_layout()
fig.subplots_adjust(top=1)
plt.show()
```

```
imprime_metricas_modelo(y_test_lbp, final_predict_svm_lbp, "SVM com LBP")
imprime_metricas_modelo(y_test_vgg, final_predict_svm_vgg, "SVM com VGG16")
imprime_metricas_modelo(y_test_lbp, final_predict_rf_lbp, "RF com LBP")
imprime_metricas_modelo(y_test_vgg, final_predict_rf_vgg, "RF com VGG16")
imprime_metricas_modelo(y_test_lbp, final_predict_mlp_lbp, "RNA com LBP")
imprime_metricas_modelo(y_test_vgg, final_predict_mlp_vgg, "RNA com VGG16")
```




```
-----------------------------------------------------
Acurácia Modelo SVM com LBP: 66.85%
Sensibilidade Modelo SVM com LBP: 67.26%
Especificidade Modelo SVM com LBP: 88.96%
F1 Modelo SVM com LBP: 67.19%
-----------------------------------------------------
Acurácia Modelo SVM com VGG16: 76.82%
Sensibilidade Modelo SVM com VGG16: 76.14%
Especificidade Modelo SVM com VGG16: 92.27%
F1 Modelo SVM com VGG16: 72.29%
-----------------------------------------------------
Acurácia Modelo RF com LBP: 69.81%
Sensibilidade Modelo RF com LBP: 70.55%
Especificidade Modelo RF com LBP: 89.96%
F1 Modelo RF com LBP: 69.40%
```

```
--------------------------------------------------
Acurácia Modelo RF com VGG16: 75.74%
Sensibilidade Modelo RF com VGG16: 75.24%
Especificidade Modelo RF com VGG16: 91.90%
F1 Modelo RF com VGG16: 72.24%
--------------------------------------------------
Acurácia Modelo RNA com LBP: 56.87%
Sensibilidade Modelo RNA com LBP: 56.52%
Especificidade Modelo RNA com LBP: 85.55%
F1 Modelo RNA com LBP: 52.88%
--------------------------------------------------
Acurácia Modelo RNA com VGG16: 76.01%
Sensibilidade Modelo RNA com VGG16: 76.22%
Especificidade Modelo RNA com VGG16: 92.07%
F1 Modelo RNA com VGG16: 75.64%
```

QUESTÃO 2

COMPARAÇÃO DOS MODELOS E MÉTRICAS UTILIZADAS

A análise comparativa entre os modelos ResNet50 e VGG16, com e sem Data Augmentation, revelou diferenças claras no desempenho e na influência dessa técnica sobre os resultados.

ResNet50	Sem Data Augmentation	Com Data Augmentation
Acurácia	93,53%	81,67%
Sensibilidade	93,79%	81,11%
Especificidade	97,86%	93,88%
F1-Score	93,58%	78,86%

Quadro 3

O modelo ResNet50 com a camada TOP adaptada treinada sem Data Augmentation, apresentou as melhores métricas, em relação ao modelo treinado utilizando Data Augmentation. Esses valores indicam que o modelo conseguiu generalizar adequadamente os padrões do conjunto de teste, mostrando um desempenho consistente sem a necessidade de técnicas adicionais para manipulação de dados. A ausência de Data Augmentation não comprometeu a sua capacidade de lidar com os dados de teste.

Observa-se que a utilização do Data Augmentation no treinamento da camada TOP da Resnet50 adaptada ao problema de quatro classes produziu métricas relativamente piores. Esta diminuição sugere que o aumento artificial da variabilidade dos dados não contribuiu significativamente para o treinamento do modelo, o que pode ter sido causado pela variabilidade de brilho introduzida, que pode ter gerado distorções de cor importantes nos exemplos de algumas classes submetidos à rede.

VGG16	Sem Data Augmentation	Com Data Augmentation
Acurácia	81,67%	81,13%
Sensibilidade	81,53%	80,59%
Especificidade	93,94%	93,74%
F1-Score	81,00%	79,01%

Quadro 4

O modelo VGG16 com a camada TOP adaptada, apresentou desempenho geral inferior ao ResNet50. Sem Data Augmentation, o VGG16 demonstra uma boa capacidade de generalização, embora significativamente inferior em comparação ao ResNet50. Com a aplicação de Data Augmentation, o VGG16 também sofreu uma

queda nas métricas, porém menos significativa. Esses resultados indicam que a técnica não apenas não ajudou os modelos, mas também prejudicou seu desempenho, possivelmente ao aumentar a dificuldade do treinamento sem um benefício proporcional.

Em suma, os resultados mostram que o **ResNet50 sem Data Augmentation** foi o modelo com o melhor desempenho em todas as métricas, inclusive na Sensibilidade, já comentada no primeiro exercício, como a melhor para a avaliação deste tipo de problema (detecção de câncer de mama)

CÓDIGO

```
# %% [markdown]
# # **Preparação do Ambiente**

# %% [markdown]
# ## Instalação de módulos necessários

# %%
# Módulo para imprimir os gráficos de treinamento de forma dinâmica.
!pip install livelossplot

# %% [markdown]
# ## Define as constantes globais que serão utilizadas

# %%
BATCH_SIZE = 32 # quantidade de imagens criadas em cada ciclo

BASE_FOLDER = '.' # Pasta base onde estão os dados

TRAINING_BASE_PATH = BASE_FOLDER + '/Bases/Train' # Caminho para a base de
treinamento
VALIDATION_BASE_PATH = BASE_FOLDER + '/Bases/Valid' # Caminho para a base de
validação
TEST_BASE_PATH = BASE_FOLDER + '/Bases/Test' # Caminho para a base de teste

WEIGHTS_FOLDER = BASE_FOLDER + '/Weights' # Pasta onde estão os pesos pré-
treinados
BEST_WEIGHTS_RESNET_DA_PATH = WEIGHTS_FOLDER +
'/img_model_resnet_da.weights.best.keras' # Caminho para salvar os melhores pesos
para a resnet com data augmentation
BEST_WEIGHTS_RESNET_PATH = WEIGHTS_FOLDER + '/img_model_resnet.weights.best.keras'
# Caminho para salvar os melhores pesos para a resnet sem data augmentation
BEST_WEIGHTS_VGG16_DA_PATH = WEIGHTS_FOLDER +
'/img_model_vgg16_da.weights.best.keras' # Caminho para salvar os melhores pesos
para a vgg16 com data augmentation
BEST_WEIGHTS_VGG16_PATH = WEIGHTS_FOLDER + '/img_model_vgg16.weights.best.keras' #
Caminho para salvar os melhores pesos para a vgg16 sem data augmentation

NUM_EPOCHS = 10 # Número de épocas para o treinamento
```

```
# %% [markdown]
# ## Importação das bibliotecas necessárias

# %%
import os
import gc
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import accuracy_score, f1_score, precision_score,
recall_score, classification_report, confusion_matrix

# importa os tipos de rede
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input as
resnet_preprocess_input
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input as
vgg16_preprocess_input

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
from keras.layers import Dense, Dropout, Flatten, AveragePooling2D
from keras.models import Model

# Otimizadores Root Mean Squared Propagation & Adam
from keras.optimizers import RMSprop, Adam
from livelossplot import PlotLossesKeras

# %% [markdown]
# Verifica se tem uma GPU disponível

# %%
import tensorflow as tf
print("Num GPUs Available: ", len(tf.config.list_physical_devices('GPU')))

# %%
# Cria os diretórios para armazenar os melhores pesos das redes, se não existirem
os.makedirs(WEIGHTS_FOLDER, exist_ok=True)

# %% [markdown]
# # **Redes ResNet50 e VGG16**

# %% [markdown]
# ## Cria e treina os modelos com Data Augmentation

# %% [markdown]
# ### Configura os geradores e fluxos das bases de Treino e Validação

# %% [markdown]
# Cria os Generators
```

```
# %%
resnet_train_generator_da = ImageDataGenerator(
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0,
    preprocessing_function=resnet_preprocess_input
)

vgg16_train_generator_da = ImageDataGenerator(
    rotation_range=90,
    brightness_range=[0.1, 0.7],
    width_shift_range=0.5,
    height_shift_range=0.5,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split=0,
    preprocessing_function=vgg16_preprocess_input
)

resnet_valid_generator =
ImageDataGenerator(preprocessing_function=resnet_preprocess_input)

vgg16_valid_generator =
ImageDataGenerator(preprocessing_function=vgg16_preprocess_input)

# %% [markdown]
# Cria os Flows de arquivos

# %%
resnet_traingen = resnet_train_generator_da.flow_from_directory(
    TRAINING_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    subset='training',
    seed=42
)

vgg16_traingen = vgg16_train_generator_da.flow_from_directory(
    TRAINING_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    subset='training',
    seed=42
)
```

```
resnet_validgen = resnet_valid_generator.flow_from_directory(
    VALIDATION_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    seed=42
)

vgg16_validgen = vgg16_valid_generator.flow_from_directory(
    VALIDATION_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    seed=42
)
```

```
Found 478 images belonging to 4 classes.
Found 478 images belonging to 4 classes.
Found 115 images belonging to 4 classes.
Found 115 images belonging to 4 classes.
```

```
# %% [markdown]
# Function para criar as camadas Fully Connected adaptadas

# %%
def monta_camada_top(model_base):
    # monta a camada Fully Connected para a rede base
    x = model_base.output
    x = AveragePooling2D(pool_size=(7, 7))(x)
    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.2)(x)
    x = Dense(512, activation='relu')(x)
    prediction = Dense(4, activation='softmax')(x)

    final_model = Model(inputs=model_base.input, outputs=prediction)

    # treinar somente a camada Fully Connected
    for i in range(0, len(final_model.layers)):
        if i >= len(model_base.layers):
            final_model.layers[i].trainable = True
        else:
            final_model.layers[i].trainable = False

    return final_model

# %% [markdown]
# ### Monta a ResNet50 já com os pesos imagenet carregados

# %% [markdown]
# Cria a estrutura original da ResNet sem a camada fully connected
```

```
# %%
# Cria a rede ResNet50 com pesos do ImageNet as camadas FC
resnet = ResNet50(
   input_shape=(224,224,3),
   weights='imagenet',
   include_top=False
)
# Marca todas as camadas da rede como não treináveis.
resnet.trainable = False

# %% [markdown]
# Monta as camadas fully connected customizadas para a ResNet

# %%
# monta a camada Fully Connected para a ResNet50
model_resnet_da = monta_camada_top(resnet)

# %%
model_resnet_da.summary() # Impressão das arquitetura da rede
```

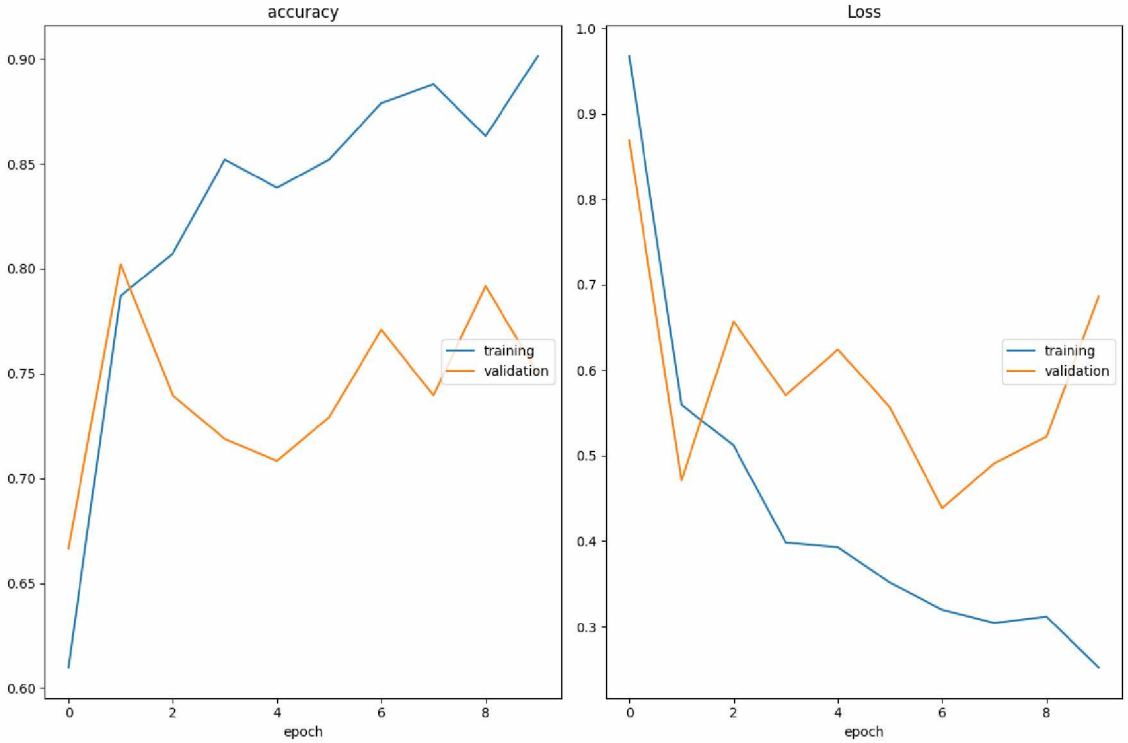
```
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 224, 224, 3)]	0	[]
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	['input_1[0][0]']
conv1_conv (Conv2D)	(None, 112, 112, 64)	9472	['conv1_pad[0][0]']
conv1_bn (BatchNormalization)	(None, 112, 112, 64)	256	['conv1_conv[0][0]']
...			
conv5_block3_out (Activation)	(None, 7, 7, 2048)	0	['conv5_block3_add[0][0]']
average_pooling2d (AveragePooling2D)	(None, 1, 1, 2048)	0	['conv5_block3_out[0][0]']
flatten (Flatten)	(None, 2048)	0	['average_pooling2d[0][0]']
dense (Dense)	(None, 1024)	2098176	['flatten[0][0]']
dropout (Dropout)	(None, 1024)	0	['dense[0][0]']
dense_1 (Dense)	(None, 512)	524800	['dropout[0][0]']
dense_2 (Dense)	(None, 4)	2052	['dense_1[0][0]']

```
Total params: 26,212,740
Trainable params: 2,625,028
Non-trainable params: 23,587,712
```

```
# %% [markdown]
# ### Treinamento da rede ResNet com Data Augmentation
```

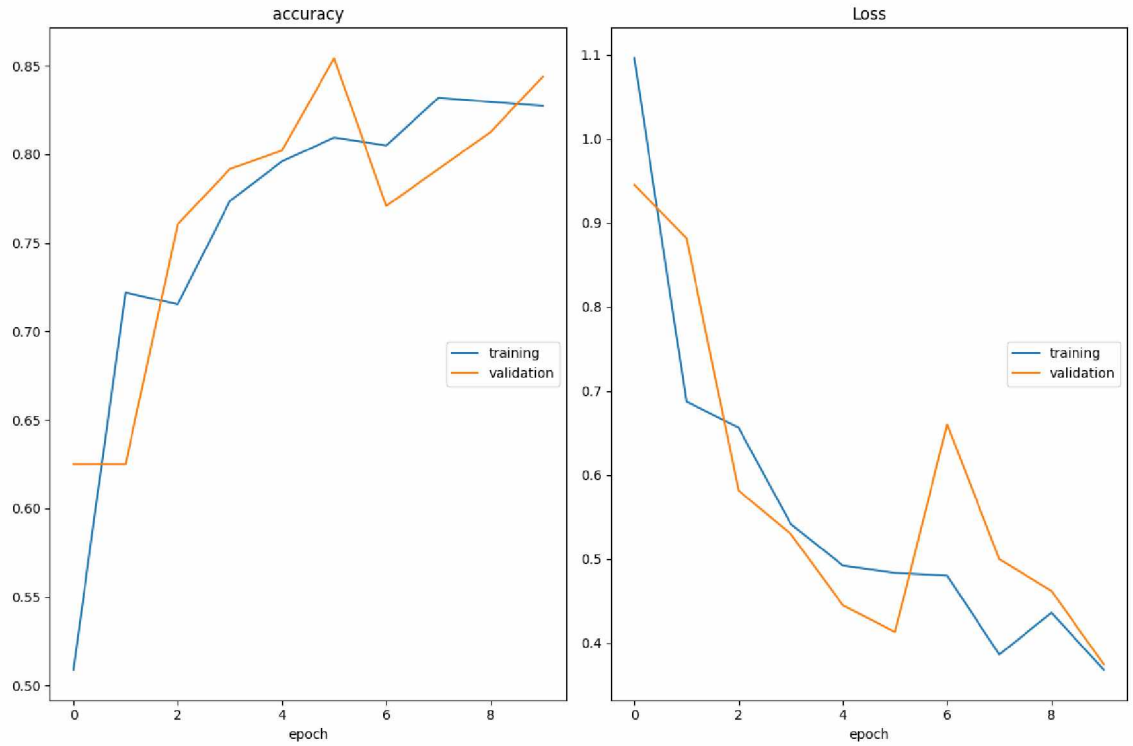
```
# %%
%%time

steps_per_epoch = resnet_traingen.samples // BATCH_SIZE
val_steps = resnet_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)
# optimizer = Adam(learning_rate=0.0001)

model_resnet_da.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath=BEST_WEIGHTS_RESNET_DA_PATH,
                               verbose=1,
                               save_best_only=True)

# Treinamento do Modelo
history_resnet_da = model_resnet_da.fit(resnet_traingen,
                                        epochs=NUM_EPOCHS,
                                        steps_per_epoch=steps_per_epoch,
                                        validation_data=resnet_validgen,
                                        validation_steps=val_steps,
                                        callbacks=[checkpointer,
                                                   PlotLossesKeras()],
                                        verbose=True)
```



```
accuracy
	training         (min:    0.610, max:    0.901, cur:    0.901)
	validation       (min:    0.667, max:    0.802, cur:    0.750)
Loss
	training         (min:    0.252, max:    0.967, cur:    0.252)
	validation       (min:    0.439, max:    0.869, cur:    0.686)
14/14 [==============================] - 18s 1s/step - loss: 0.2524 - accuracy:
0.9013 - val_loss: 0.6861 - val_accuracy: 0.7500
CPU times: total: 4min 12s
Wall time: 3min 4s

# %% [markdown]
# Deleta os objetos da rede anterior para liberar memória

# %%
del history_resnet_da
del model_resnet_da
gc.collect()

# %% [markdown]
# ### Monta a VGG16 já com os pesos imagenet carregados

# %% [markdown]
# Cria a estrutura original da VGG16 sem a camada fully connected

# %%
# Cria a rede ResNet50 com pesos do ImageNet as camadas FC
vgg = VGG16(
   input_shape=(224,224,3),
   weights='imagenet',
   include_top=False
)
# Marca todas as camadas da rede como não treináveis.
vgg.trainable = False

# %% [markdown]
# Monta as camadas fully connected customizadas para a VGG16

# %%
# monta a camada Fully Connected para a VGG16
model_vgg_da = monta_camada_top(vgg)

# %%
model_vgg_da.summary() # Impressão das arquitetura da rede
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)         [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
...
 average_pooling2d_1 (Averag   (None, 1, 1, 512)        0
 ePooling2D)

 flatten_1 (Flatten)          (None, 512)               0

 dense_3 (Dense)              (None, 1024)              525312

 dropout_1 (Dropout)          (None, 1024)              0

 dense_4 (Dense)              (None, 512)               524800

 dense_5 (Dense)              (None, 4)                 2052

=================================================================
Total params: 15,766,852
Trainable params: 1,052,164
Non-trainable params: 14,714,688
```

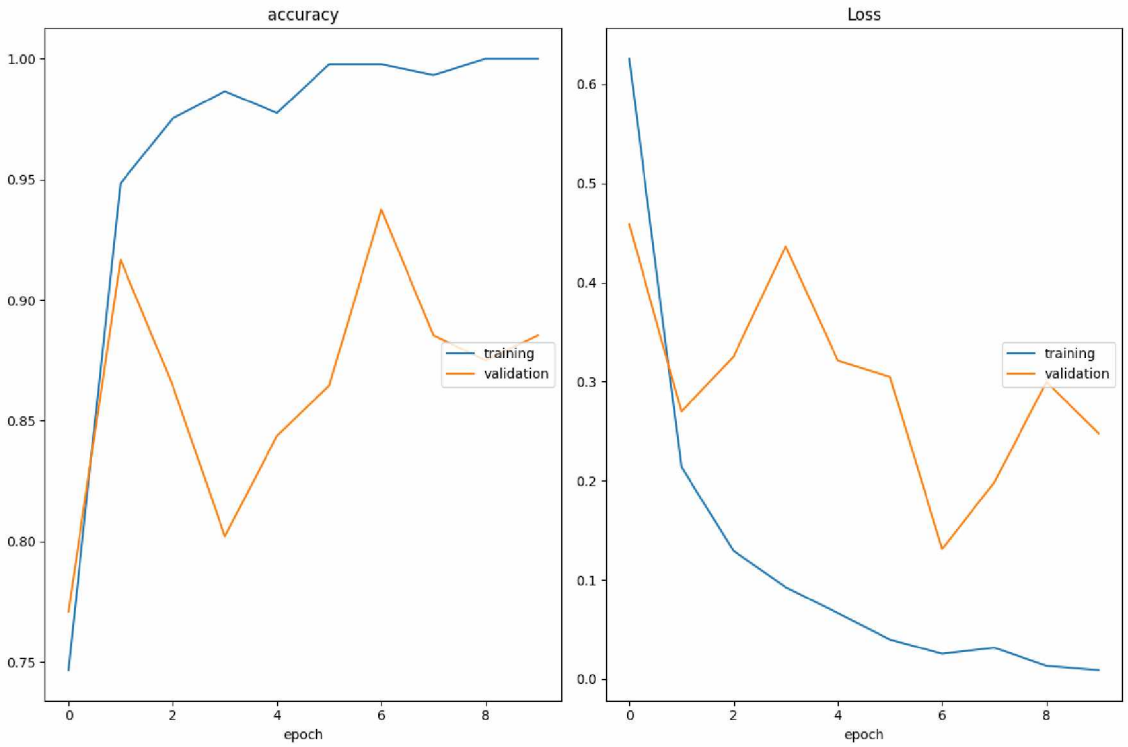
```
# %% [markdown]
# ### Treinamento da rede VGG16 com Data Augmentation

# %%
%%time

steps_per_epoch = vgg16_traingen.samples // BATCH_SIZE
val_steps = vgg16_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)
# optimizer = Adam(learning_rate=0.0001)

model_vgg_da.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath=BEST_WEIGHTS_VGG16_DA_PATH,
                               verbose=1,
                               save_best_only=True)

# Treinamento do Modelo
history_vgg_da = model_vgg_da.fit(vgg16_traingen,
                                  epochs=NUM_EPOCHS,
                                  steps_per_epoch=steps_per_epoch,
                                  validation_data=vgg16_validgen,
                                  validation_steps=val_steps,
                                  callbacks=[checkpointer, PlotLossesKeras()],
                                  verbose=True)
```



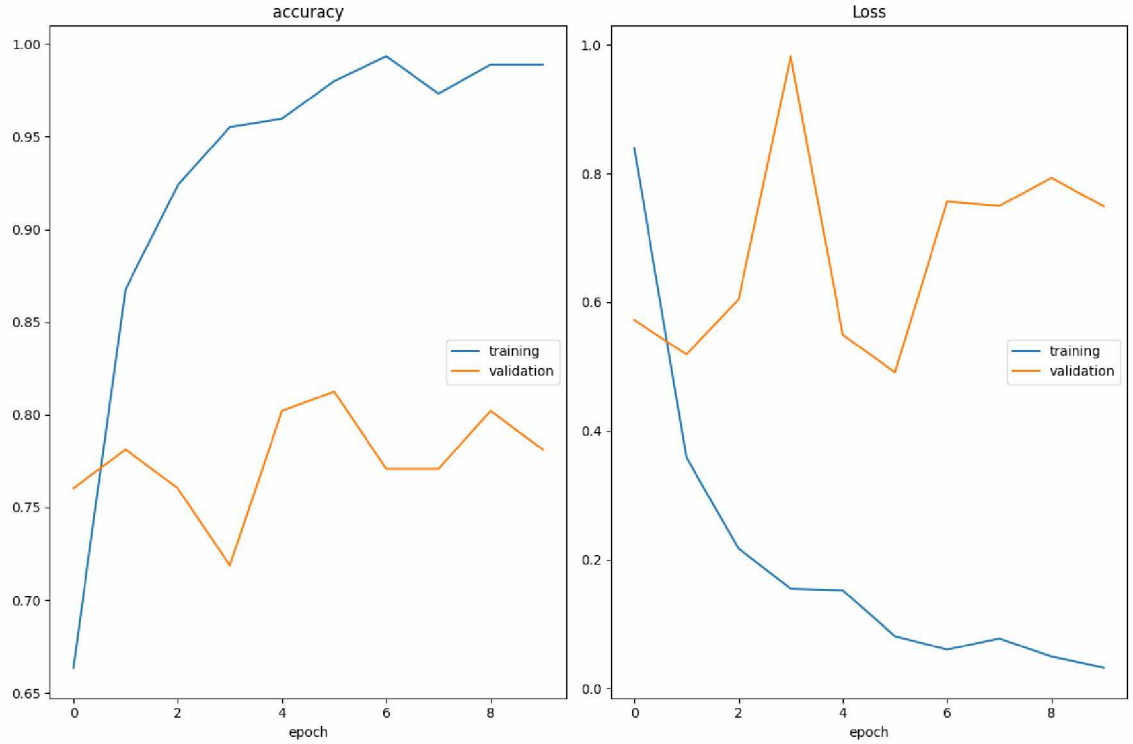

```
accuracy
      training         (min:    0.509, max:    0.832, cur:    0.827)
      validation       (min:    0.625, max:    0.854, cur:    0.844)
Loss
      training         (min:    0.368, max:    1.096, cur:    0.368)
      validation       (min:    0.375, max:    0.945, cur:    0.375)
14/14 [==============================] - 19s 1s/step - loss: 0.3681 - accuracy:
0.8274 - val_loss: 0.3746 - val_accuracy: 0.8438
CPU times: total: 4min 55s
Wall time: 3min 31s


# %% [markdown]
# Deleta os objetos da rede anterior para liberar memória

# %%
del history_vgg_da
del model_vgg_da
gc.collect()

# %% [markdown]
# ## Treina os modelos sem Data Augmentation

# %% [markdown]
# ### Configura os geradores e fluxos das bases de Treino e Validação

# %% [markdown]
# Cria os Generators
```

```
# %%
resnet_train_generator = ImageDataGenerator(
    validation_split=0,
    preprocessing_function=resnet_preprocess_input
)

vgg16_train_generator = ImageDataGenerator(
    validation_split=0,
    preprocessing_function=vgg16_preprocess_input
)

# %% [markdown]
# Cria os Flows de arquivos

# %%
resnet_traingen = resnet_train_generator.flow_from_directory(
    TRAINING_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    subset='training',
    seed=42
)

vgg16_traingen = vgg16_train_generator.flow_from_directory(
    TRAINING_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=True,
    subset='training',
    seed=42
)


Found 478 images belonging to 4 classes.
Found 478 images belonging to 4 classes.

# %% [markdown]
# ### Monta novamente as camadas fully connected customizadas para a ResNet para
resetar os pesos

# %%
# monta a camada Fully Connected para a ResNet50
model_resnet = monta_camada_top(resnet)

# %% [markdown]
# ### Treinamento da rede ResNet sem Data Augmentation
```

```
# %%
%%time

steps_per_epoch = resnet_traingen.samples // BATCH_SIZE
val_steps = resnet_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)
# optimizer = Adam(learning_rate=0.0001)

model_resnet.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath=BEST_WEIGHTS_RESNET_PATH,
                               verbose=1,
                               save_best_only=True)

# Treinamento do Modelo
history_resnet = model_resnet.fit(resnet_traingen,
                                  epochs=NUM_EPOCHS,
                                  steps_per_epoch=steps_per_epoch,
                                  validation_data=resnet_validgen,
                                  validation_steps=val_steps,
                                  callbacks=[checkpointer, PlotLossesKeras()],
                                  verbose=True)
```



```
accuracy
	training         (min:    0.747, max:    1.000, cur:    1.000)
	validation       (min:    0.771, max:    0.938, cur:    0.885)
Loss
	training         (min:    0.009, max:    0.625, cur:    0.009)
	validation       (min:    0.131, max:    0.459, cur:    0.248)
14/14 [==============================] - 10s 743ms/step - loss: 0.0089 - accuracy:
1.0000 - val_loss: 0.2478 - val_accuracy: 0.8854
CPU times: total: 2min 37s
Wall time: 1min 52s

# %% [markdown]
# Deleta os objetos da rede anterior para liberar memória

# %%
del history_resnet
del model_resnet
gc.collect()

# %% [markdown]
# ### Monta novamente as camadas fully connected customizadas para a VGG16 para
resetar os pesos

# %%
# monta a camada Fully Connected para a VGG16
model_vgg = monta_camada_top(vgg)

# %% [markdown]
# ### Treinamento da rede VGG16 sem Data Augmentation

# %%
%%time

steps_per_epoch = vgg16_traingen.samples // BATCH_SIZE
val_steps = vgg16_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)
# optimizer = Adam(learning_rate=0.0001)

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

# Salva o modelo Keras após cada época, porém só o de melhor resultado
checkpointer = ModelCheckpoint(filepath=BEST_WEIGHTS_VGG16_PATH,
                               verbose=1,
                               save_best_only=True)

# Treinamento do Modelo
history_vgg = model_vgg.fit(vgg16_traingen,
                            epochs=NUM_EPOCHS,
                            steps_per_epoch=steps_per_epoch,
                            validation_data=vgg16_validgen,
                            validation_steps=val_steps,
                            callbacks=[checkpointer, PlotLossesKeras()],
                            verbose=True)
```




```
accuracy
      training         (min:    0.664, max:    0.993, cur:    0.989)
      validation       (min:    0.719, max:    0.812, cur:    0.781)
Loss
      training         (min:    0.032, max:    0.839, cur:    0.032)
      validation       (min:    0.491, max:    0.982, cur:    0.749)
14/14 [==============================] - 12s 823ms/step - loss: 0.0323 - accuracy:
0.9888 - val_loss: 0.7490 - val_accuracy: 0.7812
CPU times: total: 2min 45s
Wall time: 2min

# %% [markdown]
# Deleta os objetos da rede anterior para liberar memória

# %%
del history_vgg
del model_vgg
gc.collect()

# %% [markdown]
# # **Predições dos modelos**

# %% [markdown]
# ## Cria o Generator de Teste

# %%
resnet_test_generator =
ImageDataGenerator(preprocessing_function=resnet_preprocess_input)

vgg16_test_generator =
ImageDataGenerator(preprocessing_function=vgg16_preprocess_input)

resnet_testgen = resnet_test_generator.flow_from_directory(
```

```
    TEST_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=False,
    seed=42
)

vgg16_testgen = vgg16_test_generator.flow_from_directory(
    TEST_BASE_PATH,
    target_size=(224, 224),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    classes=['0','1','2','3'],
    shuffle=False,
    seed=42
)
```

```
Found 371 images belonging to 4 classes.
Found 371 images belonging to 4 classes.
```

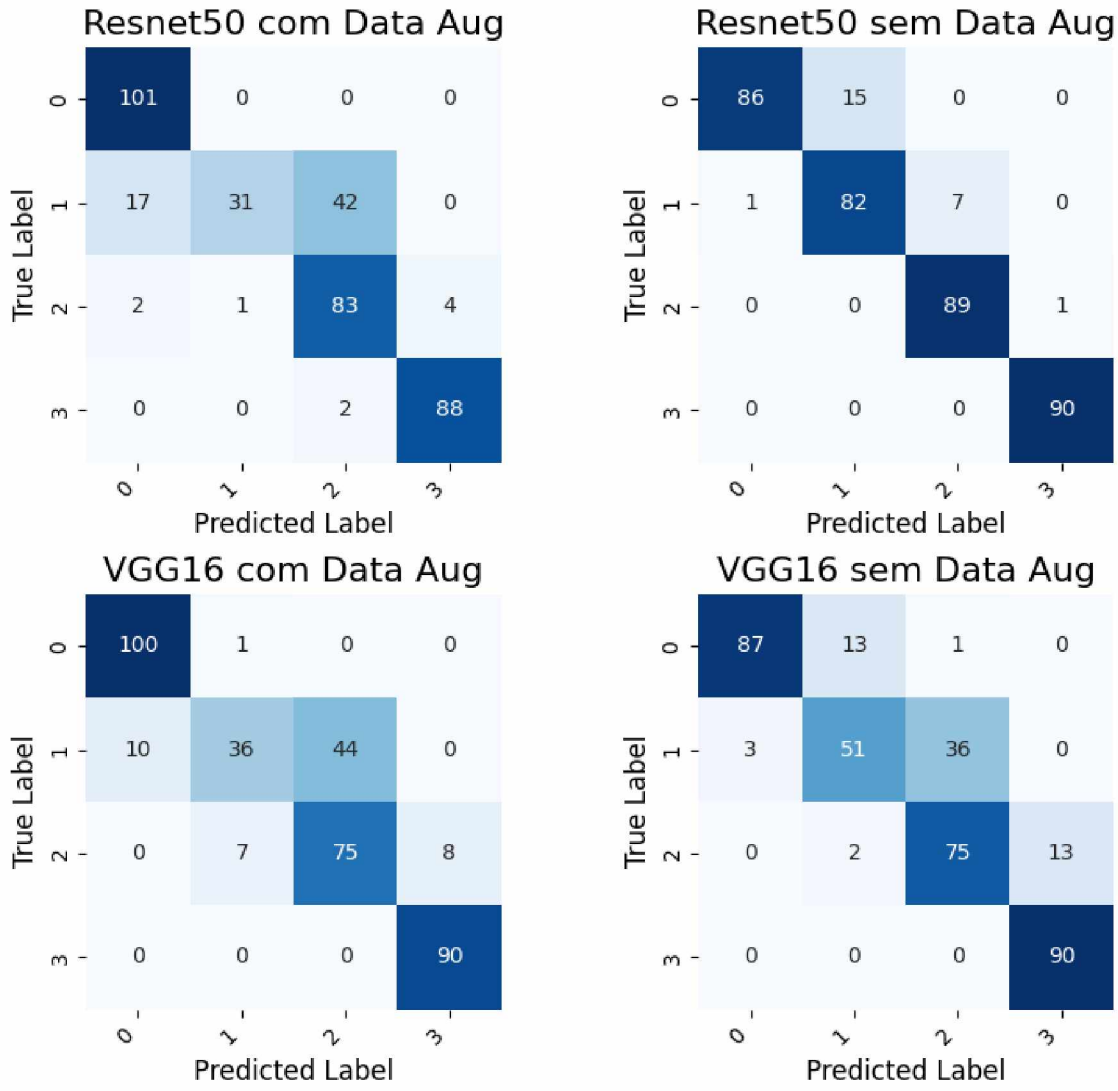
```
# %% [markdown]
# ## Novas camadas fully connected para resetar os pesos

# %%
model_resnet_pred = monta_camada_top(resnet)
model_vgg_pred = monta_camada_top(vgg)

# %% [markdown]
# ## Carrega os pesos salvos no treinamento de cada modelo

# %% [markdown]
# ### Carrega os pesos e efetua as predições com Data Augmentation

# %%
model_resnet_pred.load_weights(BEST_WEIGHTS_RESNET_DA_PATH)
model_vgg_pred.load_weights(BEST_WEIGHTS_VGG16_DA_PATH)

predicted_resnet_da = model_resnet_pred.predict(resnet_testgen)
predicted_classes_resnet_da = np.argmax(predicted_resnet_da, axis=1)

predicted_vgg_da = model_vgg_pred.predict(vgg16_testgen)
predicted_classes_vgg_da = np.argmax(predicted_vgg_da, axis=1)
```

```
12/12 [==============================] - 12s 980ms/step
12/12 [==============================] - 19s 2s/step
```

```
# %% [markdown]
# ### Carrega os pesos e efetua as predições sem Data Augmentation
```

```
# %%
model_resnet_pred.load_weights(BEST_WEIGHTS_RESNET_PATH)
model_vgg_pred.load_weights(BEST_WEIGHTS_VGG16_PATH)

predicted_resnet = model_resnet_pred.predict(resnet_testgen)
predicted_classes_resnet = np.argmax(predicted_resnet, axis=1)

predicted_vgg = model_vgg_pred.predict(vgg16_testgen)
predicted_classes_vgg = np.argmax(predicted_vgg, axis=1)

12/12 [==============================] - 6s 509ms/step
12/12 [==============================] - 6s 541ms/step


# %% [markdown]
# # **Comparação de Resultados dos modelos**

# %% [markdown]
# ## Comparação de Métricas dos modelos

# %%
def imprime_metricas_modelo(true_classes, pred_classes, nome_modelo):

    def especificidade(true_classes, pred_classes): # Define uma função para
calcular a especificidade que não existe no sklearn
        cm = confusion_matrix(true_classes, pred_classes)
        specificities = []
        for i in range(len(cm)):
            # Exclude row i and column i to get TN
            tn = np.sum(np.delete(np.delete(cm, i, axis=0), i, axis=1))
            fp = np.sum(np.delete(cm, i, axis=0)[:, i])
            specificity = tn / (tn + fp) if (tn + fp) > 0 else 0
            specificities.append(specificity)
        return np.mean(specificities)

    # Calcula as métricas
    acuracia = accuracy_score(true_classes, pred_classes)
    sensibilidade = recall_score(true_classes, pred_classes, average='macro')
    especificidade = especificidade(true_classes, pred_classes)
    f1 = f1_score(true_classes, pred_classes, average='macro')

    print("-----------------------------------------------------")
    print(f"Acurácia Modelo {nome_modelo}: {(acuracia * 100):.2f}%")
    print(f"Sensibilidade Modelo {nome_modelo}: {(sensibilidade * 100):.2f}%")
    print(f"Especificidade Modelo {nome_modelo}: {(especificidade * 100):.2f}%")
    print(f"F1 Modelo {nome_modelo}: {(f1 * 100):.2f}%")


# %%
resnet_true_classes = resnet_testgen.classes
vgg16_true_classes = vgg16_testgen.classes
resnet_class_names = resnet_testgen.class_indices.keys()
vgg16_class_names = vgg16_testgen.class_indices.keys()
```

```
def plot_heatmap(y_true, y_pred, class_names, ax, title):
    cm = confusion_matrix(y_true, y_pred)
    sns.heatmap(
        cm,
        annot=True,
        square=True,
        xticklabels=class_names,
        yticklabels=class_names,
        fmt='d',
        cmap=plt.cm.Blues,
        cbar=False,
        ax=ax
    )
    ax.set_title(title, fontsize=16)
    ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha="right")
    ax.set_ylabel('True Label', fontsize=12)
    ax.set_xlabel('Predicted Label', fontsize=12)

fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize=(8, 8))

plot_heatmap(resnet_true_classes, predicted_classes_resnet_da,
             resnet_class_names, ax1, title="Resnet50 com Data Aug")
plot_heatmap(resnet_true_classes, predicted_classes_resnet,
             resnet_class_names, ax2, title="Resnet50 sem Data Aug")
plot_heatmap(vgg16_true_classes, predicted_classes_vgg_da,
             vgg16_class_names, ax3, title="VGG16 com Data Aug")
plot_heatmap(vgg16_true_classes, predicted_classes_vgg,
             vgg16_class_names, ax4, title="VGG16 sem Data Aug")

fig.suptitle("Comparação das Matrizes de Confusão", fontsize=24)
fig.tight_layout()
fig.subplots_adjust(top=0.85)
plt.show()

imprime_metricas_modelo(resnet_true_classes,
                        predicted_classes_resnet_da, "Resnet50 com Data Aug")
imprime_metricas_modelo(resnet_true_classes,
                        predicted_classes_resnet, "Resnet50 sem Data Aug")
imprime_metricas_modelo(vgg16_true_classes,
                        predicted_classes_vgg_da, "VGG16 com Data Aug")
imprime_metricas_modelo(vgg16_true_classes,
                        predicted_classes_vgg, "VGG16 sem Data Aug")
```



Acurácia Modelo Resnet50 com Data Aug: 81.67%
Sensibilidade Modelo Resnet50 com Data Aug: 81.11%
Especificidade Modelo Resnet50 com Data Aug: 93.88%
F1 Modelo Resnet50 com Data Aug: 78.86%

Acurácia Modelo Resnet50 sem Data Aug: 93.53%
Sensibilidade Modelo Resnet50 sem Data Aug: 93.79%
Especificidade Modelo Resnet50 sem Data Aug: 97.86%
F1 Modelo Resnet50 sem Data Aug: 93.58%

Acurácia Modelo VGG16 com Data Aug: 81.13%
Sensibilidade Modelo VGG16 com Data Aug: 80.59%
Especificidade Modelo VGG16 com Data Aug: 93.74%
F1 Modelo VGG16 com Data Aug: 79.01%

```
--------------------------------------------------
Acurácia Modelo VGG16 sem Data Aug: 81.67%
Sensibilidade Modelo VGG16 sem Data Aug: 81.53%
Especificidade Modelo VGG16 sem Data Aug: 93.94%
F1 Modelo VGG16 sem Data Aug: 81.00%
```

APÊNDICE K - ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

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

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

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

B – RESOLUÇÃO

1. Introdução

O avanço da Inteligência Artificial (IA) e, mais recentemente, de chatbots como o ChatGPT, lançado em novembro de 2022 pela empresa OpenAI, tem transformado diversos setores, desde a comunicação até a saúde, passando pela educação e o entretenimento. Verdadeiras revoluções estão em andamento e por vir. No entanto, essa transformação tecnológica traz consigo uma série de dilemas éticos que precisam ser avaliados com cautela, para garantir que os benefícios sejam maximizados e os riscos minimizados. Este artigo tem como objetivo explorar as principais implicações éticas associadas ao uso do ChatGPT em diferentes contextos, abordando questões importantes como privacidade e segurança de dados, fake news e discriminação, autonomia na tomada de decisão e transparência. Além disso, propomos soluções responsáveis para enfrentar esses dilemas éticos, destacando a necessidade de uma abordagem globalizada e colaborativa que envolva governos, empresas, academia e sociedade civil. Ao adotar práticas éticas e responsáveis, podemos promover um uso mais seguro, justo e benéfico da IA, contribuindo para o desenvolvimento sustentável e igualitário da sociedade.

2. Relevância Ética

Floridi e Chiriatii (2020) descrevem GPT (Generative Pre-trained Transformer) como um modelo de linguagem projetado para gerar sequências de palavras, códigos ou qualquer outro dado a partir de uma fonte de entrada de informação do usuário, usando um banco de dados composto de textos de sites da internet como Wikipedia,

por exemplo. Contudo, o uso de chatbots como o ChatGPT, trouxe relevantes e importantes discussões sobre as implicações éticas do seu uso tornando-se uma preocupação central devido ao impacto significativo dessa tecnologia na sociedade. As respostas fornecidas pela IA são baseadas em dados até o ano em que foi construída, podendo estar desatualizadas ou conter informações falsas. Além disso, existe a possibilidade de que o ChatGPT gere respostas racistas ou discriminatórias, devido aos vieses presentes nos dados de treinamento.

Atualmente, existem muitos casos de sucesso no uso do ChatGPT, mas, como qualquer tecnologia, ele também pode ser usado para propósitos nocivos. Um exemplo é o caso de um advogado nos Estados Unidos que usou o chatbot para criar falsos precedentes em uma ação judicial, resultando em sua punição e multa. Esse incidente levanta questões éticas e morais, além de destacar a necessidade de uma política de governança que garanta a transparência no uso da ferramenta. O uso inadequado do ChatGPT pode ter consequências perigosas, levantando preocupações no campo do Direito, especialmente em relação à proteção de informações pessoais e direitos autorais.

Um caso emblemático de mau uso do ChatGPT ocorreu na China, onde um homem foi preso e pode pegar até 10 anos de prisão por espalhar falsas notícias sobre um acidente de trem. A notícia falsa, criada com o auxílio do ChatGPT, dizia que nove pessoas haviam morrido. O homem usou uma VPN para acessar o chatbot, que não é acessível na China, e contornou vários sistemas de segurança para disseminar a falsa notícia.

Outro caso envolve o Bing GPT, o chatbot da Microsoft. Em uma conversa no Reddit, o chatbot gerou conteúdo prejudicial ao discutir antissemitismo. Embora inicialmente tenha alertado sobre o perigo de exaltar figuras históricas responsáveis por atos horríveis, o chatbot Bing acabou gerando respostas automáticas prejudiciais, como uma saudação nazista.

O ChatGPT é acessado por cerca de 1,8 bilhões de pessoas por mês. Desses, 15% dos acessos diários vêm dos Estados Unidos, 6,32% da Índia e 4,01% do Japão. O Japão notificou a OpenAI sobre falhas na coleta de dados dos usuários, alegando que a plataforma estava coletando informações confidenciais sem permissão. A OpenAI se comprometeu a reduzir esse tipo de coleta. Para superar esses desafios, empresas estão trabalhando para integrar o ChatGPT a outros sistemas e mecanismos de controle, garantindo que as respostas da IA sejam precisas e seguras.

A adoção do ChatGPT também levanta preocupações sobre o futuro dos empregos. Alguns especialistas acreditam que a IA pode levar ao desaparecimento de empregos, enquanto outros veem a criação de novas oportunidades. De qualquer forma, o ChatGPT é uma tecnologia promissora que pode transformar a interação com as empresas.

Chatbots como o ChatGPT apresentam dilemas éticos significativos, sendo o viés um dos principais. Treinados em grandes conjuntos de dados de texto, eles podem refletir os vieses presentes nesses dados, resultando em respostas discriminatórias ou prejudiciais. O uso indevido também é uma preocupação. Chatbots podem ser usados para espalhar desinformação, propaganda ou para fins maliciosos. É essencial garantir o uso responsável e ético dos chatbots. Deepfakes, por exemplo, podem ser criados com chatbots, manipulando vídeos ou áudios para parecer que alguém disse ou fez algo que não fez, espalhando desinformação ou prejudicando reputações.

O impacto psicológico do uso de chatbots também merece atenção. Algumas pessoas podem se tornar dependentes dessas tecnologias, levando a problemas de saúde mental, como depressão ou ansiedade. Por exemplo, uma pessoa solitária pode se tornar dependente de um chatbot para companhia, resultando em problemas de saúde mental.

3. Análise Crítica

Há vários desafios éticos que podem ser identificados na interação dos usuários com os modelos LLM (Large Language Model), como o ChatGPT. Partindo do ponto de vista da capacidade de processamento necessária para se treinar esses modelos de aprendizado de máquina com a grande quantidade de informações (no caso, textos), em um tempo viável para se disponibilizar a aplicação para uso, pode-se inferir que o hardware necessário possui um custo muito elevado. Isso limita a sua implantação a poucas empresas que possuam essa capacidade de investimento, bem como universidades e institutos de pesquisa públicos ou privados.

As implicações disso é que há uma superconcentração desses modelos nessa minoria de empresas privadas, praticamente sem concorrência. Além disso, dificulta enormemente os testes e análises e, consequentemente, a implementação de algum tipo de governança por parte da sociedade civil.

Soma-se a isso o fato de que os modelos LLM utilizados podem incorporar toda uma sorte de vieses algorítmicos, vindo tanto da parte dos programadores do modelo em si, como dos dados utilizados no seu treinamento. A implementação desses algoritmos sem os devidos cuidados e verificações pode acabar por reforçar, ou mesmo incorporar, estereótipos étnico-raciais, socioeconômicos e reproduzir comportamentos sectários e segregacionistas. Um exemplo que ilustra bem, ainda que tenha sido com IA generativa para produção de imagens e não texto, foi o que ocorreu com a deputada estadual Renata Souza (PSOL-RJ), que ao utilizar uma ferramenta de IA para geração de imagem em forma semelhante aos posters dos filmes de animação da empresa Disney, solicitou que fosse gerada um poster "de uma mulher negra, de cabelos afro, com roupas de estampa africana num cenário de favela" e a IA gerou uma imagem de uma mulher negra com uma arma na mão.

Além dos possíveis vieses incorporados no desenvolvimento, a forma como os modelos interagem com os usuários, se não for protegida com as devidas salvaguardas, também pode fazer com que os modelos incorporem, inadvertidamente, esses mesmos vieses e passem a reproduzi-los. Um bom exemplo desta possibilidade foi o que ocorreu com o chatbot da Microsoft chamado Tay, que foi concebido para interagir com jovens entre 18 e 24 anos através de uma conta no Twitter (atual X) como se fosse um deles. Em menos de um dia de interação na rede social, a IA passou a responder e incorporar comportamentos xenófobos, racistas e genocidas, e foi então retirada do ar.

Outro caso conhecido foi o da ferramenta de chat do buscador Bing, também da Microsoft, que durante o seu período de testes, em sessões prolongadas de interação com o mesmo usuário enviando um número maior de perguntas, passava a respondê-las incorretamente e, às vezes, com linguagem considerada rude e grosseira.

Para além da questão dos vieses, outros desafios éticos também se apresentam. Entre eles, pode-se mencionar a eventual propriedade intelectual dos dados utilizados para treinar tanto modelos generativos de texto quanto de imagem. Se os textos e imagens utilizados no treinamento dos modelos não forem de domínio público, mas originalmente criados por autores humanos, na hipótese da empresa criadora da IA fazer uso comercial dela, poderá levar eventualmente a contestação judicial da propriedade intelectual desses dados de treinamento (textos e imagens).

Outro ponto importante diz respeito à segurança dos dados que são enviados pelos usuários através dos prompts, que podem receber eventualmente algum tipo de dado privado dos mesmos e, posteriormente, ser difícil a sua anonimização, ou mesmo, eliminação, uma vez incorporados às bases de treinamento dos modelos. É necessário que as regras de tratamento e a governança dos dados imputados nos prompts, inclusive do ponto de vista do algoritmo da aplicação, sejam muito claras, para que se possa fazer um uso consciente das ferramentas. Isso muitas vezes é especialmente difícil para usuários finais, pessoas físicas que não detêm o conhecimento técnico e legal para avaliar plenamente a melhor e mais correta forma de utilizá-las.

Além de todas as discussões anteriores, como os modelos GPT mais modernos incorporam cada vez mais dados advindos diretamente da internet, não se poderia deixar de mencionar a questão da possível propagação de notícias falsas (fake news) que por ventura possam estar sendo veiculadas de forma indiscriminada através de páginas ou redes sociais. Caso não haja um tratamento adequado nos algoritmos, ou a incorporação de algum tipo de filtro ou "fact checking" na incorporação dessas informações nas bases de treinamento, essa desinformação pode acabar influenciando as respostas dos modelos, sendo propagadas por eles.

4. Soluções Responsáveis

Com o surgimento contínuo de novas aplicações de Inteligência Artificial, questões éticas e filosóficas surgem, exigindo análise profunda e cuidadosa para a busca constante por soluções responsáveis. A filosofia desempenha um papel fundamental na compreensão desses desafios e na definição dessas soluções. Princípios e diretrizes baseados nos aspectos filosóficos e éticos devem ser seguidos ao projetar, desenvolver e implementar sistemas de Inteligência Artificial.

Alguns desses princípios para que tenhamos soluções responsáveis no desenvolvimento das aplicações em Inteligência Artificial incluem a transparência, equidade, privacidade, segurança e responsabilidade.

A transparência na IA envolve tornar os processos de tomada de decisão compreensíveis para os usuários e partes interessadas. Isso significa explicar como os algoritmos funcionam, quais dados são usados e como as decisões são tomadas. A transparência é fundamental para construir confiança e permitir que as pessoas entendam o impacto das decisões automatizadas.

A equidade refere-se a evitar vieses e discriminação na IA. Os algoritmos podem herdar preconceitos dos dados de treinamento, resultando em decisões injustas. Garantir a equidade significa ajustar os modelos para tratar todos os grupos de maneira justa, independentemente de raça, gênero, origem étnica ou outras características.

A privacidade é crucial na era da IA. Proteger os dados pessoais dos usuários é essencial. Isso envolve anonimização, consentimento informado e conformidade com regulamentações de privacidade, como o GDPR (Regulamento Geral de Proteção de Dados).

A segurança da IA diz respeito à robustez dos sistemas. Desenvolvedores devem criar algoritmos que resistem a ataques maliciosos, sejam resilientes a falhas e não causem danos físicos ou financeiros. Testes rigorosos e monitoramento contínuo são essenciais.

A responsabilidade envolve assumir a responsabilidade pelas ações da IA. Isso inclui considerar os impactos sociais, legais e éticos. Os criadores de IA devem estar cientes das consequências e garantir que seus sistemas sejam usados de maneira responsável.

5. Conclusão

O crescimento da Inteligência Artificial (IA), especialmente dos modelos de linguagem como o ChatGPT, está revolucionando diversos setores, incluindo comunicação, saúde, educação e entretenimento. No entanto, essa transformação tecnológica traz uma série de questões éticas que precisam ser cuidadosamente avaliadas para elevar ao máximo os benefícios e minimizar os riscos. Este artigo analisou as principais implicações éticas associadas ao uso do ChatGPT, abordando questões relevantes como privacidade e segurança de dados, disseminação de fake news, discriminação, autonomia na tomada de decisão e transparência.

Os desafios éticos identificados compreendem a possibilidade de respostas desatualizadas ou falsas, uso indevido para criar desinformação, vieses algorítmicos discriminatórios e possíveis impactos psicológicos danosos. Casos representativos de mau uso, como a criação de falsos precedentes legais e a disseminação de notícias falsas, ilustram a necessidade urgente de políticas de governança e transparência. Além disso, a concentração de poder em poucas empresas capazes de investir em IA

e a dificuldade de implementação de governança pela sociedade civil são de grande preocupação.

Para enfrentar esses problemas, é essencial adotar uma abordagem globalizada e colaborativa que envolva governos, empresas, academia e sociedade civil. Princípios éticos como transparência, equidade, privacidade, segurança e responsabilidade devem guiar o desenvolvimento e a implementação de sistemas de IA. A transparência ajuda a construir confiança, a equidade evita discriminação, a privacidade protege dados pessoais, a segurança garante força contra ataques e falhas, e a responsabilidade assegura que os impactos sociais, legais e éticos sejam considerados.

Avanços já estão ocorrendo para minimizar o impacto negativo da IA, como a regulamentação criada pela EU (União Européia), a nova lei além da aplicação de novas normas e categorias de risco, incluindo requisitos mínimos para sistemas usarem IA considerada de alto risco, fixa o que é terminantemente proibido, como uso de inteligência artificial para manipular comportamentos humanos que possam causar riscos ao próprio usuário ou a outras pessoas.

Ao estabelecer práticas éticas e responsáveis, podemos promover um uso mais seguro, justo e benéfico da IA contribuindo para o desenvolvimento sustentável e igualitário da sociedade. A filosofia e a ética desempenham papéis fundamentais na definição dessas diretrizes, garantindo que a IA seja um impulso positivo para o futuro.

6. Referências

CARTACAPITAL. Deputada denuncia 'racismo algorítmico' após IA gerar imagem com arma em uma favela. **CartaCapital**, 2023.
Disponível em: https://www.cartacapital.com.br/cartaexpressa/deputada-denuncia-racismo-algoritmico-apos-ia-gerar-i magem-com-arma-em-uma-favela/. Acesso em: 25 jun. 2024.

FLORIDI, L; CHIRIATTI, M. GPT-3: Its Nature, Scope, Limits, and Consequences. Minds & Machines. Nov. 2020.
Disponível em: https://link.springer.com/article/10.1007/s11023-020-09548-1. Acesso em: 22 abr. 2023

GONÇALVES, Matheus. O robô da Microsoft que aprende com humanos não demorou nem um dia para virar racista. **Tecnoblog**, 2016.
Disponível em: https://tecnoblog.net/especiais/tay-robo-racista-microsoft/. Acesso em: 25 jun. 2024.

GUGLIELMELLI, A. (2023, 6 de junho). País encontra GRANDE problema no ChatGPT e envia comunicado a criadora. **PronaTEC**.
Disponível em: https://pronatec.pro.br/pais-encontra-grande-problema-no-chatgpt/. Acesso em: 25 jun. 2024.

MELO, C. (2023, 16 de fevereiro). ChatGPT do Bing sugere resposta nazista a usuário. **Mundo conectado**.
Disponível em: https://www.mundoconectado.com.br/tecnologia/chatgpt-do-bing-sugere-resposta-

nazista-a-usuario/. Acesso em: 25 jun. 2024.

MELO, Cristino. Microsoft comenta mensagens rudes e mal-educadas do Bing com ChatGPT. **Mundo Conectado**, 2023.
Disponível em: https://www.mundoconectado.com.br/tecnologia/microsoft-comenta-mensagens-rudes-e-mal-educada s-do-bing-com-chatgpt/. Acesso em: 25 jun. 2024.

OLIVEIRA, D. (2023, 23 de fevereiro). Sucesso de público, ChatGPT ainda engatinha no mundo dos negócios.**Terra**.
Disponível em: https://www.terra.com.br/byte/sucesso-de-publico-chatgpt-ainda-engatinha-no-mundodos-negocios,9d 3450928e5d37845eea8bf5bc10dc07m4ioktbs.html. Acesso em: 25 jun. 2024.

SOUZA, L. P. (2023, 8 de maio). Primeiro crime por uso de ChatGPT é registrado na China. **VEJA**.
Disponível em: https://veja.abril.com.br/tecnologia/ primeiro-crime-por-uso-de-chatgpt-e-registrado-na-china. Acesso em: 25 jun. 2024.

APÊNDICE L - GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

- Quais os principais resultados obtidos pelo estudo?

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

Não altere as questões do template.

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

B – RESOLUÇÃO

- Qual o objetivo do estudo descrito pelo artigo?

O objetivo do artigo é identificar os desafios no desenho da arquitetura, as melhores práticas de mercado e as decisões que necessitam ser tomadas no desenho dessa arquitetura de sistemas de aprendizado de máquina e inteligência artificial.
Esse objetivo se desdobra em 3 questões primárias a serem pesquisadas:

1. Quais são os desafios mais comuns no desenho da arquitetura de software para sistemas de aprendizado de máquina?
2. Quais são as melhores práticas de mercado na arquitetura de sistemas baseados em aprendizado de máquina?
3. Quais são as principais decisões que precisam ser tomadas no desenho da arquitetura de sistemas baseado em inteligência artificial?

Respostas à pergunta 1 devem prover uma base comum de desafios enfrentados na definição da arquitetura para que pesquisadores e desenvolvedores possam evitá-los. As respostas à pergunta 2 devem prover um catálogo das melhores práticas que poderão ser utilizadas por pesquisadores e desenvolvedores, enquanto as respostas à pergunta 3 devem prover uma lista das decisões de desenho que poderão ajudar os pesquisadores e desenvolvedores nas definições de seus próprios projetos.

- Qual o problema/oportunidade/situação que levou a necessidade de realização deste estudo?

Já que o aprendizado de máquina tem ganhado cada vez mais projeção em diversas indústrias, sendo cada vez mais necessário e utilizado em produtos oferecidos pelas mesmas em áreas como defesa computacional, bioinformática, veículos autônomos, robótica e internet das coisas, a arquitetura desses softwares que utilizam componentes de inteligência artificial e aprendizado de máquina se coloca como um dos fatores vitais para o sucesso deles devido a alguns fatores, tais como:

- Manutenção e evolução são aspectos importantes do Ciclo de Vida de modelos de aprendizado de máquina;
- Entendimento e gerenciamento dos aspectos de qualidade de sistemas baseados em aprendizado de máquina;
- Desenho da integração com outros componentes de software e
- Gerenciamento de incertezas.

Apesar de pesquisadores e especialistas virem pesquisando melhores práticas de desenho de software baseado em aprendizado de máquina e inteligência artificial, ainda há lacunas na análise de como os desenvolvedores percebem e tomam decisões nesse sentido, que podem ser preenchidas pelo presente artigo/estudo

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

Foi utilizado um método de pesquisa que combina metodologias complementares para evitar as eventuais limitações da utilização de um único método. Foram utilizadas recomendações de pesquisas em engenharia de software de Kitchenham and Brereton (2013) e, em entrevistas e pesquisa empírica de engenharia de software de Wohlin et al. (2012). O processo desenvolvido, consiste em três etapas distintas e suas sub-etapas:

- Planejamento;
    - Estabelecer as necessidades para o trabalho;
    - Definir um objetivo geral para a pesquisa e as perguntas-chave dela;
    - Definir o protocolo de pesquisa a ser utilizado.
- Condução;
    - Pesquisa e seleção;
    - Definição do formulário para extração dos dados;

- Definição das perguntas para as entrevistas;
- Entrevistas;
- Extração de dados;
- Análise dos dados.

- Documentação.
  - Análise e documentação das possíveis ameaças à validade do estudo;
  - Documentação dos resultados;

Foram selecionados 41 estudos de pesquisa relevantes sobre o tema através de pesquisa em mecanismos de busca de sites de referência em publicação de estudos de engenharia de software (IEEE Xplore, ACM Digital, SCOPUS e Web of Science). Após a seleção, foi definido um formulário para a coleta dos dados da pesquisa, baseado nas três perguntas desejadas.

Também foram conduzidas entrevistas com 12 profissionais e/ou pesquisadores da área de aprendizado de máquina. Foram utilizadas 15 perguntas abertas para que fosse possível captar a maior quantidade de detalhes e dados qualitativos, para minimizar o risco à validade do estudo.

Finalmente, os dados foram extraídos analisados e sintetizados utilizando as recomendações contidas em Cruzes and Dyba (2011) utilizando ambas as análises quantitativa e qualitativa (Franzosi, 2010) e síntese narrativa (Rodgers et al., 2009).

Foram utilizadas melhores práticas de estudos sistemáticos para mitigar os riscos da validade da conclusão do estudo. Para isso, foram documentados todos os passos da pesquisa e produzido um pacote de replicação da pesquisa que foi disponibilizado ao público.

Quais os principais resultados obtidos pelo estudo?

Com relação às três perguntas primárias do estudo, foram encontradas algumas correlações interessantes entre as respostas e os temas trabalhados, que podem ser divididas em seis grandes categorias: Arquitetura, dados, evolução, garantia de qualidade, modelo, e Ciclo de vida do desenvolvimento do software.

- Arquitetura: O uso de padrões de arquitetura é usualmente relacionado como uma boa prática. As boas práticas e as decisões de design citadas, identificaram benefícios em algumas arquiteturas específicas, tais como:

Microserviços e Siemens four view. O uso de diferentes arquiteturas, introduz desafios diferentes na sua implementação.

- Dados: Lidar com os dados nessas aplicações é desafiador em vários aspectos, tais como: gerenciamento, visualização e privacidade. Alguns desses desafios são endereçados por decisões de design específicas. No entanto parece que para um número significativo de desafios relacionados aos dados, não há nenhuma menção nas melhores práticas e decisões de design.
- Modelo: Foram identificadas duas relações entre desafios, melhores práticas e decisões de design. Ambas dizem respeito a seleção do modelo adequado para o problema. No entanto, também para algumas questões importantes relacionadas ao modelo, não há nenhuma menção nas melhores práticas e decisões de design.

Além disso, foram encontradas também, algumas diferenças fundamentais em relação às mesmas três perguntas.

Com relação aos desafios de design (pergunta 1), a categoria dos modelos só aparece em estudos primários, enquanto os desafios de arquitetura são muito mais populares entre as publicações com revisão de pares do que entre os entrevistados. Os desafios relacionados à garantia de qualidade apareceram em ambos os estudos primários e entrevistas. Nos estudos primários o foca era mais nas técnicas e ferramentas, enquanto nas entrevistas o foco foi nas capacidades de se explicar os modelos e na falta de documentação.

Com relação às melhores práticas (pergunta 2), os desenvolvedores entrevistados não citaram nenhuma boa prática evolutiva, citadas em alguns estudos. Em compensação, boas práticas relacionadas a dados foram citadas somente pelos entrevistados. Não houve melhores práticas em comum entre os estudos e os entrevistados, tendo esses últimos citado em maior quantidade o uso de microserviços. Com relação a garantia de qualidade, entrevistados somente citaram testes, enquanto os estudos focaram em outras técnicas, p.ex. simulações, etc. Com relação às melhores práticas de ciclo de vida, entrevistados sugeriram práticas que também envolvem aspectos organizacionais enquanto os estudos focam em identificar as principais fases do desenvolvimento.

Com relação às decisões de design (pergunta 3), foram identificadas bem menos discrepâncias entre os entrevistados e os estudos.

Seria interessante aprofundar nas razões das discrepâncias, mas, com os dados coletados, não seria possível na extensão necessária. Então isso será deixado para um estudo futuro. Ainda assim, acredita-se que esses pontos já levantados já geram valor e servem de base para pesquisadores e desenvolvedores.

APÊNDICE M - FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

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

2 Regressão (RNA)

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

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

Informações:

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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"
```

```
data = pd.read_csv(url, delimiter=';')
```

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

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

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

3 Sistemas de Recomendação

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

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

Informações:

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

4 Deepdream

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

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

Informações:

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

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

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

B – RESOLUÇÃO

1 Classificação RNA

COMENTÁRIOS/EXPLICAÇÕES:

1. **Gráficos de perda e acurácia**:

   Observa-se nos gráficos que com o passar das épocas de treinamento, a perda decresce e a acurácia aumenta gradualmente para os dados de treino, até a época 10. O mesmo ocorre para os dados de validação, porém não na mesma proporção, demonstrando valores de perda maiores e acurácia menores nesses dados do que o apresentado para os dados de treino. Mesmo assim, o valor da acurácia nos dados de validação ainda pode ser considerado bom, já que fica próxima a 88% na época 10. Talvez aumentar o número de épocas de treino surtisse algum efeito de melhora nesses valores.

2. **Imagem gerada na seção "Mostrar algumas classificações erradas":**

   É selecionado aleatoriamente um caso entre as predições que foram efetuadas e que não previram a categoria corretamente. Conforme pode-se observar no exemplo selecionado, o modelo previu para a imagem a categoria 4 (coat – casaco) quando o correto seria 3 (dress – vestido). É interessante poder visualizar a imagem, pois nota- se que, a olho nu, a mesma pode ser considerada semelhante a ambas as categorias, o que sugere que se poderia tentar trabalhar a configuração do modelo de predição para melhorar a sua eficácia com amostras desse tipo.

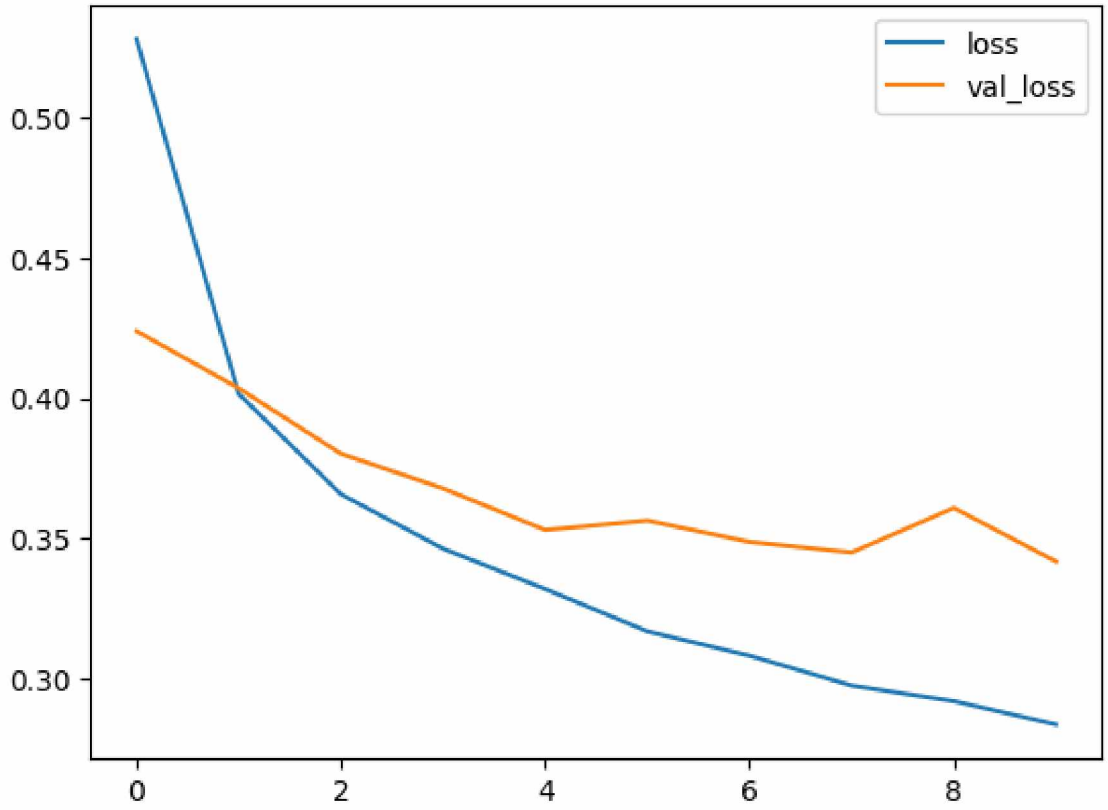
CÓDIGO

```
# %% [markdown]
# # Questão 1 - Classificação (RNA)

# %% [markdown]
# #### 1. Importação das bibliotecas

# %%
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix

tf.__version__

# %% [markdown]
# #### 2. Importação dos dados

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

# %%
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:  (60000, 28, 28)
y_train.shape:  (60000,)
x_test.shape:  (10000, 28, 28)
y_test.shape:  (10000,)
```

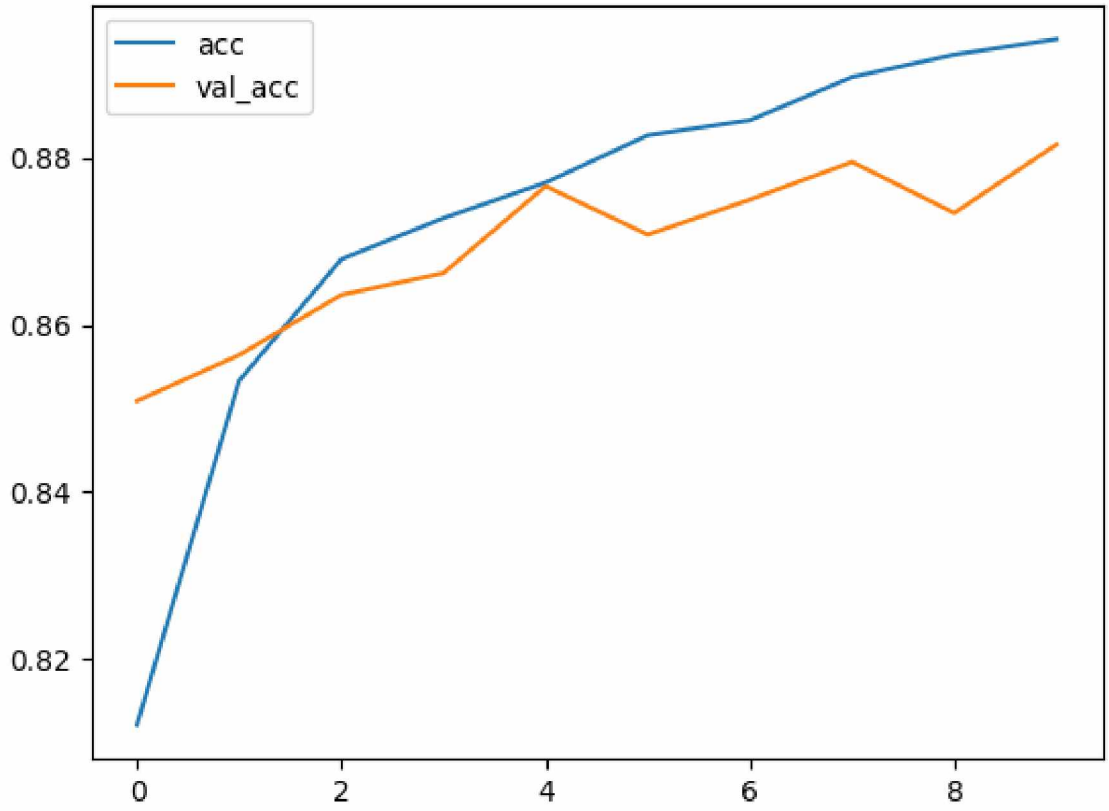
```
# %% [markdown]
# **Classes do Dataset Fashion MNIST:**
#
# | Lable | Descrição |
# |----|---|
# | 0 | T-shirt/top |
# | 1 | Trouser |
# | 2 | Pullover |
# | 3 | Dress |
# | 4 | Coat |
# | 5 | Sandal |
# | 6 | Shirt |
# | 7 | Sneaker |
# | 8 | Bag |
# | 9 | Ankle boot |

# %%
display(x_train)

# %%
display(y_train)

# %% [markdown]
# #### 3. Pré-processamento

# %%
x_train, x_test = x_train/255.0, x_test/255.0

# %% [markdown]
# #### 4. Criação do modelo

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

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

# %% [markdown]
# #### 5. Compilação e treinamento do modelo

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

# %%
r = model.fit(x_train,
              y_train,
              validation_data=(x_test, y_test),
              epochs=10)
```

```
Epoch 1/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.7615 - loss: 0.6722 -
val_accuracy: 0.8509 - val_loss: 0.4239
...
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8940 - loss: 0.2835 -
val_accuracy: 0.8816 - val_loss: 0.3418
```

```
# %% [markdown]
# #### 6. Avaliação do modelo

# %%
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
```



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




```
# %%
# Avaliar o modelo com a base de teste
print( model.evaluate(x_test, y_test) )
```

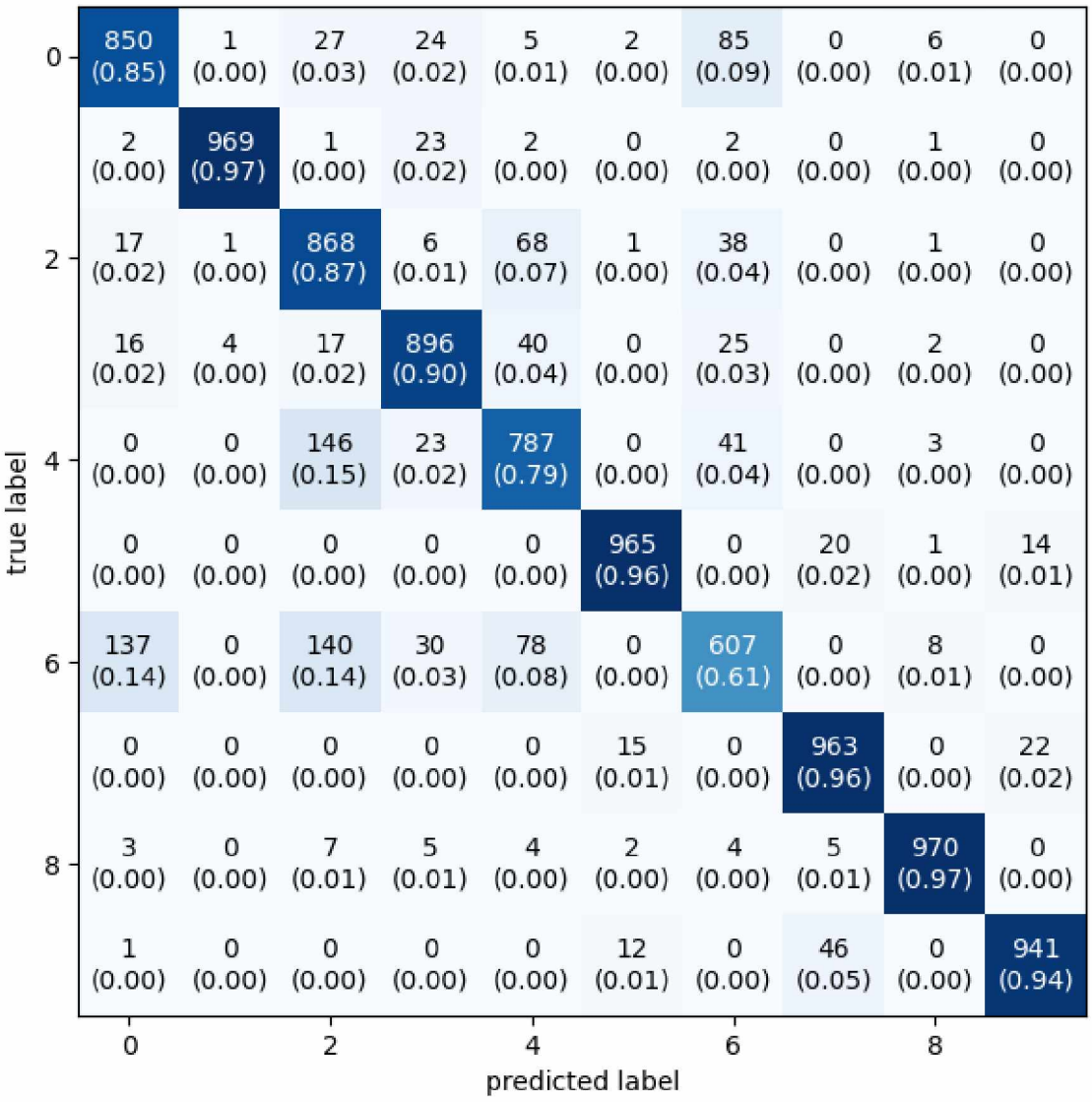
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - accuracy: 0.8839 - loss: 0.3343
[0.3418148458003998, 0.881600022315979]
```

```
# %% [markdown]
# #### 7. Predições
```

```
# %%
y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)
```

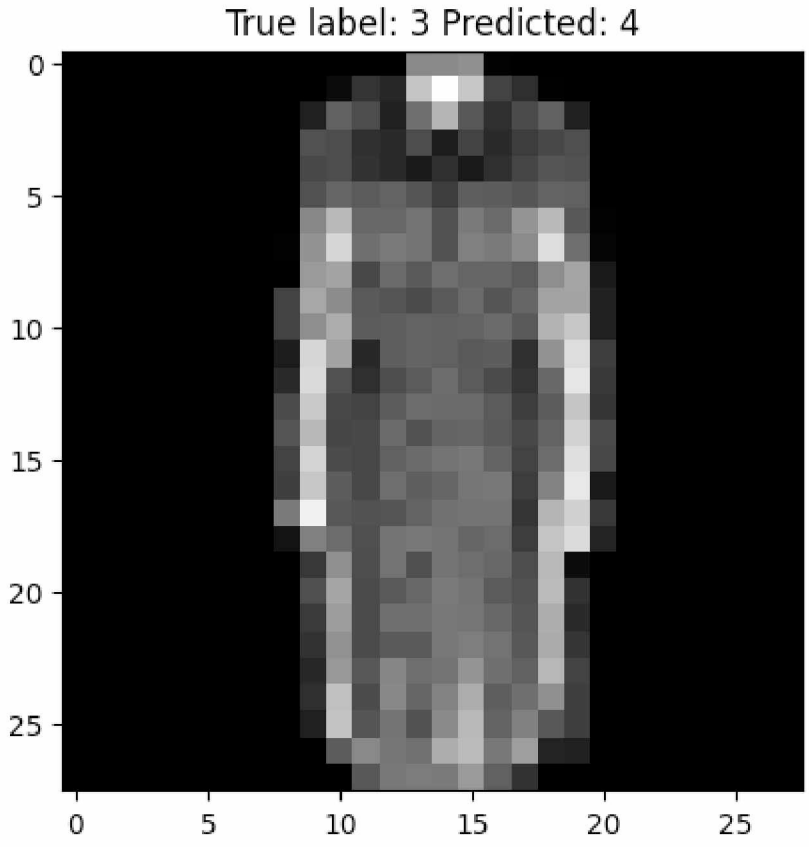
```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step
[9 2 1 ... 8 1 5]
```

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



```
# %% [markdown]
# #### 8. Mostrar algumas classificações erradas

# %%
misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)

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

Text(0.5, 1.0, 'True label: 3 Predicted: 4')



2 Regressão (RNA)

COMENTÁRIOS/EXPLICAÇÕES:

1. **Gráficos de avaliação do modelo (loss)**:

   Devido à utilização da configuração de "early stop" no treinamento do modelo, o processo de treinamento utilizou somente 87 épocas das 1500 que foram definidas na sua parametrização. O porquê disso pode ser observado tanto no gráfico de perda, quanto no de erro. A partir da época 20, os valores praticamente se estabilizam e o gráfico vira quase uma reta. Isso indica que poderiam ser utilizadas menos épocas de treinamento para o modelo, de 20 a 30 por exemplo, que, ainda assim, seria atingido o mesmo desempenho nas predições.

2. **Métricas de avaliação do modelo (pelo menos uma entre MAE, MSE, R^2):**

   Apesar dos valores apresentados nas métricas de erro serem pequenos, analisando- se o coeficiente de determinação R^2 das predições, verifica-se que o modelo apresentou uma performance muito pobre, com um resultado em torno de 22% de acurácia. Faz-se necessário buscar a melhoria dessa performance através de mudanças na parametrização do modelo, tais como: utilização de um gradiente de descida diferente, com "learning rates" diferentes, ou até mesmo o ajuste do dataset, verificando-se a correlação entre as features para que se possa desconsiderar aquelas que possuem baixa ou nenhuma correlação com o valor target.

CÓDIGO

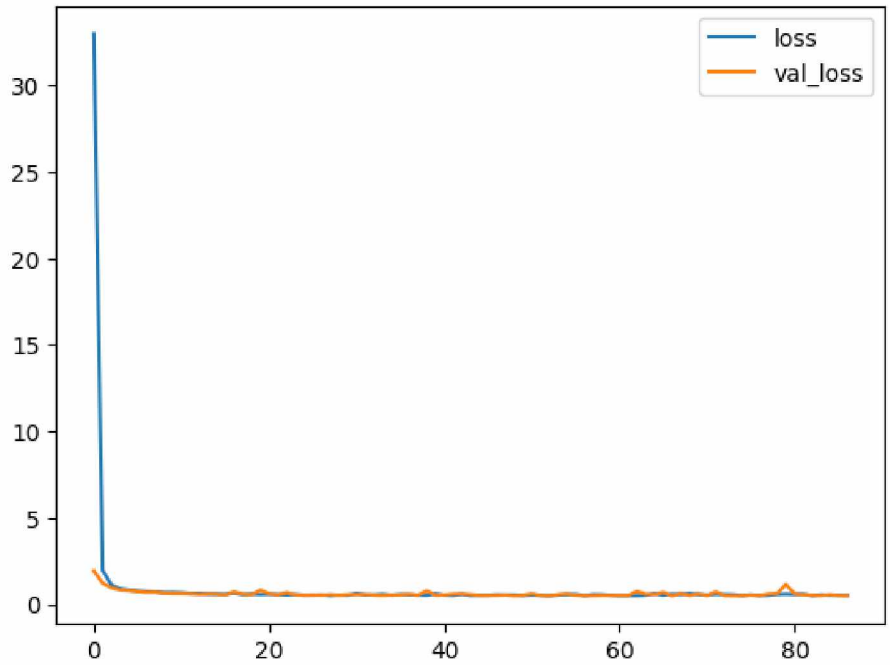
```
# %% [markdown]
# # Questão 2 - Regressão (RNA)
#
# ---

# %% [markdown]
# #### 1. Importação das bibliotecas

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

from tensorflow.python.keras import backend
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
```

```
# %% [markdown]
# ---
# #### 2. Importação dos dados

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

# %%
data.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
]

# %% [markdown]
# ---
# #### 3. Visualização dos dados

# %%
data.head()

# %%
data.shape

# %%
# Separa as colunas de features da target
X = data.iloc[:,0:10].astype(float)
y = data.iloc[:,11].astype(float)

# %% [markdown]
# #### 4. Separação da base em treino e teste (75/25)

# %%
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25)

# %% [markdown]
# #### 4. Criação do modelo
```

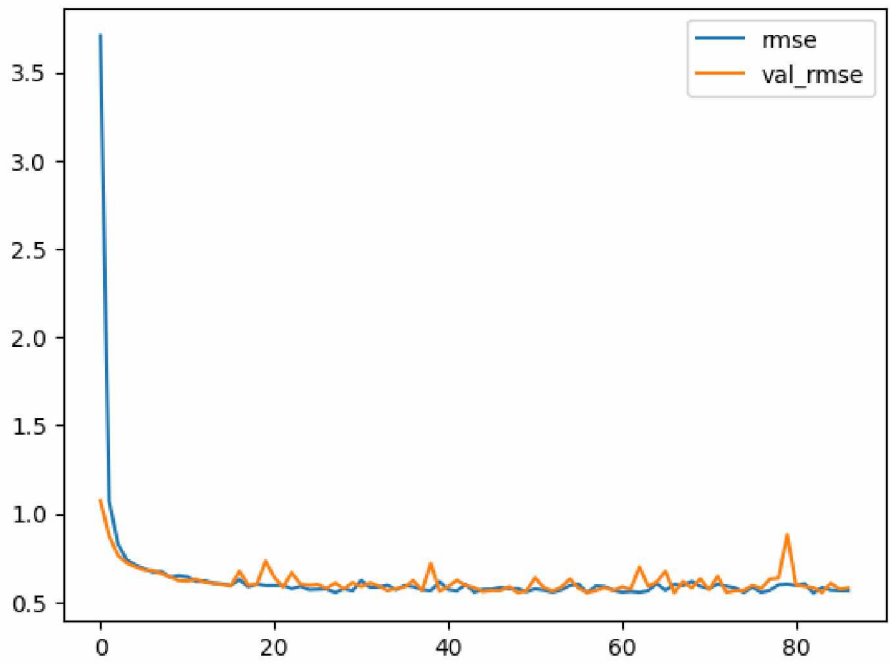
```
# %%
# 3 camadas
i = tf.keras.layers.Input(shape=(10,))
x = tf.keras.layers.Dense(50, activation="relu")(i)
x = tf.keras.layers.Dense(1)(x)

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

# %% [markdown]
# #### 5. Compilação e treinamento do modelo

# %%
# Criação de funções para as métricas R2 e RMSE serem inseridas no modelo
def rmse(y_true, y_pred):
  return backend.sqrt(backend.mean( backend.square(y_pred - y_true), axis=-1) )

def r2(y_true, y_pred):
  media = backend.mean(y_true)
  num   = backend.sum (backend.square(y_true - y_pred))
  den   = backend.sum (backend.square(y_true - media))
  return (1.0 - num/den)

# %%
# Compilação
optimizer=tf.keras.optimizers.Adam(learning_rate=0.05)
# optimizer=tf.keras.optimizers.SGD(learning_rate=0.2, momentum=0.5)
# optimizer=tf.keras.optimizers.RMSprop(0.01)

model.compile(optimizer=optimizer,
              loss=tf.keras.losses.mse,
              metrics=[rmse, r2])

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

# %%
r = model.fit(X_train, y_train,
              epochs=1500,
              validation_data=(X_test, y_test),
              callbacks=[early_stop])
```

```
Epoch 1/1500
38/38 ━━━━━━━━━━━━━━━━━━━━ 7s 56ms/step - loss: 77.3655 - r2: -114.3765 - rmse:
6.0767 - val_loss: 1.9195 - val_r2: -2.2794 - val_rmse: 1.0737
...
Epoch 87/1500
38/38 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.5137 - r2: 0.1786 - rmse: 0.5671
- val_loss: 0.5076 - val_r2: 0.1564 - val_rmse: 0.5806
```

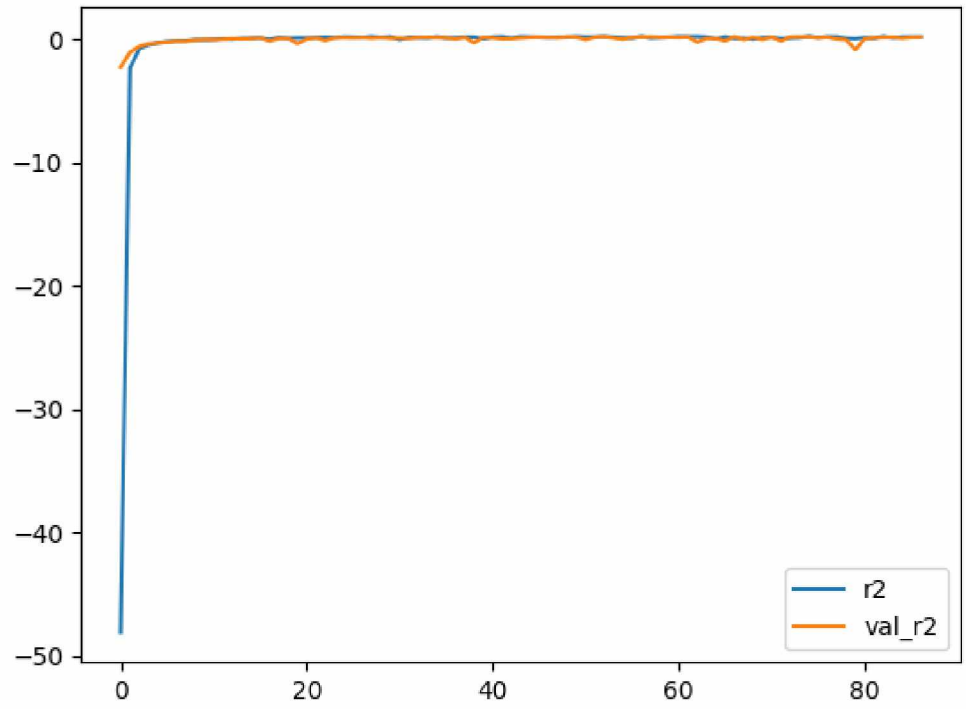
```
# %% [markdown]
# #### 6. Avaliação do modelo

# %%
plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()
```




```
# %%

plt.plot( r.history["rmse"], label="rmse" )
plt.plot( r.history["val_rmse"], label="val_rmse" )
plt.legend()
```



```
# %%
plt.plot( r.history["r2"], label="r2" )
plt.plot( r.history["val_r2"], label="val_r2" )
plt.legend()
```




```
# %% [markdown]
# #### 7. Predições

# %%
# Predição
y_pred = model.predict(X_test).flatten()
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step
```

```
# %%
# Cálculo das métricas de acurácia: mse, r2 e rmse
mse  = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2   = r2_score(y_test, y_pred)

# %%
# Resultados das métricas de acurácia
print("mse     = ", mse)
print("rmse    = ", rmse)
print("r2      = ", r2)
```

```
mse     =  0.4866209823398623
rmse    =  0.6975822405565255
r2      =  0.21903228640689731
```

3 Sistemas de Recomendação

COMENTÁRIOS/EXPLICAÇÕES:

1. **Gráficos de avaliação do modelo (loss):**

   Observando-se o gráfico, verifica-se que há uma queda gradual através das épocas no valor de perda para os dados de treino. No entanto, a partir da época 15, esse valor se estabiliza, indicando que poderiam ser utilizadas apenas em torno de 15 épocas para o treinamento, garantindo os mesmos resultados. No entanto, para os dados de validação, o valor da perda estabiliza a partir da época 10 e permanece alto, indicando que a performance do modelo não foi boa. Isso pode ser devido a pouca quantidade de dados de validação, desbalanceamento entre as classes de livro + usuário, e/ou necessidade de revisão da configuração da estrutura da rede neural (camadas e profundidade, por exemplo).

2. **Exemplo de recomendação de livro para determinado Usuário:**

   Foi testada a recomendação de livro para o usuário com ID 278851, onde o modelo seleciona os livros semelhantes aos que ele já havia lido e é então exibida a opção dentre estas que apresenta o maior rating.

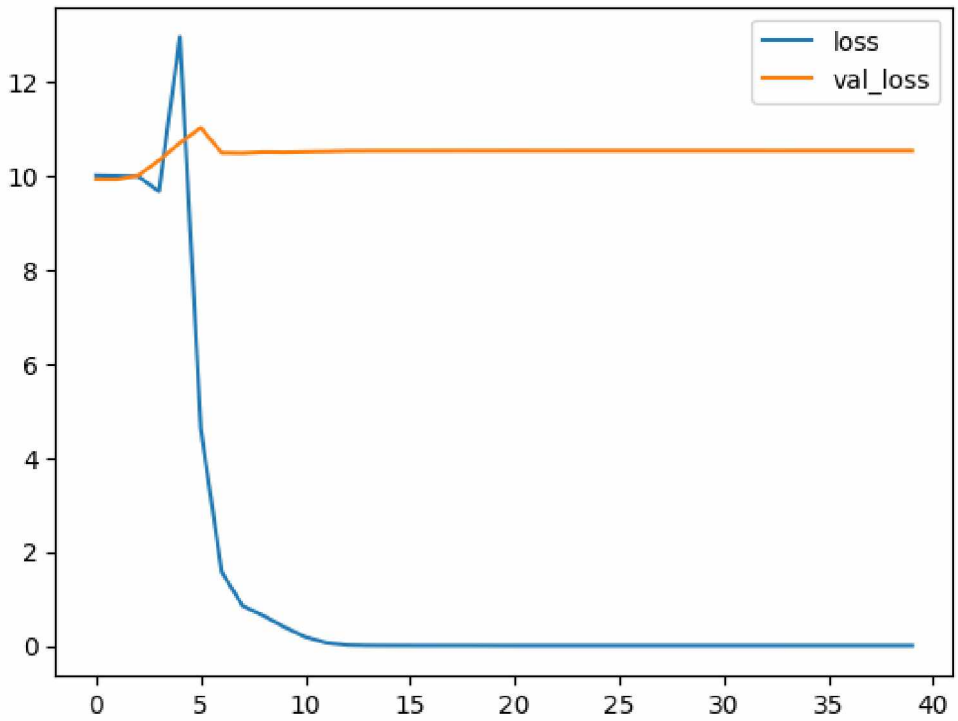
CÓDIGO

```
# %% [markdown]
# # Questão 3 - Sistemas de Recomendação
#
# ---

# %% [markdown]
# #### 1. Importação das bibliotecas

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

from sklearn.utils import shuffle

import numpy as np
import pandas as pd
from google.colab import files
import matplotlib.pyplot as plt
import io
```

```
# %% [markdown]
# ---
# #### 2. Importação dos dados

# %%
uploaded = files.upload()

# %%
df = pd.read_csv('Base_livros.csv')
df.head()

# %% [markdown]
# #### 3. Conversão de ID_usuario e ISBN para categóricos
#

# %%
df.ID_usuario = pd.Categorical(df.ID_usuario)
df['new_ID_usuario'] = df.ID_usuario.cat.codes

df.ISBN = pd.Categorical(df.ISBN)
df['new_ISBN'] = df.ISBN.cat.codes

# %%
# Dimensões
N = len(set(df.new_ID_usuario))
M = len(set(df.new_ISBN))

# dimensão do embedding (tentar outros)
K = 10

# %% [markdown]
# #### 4. Criar o modelo

# %%
# Usuário
u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

# ISBN
i = Input(shape=(1,))
i_emb = Embedding(M, K)(i)  # saída : num_samples, 1, K
i_emb = Flatten()(i_emb)    # saída : num_samples, K

x = Concatenate()([u_emb, i_emb])

x = Dense(1024, activation="relu")(x)
x = Dense(1)(x)

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

# %% [markdown]
# #### 5. Compilação do modelo
```

```
# %%
model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

# %% [markdown]
# #### 6. Separação dos dados e pré-processamento
#

# %%
user_ids, isbn_ids, ratings = shuffle(df.new_ID_usuario, df.new_ISBN, df.Notas)

Ntrain = int(0.8 * len(ratings)) # separar os dados 80% x 20%

train_user = user_ids[:Ntrain]
train_isbn = isbn_ids[:Ntrain]
train_ratings = ratings[:Ntrain]
test_user = user_ids[Ntrain:]
test_isbn = isbn_ids[Ntrain:]
test_ratings = ratings[Ntrain:]

# centralizar as notas
avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

# %% [markdown]
# #### 7. Treinamento do modelo

# %%
epochs = 40
r = model.fit(
    x=[train_user, train_isbn],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2, # não imprime o progresso
    validation_data=([test_user, test_isbn], test_ratings)
)

Epoch 1/40
101/101 - 3s - 26ms/step - loss: 10.0087 - val_loss: 9.9241
Epoch 2/40
101/101 - 0s - 3ms/step - loss: 10.0002 - val_loss: 9.9236
...
Epoch 39/40
101/101 - 0s - 3ms/step - loss: 0.0035 - val_loss: 10.5324
Epoch 40/40
101/101 - 0s - 3ms/step - loss: 0.0035 - val_loss: 10.5322
```

```
# %% [markdown]
# #### 8. Plotar a função de perda

# %%
plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()
```




```
# %% [markdown]
# ### 9. Recomendações para o usuário 278851

# %%
input_usuario = np.repeat(a=278851, repeats=M)
Livro = np.array(list(set(isbn_ids)))

preds = model.predict( [input_usuario, Livro] )

rat = preds.flatten() + avg_rating

idx = np.argmax(rat)

print("Recomendação: Livro - ", Livro[idx], " / ", rat[idx] , "*")
print('Nome do Livro: ', df.loc[df['new_ISBN'] == Livro[idx], 'Titulo'].iloc[0])
```

```
4028/4028 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step
Recomendação: Livro -  39793  /  10.507515 *
Nome do Livro:  Der Sturm / The Perfect Storm
```

4 Deepdream

COMENTÁRIOS/EXPLICAÇÕES:

1. **Imagem onírica obtida por Main Loop:**

   Este resultado mostra a amplificação de padrões locais pela rede neural, onde pequenos detalhes são intensificados e aparecem texturas e formas abstratas na imagem. O Main Loop foca em uma única escala.

2. **Imagem onírica obtida ao levar o modelo até uma oitava:**

   A adição de oitavas permite que o modelo processe a imagem em múltiplas escalas, amplificando padrões em níveis de detalhe maiores e menores. O resultado apresenta padrões mais amplos e complexos que interagem de forma mais integrada.

3. **Diferenças entre as imagens:**

   No Main Loop, as alterações são mais localizadas e detalhadas, gerando uma textura onírica intensa em pequenas áreas. Usando oitavas, o modelo cria padrões que se espalham e interagem em diferentes escalas, resultando em uma imagem mais surreal e globalmente modificada.

CÓDIGO

```
# %% [markdown]
# # 4. Deepdream

# %% [markdown]
# ### 1. Importação das bibliotecas

# %%
import tensorflow as tf
import numpy as np

import matplotlib as mpl

import IPython.display as display
import PIL.Image

# %% [markdown]
# ### 2. Importação da imagem

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

```
# %%
# Download da imagem e gravação em array Numpy
def download(url, max_dim=None):
  name = url.split('/')[-1]
  image_path = tf.keras.utils.get_file(name, origin=url)
  img = PIL.Image.open(image_path)
  if max_dim:
    img.thumbnail((max_dim, max_dim))
  return np.array(img)

# Normalização da imagem
def deprocess(img):
  img = 255*(img + 1.0)/2.0
  return tf.cast(img, tf.uint8)

# Mostra a imagem
def show(img):
  display.display(PIL.Image.fromarray(np.array(img)))


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

```
# %% [markdown]
#
# ### 3. Preparar o modelo de extração de recursos

# %% [markdown]
# Usando o modelo pré treinado
[InceptionV3](https://keras.io/api/applications/inceptionv3/)  que é semelhante ao
modelo originalmente usado no DeepDream.
```

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

# %% [markdown]
# ### Escolhe duas camadas do modelo InceptionV3
#
#

# %%
# Maximizando as ativações das camadas
names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

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

# %% [markdown]
# ### 4. Cálculo da perda (*loss*)

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

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

  return  tf.reduce_sum(losses)

# %% [markdown]
# ### 5. Subida de gradiente (*Gradient ascent*)

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

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

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

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

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

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

      return loss, img

# %%
deepdream = DeepDream(dream_model)

# %% [markdown]
# ### 6. Circuito principal (*Main Loop*)

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

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

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

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


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

  return result
```

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

```
# %% [markdown]
# ## 7. Levando o modelo até uma oitava

# %%
import time
start = time.time()

OCTAVE_SCALE = 1.30

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

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

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

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

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

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

APÊNDICE N - VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

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

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

B – RESOLUÇÃO

Conjunto de dados brutos

Esta base de dados foi adquirida através do download dos arquivos no sítio web do IBGE (Instituo Brasileiro de Geografia e Estatística), no seguinte endereço eletrônico:

https://www.ibge.gov.br/estatisticas/economicas/precos-e-custos/9256-indice-nacional-de-precos-ao-consumidor-amplo.html

A mesma demonstra os dados mês a mês do índice de inflação IPCA, calculado pelo instituto, no período de Janeiro/2024 a Janeiro/2025. Os dados foram tabulados em uma planilha eletrônica no formato de arquivos XLSX, onde também

foram criados manualmente, dados complementares das dimensões necessárias para se criar um mínimo modelo semântico dimensional, de forma a que o mesmo pudesse ser implementado na ferramenta de visualização escolhida.

Os dados brutos são demonstrados abaixo:

Índices de inflação do IPCA por período, categoria e região amostral (131 registros)

Periodo	Cod Categoria	Categoria	AJU	BH	BEL	DF	CG	CUR	FOR	GOI	VIT	POA	REC	RB	RJ	SAL	SL	SP	NAC
jan/24	1	Alimentação e bebidas	1,7	2,73	2,7	0,85	1,06	1,5	1,72	2,22	1,98	1,06	0,87	1,41	1,18	1,38	1,99	0,74	1,38
jan/24	2	Habitação	0,58	1,2	0,17	-0,39	1,03	0,01	0,39	-0,91	-0,1	-0,07	0,53	2,3	0,19	-2,26	2,19	0,6	0,25
jan/24	3	Artigos de residência	1,47	0,76	0,87	-0,64	0,11	0,22	0,85	-0,23	0,05	-0,77	0,04	0,39	0,59	-0,31	0,72	0,36	0,22
jan/24	4	Vestuário	-0,13	0	0,52	-1,58	1,48	0,75	-0,61	0,54	-0,5	-0,29	-0,29	0,1	-0,18	0,15	0,41	0,45	0,14
jan/24	5	Transportes	0,13	0,35	-1,15	-2,44	-0,66	-0,45	-0,47	1,31	-0,63	-1,32	0,79	-0,83	0,05	-0,66	-0,06	-1,23	-0,65
jan/24	6	Saúde e cuidados pessoais	0,42	1	0,67	0,82	0,39	0,74	1,34	0,52	0,87	0,84	0,84	0,68	0,53	0,97	0,64	0,9	0,83
jan/24	7	Despesas pessoais	0,93	0,63	0,88	0,56	0,79	0,58	1,06	1,01	0,71	0,9	0,68	1,09	0,87	1,22	1,19	0,83	0,82
jan/24	8	Educação	0,88	0,51	0,48	0,5	0,52	0,3	0,38	0,3	0,27	0,45	0,27	0,06	0,13	0,5	0,34	0,22	0,33
jan/24	9	Comunicação	-0,2	0,02	-0,46	-0,1	0,16	-0,17	0,33	0,22	0,03	0,29	0,47	0,15	-0,28	-0,39	-0,18	-0,16	-0,08
jan/24	10	Índice geral	0,73	1,1	0,75	-0,36	0,48	0,39	0,68	0,87	0,37	0,13	0,63	0,63	0,44	0,13	1,06	0,25	0,42
fev/24	1	Alimentação e bebidas	0,97	0,99	0,92	0,81	0,82	0,39	0,89	-0,47	0,77	0,7	1,05	0,85	1,56	1,11	0,7	1,17	0,95
fev/24	2	Habitação	0,32	0,41	0,38	0,78	0,15	0,37	0,77	0,78	0,32	0,26	0,41	-1,18	0,13	0,43	1,87	0	0,27
fev/24	3	Artigos de residência	-0,71	0,49	0,07	-0,09	-0,59	-0,81	0,61	-0,33	-0,07	0,36	-0,43	0,28	-0,7	-0,03	-0,4	0,02	-0,07
fev/24	4	Vestuário	-0,05	-0,5	0,2	-0,91	-0,35	0,3	-0,7	-0,22	-0,5	-0,3	-0,46	-0,53	-1,06	-1,45	0,4	-0,43	-0,44
fev/24	5	Transportes	2,11	0,22	-0,07	0,89	0,94	1,49	0,08	0,51	0,07	0,27	0,16	-0,22	0,51	1,11	1,63	0,93	0,72
fev/24	6	Saúde e cuidados pessoais	0,72	0,66	0,66	0,69	0,37	0,35	0,74	0,63	0,63	0,11	0,9	0,69	1	0,86	0,09	0,71	0,65
fev/24	7	Despesas pessoais	-0,07	0,73	0,4	-0,28	0,96	0,12	-0,2	0,43	0,51	0,09	-0,06	0,23	-0,2	-0,13	0,64	-0,14	0,05
fev/24	8	Educação	4,64	5,36	5,35	2,82	5,76	4,42	5,46	3,95	5,65	3,54	4,62	4,06	5,28	5,95	3,9	5,44	4,98
fev/24	9	Comunicação	0,45	1,07	1,62	1,59	1,35	1,31	1,53	1,65	1,38	1,48	1,06	1,23	1,73	1,4	1,38	1,84	1,56
fev/24	10	Índice geral	1,09	0,82	0,69	0,75	0,81	0,84	0,84	0,51	0,7	0,52	0,74	0,26	0,88	0,96	1,06	0,93	0,83
mar/24	1	Alimentação e bebidas	0,6	-0,04	1,32	1,14	0,01	0,09	0,73	0,78	0,24	0,13	1,08	-0,01	0,54	0,31	2	0,61	0,53
mar/24	2	Habitação	0,67	0,22	-0,14	0,68	0,14	0,03	0,64	1,2	0,4	-0,16	-0,26	1,12	0,46	0,12	1,01	0,02	0,19
mar/24	3	Artigos de residência	0,31	0,42	-0,5	-0,59	-0,72	0,32	-0,33	0,09	0,48	-0,17	-0,54	-0,25	-0,39	-0,15	0,2	0,04	-0,04
mar/24	4	Vestuário	0,51	-0,16	0,1	0,47	0,46	-0,42	0,45	0,41	0,43	0,31	-0,33	0,19	-0,15	0,59	0,45	-0,17	0,03
mar/24	5	Transportes	0,71	0,09	0,63	-1,15	0,22	-0,39	-0,23	-0,23	-0,59	-1,09	0,53	0,58	-0,45	-0,6	0,26	-0,36	-0,33
mar/24	6	Saúde e cuidados pessoais	0,4	0,32	0,8	0,51	0,13	0,42	0,08	0,61	0,17	0,17	0,35	-0,27	0,44	0,97	0,38	0,43	0,43
mar/24	7	Despesas pessoais	0,44	0,35	-0,01	1	0,4	0,72	0,25	-0,04	0,34	0,45	-0,06	-0,57	0,31	0,17	0,31	0,26	0,33
mar/24	8	Educação	0,01	-0,03	0,03	-0,08	-0,07	0,26	-0,03	0,03	0	0,36	0,1	0,2	0,27	0,03	0	0,19	0,14

mar/24	9	Comunicação	0,27	-0,09	0,03	-0,01	-0,15	-0,17	0,02	-0,1	-0,05	-0,15	-0,16	-0,44	-0,31	-0,06	-0,15	-0,12	-0,13
mar/24	10	Índice geral	0,5	0,12	0,54	0,21	0,11	0,03	0,28	0,36	0,05	-0,13	0,33	0,18	0,17	0,16	0,81	0,14	0,16
abr/24	1	Alimentação e bebidas	1,94	0,23	0,77	1,11	0,53	0,75	0,47	0,75	0,33	0,84	1,07	0,13	0,03	1,07	0,06	0,87	0,7
abr/24	2	Habitação	0,53	0,13	-0,65	-0,19	-0,19	-0,05	-1,02	-0,83	0,47	0,18	-0,13	0,18	0,34	0,28	-0,02	0,02	-0,01
abr/24	3	Artigos de residência	-0,19	-0,17	0,47	-0,19	0,39	-0,06	0,71	-0,53	1,01	-0,34	0,3	0,33	-0,78	-0,56	-0,33	-0,5	-0,26
abr/24	4	Vestuário	-0,5	0,73	0,53	0,59	-0,54	0,54	0,37	0,46	0,22	1,23	-0,06	0,26	0,71	0,04	0,97	0,6	0,55
abr/24	5	Transportes	0,33	0,69	0,24	0,53	0,32	-0,03	-1,54	-0,16	0,31	0,77	0,64	-0,4	-0,5	0,49	0,91	0,03	0,14
abr/24	6	Saúde e cuidados pessoais	1,03	1,04	0,53	1,28	0,95	1,26	1,13	1,62	0,76	1,11	1,13	0,91	1,43	1,42	1,39	1,08	1,16
abr/24	7	Despesas pessoais	0,38	0,1	0,01	0,23	0,97	0,23	-0,32	0,13	0,03	0,33	-0,06	0,31	-0,11	0,19	0,28	0,04	0,1
abr/24	8	Educação	0,1	0,05	0,14	0,1	-0,03	-0,03	0,03	0,02	0,04	0,13	-0,05	0,2	0,1	0,11	0,11	0,04	0,05
abr/24	9	Comunicação	0,98	0,92	1,1	0,53	0,05	1,02	0,73	0,33	1,13	0,84	0,5	0,19	-0,06	0,83	0,58	0,15	0,48
abr/24	10	Índice geral	0,78	0,45	0,33	0,55	0,36	0,37	-0,15	0,24	0,43	0,64	0,55	0,15	0,15	0,63	0,46	0,35	0,38
mai/24	1	Alimentação e bebidas	0,91	0,01	-0,88	0,33	1,25	0,48	1,25	0,32	0,85	2,63	0,82	-0,27	0,59	0,59	0,97	0,46	0,62
mai/24	2	Habitação	-0,62	0,52	0,69	0,67	-0,86	0,84	-0,41	0,15	0,07	1,19	-0,17	-0,59	0,67	1,31	0,43	0,86	0,67
mai/24	3	Artigos de residência	-0,19	-0,18	0,37	0,21	0,38	-1,02	0,6	-0,01	0,43	-1,54	0	0,11	-0,6	-0,38	0,2	-0,86	-0,53
mai/24	4	Vestuário	0,83	0,46	1,12	-0,66	1,5	0,86	0,1	0,37	0,79	0,53	0,26	0,74	0,42	0,33	0,79	0,49	0,5
mai/24	5	Transportes	1,34	1,52	0,02	0,16	0,38	0,5	0,7	-1,26	0,55	0,74	0,04	0,48	0,24	0,64	0,57	0,38	0,44
mai/24	6	Saúde e cuidados pessoais	0,9	0,97	0,81	0,92	0,73	0,9	0,89	0,68	0,98	-0,02	1,21	0,8	0,8	0,76	0,92	0,51	0,69
mai/24	7	Despesas pessoais	0,09	0,59	0,6	0,34	0,2	0,13	0,23	0,48	-0,27	0,36	0,55	0,46	0,33	0	0,03	0,03	0,22
mai/24	8	Educação	0,06	0,2	0,23	0,14	-0,03	0,14	0,08	0,07	0,03	0,12	-0,03	0,37	0,08	0,13	0,09	0,05	0,09
mai/24	9	Comunicação	0,63	0,56	0,49	-0,11	-0,25	0,52	0,34	-0,08	0,66	-0,41	0,28	0,15	0,09	0,46	0,58	-0,01	0,14
mai/24	10	Índice geral	0,6	0,63	0,13	0,34	0,42	0,49	0,55	-0,06	0,51	0,87	0,43	0,19	0,44	0,58	0,63	0,37	0,46
jun/24	1	Alimentação e bebidas	0,06	0,47	-0,83	0,5	0,38	0,97	0,24	0,69	1,76	-0,52	0	0,27	0,9	0	-0,14	0,78	0,44
jun/24	2	Habitação	0,22	2,18	1,19	0,88	0,33	0,01	0,66	0,17	-1,27	-0,48	0,06	1,03	-0,34	-0,62	0,67	0,22	0,25
jun/24	3	Artigos de residência	-0,39	-0,15	0,94	-0,55	-0,75	-0,5	-0,51	0,69	0,6	0,55	-0,74	0,17	0,19	-0,13	0,02	0,62	0,19
jun/24	4	Vestuário	-0,26	0,06	-0,37	0,27	-0,29	0,28	0,59	0,92	-0,09	-0,44	0,42	-0,42	0,6	0,26	0,16	-0,31	0,02
jun/24	5	Transportes	-0,47	-0,4	0,42	0,34	-0,23	-0,33	0,13	0,65	-0,69	0,06	-0,52	0,3	-0,4	-0,02	0,18	-0,36	-0,19
jun/24	6	Saúde e cuidados pessoais	0,83	0,55	0,26	0,53	0,42	0,97	0,49	0,52	0,38	0,5	0,15	0,46	0,37	0,31	0,08	0,7	0,54
jun/24	7	Despesas pessoais	0,32	0,51	0,58	-0,19	0,25	0,29	0,45	0,29	0,22	-0,37	0,04	0,64	-0,15	-0,01	0,54	0,6	0,29

jun/24	8	Educação	0	0,1	0,17	0,01	0,13	0,06	-0,02	0	0,02	-0,04	0,01	0	0,01	0,09	-0,11	0,1	0,06
jun/24	9	Comunicação	-0,43	0,25	-0,18	0,16	-0,09	0,06	-0,38	0,06	0	0,2	-0,62	-0,33	0,02	-0,14	-0,85	-0,25	-0,08
jun/24	10	Índice geral	0,08	0,46	0,13	0,34	0,12	0,25	0,28	0,5	0,05	-0,14	-0,09	0,34	0,11	-0,04	0,11	0,29	0,21
jul/24	1	Alimentação e bebidas	-1,02	-1,37	-0,66	-1,31	-1,32	-0,94	-0,59	-1,17	-1,95	-1,62	-1,01	-0,3	-1,17	-0,67	-1,15	-0,72	-1
jul/24	2	Habitação	1,16	0,63	0,98	0,56	1,18	1,05	0,17	0,8	0,72	0,71	1,66	0,71	1,15	0,52	1,42	0,57	0,77
jul/24	3	Artigos de residência	0,06	-0,14	0,63	0,44	0,57	0,5	0,22	0,55	0,53	1,57	0,25	1,31	-0,13	0,16	-0,33	0,66	0,48
jul/24	4	Vestuário	0,46	-0,4	0,61	-0,33	1,11	-1,03	-0,19	0,06	0,04	-0,37	-0,31	-0,21	-0,06	-0,22	1,22	0,41	-0,02
jul/24	5	Transportes	0,99	2,1	1,82	1,76	1,29	1,23	2,48	1,93	1,62	2	1,13	1,4	1,12	0,8	2,15	2,33	1,82
jul/24	6	Saúde e cuidados pessoais	0,21	0,12	-0,22	0,33	-0,01	0,03	0,37	0,16	0,21	0,67	0,34	0,81	0,26	0,01	0,56	0,21	0,22
jul/24	7	Despesas pessoais	0,17	0,57	0,36	0,39	0,49	0,37	1,01	0,58	0,49	0,13	0,74	0,35	0,47	1,13	1,04	0,5	0,52
jul/24	8	Educação	-0,06	0,11	0,08	-0,02	0,02	0,21	0,05	0,05	0,07	0,32	0,17	0,22	0,01	0,01	-0,2	0,04	0,08
jul/24	9	Comunicação	0,29	0,15	0,23	0,41	-0,21	0,36	-0,13	0	0,23	0,67	0,54	0	0,3	0,27	0,39	-0,09	0,18
jul/24	10	Índice geral	0,18	0,26	0,39	0,36	0,29	0,3	0,47	0,43	0,25	0,36	0,33	0,53	0,28	0,18	0,53	0,52	0,38
ago/24	1	Alimentação e bebidas	-1,73	-0,12	-0,6	0,24	-0,6	-0,21	-0,89	-0,79	-0,94	-0,76	-0,49	-0,06	-0,41	-1,44	-1,28	-0,13	-0,44
ago/24	2	Habitação	-0,56	-0,42	-2,22	-0,65	-1,04	-0,58	0,19	-0,56	0,18	-0,25	-0,82	-1,17	-0,56	0	-2,13	-0,39	-0,51
ago/24	3	Artigos de residência	0,12	0,63	0,75	1,53	-0,35	1,4	0,07	0,75	1,22	0,92	-0,55	1,46	0,82	0,41	-1,2	0,89	0,74
ago/24	4	Vestuário	0,45	0,43	0,24	0,3	0,88	0,09	0,03	-0,68	0,77	0,27	0,72	-1,52	-0,02	0,23	-0,07	0,88	0,39
ago/24	5	Transportes	-0,15	0,48	0,07	0,04	1,12	-1,26	0,11	-1,65	0,29	0,95	0,09	0,02	0,08	1,14	0,17	-0,07	0
ago/24	6	Saúde e cuidados pessoais	0,21	0,27	0,03	-0,03	0,52	-0,29	0,02	0,01	0,42	0,45	0,71	-0,52	0,2	0,33	0,52	0,34	0,25
ago/24	7	Despesas pessoais	0,11	0,04	0,33	0,65	0,01	0,42	0,48	0,51	0,45	0,41	0,18	0,51	0,24	0,26	0,04	0,14	0,25
ago/24	8	Educação	0,62	0,3	0,58	0,95	0,01	0,66	1,64	1,06	0,35	0,73	0,03	0,82	0,38	0,5	1,04	0,92	0,73
ago/24	9	Comunicação	0,64	0,09	0,28	-0,09	-0,34	-0,21	0,34	0,66	0,2	-0,33	-0,02	-0,25	0,01	0,32	-0,11	0,26	0,1
ago/24	10	Índice geral	-0,33	0,13	-0,4	0,17	0,03	-0,36	0	-0,51	0,14	0,18	-0,07	-0,21	-0,08	0,03	-0,54	0,1	-0,02
set/24	1	Alimentação e bebidas	-1,04	0,95	-0,33	0,57	1,13	1,23	0,23	1,2	0,5	-0,12	-0,11	0,27	0,63	-0,26	0,82	0,6	0,5
set/24	2	Habitação	1,82	1,51	1,44	1,78	2,06	1,47	2,54	1,53	2,33	2,17	2,07	3,5	1,88	2,38	2,12	1,62	1,8
set/24	3	Artigos de residência	-0,68	0,27	0,33	-0,71	0,33	-0,03	-0,52	0,46	-0,72	0,25	0,33	0,38	-0,05	0,52	1,38	-0,92	-0,19
set/24	4	Vestuário	0,95	0,17	0,21	0,13	-0,24	0,16	0,22	0,76	0,12	0,51	-0,62	0,08	0,12	-0,1	0,24	0,21	0,18
set/24	5	Transportes	-0,01	0,05	-0,45	-0,61	0,03	0,94	-1,11	2,21	0,04	0,19	-0,97	0,51	0,02	-0,23	-0,61	0,15	0,14
set/24	6	Saúde e cuidados pessoais	0,24	0,18	0,78	0,44	0,3	0,47	0,84	0,47	0,33	0,4	0,87	0,73	0,49	0,36	0,92	0,44	0,46

set/24	7	Despesas pessoais	-0,02	0,45	0,02	0,23	-0,25	-0,05	-0,67	-0,19	0,05	-0,2	-0,02	0,03	-0,12	-0,35	-0,02	-0,81	-0,31
set/24	8	Educação	0,03	0,07	0,14	0,04	0,15	0,05	-0,03	0,08	0,02	0,02	0,11	0,18	0,02	0,04	0,08	0,05	0,05
set/24	9	Comunicação	0,01	0,03	-0,16	-0,17	-0,22	0,18	-0,23	-0,1	-0,02	0,02	-0,3	-0,04	-0,22	0,09	-0,01	-0,02	-0,05
set/24	10	Índice geral	0,07	0,51	0,18	0,26	0,58	0,77	0,3	1,08	0,49	0,39	0,17	0,75	0,53	0,28	0,6	0,36	0,44
out/24	1	Alimentação e bebidas	0,5	1,28	1,06	1,04	2,36	0,97	0,45	1,68	0,52	0,16	0,77	1,56	1,17	0,49	0,93	1,37	1,06
out/24	2	Habitação	1,47	1,36	1,97	1,45	0,81	1,2	1,49	2,59	1,44	1,26	2,28	1,1	1,24	1,27	2,75	1,49	1,49
out/24	3	Artigos de residência	0,56	0,23	1,34	0,38	-0,75	0,37	0,75	-0,17	-0,38	0,36	-0,29	0,16	-0,12	0,81	-0,57	0,82	0,43
out/24	4	Vestuário	0,72	0,61	1,4	0,62	0,77	0,81	0,65	1,2	-0,37	-0,21	0,92	0,75	0,37	0,91	0,25	-0,26	0,37
out/24	5	Transportes	-1,68	-0,65	-0,27	0,6	-0,16	-0,41	-0,43	-0,41	0,49	-1,11	-0,81	0,04	-0,27	-0,24	-0,81	-0,24	-0,38
out/24	6	Saúde e cuidados pessoais	0,12	0,38	0,14	0,32	0,04	0,17	0,37	0,49	0,15	0,62	0,17	-0,8	0,39	0,17	0,62	0,5	0,38
out/24	7	Despesas pessoais	0,47	0,54	0,76	0,65	0,73	0,52	0,71	0,66	0,63	0,63	0,79	0,7	0,8	1,11	0,17	0,74	0,7
out/24	8	Educação	0,01	0,13	0,18	-0,17	0	0,06	0,03	0,03	-0,01	0,08	0,01	0,25	0,05	0,05	-0,11	0,04	0,04
out/24	9	Comunicação	-0,01	0,23	0,41	0,41	0,43	0,56	0,57	0,38	0,43	0,42	0,43	0,43	0,7	0,29	0,08	0,69	0,52
out/24	10	Índice geral	0,11	0,5	0,78	0,68	0,7	0,42	0,46	0,8	0,5	0,16	0,5	0,55	0,6	0,48	0,57	0,67	0,56
nov/24	1	Alimentação e bebidas	1,41	1,93	2,51	1,2	2,63	1,23	1,35	2,9	1,25	0,71	1,04	2,71	1,68	1,5	2,35	1,41	1,55
nov/24	2	Habitação	-1,55	-1,44	-2,09	-2,05	-0,78	-1,15	-1,72	-0,73	-2,23	-2,02	-2,03	-0,61	-1,44	-1,38	-2,07	-1,47	-1,53
nov/24	3	Artigos de residência	-0,97	-0,3	-1,71	-1,04	0	0,08	0,21	-1,14	-0,87	-0,18	-0,5	-0,2	-0,87	-0,54	0,46	0,06	-0,31
nov/24	4	Vestuário	0,06	0,78	-0,62	-0,09	0,06	0,44	-0,03	-0,38	0,02	-0,38	-0,06	-1,27	-0,34	0,6	-0,89	-0,46	-0,12
nov/24	5	Transportes	0,42	0,66	1,39	1,37	0,36	0,82	1,3	-1,06	0,68	0,13	2,48	2	1,45	0,12	0,19	1,19	0,89
nov/24	6	Saúde e cuidados pessoais	0,08	0,23	-0,35	0,18	0,04	-0,17	0,39	0,15	0,42	0,11	-0,26	-0,51	0,03	0,17	-0,26	-0,31	-0,06
nov/24	7	Despesas pessoais	0,47	1,46	0,45	0,73	0,96	0,8	1,28	1,94	1,18	1,63	0,67	0,98	1,73	0,89	0,59	1,8	1,43
nov/24	8	Educação	0,01	-0,01	0,18	-0,03	-0,02	0,13	-0,02	-0,04	-0,05	-0,05	0,03	-0,11	0,03	-0,05	0,07	-0,13	-0,04
nov/24	9	Comunicação	0,54	-0,13	-0,5	-0,17	-0,26	-0,13	0,1	-0,1	0,01	-0,17	0,05	-0,26	0,06	0,13	0,5	-0,17	-0,1
nov/24	10	Índice geral	0,24	0,57	0,46	0,3	0,63	0,39	0,44	0,41	0,16	0,03	0,42	0,92	0,49	0,28	0,33	0,4	0,39
dez/24	1	Alimentação e bebidas	1,02	0,81	1,77	1,18	1,36	0,56	1,17	1,31	1,63	0,62	1,18	1,79	1,26	1,46	1,37	1,37	1,18
dez/24	2	Habitação	-0,63	-0,66	-0,58	-0,95	-0,52	0,06	-0,4	-1,37	-0,32	-0,77	-0,9	-1,79	0	-0,04	-0,6	-0,74	-0,56
dez/24	3	Artigos de residência	1,02	0,21	0,52	-0,25	0,86	0,76	0,96	0,57	0,97	0,88	0,84	0,89	0,35	0,45	-0,1	0,89	0,65
dez/24	4	Vestuário	0,61	0,93	1,18	1,2	1,13	1,02	0,79	0,92	1,13	1	1,4	1,33	1,39	0,71	0,43	1,43	1,14
dez/24	5	Transportes	1,99	-0,23	0,49	0,11	0,38	0,15	1,78	2,27	0,4	1,05	-0,12	0,47	0,5	2,44	2,11	0,52	0,67

dez/24	6	Saúde e cuidados pessoais	0,24	0,43	-0,11	0,48	-0,18	1,05	-0,31	0,25	0,03	0,38	0,13	-0,02	0,38	0,2	-0,31	0,53	0,38
dez/24	7	Despesas pessoais	0,59	0,6	0,73	0,12	0,45	0,78	0,35	0,57	0,75	0,83	0,88	0,82	0,66	0,32	0,27	0,66	0,62
dez/24	8	Educação	0,25	0,05	0	0,03	0,06	0,06	0,23	0,07	0,23	0,03	0,06	0,51	0,09	0,17	0,25	0,15	0,11
dez/24	9	Comunicação	-0,42	0,44	0,42	0,6	0,19	0,55	0,61	0,09	0,32	0,64	-0,1	0,23	0,68	-0,28	0,84	0,26	0,37
dez/24	10	Índice geral	0,67	0,25	0,63	0,26	0,43	0,46	0,65	0,8	0,52	0,5	0,34	0,53	0,58	0,89	0,71	0,52	0,52
jan/25	1	Alimentação e bebidas	1,89	1,15	1,64	1,08	0,1	0,68	1,25	0,72	1,64	1	0,57	0,7	0,8	1,81	1,22	0,74	0,96
jan/25	2	Habitação	-3,4	-2,12	-4,28	-2,83	-2,6	-3,6	-2,94	-4,57	-4,47	-3,45	-4,24	-8,03	-3,13	-3,41	-5,71	-2,42	-3,08
jan/25	3	Artigos de residência	0,72	0,09	0,28	0,73	0,11	-0,13	0,09	0,34	0,63	0,56	-0,23	0,99	0,07	-0,06	0,67	-0,74	-0,09
jan/25	4	Vestuário	-0,03	-0,06	0,18	-0,97	0,4	0,8	-0,36	0,12	0,2	0,02	0,24	0,16	-0,36	-0,4	0,15	-0,51	-0,14
jan/25	5	Transportes	1,66	1,45	0,77	2,88	0,82	0,6	0,9	1,46	2,08	0,19	2,17	1,19	1,58	1,22	1,08	1,44	1,3
jan/25	6	Saúde e cuidados pessoais	0,75	0,58	1,65	0,54	1,03	0,75	0,05	0,38	0,84	0,83	0,25	0,94	0,46	0,64	1,23	0,81	0,7
jan/25	7	Despesas pessoais	1,24	0,44	0,56	0,08	0,63	0,14	1,24	-0,06	0,82	0,56	0,94	0,58	1,32	1,35	-0,06	0,28	0,51
jan/25	8	Educação	0,23	0,53	0,03	0,16	0,26	0,21	0,41	0,02	0,17	0,22	0,24	0,02	0,15	0,23	0,36	0,28	0,26
jan/25	9	Comunicação	0,26	-0,1	0,25	0,02	0,04	-0,34	-0,02	0,05	-0,01	0	0,28	-0,1	-0,46	-0,05	-0,41	-0,3	-0,17
jan/25	10	Índice geral	0,59	0,43	0,22	0,56	0,04	-0,09	0,11	-0,03	0,35	-0,03	0,12	-0,34	0,06	0,38	-0,08	0,15	0,16

- Dimensão de Categorias, criada manualmente (10 registros):

Cod Categoria	Categoria
1	Alimentação e bebidas
2	Habitação
3	Artigos de residência
4	Vestuário
5	Transportes
6	Saúde e cuidados pessoais
7	Despesas pessoais
8	Educação
9	Comunicação
10	Índice geral

- Dimensão de Cidades, criada manualmente (17 registros):

Cod Cidade	Cidade	UF	País	Latitude	Longitude
AJU	Aracaju	SE	Brazil	-10,57	-37,45
BH	Belo Horizonte	MG	Brazil	-18,1	-44,38
BEL	Belém	PA	Brazil	-3,79	-52,48
DF	Brasília	DF	Brazil	-15,83	-47,86
CG	Campo Grande	MS	Brazil	-12,64	-55,42
CUR	Curitiba	PR	Brazil	-24,89	-51,55
FOR	Fortaleza	CE	Brazil	-5,2	-39,53
GOI	Goiânia	GO	Brazil	-15,98	-49,86
VIT	Vitória	ES	Brazil	-19,19	-40,34
POA	Porto Alegre	RS	Brazil	-30,17	-53,5
REC	Recife	PE	Brazil	-8,38	-37,86
RB	Rio Branco	AC	Brazil	-8,77	-70,55
RJ	Rio de Janeiro	RJ	Brazil	-22,25	-42,66
SAL	Salvador	BA	Brazil	-13,29	-41,71
SL	São Luís	MA	Brazil	-5,42	-45,44
SP	São Paulo	SP	Brazil	-22,19	-48,79
NAC	Nacional	BR	Brazil	-15,83	-47,86

2 Contexto e Público-alvo

O conceito do trabalho foi desenvolver um dashboard que demonstre as taxas de inflação mensais medidas pelo IPCA sob uma ótica mais ampla, não só demonstrando puramente os seus valores mensais, mas também a composição regional e categórica do índice, para enriquecer e aprofundar a análise dos dados. O detalhamento categórico, se limitou ao nível das categorias que compõem o índice, não chegando até as subcategorias. Ainda que isso acabe penalizando o nível de detalhamento dos dados, optou-se por essa abordagem para reduzir um pouco o grau de complexidade na análise, visando a melhor utilização pelo público-alvo prioritário.

A persona prioritária imaginada como público-alvo do relatório são as pessoas que desejam estudar o comportamento da inflação, mas que ainda não tinham tido a oportunidade de visualizar a composição mais detalhada dos seus dados, o que acaba por empobrecer sua capacidade de análise crítica com relação aos valores observados.

Além disso, um público-alvo adicional seriam pessoas já familiarizadas com a composição do índice, mas que desejam uma ferramenta mais acessível e rápida para visualizar e comparar os dados com um nível de detalhamento um pouco maior.

3 Storytelling Desenvolvido

O storytelling desenvolvido no relatório visa levar o usuário a entender o que é o IPCA, o que ele representa e como seus dados são coletados e posteriormente compilados. Além disso, procura demonstrar que há significativas diferenças regionais nas amostras dos dados coletados, bem como discrepâncias importantes nas variações apresentadas para cada uma das categorias de produtos, bens e serviços medidos.

A sequência de visualizações diferentes indo do nível menos detalhado até os níveis mais detalhados da composição do índice, pretendem gerar o senso crítico no usuário de que é muito superficial analisar somente o valor final do índice que é divulgado.

4 Escolha das visualizações e da ferramenta

A ferramenta selecionada para o desenvolvimento das visualizações foi o Microsoft Power BI, pela sua facilidade de utilização, produtividade na produção dos relatórios e disponibilidade de recursos com relação a navegação entre as telas desenvolvidas. Além disso, o mesmo possui uma versão desktop gratuita, que possibilita a visualização do relatório desenvolvido, bastando apenas que o usuário tenha a ferramenta instalado em seu computador e esteja de posse do arquivo extensão PBIX desenvolvido.

Como os dados se apresentam basicamente como uma série temporal, foram utilizados prioritariamente gráficos de linha para demonstrar as variações dos valores do índice no período de visualização selecionado. Foram disponibilizados filtros de data para que o usuário possa eventualmente selecionar um intervalo de meses específicos para a sua análise.

Além disso, nas visualizações demonstrativas das composições regionais e categóricas, foram utilizadas visualizações que possibilitam reforçar graficamente as comparações entre os valores do índice geral e os valores que que o compõe, tais como um gráfico de linhas comparativo e um gráfico de mapa demonstrando as regiões onde foram coletados os dados com bolhas dimensionadas de acordo com os valores levantados em cada região.

5 Descrição da narrativa

O relatório inicia com uma página que explica o que é o IPCA e porque é importante analisar os dados da sua composição e não somente o índice geral do período.

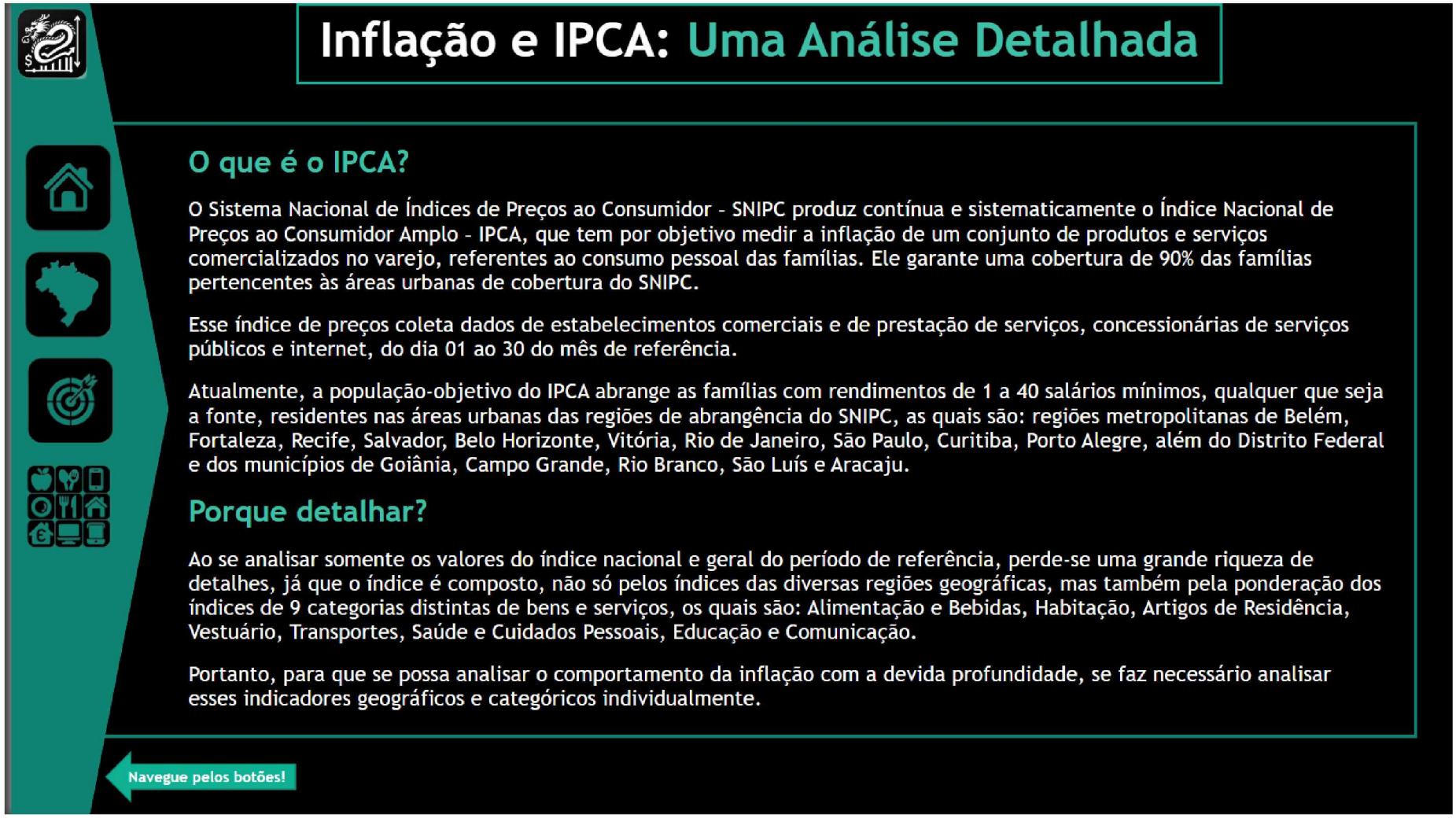
Navegando para a próxima tela, é demonstrado somente o índice geral ao longo do tempo, bem como o acumulado geral da inflação do período selecionado/visualizado.

Na tela seguinte, são demonstrados os índices de cada região pesquisada ao longo do tempo, sobrepostos com o índice geral, para demonstrar graficamente as diferenças de valores e comportamento para cada região, bem como um mapa que permite visualizar as diferenças de valor de todas as regiões ao mesmo tempo.

Na última tela então, são demonstrados e comparados os valores dos índices de cada categoria diferente de produtos, bens e serviços analisados, e o valor do índice geral do período, para que o usuário possa perceber como esse valor geral é na verdade composto pelo valor das diversas categorias.

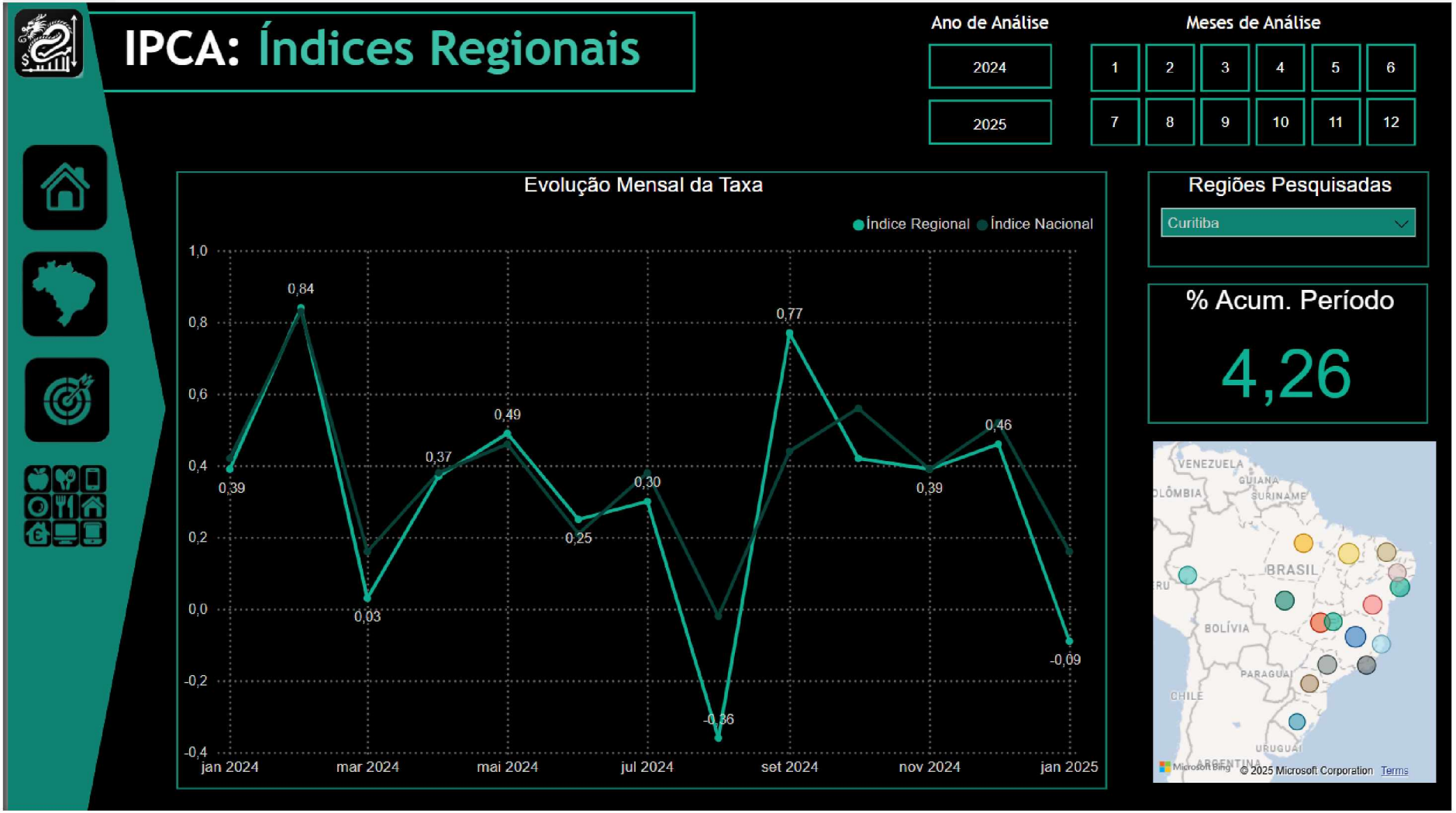
6 Telas do relatório

- Tela Inicial:

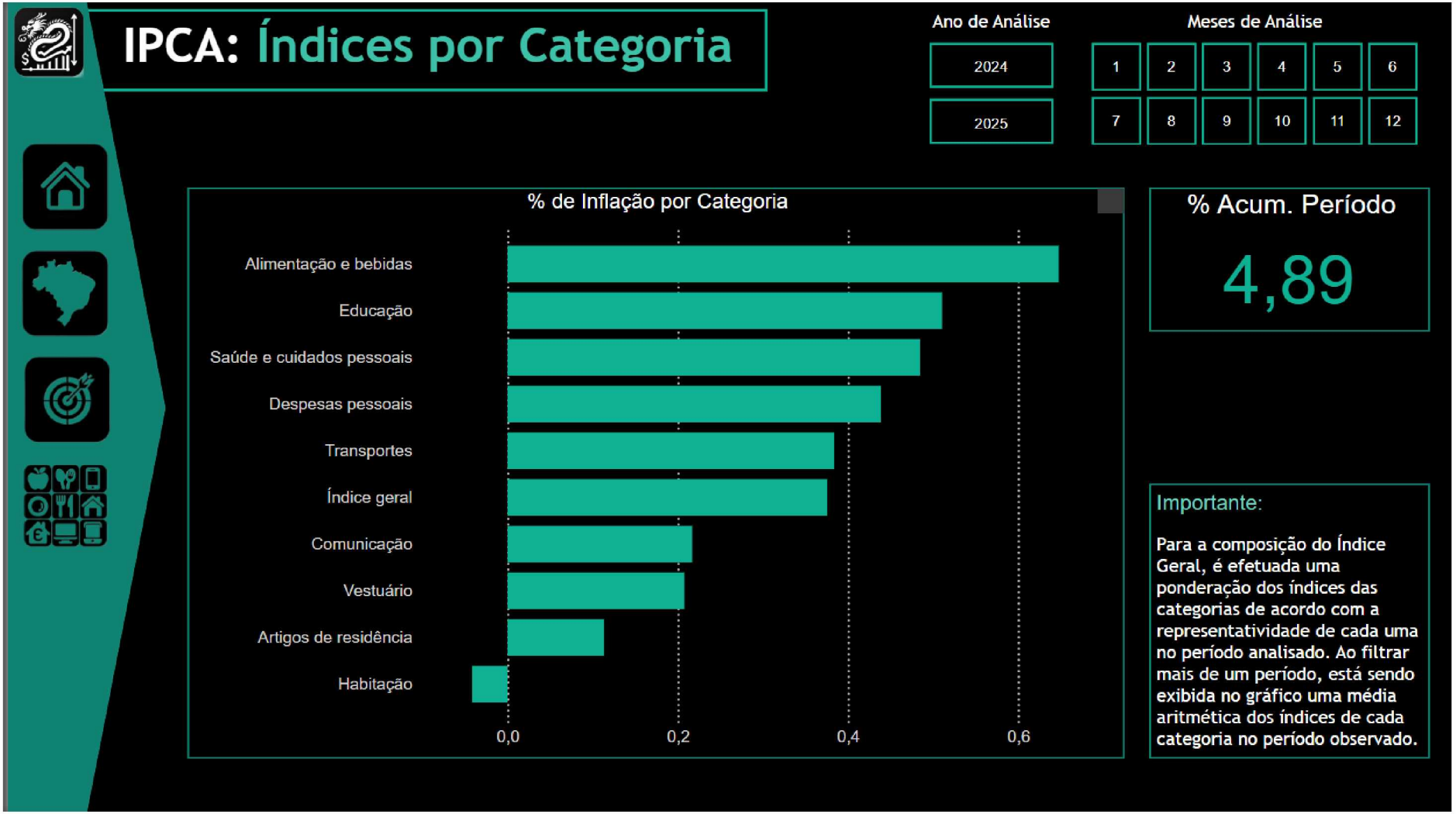


- Tela do Índice Geral:

- Tela dos Índices Regionais x Geral:



- Tela dos Índices por Categoria:



APÊNDICE O - TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

2) Compare a representação de dois modelos vetoriais

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

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

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

B – RESOLUÇÃO

1) Algoritmo Genético

CÓDIGO

```
import matplotlib.pyplot as plt import numpy as np
import random as rd
#from collections import deque as dq

#######################################################
# Funções auxiliares
#######################################################
def plotaGrafico(individuo : list, coordenadas : list, numGeracao : int, \
                 numCidades : int, distancia : float):
    x_caminho = []
    y_caminho = []
    for c in individuo:
        x, y = coordenadas[c]
        x_caminho.append(x)
        y_caminho.append(y)

    fig, ax = plt.subplots()
    ax.plot(x_caminho, y_caminho,'--go', mfc='r', mec='r', \
            label='Melhor Rota', linewidth=2)
    plt.legend()
    plt.title(label='Caixeiro Viajante: Melhor Rota usando GA', \
              fontsize=12,color='k')
    txtParams = 'Diatância Total: '+str(round(distancia,3)) + '\n' + \
                'Núm.Gerações: ' + str(numGeracao) + '\n' + \
                'Qtde.Cidades: '+ str(numCidades)
    plt.suptitle(txtParams, fontsize=10, y=1)
    for i in individuo:
        ax.annotate(str(i) if i!=100 else 0, (coordenadas[i][0], \
            coordenadas[i][1]), fontweight='bold' if i==0 else 'normal', \
            fontsize=10 if i==0 else 8, color='#000000' if i==0 else '#999999')
```

```
    fig.set_size_inches(16,10)
    plt.grid(color='#888888', linestyle='dotted')
    plt.savefig('solucao_ger'+str(numGeracao)+'.png')
    plt.show()

def criaCoordenadas(qtdeCidades : int, xMax : int, yMax : int) -> list:
    """ Cria as coordenadas das cidades num plano cartesiano de tamanho xMax x
yMax de forma aleatória """
    ret = []
    while len(ret) < qtdeCidades:
        # Gera as coordenadas da cidade
        cidade = (rd.randint(1, xMax), rd.randint(1, yMax))
        # Evita incluir cidades que já existam
        while cidade in ret:
            cidade = (rd.randint(0, xMax), rd.randint(0, yMax))
        ret.append(cidade)
    # Inclui a cidade de origem na última posição da lista
    ret.append(ret[0])
    return ret

def distanciaEuclidiana(pontoA : tuple, pontoB : tuple) -> float:
    """ Calcula a distância euclidiana entre os dois pontos """
    return ( np.sqrt(np.sum((np.array(pontoA) - np.array(pontoB))**2)) )

def distanciaTotal(individuo : list, coordenadas: list) -> float:
    """ Calcula a distância total dos pontos de um determinado individuo """
    coordInd = [ coordenadas[c] for c in individuo ]
    return sum([ distanciaEuclidiana(c1, c2)
                 for c1, c2 in list(zip(coordInd[:-1],coordInd[1:])) ])

######################################################
# Passo 1 do AG -> Cria a população Inicial
######################################################
def criaPopulacaoInicial(qtdeIndividuos : int, qtdeCidades : int) -> list:
    """
    Cria a população inicial de cidades de forma randomica
    qtdeIndividuos -- A quantidade de indivíduos na população
    qtdeCidades -- A quantidade de cidades
    xMax -- O tamanho máximo de pixels do eixo X do plano cartesiano
    yMax -- O tamanho máximo de pixels do eixo Y do plano cartesiano
    """
    from math import factorial
    ret = []
    # Inicializa o primeiro indivíduo
    individuo = list(range(1, qtdeCidades))
    ret.append([0]+individuo+[qtdeCidades])
    # Cria os demais indivíduos
    # Se necessário, ajusta qtdeIndividuos para o máximo de permutações possíveis
    if qtdeIndividuos > factorial(qtdeCidades-1):
        qtdeIndividuos = factorial(qtdeCidades-1)
        print(f'Quantidade de indivíduos maior que a permnutação possível!
População Inicial ajustada para {qtdeIndividuos} indivíduos')

    while len(ret) < qtdeIndividuos:
        rd.shuffle(individuo)
        while (individuo in ret): # previne permutações repetidas
            rd.shuffle(individuo)
        ret.append([0]+individuo+[qtdeCidades])
```

```
    return ret

#####################################################
# Passo 2 do AG -> Avalia as soluções
#####################################################
def avaliaSolucao(populacao : list, coordenadas : list, \
                  xMax : int, yMax : int) -> list:
    """Avalia a população e retorna os valores da avaliação de cada individuo como
uma lista"""
    ret=[]
    # Retorna a maior diatancia euclidiana existente no plano (a diagonal)
    maiorDistancia = distanciaEuclidiana((1, 1), (xMax, yMax))
    # Efetua um score para que as menores distâncias tenhas as maiores avaliações
    # Calcula a distancia dos pontos como proporção da maior distância
    for individuo in populacao:
        coordInd = [coordenadas[c] for c in individuo]
        ret.append( sum([maiorDistancia/distanciaEuclidiana(c1, c2) \
                   for c1, c2  in list(zip(coordInd[:-1],coordInd[1:]))]) )

    return ret

#####################################################
# Passo 3 do AG -> Seleciona/Descarta Indivíduos
#####################################################
def ranqueiaPopulacao(populacao : list, coordenadas : list, \
                      xMax : int, yMax : int) -> list:
    """Ranqueia a população do melhor para o pior individuo"""
    return np.argsort(avaliaSolucao(populacao, coordenadas, xMax , yMax))[::-1]

def selecionaMelhores(nMelhores : int, populacao : list, \
                      coordenadas : list, xMax : int, yMax : int) -> list:
    """Seleciona os n melhores indivíduos da geração e retorna"""
    ret = [ populacao[i] \
            for i in np.argsort(avaliaSolucao(populacao, coordenadas, \
                                              xMax , yMax))[::-1]][:nMelhores]
    return ret

#####################################################
# Passo 4 do AG -> Aplicar Operadores de Reprodução
#####################################################
def cruzamento(populacaoOrigem : list, populacaoDestino : list, \
               qtde : int = 1) -> list:
    """
    Efetua o cruzamento da população e retorna a quantidade desejada de indivíduos
em forma de lista.
    Impede a geração de novos indivíduos que já existam na população de Destino.
    """
    ret = []
    while len(ret) < qtde:
        # Seleciona aleatoriamente o primeiro indivíduo
        posicaoA = rd.randrange(0, len(populacaoOrigem))
        #Garante que as posições selecionadas nunca seja iguais
        posicaoB = posicaoA
        while posicaoA == posicaoB:
            posicaoB = rd.randrange(0, len(populacaoOrigem))
        individuoA = populacaoOrigem[posicaoA]
        individuoB = populacaoOrigem[posicaoB]
        # Elimina o primeiro e o último elemento porque é a cidade de origem
```

```
        indA = populacaoOrigem[posicaoA][1:len(populacaoOrigem[posicaoA])-1]
        indB = populacaoOrigem[posicaoB][1:len(populacaoOrigem[posicaoB])-1]
        # Define em que posição do vetor vai efetuar o corte para cruzar os genes
        corte = rd.randrange( round(len(indA)/2), len(indA) )
        # Efetua o cruzamento OX
        novoIndA = indA[:corte]
        for gene in indB[corte:]:
            if gene in novoIndA:
                novoIndA.append(list(set(indB[:corte]) - set(novoIndA))[0])
            else:
                novoIndA.append(gene)

        novoIndB = indB[:corte]
        for gene in indA[corte:]:
            if gene in novoIndB:
                novoIndB.append(list(set(indA[:corte]) - set(novoIndB))[0])
            else:
                novoIndB.append(gene)

        # Retorna os indivíduos
        novoIndividuoA = [individuoA[0]] + novoIndA + \
                         [individuoA[len(individuoA)-1]]
        novoIndividuoB = [individuoB[0]] + novoIndB + \
                         [individuoB[len(individuoB)-1]]
        if novoIndividuoA not in populacaoDestino: # valida se o novo individuo já
existe
            ret.append(novoIndividuoA)
        # Se não atingiu a qtde de itens desejada e o novo individuo não existe
ainda, adiciona o segundo indivíduo
        if (len(ret) < qtde) and (novoIndividuoB not in populacaoDestino):
            ret.append(novoIndividuoB)

    return ret
#####################################################
# Passo 5 do AG -> CriaAplicar Operadores de Mutação
#####################################################
def mutacao(populacaoOrigem : list, populacaoDestino : list, \
            qtde : int = 1) -> list:
    """
    Efetua a mutação da população de origem e retorna a quantidade desejada de
indivíduos,
    que não existem na população de destino, em forma de lista
    """
    ret = []
    while len(ret) < qtde:
        #Seleciona aleatoriamente o primeiro indivíduo
        posicao = rd.randrange(0, len(populacaoOrigem))
        individuo = populacaoOrigem[posicao]
        # Elimina o primeiro e o último elemento porque é a cidade de origem
        # Define os genes que vão gerar a mutação
        gene1 = rd.randrange( 1, round(len(individuo)/2) )
        gene2 = rd.randrange( round(len(individuo)/2)+1, len(individuo)-1 )
        # Gera a mutação
        if gene2 != len(individuo)-1:
            individuo = individuo[:gene1] + [individuo[gene2]] + \
                        individuo[gene1+1:gene2] + [individuo[gene1]] + \
                        individuo[gene2+1:]
        else:
```

```
            individuo = individuo[:gene1] + [individuo[gene2]] + \
                        individuo[gene1+1:gene2] + [individuo[gene1]]
        # Testa se o indivíduo não existe na população de destino
        if individuo not in populacaoDestino: # valida se o novo individuo já
existe
            ret.append(individuo)

    return ret

######################################################
# Executa o Algoritmo Genético chamando os passos
######################################################
# Define os parâmetros globais para a execução do AG
qtdeIndividuos = 300
qtdeCidades = 100
xTamMax = 100
yTamMax = 100
percMuta = 0.02
percCruza = 0.9
totalGeracoes = 4000

# Passo 1
populacaoInicial = criaPopulacaoInicial(qtdeIndividuos, qtdeCidades)
coordenadas = criaCoordenadas(qtdeCidades, xTamMax, yTamMax)
# print("População inicial: ",end='')
# print(populacaoInicial)
print("Coordenadas das Cidades: ",end='')
print(coordenadas)

# Iteração entre os passos 3, 4 e 5 do AG
qtdeCruza = round(percCruza*qtdeIndividuos)
qtdeMuta = round(percMuta*qtdeIndividuos)
qtdeMelhores = qtdeIndividuos - qtdeCruza - qtdeMuta

numGeracao = 1
# Pega o melhor resultado da população inicial


melhorIndividuo = selecionaMelhores(len(populacaoInicial), \
                  populacaoInicial, coordenadas, xTamMax, yTamMax)[0]
menorDistancia = distanciaTotal(melhorIndividuo, coordenadas)
# Exibe a melhor solução encontrada (melhor valor da população inicial)
print( f'Geracao {numGeracao}: {menorDistancia}\nMelhor solucao:
{melhorIndividuo}')
plotaGrafico(melhorIndividuo, coordenadas, numGeracao, \
             qtdeCidades, menorDistancia)
geracaoAtual = populacaoInicial

while numGeracao <= totalGeracoes-1:
    proximaGeracao = []
    # Seleciona a qtde dos melhores indivíduos para a próxima geração
    proximaGeracao.extend( selecionaMelhores(qtdeMelhores, geracaoAtual, \
                                             coordenadas, xTamMax, yTamMax) )
    # Efetua o cruzamento da geração atual, gerando X% de indivíduos novos para a
próxima geração
    proximaGeracao.extend( cruzamento(geracaoAtual, proximaGeracao, qtdeCruza) )
    # Efetua X% de mutação na próxima Geração
    proximaGeracao.extend( mutacao(geracaoAtual, proximaGeracao, qtdeMuta) )
    # Ordena a próxima geracao com base na avaliacao e atribui com a geração atual
```

```
    geracaoAtual = selecionaMelhores(len(proximaGeracao), proximaGeracao, \
                                     coordenadas, xTamMax, yTamMax)
    numGeracao += 1
    print('.',end='')
    # Exibe os dados da melhor solução a cada 50 gerções
    if numGeracao%50 == 0:
        melhorIndividuo = geracaoAtual[0]
        menorDistancia = distanciaTotal(melhorIndividuo, coordenadas)
        print( f'\nGeracao {numGeracao}: {menorDistancia}')

print( f'Geracao {numGeracao}: {menorDistancia}')

# Exibe a melhor solução encontrada (melhor valor da última geração)
print(f'Melhor solucao: {melhorIndividuo} com {menorDistancia} unidades')
plotaGrafico(melhorIndividuo, coordenadas, numGeracao, qtdeCidades,
menorDistancia)
```

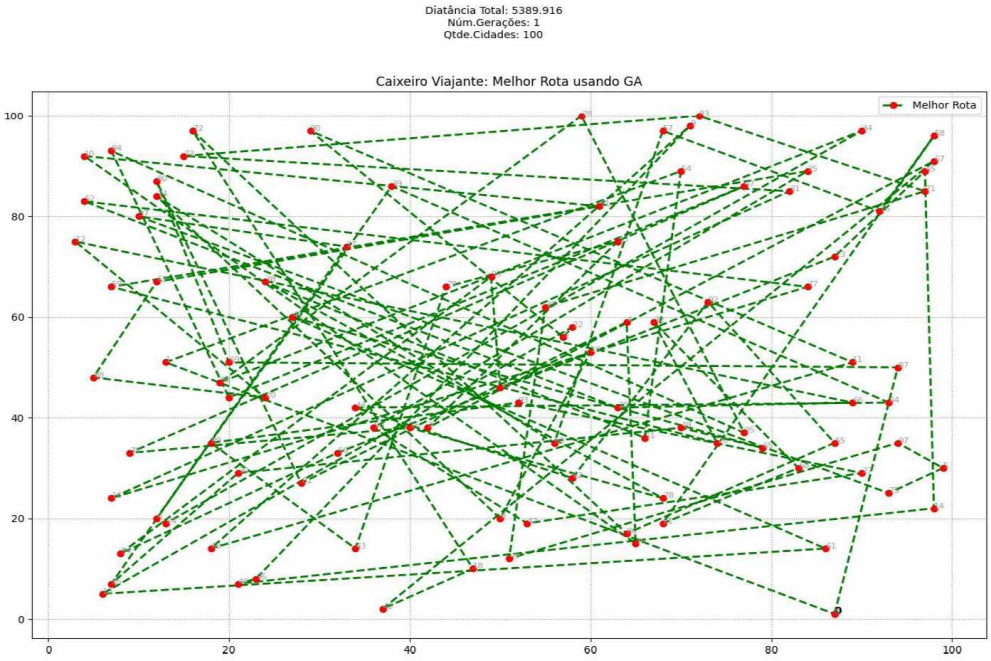


Imagem 1: primeira melhor solução (geração 1).

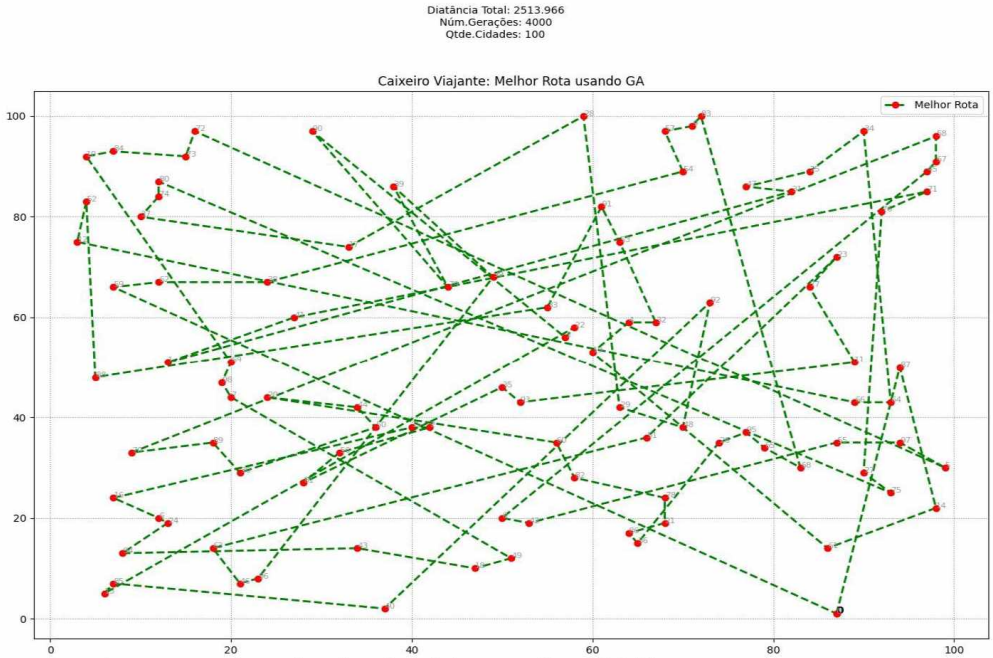


Imagem 2: solução final (geração 4000).

Log de Resultados:

```
Coordenadas das Cidades: [(87, 1), (13, 51), (42, 38), (57, 56), (64, 59), (99, 30), (12, 20), (20,
44), (50, 20), (71, 98), (4, 92), (89, 51), (3, 75), (6, 5), (98, 22), (40, 38), (7, 24), (33, 74),
(47, 10), (60, 53), (24, 44), (68, 19), (58, 58), (87, 72), (13, 19), (84, 89), (74, 35), (90, 29),
(59, 100), (63, 42), (21, 29), (82, 85), (67, 59), (55, 62), (90, 97), (50, 46), (65, 15), (10, 80),
(24, 67), (38, 86), (37, 2), (27, 60), (53, 19), (34, 14), (34, 42), (97, 89), (21, 7), (77, 86),
(70, 38), (51, 12), (36, 38), (66, 36), (4, 83), (63, 75), (70, 89), (87, 35), (32, 33), (68, 97),
(98, 96), (7, 66), (56, 35), (86, 14), (12, 67), (18, 14), (93, 43), (79, 34), (89, 43), (98, 91),
(83, 30), (20, 51), (9, 33), (97, 85), (16, 97), (15, 92), (12, 84), (93, 25), (92, 81), (84, 66),
(68, 24), (44, 66), (12, 87), (28, 27), (58, 28), (72, 100), (7, 93), (7, 7), (64, 17), (94, 50), (5,
48), (18, 35), (29, 97), (61, 82), (73, 63), (52, 43), (8, 13), (77, 37), (23, 8), (94, 35), (19, 47),
(49, 68), (87, 1)]

Geracao 1: 5389.91580805757

Melhor solucao: [0, 87, 69, 80, 81, 4, 36, 10, 91, 62, 88, 20, 12, 66, 29, 52, 77, 16, 31, 73, 83, 71,
33, 49, 97, 5, 75, 37, 17, 85, 30, 48, 65, 41, 61, 13, 22, 3, 90, 11, 63, 9, 96, 46, 14, 45, 23, 56,
94, 99, 35, 19, 8, 74, 60, 57, 76, 58, 21, 68, 32, 26, 59, 25, 50, 42, 27, 93, 89, 43, 79, 53, 2,
67, 40, 18, 72, 86, 55, 92, 82, 44, 64, 39, 6, 24, 34, 98, 28, 95, 84, 7, 47, 70, 15, 78, 38, 51, 54,
1, 100]

Geracao 50: 4822.430448503724
.................................................
Geracao 100: 4497.947104483611
.................................................
Geracao 3900: 2517.279360502755
.................................................
Geracao 3950: 2513.966382801496
.................................................
Geracao 4000: 2513.966382801496
Geracao 4000: 2513.966382801496
Melhor solucao: [0, 87, 14, 61, 19, 4, 32, 53, 91, 33, 88, 52, 12, 66, 64, 34, 25, 47, 31, 1, 41, 71,
76, 27, 75, 80, 74, 37, 17, 28, 29, 48, 92, 40, 85, 13, 22, 3, 90, 79, 39, 99, 96, 46, 63, 51, 23,
77, 11, 93, 35, 81, 56, 2, 15, 16, 6, 24, 94, 43, 18, 49, 7, 98, 69, 10, 84, 73, 72, 5, 97, 55, 42,
8, 45, 67, 58, 70, 89, 30, 50, 44, 20, 60, 82, 78, 21, 86, 36, 26, 95, 65, 68, 83, 9, 57, 54,
38, 62, 59, 100] com 2513.966382801496 unidades
```

2) Compare a representação de dois modelos vetoriais

Sentenças:

1. O gato preto pulou o muro alto.
2. O cachorro marrom correu no parque grande.
3. O gato preto saltou sobre o muro baixo.
4. As flores coloridas desabrocharam no jardim ensolarado.
5. O cachorro marrom brincou no gramado amplo.
6. As crianças felizes riram na festa animada.

Vetores simplificados:

Vetor 1: [gato, preto, pulou, muro, alto]

Vetor 2: [cachorro, marrom, correu, parque, grande]

Vetor 3: [gato, preto, saltou, muro, baixo]

Vetor 4: [flores, coloridas, desabrocharam, jardim, ensolarado]

Vetor 5: [cachorro, marrom, brincou, gramado, amplo]

Vetor 6: [crianças, felizes, riram, festa, animada]

Vetorização das Frases Dado o exemplo:

- Vocabulário Total: gato, preto, pulou, muro, alto, cachorro, marrom, correu, parque, grande, saltou, baixo, flores, coloridas, desabrocharam, jardim, ensolarado, brincou, gramado, amplo, crianças, felizes, riram, festa, animada

Aqui, cada vetor será uma lista binária indicando a presença (1) ou ausência (0) das palavras na sentença.

Aplicação do PCA

Após transformar as frases em vetores binários, aplicaremos o PCA para reduzir suas dimensões e permitir a projeção gráfica.

CÓDIGO

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

```
# Sentenças
sentences = [
    "O gato preto pulou o muro alto.",
    "O cachorro marrom correu no parque grande.",
    "O gato preto saltou sobre o muro baixo.",
    "As flores coloridas desabrocharam no jardim ensolarado.",
    "O cachorro marrom brincou no gramado amplo.",
    "As crianças felizes riram na festa animada."
]

# Vetorização
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sentences).toarray()

# PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Plotagem
plt.figure(figsize=(8, 6))
plt.title('Projeção de Vetores usando PCA')
plt.scatter(X_pca[:, 0], X_pca[:, 1], c='r', marker='o')

for i, sent in enumerate(sentences):
    plt.annotate(f'S{i + 1}', (X_pca[i, 0], X_pca[i, 1]))

plt.xlabel('Componente Principal 1')
plt.ylabel('Componente Principal 2')
plt.grid(True)
plt.show()
```