UNIVERSIDADE FEDERAL DO PARANÁ

DARIO CALDAS MARTINS

MEMORIAL DE PROJETOS: A ÉTICA NA GERAÇÃO DE CONTEÚDO DA INTELIGÊNCIA ARTIFICIAL, LIMITES E RESPONSABILIDADES NO USO DO CHATGPT

CURITIBA

2025

DARIO CALDAS MARTINS

MEMORIAL DE PROJETOS: A ÉTICA NA GERAÇÃO DE CONTEÚDO DA INTELIGÊNCIA ARTIFICIAL, LIMITES E RESPONSABILIDADES NO USO DO CHATGPT

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

Orientador: Prof. Dr. Razer Anthom Nizer Rojas Montaño

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 **DARIO CALDAS MARTINS**, intitulada: **MEMORIAL DE PROJETOS: A ÉTICA NA GERAÇÃO DE CONTEÚDO DA INTELIGÊNCIA ARTIFICIAL, LIMITES E RESPONSABILIDADES NO USO DO CHATGPT**, 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, 12 de Novembro de 2025.

RAZER ANTHOM NIZER ROJAS MONTAÑO
Presidente da Banca Examinadora

JAIME WOJCIECHOWSKI
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

RESUMO

A Inteligência Artificial (IA), originada na década de 1950, evoluiu significativamente ao longo das décadas, transformando-se em uma tecnologia disruptiva com impacto profundo em diversos setores da sociedade. Inicialmente concebida para simular funções cognitivas humanas, a IA passou por quatro fases históricas: geração inicial, entusiasmo e expectativas, sistemas baseados em conhecimento e consolidação como indústria. Atualmente, destaca-se pela capacidade preditiva, análise de dados e personalização de experiências, com aplicações que vão da medicina à educação. O avanço recente da IA é exemplificado por ferramentas como o ChatGPT, que têm revolucionado a comunicação e o acesso à informação. No entanto, esse progresso também levanta questões éticas e jurídicas, especialmente no contexto brasileiro, onde ainda há lacunas regulatórias para lidar com os danos potenciais causados por sistemas inteligentes. Este trabalho tem como objetivo analisar a ética na geração de conteúdo por meio da IA, com foco no ChatGPT, propondo uma reflexão sobre o uso responsável e justo dessas tecnologias. A adoção de práticas éticas é essencial para garantir que os benefícios da IA sejam maximizados, enquanto seus riscos sejam mitigados, contribuindo para um desenvolvimento social sustentável e equitativo.

Palavras-chave: Inteligência artificial; ética; chatgpt; inovação tecnológica; geração de conteúdo.

ABSTRACT

Artificial Intelligence (AI), which originated in the 1950s, has evolved significantly over the decades, becoming a disruptive technology with a profound impact on various sectors of society. Initially designed to simulate human cognitive functions, AI has gone through four historical phases: initial generation, enthusiasm and expectations, knowledge-based systems, and consolidation as an industry. Today, it stands out for its predictive capabilities, data analysis, and personalized experiences, with applications ranging from medicine to education. The recent advancement of AI is exemplified by tools such as ChatGPT, which have revolutionized communication and access to information. However, this progress also raises ethical and legal issues, especially in the Brazilian context, where regulatory gaps still exist to address the potential harm caused by intelligent systems. This paper aims to analyze the ethics of content generation through AI, focusing on ChatGPT, and proposes a reflection on the responsible and fair use of these technologies. The adoption of ethical practices is essential to ensure that the benefits of AI are maximized while its risks are mitigated, contributing to sustainable and equitable social development.

Keywords: Artifical intelligence; ethics; chatgpt; technological innovation; content generation.

SUMÁRIO

1. PARECER TÉCNICO

1.1 INTRODUÇÃO

A inteligência artificial (IA) tem sua origem na década de 1950, se desenvolvendo em diversos ramos da ciência e áreas de pesquisa, tendo como objetivo “fornecer ao computador a habilidades para efetuar funções que apenas o cérebro humano é capaz de solucionar” (Gomes, 2010, p. 234). O termo criado por John McCarthy (1962) define IA como aquela ciência e engenharia capaz de produzir sistemas inteligentes.

Corroborando essa definição, Barbosa e Portes (2023) afirmam que a IA é um ramo da ciência da computação que busca construir mecanismos físicos ou digitais que simulem a capacidade humana e ajudem a tomar decisões.

Outrossim, a IA na atual sociedade tem desempenhado cada vez mais um papel significativo na composição das fronteiras do conhecimento e da tecnologia. O que era anteriormente realizado de forma manual e demorava para ser concluído, hoje leva minutos ou até segundos, isso a partir da implementação de máquinas e dos sistemas computadorizados.

A trajetória da evolução tecnológica encontra-se permeada pela capacidade preditiva, habilidade em analisar dados, identificação de padrões, além da capacidade de previsões com base nas análises (Maarouf, 2023) (Protázio; Amorim Filho; Bottentuit Junior, 2024). Nesse cenário de inovações tecnológicas, novos sistemas são desenvolvidos para a realização das atividades, a exemplo do deep learning, que é um método de aprendizado de máquina que se utiliza de uma rede neural baseada em algoritmos, combinando novos parâmetros de treinamento (Maarouf, 2023).

Recentemente, o avanço da IA tem sido marcado também pelos *chatbots*, tal como o ChatGPT, lançado em novembro de 2022 pela empresa OpenAI, que vem transformando diversos setores, desde a comunicação até a saúde, passando pela educação e o entretenimento (OpenAI, 2021).

Como pondera Maarouf (2023), o uso da inteligência artificial proporciona diversos benefícios, como a redução de custos operacionais, aumento da produtividade, entre outros. No entanto, também pode acarretar vários danos. Ainda, o ordenamento jurídico brasileiro não possui normativas de regramentos suficientes

para a demanda dos danos, que podem ser materiais ou imateriais, os quais consistem na violação dos direitos, direitos autorais e outros.

Pode-se aferir que essa transformação tecnológica traz consigo uma série de dilemas que precisam ser avaliados com cautela, para garantir que os benefícios sejam maximizados e os riscos minimizados.

Assim, o presente trabalho tem como objetivo analisar a ética na geração de conteúdo da inteligência artificial, em especial o ChatGPT, tendo como compreensão que, ao adotar práticas éticas e responsáveis em seu uso, pode-se promover um uso mais seguro, justo e benéfico da IA, contribuindo para o desenvolvimento sustentável e igualitário da sociedade.

1.2 INTELIGÊNCIA ARTIFICIAL

Gomes (2010) traz em seu artigo a história da inteligência artificial dividida em quatro fases, isso a partir da noção construída por Russel e Norvig (2004), a saber: 1) Geração da inteligência artificial (1943-1955); 2) Entusiasmo inicial e grandes expectativas (1952-1969); 3) Sistemas baseados em conhecimento (1966-1979); e 4) Inteligência artificial se torna uma indústria (1980 até os dias atuais). A *geração da* inteligência artificial *(1943-1955),* foi marcada por grandes trabalhos com ia destacando-se o trabalho de Warrem Macculloch e Walter Pitts, no ano de 1943 (Gomes, 2010). Conforme Russell e Norvig (2004), a pesquisa realizada por Macculloch e Pitts encontrava-se baseada em três pontes: fisiologia básica, função dos neurônios no cérebro humano e a análise forma da lógica proposicional. Com isso, os pesquisadores criaram um modelo que representava neurônios artificiais. Entretanto, o primeiro pesquisador a apresentar uma visão completa da ia foi Alan Turin, no ano de 1950, em seu artigo *computing machinery and intelligency*, onde demonstrava o Teste de Turing (1950). Esse teste era baseado na impossibilidade de distinção das respostas dadas pelos seres humanos (entidades inteligentes) e as originadas por uma máquina.

A fase do entusiasmo inicial e grandes expectativas (1952-1969) foi marcada de grande expectativa, mas poucos progressos na área da IA. Contudo, foi um período em que apareceram aqueles que seriam os destaques da tecnologia, Arthur Samuel , Allen Newell e Herbet Simon, Ray Solomonoff e Oliver Selfridge.

Na fase sistemas baseados em conhecimento (1966-1979), a Universidade de Stanford, no ano de 1969, desenvolveu o programa Dendral, esse capaz de encontrar as estruturas moleculares orgânicas a partir da espectrometria de massa das ligações químicas presentes em uma molécula desconhecida, isso foi possível pela capacidade do programa em solucionar problemas de forma automática e a capacidade de tomada de decisões. O Dendral foi revolucionário no que tange aos programas inteligentes, já que foi o primeiro sistema que obteve sucesso na aplicação do conhecimento intensivo (Russel; Norvig, 2004) (Gomes, 2010).

Por fim, passa-se por uma fase onde a Inteligência Artificial se torna uma indústria (1980 – até os dias atuais)", nesse momento teremos o primeiro sistema especialista comercial, o R1, da Digital Equipament Corporation. Esse programa permitiu que novos sistemas de computador fossem solicitados/revendidos, a exemplo de que até o ano de 1986, ele já havia faturado aproximadamente 40 milhões de dólares/ano para (Russel; Norvig, 2004) (Gomes, 2010). Igualmente, foi um período de construção de novos sistemas especialistas e de empresas de tecnologia no mundo, culminando no que temos presenciado nos últimos anos, uma revolução na área da inteligência artificial.

Nesse cenário, a IA é atualmente reconhecida como uma tecnologia disruptiva, pois ela altera o funcionamento e a lógica da sociedade, tendo potencial para redefinir vários aspectos da vida dos indivíduos, ou ainda, promover nossas capacidades em diversas esferas da vida. A capacidade preditiva da IA transcende o mero processamento de dados, ela se tornou uma ferramenta capaz de antecipar cenários e oferecer soluções. As plataformas de IA acabam personificando a experiência dos usuários, já que os algoritmos conseguem analisar os comportamentos, adaptando-se às preferências destes (Protázio; Amorim Filho; Bottentuit Junior, 2024). Para Fava (2018, p. 680), o uso de IA pode ser compreendido por "grandes mudanças nas práticas culturais em todas as sociedades por onde, de uma forma ou outra, se estabeleceu, criando no indivíduo a noção de tempo linear, progresso e atualização das informações".

A IA pode ser considerada uma extensão das capacidades humanas, que oferece uma maior eficiência para a obtenção de informações e aprimoramento na resolução dos problemas (Hassabis, 2017). Em diversos setores, o uso de IA causou revolução e vem ajudando na melhoria das soluções, tais como a área da medicina,

com diagnósticos rápidos e precisos, na educação, traduções mais rápidas e fidedignas (Heldwein; de Almeida, 2023).

No entanto, essa capacidade também suscita debates sobre questões éticas. Segundo Kaufman (2022), ao adotar um sistema de IA, deve-se dispor de conhecimento para identificar e evitar resultados tendenciosos. A autora ainda acrescenta que os algoritmos de IA não são agentes morais, pois esses carecem de sentimentos, consciência e intencionalidade. Pensamento também compartilhado por Floridi *et al*. (2018), No momento se trata das questões éticas envolvendo o uso da IA, para os autores precisamos antecipar, evitar e minimizar os riscos advindos do uso da tecnologia e não somente lidar com os problemas após a ocorrência. Para Tegmark (2017), essas discussões não devem ser pautadas somente no uso da tecnologia, mas quais serão os caminhos que a sociedade deverá seguir. Em outras palavras, quais as normativas éticas e jurídicas sobre o assunto.

De forma resumida, a IA tem alterado o estilo de vida da sociedade contemporânea, se tornando uma tecnologia que proporciona soluções para diversos problemas, bem como facilita diversas áreas do conhecimento. Dentre essas ferramentas destaca-se o lançamento do ChatGPT, que como Zhuo *et al.* (2023) apontam, se tornou um dos maiores *breakthroughs* da atualidade.

1.3 CHATGPT

1.3.1 CHATGPT: REVISÃO DE LITERATURA

O ChatGPT foi desenvolvido pela OpenAI, sendo considerada uma ferramenta pública (Brockman *et al*., 2016), baseada no modelo de linguagem G*enerative Pre-trained Transformer* (GPT) (Kirmani, 2022), um *chatbot* 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 (Floridi; Chiriatii, 2020); (Santos, 2022). Aliás, uma ferramenta diferente das anteriores, tais como *chatbots* repetitivos ou assistentes pessoais, que eram automatizados e restritos (Santos, 2022).

Um *chatbot*, por sua definição, seria um software projetado para simular interações com usuários humanos, em especial, conversas pela internet (King, 2023). Já o GPT é um modelo de aprendizado, que usa técnicas de aprendizado não

supervisionado e supervisionado para compreender e gerar uma linguagem similar a linguagem humana (Radford *et al*., 2018).Nesse contexto, o ChatGPT pode ser considerado um exemplar da evolução tecnológica da IA (Brown, 2020) (Lund, 2023), baseada na rede neural chamada GPT-3 (G*enerative Pre-trained Transformer 3*), que permite desempenhar diversas atividades na linguagem chamada de natural (Brown 2020). Para Liu (2021), o ChatGPT é capaz de responder desde uma pergunta simples até perguntas que exigem respostas mais elaboradas. Ademais, Baidoo-anu e Ansah (2023) acrescentam que essa ferramenta é uma evolução do *machine learning*. Por isso, Mollman (2022) reflete que esse seria um dos fatores responsáveis pela rápida disseminação dessa ferramenta pela população.

No entanto, como as demais tecnologias, além dos benefícios e oportunidades, aparecem riscos e incertezas sobre os limites e potências do uso da IA e do ChatGPT (Bertoncini; Serafim, 2023). Os erros de programação e o seu mau uso, a exemplo em trabalhos acadêmicos/científicos ou uso para disseminar desinformações, bem como problemas sociais e questões de privacidade (Daza; Ilozumba; Sok; Heng, 2023), suscitam questionamentos éticos (Sok; Heng, 2023). Destarte, a questão ética e o uso do ChatGPT têm frequentemente sido discutida na área de ensino-aprendizagem e na pesquisa, pois existe a preocupação em relação à redução da agência moral dos discentes, docentes e pesquisadores, tendo em vista o comprometimento da própria essência do ensino e da pesquisa (Jacinto, 2023).

Através dessa ferramenta está apto a construir, de maneira rápida e fácil, respostas e informações, mas essas podem ser equivocadas ou ferir algum princípio ético, ou gerar um resultado falso (Donato; Escada; Villanueva, 2023). Além disso, existem outras questões a serem consideradas, a autenticidade e originalidade, expondo claramente que um conteúdo foi gerado por IA (Farias, 2023). Desse modo, é fundamental que os governos e instituições criem políticas e normas claras para o uso de ias definindo padrões éticos e responsáveis, bem como estipule diretrizes que exijam transparência no uso de IA, em especial ChatGPT (Sampaio *et al*., 2024).

1.3.2 CHATGPT: ALGUMAS CONSIDERAÇÕES

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 (Hua; Jin; Jiang, 2024). As respostas fornecidas pela IA são baseadas em dados até o ano em

que foi construída, podendo estar desatualizadas ou conter informações falsas (Hua; Jin; Jiang, 2024). Além disso, existe a possibilidade de que o ChatGPT gere respostas racistas ou discriminatórias, devido aos vieses presentes nos dados de treinamento (Li *et al*., 2025).

Atualmente, existem muitos casos de sucesso no uso do ChatGPT, mas, como qualquer tecnologia, ela também pode ser usada 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 (Schwartz e LoDuca, 2023).

O uso inadequado dessa ferramenta pode ter consequências perigosas, levantando preocupações no campo do direito, especialmente em relação à proteção de adoção do ChatGPT. Também levantam-se preocupações sobre o futuro dos empregos, pois 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, eles são treinados em grandes conjuntos de dados de texto, que refletem os vieses presentes nesses dados, resultando em respostas discriminatórias ou prejudiciais. Assim, o uso indevido também é uma preocupação (Ferrara, 2023). Ainda, a ferramenta pode ser usada para disseminar 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 (Mengying, 2025).

1.3.3 CHATGPT: ANÁLISE CRÍTICA

Existem vários desafios éticos que podem ser identificados na interação dos usuários com os modelos *Large Language Model* (*LLM*), 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 (Cottier, 2025).

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. (Yang; Zhou; Zhu; Mengqiu, 2024)

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, que ao utilizar uma ferramenta de IA para geração de imagem em forma semelhante aos pôsteres dos filmes de animação da empresa Disney, solicitou que fosse gerada um pôster “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 (O Globo, 2023).

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 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 (CBS, 2016).

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) (Artists, 2024).

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 de entrada 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.

Logo, todas essas discussões, 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 verificação na incorporação dessas informações nas bases de treinamento, essa desinformação pode acabar influenciando as respostas dos modelos, sendo propagadas por eles.

1.3.4 CHAT GPT: 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 (Radanliev, 2025).

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 (Cheong; Cheong, 2024).

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 a Lei Geral de Proteção de Dados (LGPD). 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.

1.4 CONSIDERAÇÕES FINAIS

Os desafios éticos 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, bem como a responsabilidade que assegura os impactos sociais, legais e éticos sejam considerados.

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.

REFERÊNCIAS

BAIDOO-ANU, David; ANSAH, L. O. **Education in the era of generative artificial intelligence (ai)**: understanding the potential benefits of chatgpt in promoting teaching and learning. Journal of ai, v. 7, n. 1, p. 52-62, 2023.

BARBOSA, L. M.; PORTES, L. A. F. **A inteligência artificial**. Revista tecnologia educacional, n. 236, p. 16-27, 2023.

BERTONCINI, A. L. C; SERAFIM, M. C. **Ethical content in artificial intelligence systems**: a demand explained in three critical points. Frontiers in psychology, v. 14, p. 1074787, 2023.

BROCKMAN, G. **Openai gym**. **2016.** Disponível em: https://arxiv.org/abs/1606.01540. Acesso em 22 set. 2025. Doi: https://doi.org/10.48550/arxiv.1606.01540

BROWN, T. **Language models are few-shot learners**. Advances in neural information processing systems, v. 33, p. 1877-1901, 2020.

DAZA, M. T; ILOZUMBA, U. J. **A survey of ai ethics in business literature**: maps and trends between 2000 and 2021. Frontiers in psychology, v. 13, p. 1042661, 2022.

DONATO, H; ESCADA, P.; VILLANUEVA, T. **The transparency of science with chatgpt and the emerging artificial intelligence language models**: where should medical journals stand?. Acta médica portuguesa, v. 36, n. 3, p. 147-148, 2023.

FARIAS, S. A. de. Pânico na academia! **Inteligência artificial na construção de textos científicos com o uso do ChatGPT**. Revista interdisciplinar de marketing (rimar), v. 13, n. 1, 2023.

FLORIDI, L.; CHIRIATTI, M. **Gpt-3: its nature, scope, limits, and consequences**. Minds and machines, v. 30, n. 4, p. 681-694, 2020. Doi:https://doi.org/10.1007/s11023-020-09548-1

FLORIDI, L. **Ai4people—an ethical framework for a good ai society**: opportunities, risks, principles, and recommendations. Minds and machines, v. 28, n. 4, p. 689-707, 2018.

GOMES, D. dos S. **Inteligência artificial: conceitos e aplicações**. Revista olhar científico, v. 1, n. 2, p. 234-246, 2010.

HASSABIS, D. **Neuroscience-inspired artificial intelligence**. Neuron, v. 95, n. 2, p. 245-258, 2017.

HELDWEIN, F. L.; DE ALMEIDA, S. H. M. **Chatgpt na publicação científica–a era da ia chegou**: oportunidades, desafios e ética: a era do ChatGPT na publicação científica. Recet, v. 10, n. 1, p. 4-7, 2023.

KAUFMAN, D. **Desmistificando a inteligência artificial.** Belo horizonte: autêntica, 2022.

KING, M. R. **The future of ai in medicine:** a perspective from a *chatbot*. Annals of biomedical engineering, v. 51, n. 2, p. 291-295, 2023.

KIRMANI, A. R. **Artificial intelligence-enabled science poetry**. Acs energy letters, v. 8, n. 1, p. 574-576, 2022.

JACINTO, E. **Os desafios do uso do ChatGPT no ensino e pesquisa em administração**: uma discussão baseada na ética das virtudes. 2023. Doi: https://doi.org/10.21714/2177-2576enanpad2023

LUND, B D. **Chatgpt and a new academic reality**: artificial intelligence-written research papers and the ethics of the large language models in scholarly publishing. Journal of the association for information science and technology, v. 74, n. 5, p. 570-581, 2023.

MAAROUF, A. C. R. **A responsabilidade civil pelo uso do ChatGPT**: uma análise dos reflexos jurídicos causados pela utilização da inteligência artificial. 2023. 80 f. Monografia (bacharel em ciências jurídicas e sociais) - universidade federal do rio grande do sul, porto alegre, rs, 2025.

MCCARTHY, J. **Manual do programador lisp 1.5. Massachusetts**: imprensa do mit, 1962.

MOLLMAN, S. **Chatgpt gained 1 million users in under a week. Here's why the ai *chatbot* is primed to disrupt search as we know it.** Yahoo! Finance, v. 9, 2022.

PROTÁZIO, L. M. C. G.; AMORIM, L. C. S. F; BOTTENTUIT J.; BATISTA, J. **Ética e responsabilidade na era da inteligência artificial:** uma análise do ChatGPT e seus impactos. Pluralidades epistemológicas e metodológicas no fazer interdisciplinar. In: ana oliveira, caroline amorim *et al*. (orgs.). 2024. Disponível em: https://www.researchgate.net/profile/ozeias-oliveira-junior/publication/381966987_orientacao_argumentativa_e_mobilizacoes_intertextuais_em_postagens_contra_o_projeto_de_lei_5802007/links/66869c2d2aa57f3b8269f86a/orientacao-argumentativa-e-mobilizacoes-intertextuais-em-postagens-contra-o-projeto-de-lei-580-2007.pdf#page=43. Acesso em: 15 set. 2025.

RADFORD, A. **Improving language understanding by generative pre-training. 2018.** Disponível em: https://www.mikecaptain.com/resources/pdf/gpt-1.pdf. Acesso em: 22 set. 2025.

RUSSEL, S; NORVIG, P. **Inteligência artificial.** 2. Ed. Rio de janeiro: campos, 2004.

SAMPAIO, R. C. **Chatgpt e outras ias transformarão a pesquisa científica**: reflexões sobre seus usos. Revista de sociologia e política, v. 32, p. E008, 2024.

SANTOS, L. C. dos. **Inteligência artificial conversacional e o paradigma simulativo**: pistas antropomórficas nas assistentes digitais. 2022. Disponível em: https://proceedings.science/compos/compos-2022/trabalhos/inteligenciaartificial-conversacional-e-o-paradigma-simulativo-pistas-antropomo?lang=pt-br. Acesso em: 22 set. 2025.

SOK, S; HENG, K. **Chatgpt for education and research**: a review of benefits and risks. Cambodian journal of educational research, v. 3, n. 1, p. 110–121-110–121, 2023.

TEGMARK, M. **Life 3.0: being human in the age of artificial intelligence.** Nova iorque: knopf, 2017.

ZHUO, T. Y. **Red teaming chatgpt via jailbreaking**: bias, robustness, reliability and toxicity. Arxiv preprint arxiv:2301.12867, 2023.

APÊNDICE 1 – INTRODUÇÃO À INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 ChatGPT

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

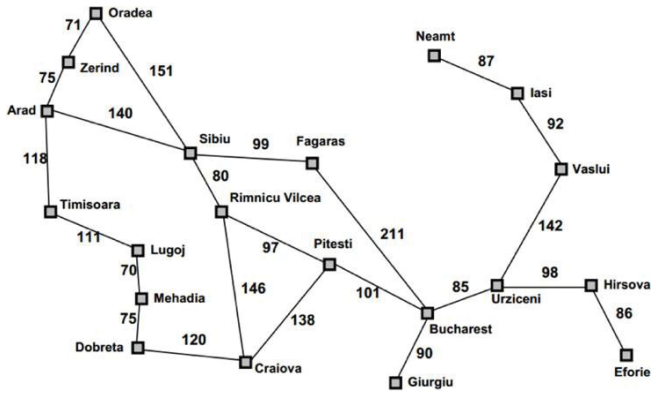
2 Busca Heurística

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

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

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

NÃO É NECESSÁRIO IMPLEMENTAR O ALGORITMO.



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

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

3 Lógica

Verificar se o argumento lógico é válido.

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

Logo

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

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

Dicas:

1. Transformar as afirmações para lógica:

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

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

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

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

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

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

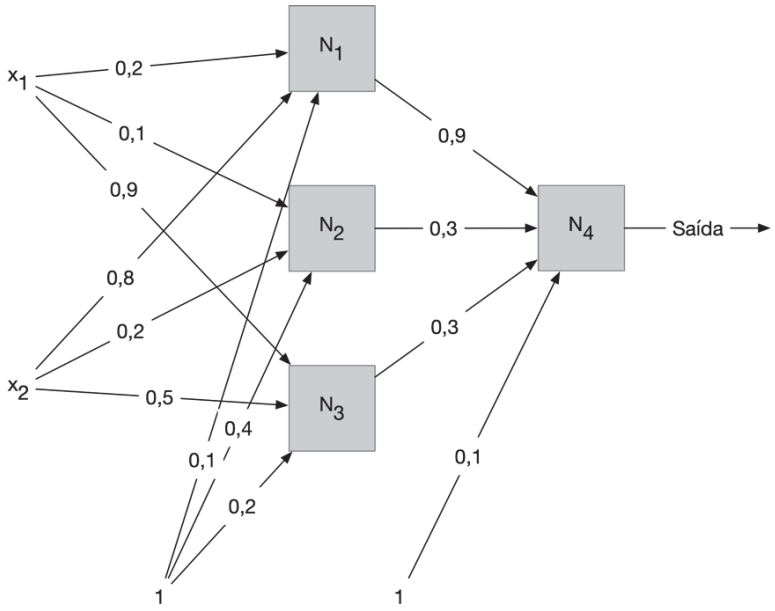
Conjunção: $\alpha, \beta \vdash \alpha \wedge \beta$

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

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

4 Redes Neurais Artificiais

Seja a RNA da figura abaixo.



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

Dada a entrada x_1=-3, x_2=1, dê:

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

B – RESOLUÇÃO

1

a) Inteligência Artificial (IA) refere-se à capacidade de sistemas computacionais de realizar tarefas que, normalmente, requerem inteligência humana. Isso inclui atividades como aprendizado, raciocínio, percepção, resolução de problemas, tomada de decisões e até mesmo reconhecimento de padrões e fala. O objetivo da IA é criar máquinas ou softwares capazes de realizar essas tarefas de forma autônoma ou com interação mínima com seres humanos. A IA pode ser dividida em duas grandes categorias: IA Fraca (ou IA Narrow): Refere-se a sistemas que são projetados para realizar tarefas específicas, como assistentes virtuais (Siri, Alexa), sistemas de recomendação (Netflix, YouTube) ou programas de jogos de xadrez. Esses sistemas não possuem consciência ou capacidade de raciocínio além de suas funções específicas. IA Forte: Refere-se a sistemas com capacidades cognitivas semelhantes às humanas, ou seja, com a habilidade de compreender, aprender, raciocinar e tomar decisões de forma geral. Essa IA ainda é um conceito mais teórico e está em desenvolvimento. Além disso, a IA pode ser implementada por meio de várias técnicas, como aprendizado de máquina (machine learning), redes neurais, algoritmos genéticos, processamento de linguagem natural (PNL), entre outros.

b) Na abordagem de Agentes Racionais, o GPT pode ser considerado um tipo de agente que responde de maneira otimizada aos prompts que recebe. Embora não tenha objetivos próprios, ele tenta gerar respostas que pareçam racionais e úteis com base nos dados que foram pré-carregados durante seu treinamento. No entanto, ao contrário de agentes racionais clássicos, que têm um comportamento orientado por metas e uma forma explícita de otimização, o GPT não toma decisões por si mesmo, mas sim gera respostas baseadas em probabilidades derivadas de um grande volume de dados. Na abordagem baseada em conhecimento, o GPT se encaixa de maneira parcial. O modelo não segue um conjunto fixo de regras ou possui um banco de dados explícito de conhecimento, como em sistemas tradicionais baseados em regras. No entanto, ele é treinado com uma enorme quantidade de textos de diversas fontes e, com isso, adquire um tipo de “conhecimento implícito” sobre padrões de linguagem, o que lhe permite gerar respostas relevantes para os usuários. Embora isso se aproxime de um sistema baseado em conhecimento, o GPT não possui um entendimento real ou consciente do conteúdo que gera. A abordagem empírica é onde o GPT se encaixa com mais clareza. Como um modelo treinado por meio de aprendizado de máquina (machine learning), ele aprende diretamente com grandes volumes de dados, identificando padrões e probabilidades para gerar respostas. Esse processo é empírico porque, ao invés de ser programado com regras fixas, o GPT se baseia na experiência de interagir com imensos conjuntos de dados para aprimorar sua capacidade de gerar textos. Sua aprendizagem vem da observação de dados e da modelagem dessas informações para realizar tarefas específicas de linguagem. Finalmente, na abordagem naturalista, o GPT se destaca como uma tentativa de imitar a comunicação humana. Ele utiliza técnicas de processamento de linguagem natural (PNL) para gerar respostas que são, em muitos casos, indistinguíveis das produções humanas, em termos de fluidez e coerência. No entanto, o GPT não possui uma compreensão real ou semântica do que está gerando, o que diferencia sua “habilidade” de linguagem de um verdadeiro entendimento humano. Mesmo assim, ele representa uma aproximação interessante do comportamento humano em

interações de linguagem, algo que é uma das principais propostas dessa abordagem.

c) O Chat GPT, desenvolvido pela OpenAI, é uma inteligência artificial avançada baseada na arquitetura GPT (Generative Pre-trained Transformer), utilizando algoritmos de aprendizado profundo para compreender e gerar texto de forma mais natural e coerente em comparação com tecnologias anteriores (CUSTÓDIO, 2023).

Esse modelo se destaca por sua capacidade de realizar uma variedade de tarefas, como responder perguntas, redigir textos, traduzir idiomas, entre outras funções. Ao ser integrado a diversas plataformas, o Chat GPT tem se mostrado uma ferramenta poderosa para facilitar a interação humana com a tecnologia. Esse modelo de linguagem faz parte de um grupo de tecnologias conhecidas como Large Language Models, sendo especificamente um modelo GPT. Seu principal objetivo é a geração de texto por meio de interações com o usuário, que insere prompts (perguntas, instruções, ordens, comandos, etc.) de forma sucessiva. Como apontam Silva e Vicentin (2023), o funcionamento do ChatGPT pode gerar a impressão de que o "robô" possui um bom entendimento semântico e uma capacidade de escrita que se aproxima da humana. Isso ocorre graças à enorme base de dados com a qual o modelo foi pré-treinado, permitindo que ele gere respostas com alta coerência e fluidez.

Por ser um modelo de Processamento de Linguagem Natural (PNL), o Chat GPT "aprende" a definir padrões e probabilidades de ocorrência das palavras de forma coerente com a comunicação humana. Contudo, como ressaltam Foletto, Bentes e Maia (2023), apesar de sua aparência convincente e verossímil, o modelo não entende o significado real do que está gerando, o que representa um dos aspectos sensíveis de seu uso. Essa falta de compreensão semântica do Chat GPT levanta questões sobre sua aplicabilidade em contextos que exigem um entendimento profundo do conteúdo gerado.

Além disso, os modelos de inteligência artificial, como o Chat GPT, frequentemente apresentam vieses devido à maneira como são treinados. Esses vieses surgem a partir das escolhas feitas durante o processo de criação, como a seleção dos dados utilizados para o treinamento e as decisões tomadas pelos desenvolvedores e trabalhadores responsáveis. Sampaio et al. (2023) apontam que muitas empresas se justificam pela proteção de segredos comerciais, alegando que não podem revelar os detalhes dos modelos para não prejudicar sua competitividade no mercado. No entanto, essa falta de transparência sobre os algoritmos e as decisões tomadas levanta sérias questões, já que os processos envolvidos muitas vezes são obscuros e difíceis de entender.

Em conclusão, embora o Chat GPT e outras IAs similares representem avanços significativos no campo da tecnologia e do processamento de linguagem natural, é essencial abordar as questões éticas e técnicas envolvidas no seu uso. A transparência nos processos de treinamento, a mitigação de vieses e a compreensão dos limites do modelo são pontos fundamentais para garantir que essas ferramentas possam ser utilizadas de maneira eficaz e responsável, sem comprometer a integridade dos dados e dos resultados gerados.

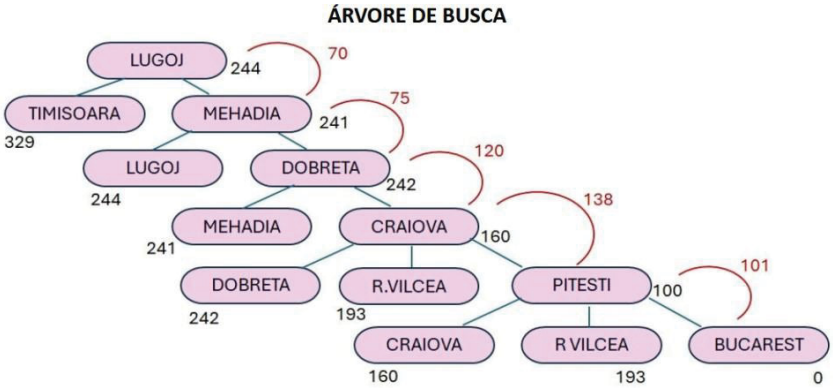
d) Na abordagem de Agentes Racionais, o Chat GPT pode ser visto como um agente que busca otimizar suas respostas com base nos dados que recebe. Ele não possui objetivos próprios, mas responde aos prompts de forma a gerar respostas que pareçam racionais e coerentes, dependendo das informações

fornecidas. No entanto, ao contrário de agentes racionais tradicionais, o Chat GPT não tem uma percepção do mundo real nem objetivos de longo prazo, o que limita sua aplicação como um agente totalmente racional.

Na abordagem baseada em conhecimento, o Chat GPT pode ser considerado uma ferramenta que, embora não possua um banco de dados explícito como em sistemas baseados em regras, é treinado com uma enorme quantidade de dados. Isso permite que o modelo gere respostas coerentes e plausíveis em muitas situações, baseando-se nos padrões e probabilidades de ocorrência das palavras, adquiridos durante o treinamento. Porém, o modelo não possui um conhecimento explícito estruturado ou uma base de regras que ele siga conscientemente, o que diferencia essa abordagem das tradicionais em sistemas baseados em conhecimento.

A abordagem empírica se aplica bem ao Chat GPT, pois ele é treinado com grandes quantidades de dados reais, usando aprendizado de máquina para melhorar suas respostas. Ao invés de seguir regras predefinidas ou lógicas abstratas, o modelo aprende a partir de exemplos e experiência direta com dados. Dessa forma, ele gera respostas mais precisas conforme recebe mais interações e mais dados, o que é típico de sistemas empíricos que dependem da observação e análise de grandes volumes de informações para realizar inferências e gerar respostas. Por fim, na abordagem naturalista, o Chat GPT é um exemplo de tentativa de simulação da comunicação humana natural. Ele se baseia no processamento de linguagem natural para gerar respostas que imitam a interação humana, com uma fluidez e coerência que, em muitos casos, são difíceis de distinguir das produções humanas. Contudo, apesar de sua capacidade de gerar textos plausíveis, o modelo não possui uma compreensão real do conteúdo, o que destaca a diferença entre a inteligência artificial e a inteligência humana, um ponto que caracteriza a crítica dentro dessa abordagem.

2



3

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

O argumento lógico é válido.

4

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 2 – LINGUAGEM DE PROGRAMAÇÃO APLICADA

A – ENUNCIADO

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

Informações sobre a base de dados:

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

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

Metadados:

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

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

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

1 Análise Exploratória dos dados

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

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

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
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
from xgboost import XGBRegressor

dados_df = pd.read_csv('media_precos_carros_brasil.csv')
print(dados_df.shape)

(267542, 11)

print(dados_df.dtypes)

year_of_reference     float64
month_of_reference     object
fipe_code              object
authentication         object
brand                  object
model                  object
fuel                   object
gear                   object
engine_size            object
year_model            float64
avg_price_brl         float64
dtype: object
dados_df.dropna(axis=0,               how='all',               inplace=True)
dados_df.duplicated().sum()
```

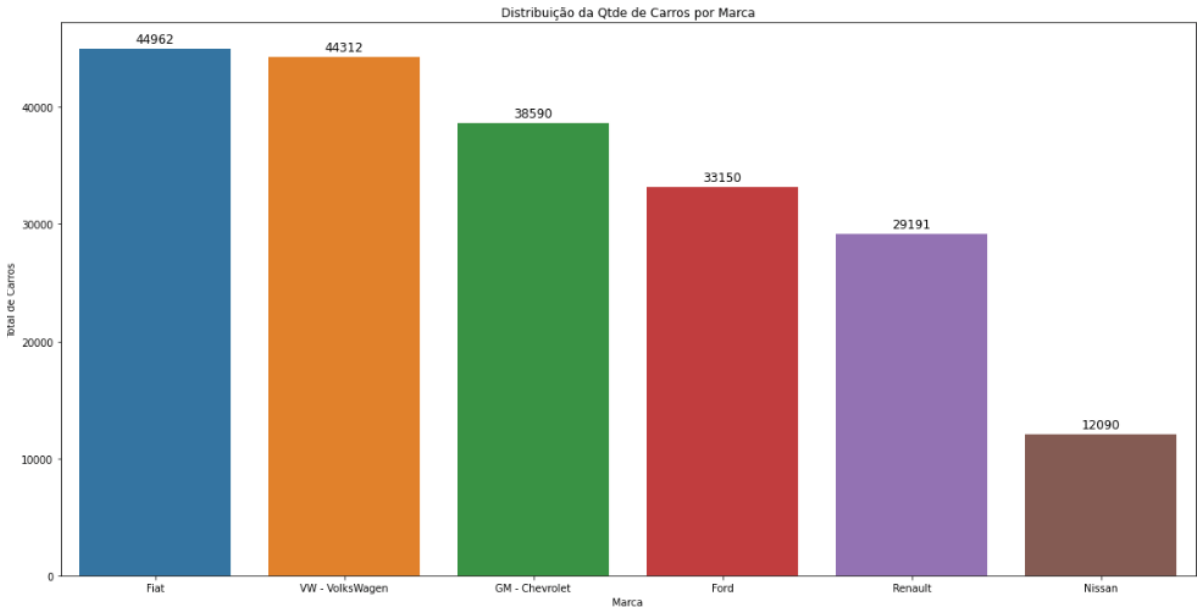
```
2

dados_df.drop_duplicates(inplace=True)

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']
dados_df[num_cols].describe()
dados_df[categ_cols].describe()
dados_df['model'].value_counts()

Palio Week. Adv/Adv TRYON 1.8 mpi Flex     425
Focus 1.6 S/SE/SE Plus Flex 8V/16V 5p      425
Focus 2.0 16V/SE/SE Plus Flex 5p Aut.      400
Saveiro 1.6 Mi/ 1.6 Mi Total Flex 8V       400
Corvette 5.7/ 6.0, 6.2 Targa/Stingray      375
                                          ...
STEPWAY Zen Flex 1.0 12V Mec.                2
Saveiro Robust 1.6 Total Flex 16V CD         2
Saveiro Robust 1.6 Total Flex 16V            2
Gol Last Edition 1.0 Flex 12V 5p             2
Polo Track 1.0 Flex 12V 5p                   2
Name:         model,         Length:         2112,         dtype:         int64

dados_df['brand'].value_counts()

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

A base de dados, que possui 267.542 registros e 11 colunas, apresenta no final 65.245 linhas vazias, provavelmente devido ao processo de exportação para o formato CSV. Após a eliminação dessas linhas, não há valores faltantes. Foram encontradas duas linhas duplicadas na base. Entre as 11 colunas, 3 são numéricas e 8 categóricas. As informações estão distribuídas por 5 marcas de veículos, com uma grande quantidade de modelos diferentes.

2

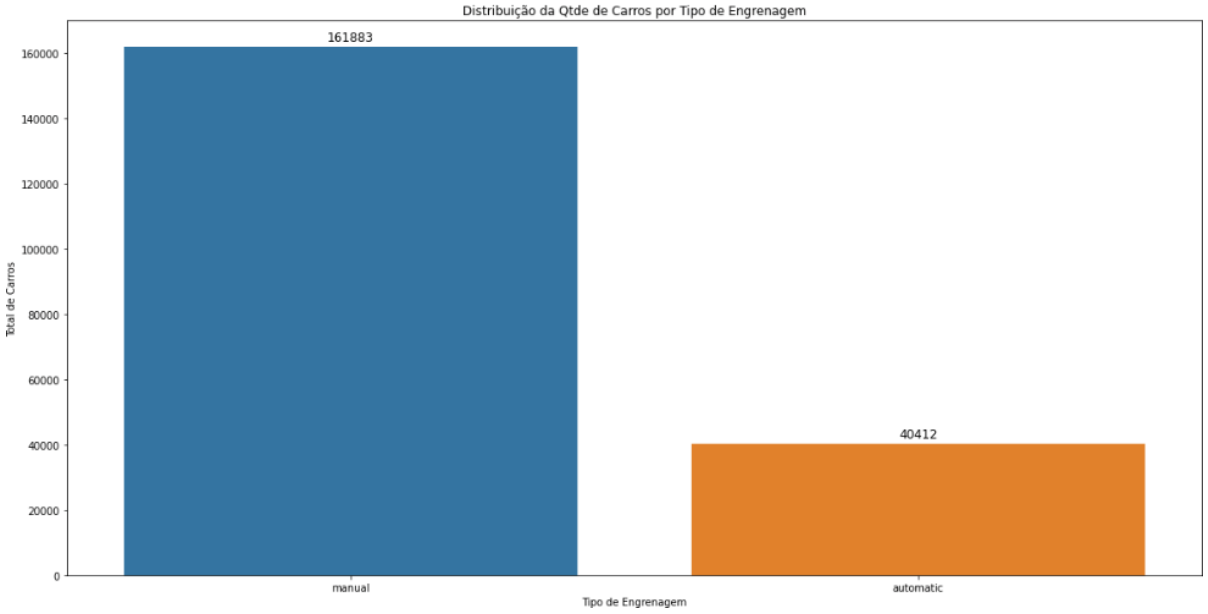
```
plt.figure(figsize=(20,10))
marca_plt = sns.countplot(x="brand", data=dados_df,
order=dados_df['brand'].value_counts().index)
marca_plt.set_xlabel('Marca')
marca_plt.set_ylabel('Total de Carros')
marca_plt.bar_label(marca_plt.containers[0], size=12, padding=3)
marca_plt.set_title('Distribuição da Qtde de Carros por Marca')
```

```
Text(0.5, 1.0, 'Distribuição da Qtde de Carros por Marca')
```



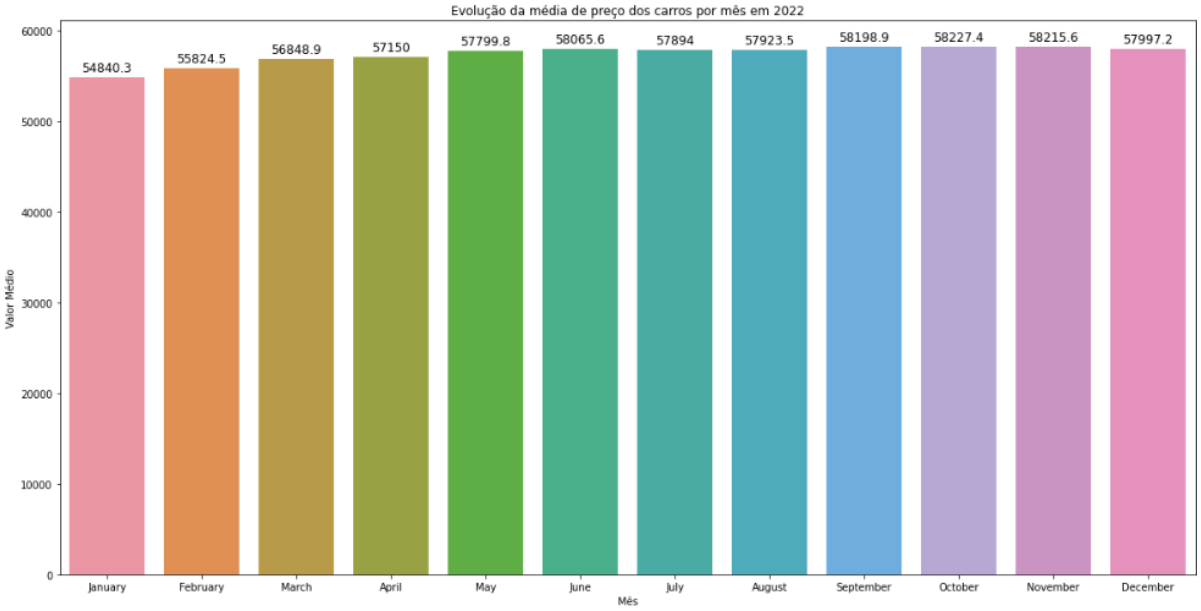

```
plt.figure(figsize=(20,10))
marca_plt = sns.countplot(x="gear", data=dados_df,
order=dados_df['gear'].value_counts().index)
marca_plt.set_xlabel('Tipo de Engrenagem')
marca_plt.set_ylabel('Total de Carros')
marca_plt.bar_label(marca_plt.containers[0], size=12, padding=3)
marca_plt.set_title('Distribuição da Qtde de Carros por Tipo de Engrenagem')
```

```
Text(0.5, 1.0, 'Distribuição da Qtde de Carros por Tipo de Engrenagem')
```

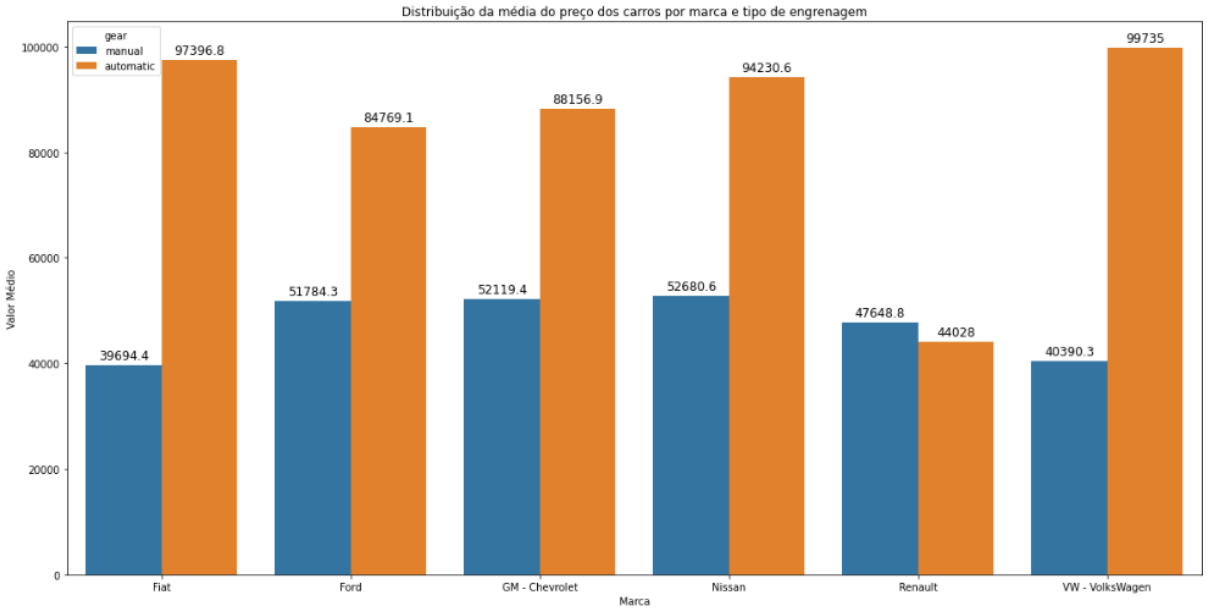



```
plt.figure(figsize=(20,10))
marca_plt = sns.barplot(x="month_of_reference", y='avg_price_brl', \
                        data=dados_df[dados_df['year_of_reference']       ==
2022].groupby('month_of_reference')['avg_price_brl'].mean().round(2).reset_
index(), \
                        order=dados_df[dados_df['year_of_reference']      ==
2022]['month_of_reference'].unique())
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')
```

```
Text(0.5, 1.0, 'Evolução da média de preço dos carros por mês em 2022')
```

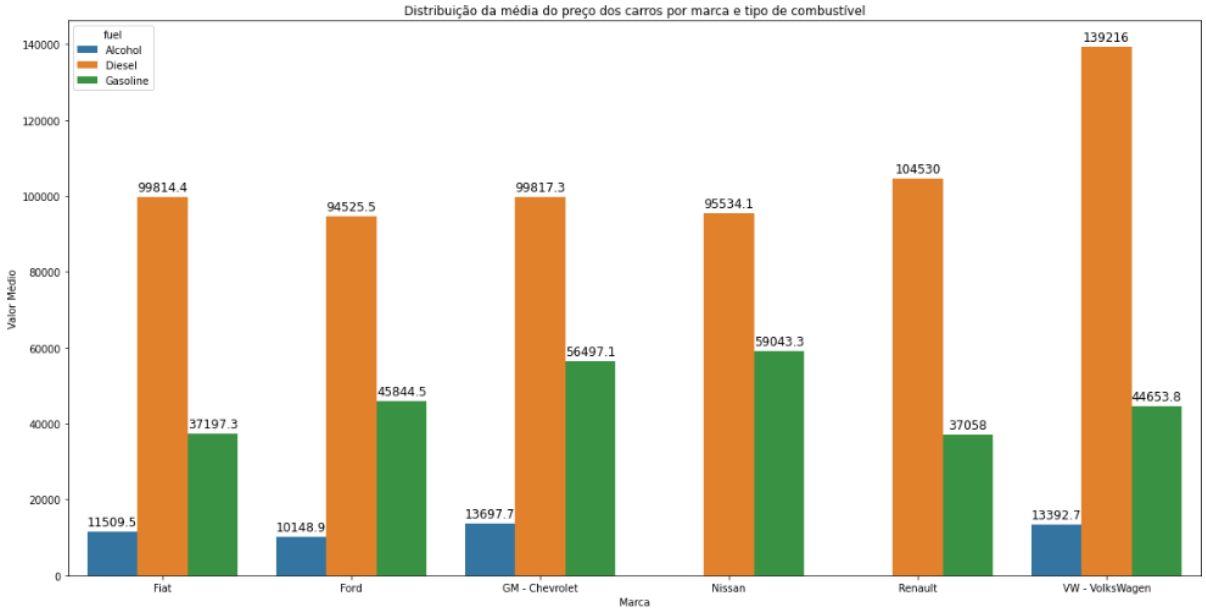



```
plt.figure(figsize=(20,10))
marca_plt = sns.barplot(x='brand', y='avg_price_brl', hue='gear', \
                        data=dados_df.groupby(['brand',
'gear'])['avg_price_brl'].mean().round(2).reset_index(), \
                        hue_order=dados_df['gear'].unique())
marca_plt.set_xlabel('Marca')
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')

Text(0.5, 1.0, 'Distribuição da média do preço dos carros por marca e tipo
de                                                                engrenagem')
```



Em quase todas as marcas, observa-se uma grande disparidade entre o preço médio dos carros automáticos e dos manuais. No entanto, a Renault foge dessa regra, pois, além dos preços serem bem próximos, o valor médio dos carros manuais é ligeiramente superior ao dos automáticos. A Volkswagen é a marca com o maior preço médio para os modelos automáticos, seguida pela Fiat e Nissan, respectivamente.

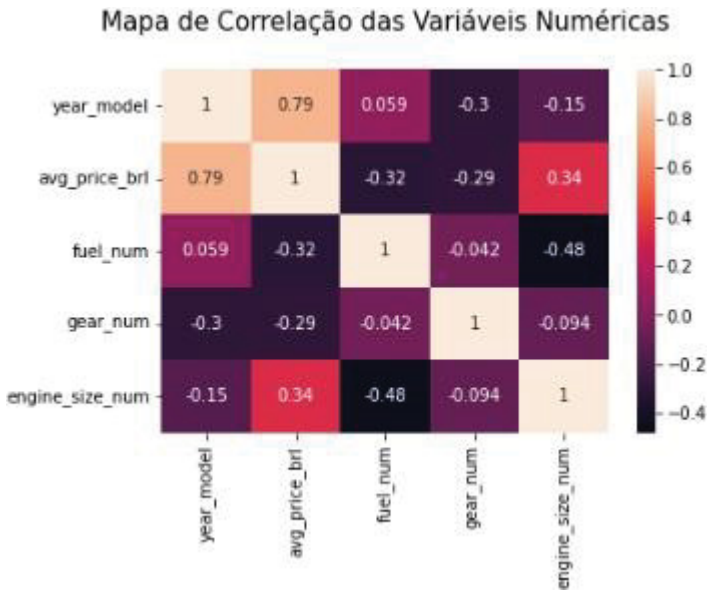
```
plt.figure(figsize=(20,10))
marca_plt = sns.barplot(x='brand', y='avg_price_brl', hue='fuel', \
                        data=dados_df.groupby(['brand',
'fuel'])['avg_price_brl'].mean().round(2).reset_index(), \
                        hue_order=dados_df['fuel'].unique().sort())
marca_plt.set_xlabel('Marca')
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')

Text(0.5, 1.0, 'Distribuição da média do preço dos carros por marca e tipo
de                                                                combustível')
```



Em todas as marcas, os carros movidos a Diesel apresentam um valor médio significativamente superior aos de outros combustíveis, seguidos pelos modelos a Gasolina, que têm preços médios maiores do que os carros a Álcool. Vale ressaltar que a Renault e a Nissan não possuem nenhum veículo movido a Álcool em sua base de dados, logo, não há valores médios para esse combustível nas respectivas marcas.

3

```
dados_df['year_model'] = dados_df['year_model'].astype(int)
dados_df.head()
dados_df['fuel_num'] = LabelEncoder().fit_transform(dados_df['fuel'])
dados_df.head()
dados_df['gear_num'] = LabelEncoder().fit_transform(dados_df['gear'])
dados_df.head()
dados_df['engine_size_num'] = dados_df['engine_size'].str.replace(',',
'.').astype(float)
dados_df.head()
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()
sns.heatmap(dados_num_df.corr("spearman"), annot = True)
plt.title("Mapa de Correlação das Variáveis Numéricas\n", fontsize = 15)
plt.show()
```




```
X = dados_num_df.drop('avg_price_brl',axis = 1)
Y = dados_num_df['avg_price_brl']
Y.head()
```

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

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
random_state = 42)
print(X_train.shape)
X_train.head(1)
```

```
(151721, 4)
```

```
print(X_test.shape)
X_test.head(1)
```

```
(50574, 4)
```

```
model_rf = RandomForestRegressor()
```

```
model_rf.fit(X_train,                                                    Y_train)

RandomForestRegressor()

model_xgboost = XGBRegressor()
model_xgboost.fit(X_train,                                               Y_train)

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

valores_preditos_rf = model_rf.predict(X_test)
valores_preditos_xgboost = model_xgboost.predict(X_test)
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
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
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)

191532975.81806594
7820.921055193546
```

```
0.9288314149027983

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)

191424816.71144438
7819.671020396782
0.928871603964503
```

APÊNDICE 3 – LINGUAGEM R

A – ENUNCIADO

1 Pesquisa com Dados de Satélite (Satellite)

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

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

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

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

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

Tarefas:

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

2 Estimativa de Volumes de Árvores

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

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

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

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

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

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

Tarefas

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

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

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

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

    - Coeficiente de determinação: R^2

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

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

- Erro padrão da estimativa: S_{yx}

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

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

- Syx%

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

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

7. Escolha o melhor modelo.

B – RESOLUÇÃO

1

RandomForest						
	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 foi 0.8411215 (84,11%).

SVM						
	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 foi 0.8660436 (86,60%).

RNA						
	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 foi 0.7764798 (77,64%).

Baseado nos resultados obtidos, o que resultou no maior índice de acurácia foi o modelo SVM. Portanto, o melhor modelo para este exercício é o SVM.

```
install.packages("e1071")
install.packages('knitr')
install.packages("randomForest")
install.packages("kernlab")
install.packages('caret')
install.packages("mlbench")
library('knitr')
library('mlbench')
library('caret')
seed <- 4
set.seed(seed)
data(Satellite)
database <- Satellite
indexes <- createDataPartition(database$classes, p=0.80, list=F)
train <- database[indexes,]
test <- database[-indexes,]
formula <- (classes ~ x.17 + x.18 + x.19 + x.20)
rf <- train(formula, data=train, method='rf')
svm <- train(formula, data=train, method='svmRadial')

rna <- train(formula, data=train, method='nnet', trace=F)
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.rna <- predict(rna, test)
generatePresentation <- function (predicted, testData, modelName, fileName)
{
cm <- confusionMatrix(predicted, testData$classes)
matrixTable <- paste(knitr::kable(cm$table, 'pipe'), collapse='\n')
```

```
overall <- paste(knitr::kable(as.data.frame(cm$overall), 'pipe',
col.names=c('Propriedade', 'Valor')),
collapse='\n')
text <- paste(c("# ", toupper(modelName), "\n\n## Matriz de Confusão\n\n",
matrixTable, "\n\n##
Resultados\n\n", overall), collapse= ')
write.table(text, fileName, quote=F, row.names=F, col.names=F)
}
generatePresentation(predict.rf, test, 'random forest',
'IAA003RandomForest.md')
generatePresentation(predict.svm, test, 'svm', 'IAA003SVM.md')
generatePresentation(predict.rna, test, 'rna', 'IAA003RNA.md')
```

2

EXERCÍCIO 2				
	RandomForest	**SVM**	**Redes Neurais**	**Modo Alométrico de SPURR**
R^2	0,8098753	0,7099086	-0,8242563	0,8147245
Erro Padrão da Estimativa	0,1637666	0,2022896	0,5072815	0,1616646
Percentual de Erro Padrão da Estimativa	12,3738611	15,2845763	38,3291257	12,2150426

Baseado nos resultados apresentados na Tabela 4, o melhor modelo para este exercício, devido ao índice R^2 mais próximo de 1 e erro padrão de estimativa e percentual de erro padrão de estimativa mais próximos de 0, é o Modo Alométrico de SPURR.

```
install.packages("e1071")
install.packages('knitr')
install.packages("randomForest")
install.packages("kernlab")
install.packages('caret')
library('knitr')
library('caret')
seed <- 8
set.seed(seed)
dataset <- read.csv2('http://www.razer.net.br/datasets/Volumes.csv',
header=T, dec=',', sep=';')
dataset$NR <- NULL
indexes <- createDataPartition(y=dataset$VOL, p=0.80, list=F)

train <- dataset[indexes,]
```

```
test <- dataset[-indexes,]
rf <- caret::train(VOL ~ DAP * DAP * HT, data=train, method='rf')
svm <- caret::train(VOL~ DAP * DAP * HT, data=train, method='svmRadial')
rna <- caret::train(VOL~ DAP * DAP * HT, data=train, method='nnet')
spurr <- nls(VOL ~ b0 + b1 * DAP * DAP * HT, train, start=list(b0=0.5,
b1=0.5))
predict.rf <- predict(rf, test)
predict.svm <- predict(svm, test)
predict.rna <- predict(rna, test)
predict.spurr <- predict(spurr, test)
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 )
}
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)),
'RNA'=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')
)
table <- knitr::kable(results, 'pipe')
write.table(table, 'IAA003EX2.md', quote=F, row.names=F, col.names=F)
```

APÊNDICE 4 – ESTATÍSTICA APLICADA I

A – ENUNCIADO

1) Gráficos e tabelas

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

(15 pontos) Elaborar a tabela de 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.

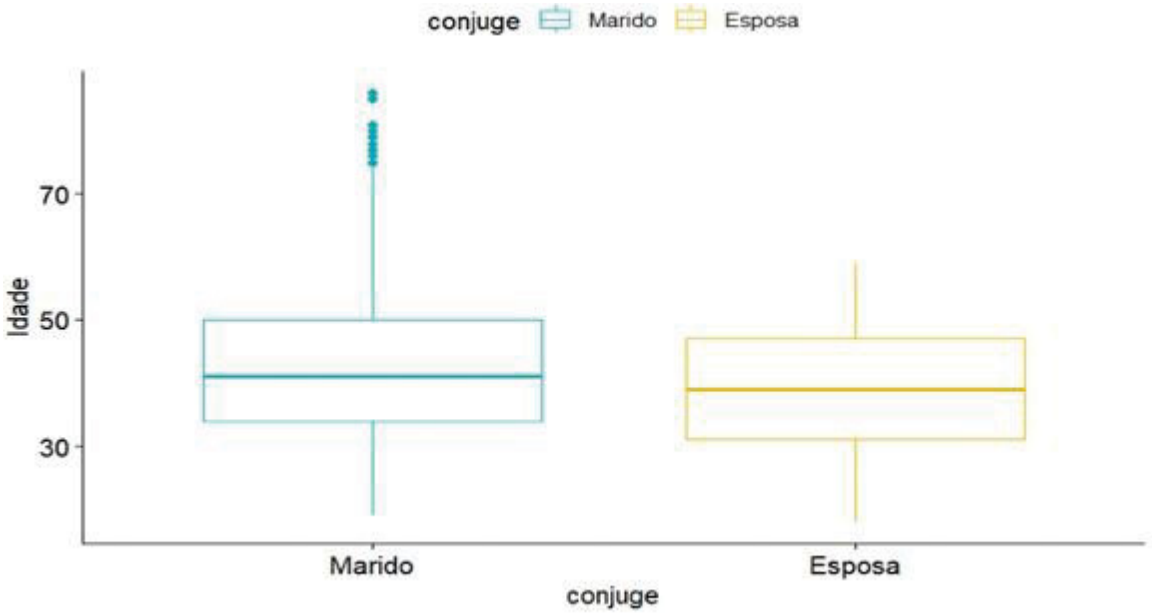
B – RESOLUÇÃO

1

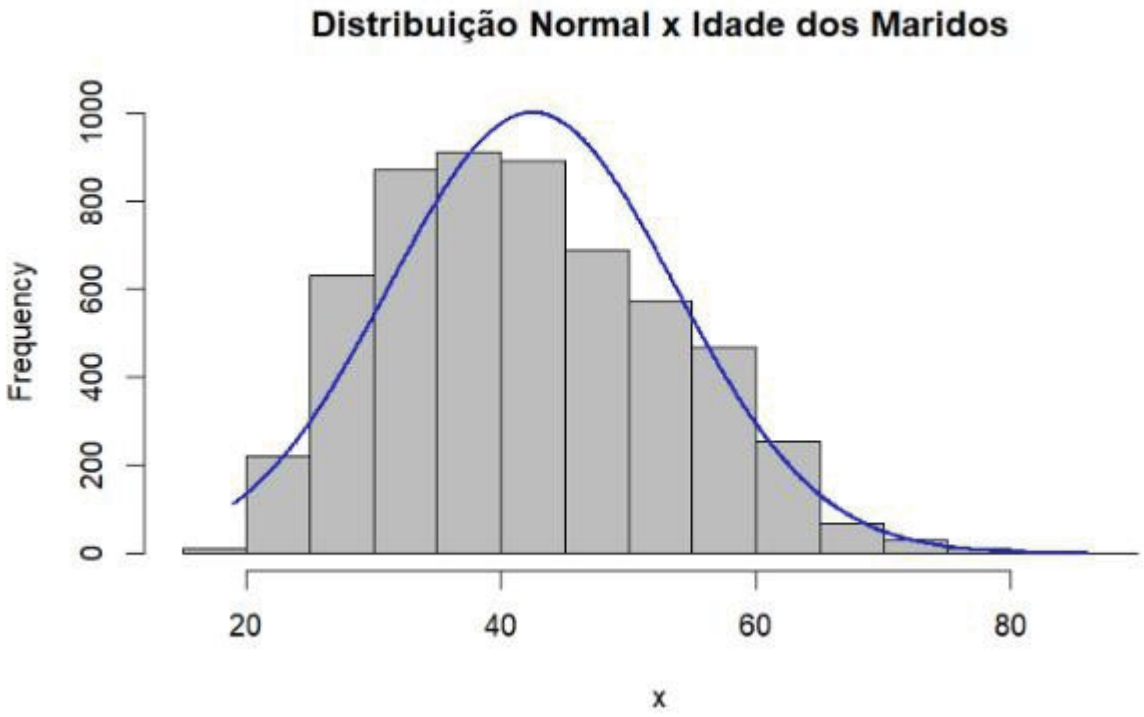
Código:

```
library(rcompanion)
library(car)
library(fdth)
library(nortest)
```

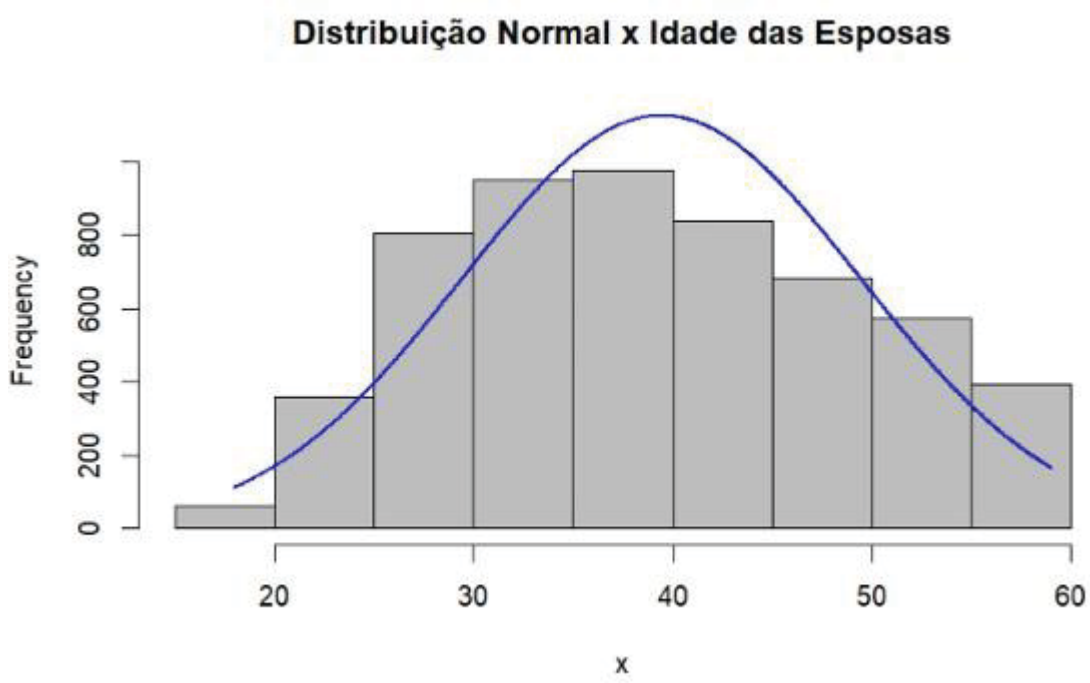
```
library(DescTools)
options(scipen=999)
load('salarios.RData')
idades <- data.frame(
conjuge = rep(c("Marido", "Esposa"), each = 5634),
idade = c(salarios$husage,  salarios$age)
)

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




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




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



A análise dos boxplots revela que a idade das esposas varia entre 18 e 59 anos, enquanto a dos maridos vai de 19 até 86 anos. A mediana das idades, representada pela linha central da caixa, é de 39 anos para as esposas e 41 anos para os maridos. Esse valor corresponde ao segundo quartil, ou seja, o ponto que separa a metade inferior da superior dos dados. Embora o tamanho das caixas — que representa o intervalo interquartil — seja semelhante para ambos, a distribuição etária dos maridos mostra maior dispersão, inclusive com presença de outliers, o que não ocorre entre as esposas.

Nos histogramas, observa-se que a maior concentração de idades das esposas está entre 25 e 65 anos, enquanto a dos maridos se concentra de forma mais expressiva entre 25 e 60 anos. No entanto, as distribuições seguem padrões diferentes: a curva referente aos maridos é mais estreita e elevada (leptocúrtica), com assimetria voltada para a esquerda, enquanto a das esposas é mais achatada (platicúrtica) e tende a se assimetria à direita.

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

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:

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

with(idades,
     range(idade[conjuge=="Esposa"])
 )

[1] 18 59
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    1       3.07
      [25,30) 532 0.09  9.44    1       12.51
      [30,35) 868 0.15  15.41   15     27.92
      [35,40) 899 0.16  15.96   24     43.88
      [40,45) 918 0.16  16.29   33     60.17
      [45,50) 710 0.13  12.60   41     72.77
      [50,55) 573 0.10  10.17   46     82.94
      [55,60) 512 0.09  9.09    5      92.03
      [60,65) 307 0.05  5.45    5      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
```

Com base nos dados apresentados nas Tabelas 1 e 2, observa-se que a maioria das esposas tem entre 25 e 55 anos, enquanto a faixa etária predominante entre os maridos está entre 20 e 60 anos. Analisando a frequência acumulada, nota-se que 18,44% das esposas estão na faixa de 25 a 30 anos, e esse percentual sobe para 91,16% até a faixa de 50 a 55 anos. Já entre os maridos, 12,51% têm entre 25 e 30 anos, e a frequência acumulada atinge 99,06% até a faixa de 65 a 70 anos.Em relação à frequência relativa, a faixa etária mais representativa entre as esposas é a de 35 a 40 anos, com 17,43% dos casos (982 mulheres). Para os maridos, o maior percentual está na faixa de 40 a 45 anos, que concentra 16,29% dos dados (918 homens).

```
conjuge    count  mean    median  var        sd       c.var    IQR
Esposa     5634    39.4         39      99.8      9.99     25.3   16
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
```

Na amostra analisada, a idade média dos maridos é de 42,5 anos, o que representa um valor 7,87% superior à média das esposas, que é de 39,4 anos. A mediana também apresenta essa diferença: os maridos têm uma mediana de 41 anos, enquanto entre as esposas esse valor é de 39 anos — uma diferença de 5,13%. Já em relação à moda, observa-se a maior disparidade: a idade mais frequente entre os maridos é 44 anos (com 201 registros), o que equivale a um aumento de 18,92% em comparação com a moda das esposas, que é de 37 anos (registrada por 217 mulheres).

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)
)
conjuge   count  mean   median  var       sd     c.var     IQR
Esposa    5634   39.4        39     99.8     9.99    25.3      16
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
```

Entre os grupos analisados, a variância da idade dos maridos foi de 126 anos², valor 26,25% superior à variância observada entre as esposas, que foi de 99,8 anos². O desvio padrão também seguiu essa tendência: os maridos apresentaram um desvio padrão de 11,2 anos, o que representa uma diferença de 12,11% em relação ao das esposas, que ficou em 9,99 anos. No que diz respeito ao coeficiente de variação, os maridos novamente apresentaram maior dispersão relativa, com um índice de 26,4%, frente aos 25,3% registrados entre as esposas.

3

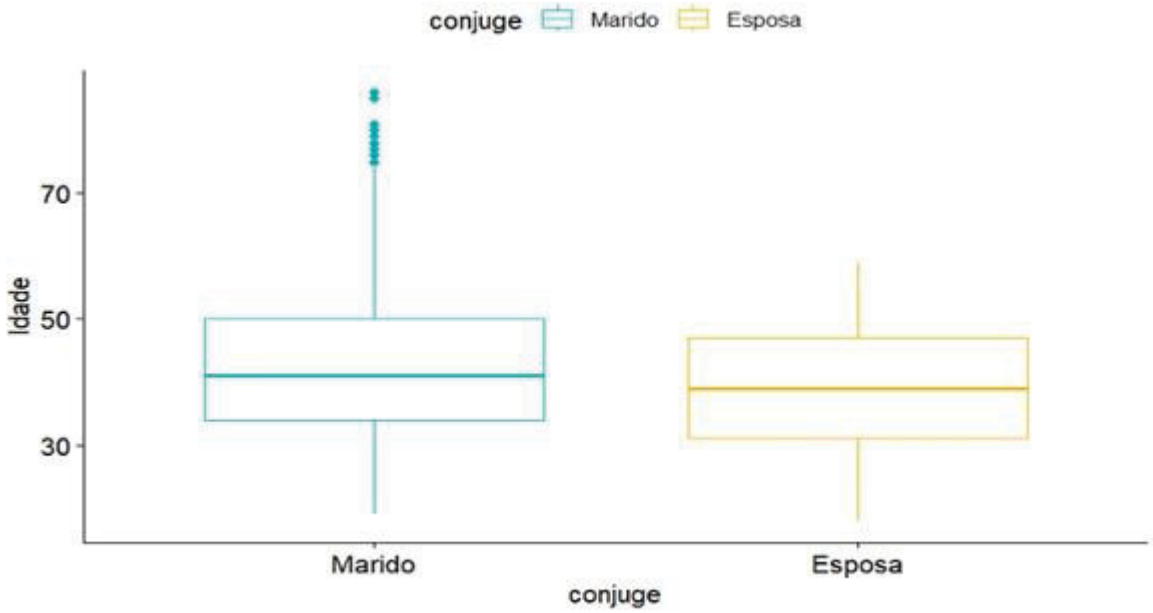
Código:

```
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
```

Como o p-valor obtido para ambos os grupos foi inferior a 0,05, conclui-se que as amostras não seguem uma distribuição normal. Diante disso, a aplicação de um teste paramétrico não é apropriada. Portanto, opta-se por um teste não paramétrico. Considerando que as amostras são independentes, o teste indicado é o Mann-Whitney U. Testando se a mediana da idade dos maridos e esposas são estatisticamente iguais:

```
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)
)
conjuge   count  mean    median  var         sd      c.var      IQR
Esposa      5634    39.4        39       99.8      9.9925.3           16
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")
```



Hipóteses do teste Mann-Whitney U:

H_0: A mediana das idades dos maridos é estatisticamente igual à mediana das idades das esposas.

H_a: A mediana das idades dos maridos é estatisticamente diferente da mediana das idades 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
```

Como o p-valor obtido foi menor que 0,05, rejeita-se a H_0, indicando que há diferença estatisticamente significativa entre as medianas das idades dos maridos e das esposas.

O intervalo de confiança para a diferença entre as medianas está aproximadamente entre -3 e -2, com uma mediana estimada da diferença em torno de -3. Isso sugere que, em média, a idade das esposas é de 2 a 3 anos inferior à dos maridos.

Teste unilateral para comparar as medianas das idades

Hipóteses:

H_0: A mediana da idade das esposas não é menor que a mediana da idade dos maridos (ou seja, é igual ou maior).

H_a: A mediana da idade das esposas é menor que a mediana da idade 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
```

Como o p-valor foi inferior a 0,05, rejeita-se a hipótese nula (H_0). Isso confirma que há evidência estatística suficiente para afirmar que a mediana da idade das esposas é menor que a mediana da idade dos maridos.

O intervalo de confiança para a diferença entre as medianas encontra-se abaixo de aproximadamente -2, com uma estimativa central em torno de -3. Isso indica que, em média, a idade das esposas é de 2 a 3 anos inferior à dos maridos, com um alto grau de confiança estatística.

Teste unilateral para comparar as medianas das idades

Hipóteses:

H_0: A mediana da idade das esposas não é maior que a mediana da idade dos maridos (ou seja, é igual ou
menor).
H_a: A mediana da idade das esposas é maior que a mediana da idade 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
```

Como o p-valor foi maior que 0,05, não há evidência suficiente para rejeitar a hipótese nula (H_0). Dessa forma, não se pode afirmar que a mediana da idade das esposas seja maior que a mediana da idade dos maridos. Além disso, o intervalo de confiança para a diferença entre as medianas apresenta valores acima de aproximadamente -3, com uma mediana estimada em torno de -3, indicando que a diferença entre as medianas não apoia a ideia de que a mediana das esposas seja superior à dos maridos.

Neste caso, foi necessário verificar as premissas para definir se seria adequado aplicar um teste paramétrico ou não paramétrico. Após confirmar que as amostras são independentes, realizou-se o teste de normalidade por meio do teste de Jarque-Bera. Os resultados indicaram que os dados não seguem uma distribuição normal, o que levou à escolha de um teste não paramétrico.

Dado que os grupos são independentes e não pareados, o teste mais apropriado foi o Mann-Whitney U. Os resultados desse teste confirmaram que há, de fato, uma diferença estatisticamente significativa entre as medianas das idades, sendo a mediana das esposas inferior à dos maridos.

APÊNDICE 5 – ESTATÍSTICA APLICADA II

A – ENUNCIADO

Regressões Ridge, Lasso e ElasticNet

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

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

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

B – RESOLUÇÃO

1. Carregando os pacotes

```
# Para Regressão Ridge/Lasso/Elastic-Net

library(glmnet)
require(dplyr)
library(tidyverse)
library(caret)
library(car)
```

```
library(lmtest)
library(olsrr)
```

Nessa primeira parte carregou os pacotes necessários para a análise dos modelos de regressão Ridge, Lasso e Elastic-Net.

2. Buscando o conjunto de dados

```
load("C:/Users/pamar/MODELOS REGRESSAO/trabalhosalarios.RData")
attach(trabalhosalarios)

# Lendo as 6 primeiras linhas dos dados
head(trabalhosalarios)

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

##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':   2065 obs. of  17 variables:
 $ husage  : num  56 31 33 34 42 45 33 31 45 22 ...
 $ 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 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 40 60 50 40 38 40 40 50 40 ...
 $ kidge6  : num  1 0 0 1 1 1 0 0 0 0 ...
 $ earns   : num  100 480 455 102 300 425 770 245 300 299 ...
 $ age     : num  49 29 30 31 41 45 32 30 42 23 ...
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ...
 $ educ    : num  12 14 12 12 12 18 12 15 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  31 9 12 13 23 21 14 9 24 4 ...
 $ kidlt6  : num  0 0 1 0 0 0 0 1 0 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" ...

str(dados.teste)

'data.frame':   509 obs. of  17 variables:
 $ husage  : num  31 44 66 26 38 25 24 46 42 46 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  769 750 500 520 1500 350 390 1000 1250 800 ...
 $ huseduc : num  18 12 16 14 16 11 12 12 18 18 ...
 $ husblck : num  0 0 0 0 0 1 1 0 0 0 ...
 $ hushisp : num  0 0 0 0 0 0 0 0 0 0 ...
 $ hushrs  : num  55 40 40 36 40 40 0 24 50 40 ...
 $ kidge6  : num  0 1 0 0 0 0 0 0 1 0 ...
 $ earns   : num  125 539 500 345 550 141 205 992 769 400 ...

 $ age     : num  27 42 55 27 38 33 22 33 42 44 ...
 $ black   : num  0 0 0 0 0 1 1 0 0 0 ...
 $ educ    : num  14 12 12 14 16 10 12 12 16 18 ...
 $ hispanic: num  0 0 0 0 0 0 0 0 0 0 ...
 $ union   : num  0 0 0 0 0 0 0 1 0 0 ...
 $ exper   : num  7 24 37 7 16 17 4 15 20 20 ...
 $ kidlt6  : num  1 0 0 0 1 0 1 0 0 1 ...
 $ lwage   : num  1.78 2.6 2.53 2.15 2.62 ...
 - 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" ...

# Buscando preditores (x e y - lwage) 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()
```

Na segunda parte buscou os dados no programa R, fez-se a leitura das primeiras seis linhas dos dados, a separação em dados de treino (80% dos dados) e teste (20% dos dados) e a definição

dos da variável resposta (dependente) e as variáveis independentes. Na seção 3 tem a análise dos modelos de regressão *Ridge*, Lasso e *Elastic-Net*.

3. Modelos de Regressão *Ridge*, Lasso e *Elastic-Net*

a) Regressão *Ridge*

```
# Validação cruzada para obter melhor valor de lambda para Regressão Ridge
(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 0.04162   100 0.07837 0.002879      16
1se 0.09615    91 0.08099 0.002414      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 0.7041207 0.2766061

# 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 0.6485221 0.3518183

# Validação cruzada Regressão Ridge
set.seed(123)
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, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2903418  0.7019108  0.1888077

Tuning parameter 'alpha' was held constant at a value of 0
Tuning
 parameter 'lambda' was held constant at a value of 0.04162196
```

Em média, o Modelo de Regressão *Ridge* apresentou um R^2 de 70,19% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo *Ridge* e o RMSE para o modelo é 0,2903418.

b) Regressão Lasso

```
# Validação cruzada para obter melhor valor de lambda para Regressão Lasso
(alpha = 1)
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 0.00172    60 0.07786 0.007279      15
1se 0.05900    22 0.08437 0.006790       3

best_lambda = cv_best_lambda$lambda.min
# Modelo de 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 0.7050773 0.2750101

# 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 0.6507244 0.3469033

# # Validação cruzada Regressão Lasso
set.seed(123)
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, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.2888847  0.703819  0.1831222

Tuning parameter 'alpha' was held constant at a value of 1
Tuning
 parameter 'lambda' was held constant at a value of 0.001719824
```

Em média, o Modelo de Regressão Lasso apresentou um R^2 de 70,38% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo Lasso e o RMSE para o modelo é 0,2888847.

c) Regressão *Elastic-Net*

```
# Validação cruzada para Regressão Elastic Net

set.seed(123)
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 0.4089769 0.001472246

# 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 0.7051887 0.2749518
```

```
# 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 0.6506062 0.346834

# Validação cruzada para Regressão Elastic Net
set.seed(123)
train.control <- trainControl(method = "repeatedcv", number = 10, repeats =
3)

# Modelo final Regressão Elastic Net
Elasticnet_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(Elasticnet_modelo_cv)

glmnet

2574 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2890305  0.7034141  0.1833517

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

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

Em média, o Modelo de Regressão *Elastic-Net* apresentou um R^2 de 70,34% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo *Elastic-Net* e o RMSE para o modelo é 0,2890305.

4. Comparação dos diferentes modelos (*Ridge*, Lasso e *Elastic-Net*)

```
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, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2903418  0.7019108  0.1888077

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

```
Tuning
 parameter 'lambda' was held constant at a value of 0.04162196

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, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.2888847  0.703819  0.1831222



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


Tuning
 parameter 'lambda' was held constant at a value of 0.001719824

print(Elasticnet_modelo_cv)

glmnet

2574 samples
  16 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2890305  0.7034141  0.1833517

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(0.2903418,0.2888847,0.2890305),
                         R2 = c(0.7019108, 0.703819, 0.7034141))

knitr::kable(resultados, align = "c", caption = "Métricas dos modelos")
```

Métricas dos modelos

Método	RMSE	R2
Ridge	0,2903418	0.7019108
Lasso	0,2888847	0,703819
Elastic-Net	0,2890305	0,7034141

Dos três modelos avaliados o Modelo de Regressão Lasso captura 70,38% da variabilidade do salário-hora da esposa em logaritmo neperiano e o RMSE para o modelo é 0,2888847, sendo o modelo escolhido por apresentar menor valor de RMSE e maior valor R^2 (coeficiente de determinação do modelo) em relação a Regressão *Ridge* e o modelo de Regressão *Elastic-Net*.

5. Obtendo melhor modelo

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

16 x 1 sparse Matrix of class "dgCMatrix"
                    s0
husage     0.0009141029
husunion   0.0124011174
husearns   0.0001232746
huseduc    0.0004232420
husblck   -0.0173725642
hushisp   -0.0211190175
hushrs    -0.0011798022
kidge6     0.0062343995
earns      0.0015600008
age        0.0005880356
black     -0.0295717695
educ       0.0250478068
hispanic -0.0244298369
union      0.0535280183
exper      .
kidlt6     0.0465055001

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

 [1] "husage"    "husunion" "husearns" "huseduc"  "husblck"  "hushisp"
 [7] "hushrs"    "kidge6"   "earns"    "age"      "black"    "educ"
[13] "hispanic" "union"    "kidlt6"
```

Na Seção 5 têm os coeficientes do Modelo de Regressão Lasso e a seguir tem a predição e o intervalo de confiança (Seção 6).

6. Predição e intervalo de confiança

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 você não precisa aplicar o logaritmo nela para fazer as regressões, mas é necessário aplicar o antilog para obter o resultado da predição.

```
#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,] 1.602809

exp(valorpreditoIwage)

           s1
[1,] 4.966963

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

[1] 0.2790276

#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: 4.372229

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

IC superior: 5.561696
```

7. Buscando o conjunto de dados sem a variável *earns*

```
##Retirar a varíavel earns do conjunto de dados
trabalhosalarios<-trabalhosalarios[,-c(9)]

##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':   2047 obs. of  16 variables:
 $ husage  : num  56 34 42 33 31 31 44 22 66 43 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  1500 1000 730 1350 769 340 750 249 500 400 ...
 $ huseduc : num  14 14 14 16 18 12 12 12 16 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 ...
                                          40 50 ...
 $ hushrs  : num  40 50 40 40 55 40 40 40
 $ kidge6  : num  1 1 1 0 0 0 1 0 0 1 ... 55 31 ...
 $ age     : num  49 31 41 32 27 30 42 23
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ... 12 12 ...
 $ educ    : num  12 12 12 12 14 15 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  31 13 23 14 7 9 24 4 37 13 ...
 $ kidlt6  : num  0 0 0 0 1 1 0 0 0 0 ...
 $ lwage   : num  1.9 1.63 2.3 2.96 1.78 ...

str(dados.teste)

'data.frame':   527 obs. of  16 variables:
 $ husage  : num  31 33 45 45 26 37 35 44 33 39 ...
 $ husunion: num  0 0 0 0 0 0 0 0 0 0 ...
 $ husearns: num  800 950 1154 1200 520 ...
 $ huseduc : num  17 13 16 12 14 12 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 ...
                                          35 50 ...
 $ hushrs  : num  40 60 38 50 36 50 42 40
 $ kidge6  : num  0 0 1 0 0 0 0 1 1 0 ... 31 28 ...
 $ age     : num  29 30 45 42 27 32 31 45
 $ black   : num  0 0 0 0 0 0 0 0 0 0 ... 12 12 ...
 $ educ    : num  14 12 18 12 14 17 12 18
```

```
 $ hispanic: num  0 0 0 0 0 0 0 0 0 0 ...
 $ union   : num  0 0 0 0 0 1 0 0 0 0 ...
 $ exper   : num  9 12 21 24 7 9 13 21 13 10 ...
 $ kidlt6  : num  0 1 0 0 0 1 0 0 0 0 ...
 $ lwage   : num  2.48 2.43 2.5 2.01 2.15 ...

# Buscando preditores (x e y - lwage) 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()
```

Nessa seção retirou a variável *earns* do conjunto de dados, a separação em dados de treino (80% dos dados) e teste (20% dos dados) e a definição dos da variável resposta (dependente) e as variáveis independentes. Na Seção 8 tem a análise dos modelos de regressão *Ridge*, Lasso e *Elastic-Net* sem a variável *earns* do conjunto de dados

8. Modelos de Regressão *Ridge*, Lasso e *Elastic-Net* sem a variável *earns* do conjunto de dados

a) Regressão *Ridge*

```
# Validação cruzada para obter melhor valor de lambda para Regressão Ridge
(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 0.0228   100  0.1823 0.007804      15
1se 0.3714    70  0.1899 0.007429      15

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 0.2933104 0.4232941

# 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 0.2637242 0.5083858
```

```
# Validação cruzada Regressão Ridge
set.seed(123)
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
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.4417246  0.2839737  0.3310697

Tuning parameter 'alpha' was held constant at a value of 0
Tuning
 parameter 'lambda' was held constant at a value of 0.02279055
```

Em média, o Modelo de Regressão *Ridge* apresentou um R^2 de 28,39% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo *Ridge* e o RMSE para o modelo é 0,4417246.

b) Regressão Lasso

```
# Validação cruzada para obter melhor valor de lambda para Regressão Lasso
(alpha = 1)
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 0.00417    44  0.1813 0.007497      10
1se 0.04271    19  0.1881 0.008248       3

best_lambda = cv_best_lambda$lambda.min
# Modelo de 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 0.2920455 0.4236468

# 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 0.2647143 0.5080806

# # Validação cruzada Regressão Lasso
set.seed(123)
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
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.4415596  0.284645  0.3307113

Tuning parameter 'alpha' was held constant at a value of 1
Tuning
 parameter 'lambda' was held constant at a value of 0.004172353
```

Em média, o Modelo de Regressão Lasso apresentou um R^2 de 28,46% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo Lasso e o RMSE para o modelo é 0,4415596.

c) Regressão *Elastic-Net*

```
# Validação cruzada para Regressão Elastic Net

set.seed(123)
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 0.4089769 0.001472246

# 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 0.2936312 0.4231086

# 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 0.2612618 0.508639

# Validação cruzada para Regressão Elastic Net
set.seed(123)
train.control <- trainControl(method = "repeatedcv", number = 10, repeats =
3)

# Modelo final Regressão Elastic Net
Elasticnet_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(Elasticnet_modelo_cv)

glmnet

2574 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE      Rsquared   MAE
  0.441855  0.2835998  0.3309394

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

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

Em média, o Modelo de Regressão *Elastic-Net* apresentou um R^2 de 28,36% sendo que do total da variabilidade do salário-hora da esposa em logarítmo neperiano foi explicado pelo modelo *Elastic-Net* e o RMSE para o modelo é 0,441855.

9. Comparação dos diferentes modelos (*Ridge*, Lasso e *Elastic-Net*) sem a variável *earns*

do conjunto de dados

```
print(Ridge_modelo_cv)

glmnet

2574 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.4417246  0.2839737  0.3310697

Tuning parameter 'alpha' was held constant at a value of 0
Tuning
 parameter 'lambda' was held constant at a value of 0.02279055
```

```
print(Lasso_model_cv)

glmnet

2574 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE       Rsquared  MAE
  0.4415596  0.284645  0.3307113

Tuning parameter 'alpha' was held constant at a value of 1
Tuning
 parameter 'lambda' was held constant at a value of 0.004172353

print(Elasticnet_modelo_cv)

glmnet

2574 samples
  15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 2317, 2315, 2316, 2317, 2317, 2316, ...
Resampling results:

  RMSE      Rsquared   MAE
  0.441855  0.2835998  0.3309394

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(0.4417298,0.441568,0.441855),
                         R2 = c(0.2839878, 0.284641, 0.2835998))

knitr::kable(resultados, align = "c", caption = "Métricas dos modelos")
```

Métricas dos modelos

Método	RMSE	R^2
Ridge	0,4417298	0,2839878
Lasso	0,4415680	0,2846410
Elastic-Net	0,4418550	0,2835998

Dos três modelos avaliados sem a variável *earns* o Modelo de Regressão Lasso captura 28,46% da variabilidade do salário-hora da esposa em logaritmo neperiano e o RMSE para o modelo é 0,4415680, sendo o modelo escolhido por apresentar menor valor de RMSE e maior valor R^2 (coeficiente de determinação do modelo) em relação a Regressão *Ridge* e o modelo de Regressão *Elastic-Net*.

10. Obtendo melhor modelo sem a variável *earns* no conjunto de dados

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

```
15 x 1 sparse Matrix of class "dgCMatrix"
                    s0
husage    .
husunion  .
husearns  0.0003533008
huseduc   0.0061869358
husblck   .
hushisp   0.0450670456
hushrs   -0.0026212237
kidge6   -0.0507666466
age       0.0027096414
black    -0.0044682028
educ      0.0728893438
hispanic  .
union     0.1664592340
exper     .
kidlt6   -0.0226263904
```

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

```
 [1] "husearns" "huseduc"   "hushisp"   "hushrs"    "kidge6"    "age"
 [7] "black"    "educ"      "union"     "kidlt6"
```

Na Seção 10 têm os coeficientes do Modelo de Regressão Lasso e a seguir tem a predição e o intervalo de confiança (Seção 11).

11. Predição e intervalo de confiança sem a variável *earns* no conjunto de dados

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.

```
dadospredicao = matrix(c(40,0,600,13,1,0,40,1,38,0,13,1,0,18,1), nrow=1,
ncol=15)

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

           s0
[1,] 2.100205

exp(valorpreditoIwage)

           s0
[1,] 8.167844

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

[1] 0.4258416

#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: 7.21901

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

IC superior: 9.11667
```

12. Conclusão

Foram ajustados três métodos de Regressão *Ridge*, Lasso e *Elastic-Net*. O melhor modelo de regressão ajustado foi o Lasso para a variável dependente “lwage” que estava em logaritmo neperiano e fez a volta aplicando a exponencial no valor ajustado na qual obteve uma estimativa de 4,97 com um intervalo de confiança de 95%variando de 4,37 a 5,56. Também fez o ajuste sem a variável *earns* e o melhor modelo também foi o com regressão Lasso com um valor ajustado com estimativa de 8,17 com um intervalo de confiança de 95%variando de 7,22 a 9,12. Quando retirou a variável *earns* o ajuste piorou em termos de R^2 e RMSE, sendo que sem a variável foi 28,46% o R^2 e o RMSE 0,4415680, enquanto com a variável *earns* no modelos os valores de R^2 e RMSE foram 70,38% e 0,2888847,

respectivamente, sendo o modelo Lasso com um melhor ajuste quando a variável *earns* estava presente no conjunto de dados.

APÊNDICE 6 – ARQUITETURA DE DADOS

A – ENUNCIADO

1 Construção de Características: Identificador automático de idioma

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

Parta do exemplo (notebook produzido no Colab) que foi 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

```
ingles = [
```

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

espanhol = [
"Hola, ¿cómo estás?",
"Me encanta leer libros.",
"El clima está agradable hoy.",
"¿Dónde está el restaurante más cercano?",
"¿Qué hora es?",
"Voy al parque todos los días.",
"¿Puedes ayudarme con esto?",
"Me gustaría ir de vacaciones.",
```

```
"Este es mi libro favorito.",
"Me gusta bailar salsa.",
"¿Hablas español?",
"¿Cuál es tu comida favorita?",
"Estoy aprendiendo a tocar el piano.",
"¡Que tengas un buen día!",
"Necesito comprar algunas frutas.",
"Vamos a dar un paseo.",
"¿Cómo estuvo tu fin de semana?",
"Estoy emocionado por el concierto.",
"¿Me pasas la sal, por favor?",
"Tengo una reunión a las 2 PM.",
"Estoy planeando unas vacaciones.",
"Ella canta hermosamente.",
"El perro está jugando.",
"Quiero aprender italiano.",
"Disfruto ir a la playa.",
"¿Dónde puedo encontrar un taxi?",
"Lamento las molestias.",
"Estoy estudiando para mis exámenes.",
"Me gusta cocinar la cena en casa.",
"¿Tienes alguna recomendación de restaurantes?",
]

portugues = [
"Estou indo para o trabalho agora.",
"Adoro passar tempo com minha família.",
"Preciso comprar leite e pão.",
"Vamos ao cinema no sábado.",
"Gosto de praticar esportes ao ar livre.",
"O trânsito está terrível hoje.",
"A comida estava deliciosa!",
"Você já visitou o Rio de Janeiro?",
"Tenho uma reunião importante amanhã.",
"A festa começa às 20h.",
"Estou cansado depois de um longo dia de trabalho.",
"Vamos fazer um churrasco no final de semana.",
"O livro que estou lendo é muito interessante.",
"Estou aprendendo a cozinhar pratos novos.",
"Preciso fazer exercícios físicos regularmente.",
"Vou viajar para o exterior nas férias.",
```

```
"Você gosta de dançar?",
"Hoje é meu aniversário!",
"Gosto de ouvir música clássica.",
"Estou estudando para o vestibular.",
"Meu time de futebol favorito ganhou o jogo.",
"Quero aprender a tocar violão.",
"Vamos fazer uma viagem de carro.",
"O parque fica cheio aos finais de semana.",
"O filme que assisti ontem foi ótimo.",
"Preciso resolver esse problema o mais rápido possível.",
"Adoro explorar novos lugares.",
"Vou visitar meus avós no domingo.",
"Estou ansioso para as férias de verão.",
"Gosto de fazer caminhadas na natureza.",
"O restaurante tem uma vista incrível.",
"Vamos sair para jantar no sábado.",
]

import random

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

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

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

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

```
[['Preciso fazer exercícios físicos regularmente.', 'português'], ['El perro está jugando.', 'espanhol'], ['Voy al parque todos los días.', 'espanhol'], ['O restaurante tem uma vista incrível.', 'português'], ['O trânsito está terrível hoje.', 'português'], ['Estou ansioso para as férias de verão.', 'português'], ['Tengo una reunión a las 2 PM.', 'espanhol'], ['Estoy aprendiendo a tocar el piano.', 'espanhol'], ['Hello, how are you?', 'inglês'], ['The weather is nice today.', 'inglês'], ['What time is it?', 'inglês'], ['Tenho uma reunião importante amanhã.', 'português'], ['A festa
```

começa às 20h.', 'português'], ['Meu time de futebol favorito ganhou o jogo.', 'português'], ['Este es mi libro favorito.', 'espanhol'], ['Hoje é meu aniversário!', 'português'], ["I'm sorry for the inconvenience.", 'inglês'], ['¿Hablas español?', 'espanhol'], ['Vamos sair para jantar no sábado.', 'português'], ['Where can I find a taxi?', 'inglês'], ['Quero aprender a tocar violão.', 'português'], ['Have a great day!', 'inglês'], ['Me gusta cocinar la cena en casa.', 'espanhol'], ['Do you speak English?', 'inglês'], ['Vou visitar meus avós no domingo.', 'português'], ['Can you help me with this?', 'inglês'], ['Could you pass me the salt, please?', 'inglês'], ["I'm planning a vacation.", 'inglês'], ['She sings beautifully.', 'inglês'], ['Lamento las molestias.', 'espanhol'], ['Estou cansado depois de um longo dia de trabalho.', 'português'], ['Me encanta leer libros.', 'espanhol'], ['Você gosta de dançar?', 'português'], ['What is your favorite color?', 'inglês'], ['¿Qué hora es?', 'espanhol'], ['Estou estudando para o vestibular.', 'português'], ['O livro que estou lendo é muito interessante.', 'português'], ['Ella canta hermosamente.', 'espanhol'], ['Gosto de praticar esportes ao ar livre.', 'português'], ['Necesito comprar algunas frutas.', 'espanhol'], ['I love to read books.', 'inglês'], ['¡Que tengas un buen día!', 'espanhol'], ['Gosto de ouvir música clássica.', 'português'], ['El clima está agradable hoy.', 'espanhol'], ['A comida estava deliciosa!', 'português'], ['Vou viajar para o exterior nas férias.', 'português'], ['Quiero aprender italiano.', 'espanhol'], ['Disfruto ir a la playa.', 'espanhol'], ['I need to buy some groceries.', 'inglês'], ['Vamos a dar un paseo.', 'espanhol'], ['¿Tienes alguna recomendación de restaurantes?', 'espanhol'], ['Preciso comprar leite e pão.', 'português'], ['¿Cómo estuvo tu fin de semana?', 'espanhol'], ['Estou aprendendo a cozinhar pratos novos.', 'português'], ["I'm excited for the concert.", 'inglês'], ['Estoy emocionado por el concierto.', 'espanhol'], ['¿Dónde puedo encontrar un taxi?', 'espanhol'], ['I enjoy playing soccer.', 'inglês'], ['I enjoy going to the beach.', 'inglês'], ['I like listening to music.', 'inglês'], ['Vamos fazer uma viagem de carro.', 'português'], ['I like to cook dinner at home.', 'inglês'], ['Estou indo para o trabalho agora.', 'português'], ['Preciso resolver esse problema o mais rápido possível.', 'português'], ['I have a meeting at 2 PM.', 'inglês'], ['Estoy planeando unas vacaciones.', 'espanhol'], ['Vamos fazer um churrasco no final de semana.', 'português'], ['Me gusta bailar salsa.', 'espanhol'], ['I want to learn French.', 'inglês'], ['¿Me pasas la sal, por favor?', 'espanhol'], ['¿Cuál es tu comida favorita?', 'espanhol'], ['Adoro passar tempo com minha família.', 'português'], ['Me gustaría ir de vacaciones.', 'espanhol'], ['Where is the nearest restaurant?', 'inglês'], ['O parque fica cheio aos finais de semana.', 'português'], ['This is a beautiful place.', 'inglês'], ['Do you have any

```
recommendations for restaurants?', 'inglês'], ["I'm learning to play the guitar.", 'inglês'], ['¿Puedes ayudarme con esto?', 'espanhol'], ['Gosto de fazer caminhadas na natureza.', 'português'], ['Você já visitou o Rio de Janeiro?', 'português'], ['Hola, ¿cómo estás?', 'espanhol'], ['¿Dónde está el restaurante más cercano?', 'espanhol'], ['Adoro explorar novos lugares.', 'português'], ['Estoy estudiando para mis exámenes.', 'espanhol'], ['Vamos ao cinema no sábado.', 'português'], ["I'm going to the movies tonight.", 'inglês'], ["I'm studying for my exams.", 'inglês'], ['How was your weekend?', 'inglês'], ["Let's go for a walk.", 'inglês'], ['The cat is sleeping.', 'inglês'], ['O filme que assisti ontem foi ótimo.', 'português']]

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

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

padroes = geraPadroes(pre_padroes)

print(padroes)

dados = pd.DataFrame(padroes)
dados
```

```
[[8.2, 0, 2, 0, 0, 1, 1, 0, 'português'], [4.5, 0, 1, 0, 0, 1, 0, 0,
'espanhol'], [3.8333333333333335, 0, 1, 0, 0, 2, 0, 0, 'espanhol'],
[5.166666666666667, 0, 1, 0, 0, 2, 0, 0, 'português'], [5.0, 0, 2, 0, 0, 2,
0, 1, 'português'], [4.428571428571429, 0, 1, 0, 0, 3, 0, 1, 'português'],
```

[3.142857142857143, 0, 1, 0, 1, 1, 3, 0, 'espanhol'], [4.833333333333333, 0, 0, 0, 0, 2, 1, 0, 'espanhol'], [3.5, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [4.2, 0, 0, 0, 0, 4, 0, 0, 'inglês'], [3.0, 0, 0, 0, 0, 3, 0, 0, 'inglês'], [6.2, 0, 0, 0, 0, 0, 0, 1, 'português'], [3.4, 0, 2, 0, 2, 0, 0, 0, 'português'], [4.375, 0, 0, 0, 0, 2, 0, 0, 'português'], [4.2, 0, 0, 0, 0, 2, 0, 0, 'espanhol'], [4.75, 0, 2, 0, 0, 0, 0, 1, 'português'], [4.333333333333333, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [6.5, 1, 0, 0, 0, 1, 2, 0, 'espanhol'], [4.5, 0, 1, 0, 0, 0, 0, 0, 'português'], [3.0, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [5.0, 0, 0, 0, 0, 1, 0, 1, 'português'], [3.25, 0, 0, 0, 0, 1, 0, 0, 'inglês'], [3.7142857142857144, 0, 0, 0, 0, 2, 2, 0, 'espanhol'], [4.25, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [4.5, 0, 1, 0, 0, 3, 0, 1, 'português'], [3.3333333333333335, 0, 0, 0, 0, 3, 0, 0, 'inglês'], [3.857142857142857, 0, 0, 0, 0, 4, 0, 0, 'inglês'], [3.8, 0, 0, 0, 0, 1, 1, 0, 'inglês'], [6.333333333333333, 0, 0, 0, 0, 1, 0, 0, 'inglês'], [6.333333333333333, 0, 0, 0, 0, 3, 3, 0, 'espanhol'], [4.444444444444445, 0, 0, 0, 0, 2, 0, 1, 'português'], [4.75, 0, 0, 0, 0, 0, 0, 0, 'espanhol'], [4.25, 0, 1, 0, 0, 0, 0, 1, 'português'], [4.6, 0, 0, 0, 0, 4, 0, 0, 'inglês'], [3.0, 0, 1, 0, 0, 0, 0, 0, 'espanhol'], [5.8, 0, 0, 0, 0, 2, 1, 1, 'português'], [4.625, 0, 1, 0, 0, 4, 0, 2, 'português'], [7.0, 0, 0, 0, 0, 0, 0, 1, 0, 'espanhol'], [4.571428571428571, 0, 0, 0, 0, 1, 0, 1, 'português'], [7.0, 0, 0, 0, 0, 4, 1, 0, 'espanhol'], [3.2, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [3.6, 0, 1, 0, 0, 1, 0, 0, 'espanhol'], [5.2, 0, 2, 0, 0, 1, 0, 1, 'português'], [4.6, 0, 1, 0, 0, 0, 1, 0, 'espanhol'], [5.5, 0, 0, 0, 0, 0, 0, 0, 'português'], [4.428571428571429, 0, 1, 0, 0, 2, 0, 1, 'português'], [7.333333333333333, 0, 0, 0, 0, 1, 1, 0, 'espanhol'], [3.6, 0, 0, 0, 0, 2, 3, 0, 'espanhol'], [3.8333333333333335, 0, 0, 0, 0, 1, 0, 0, 'inglês'], [3.2, 0, 0, 0, 0, 1, 0, 0, 'espanhol'], [7.8, 0, 1, 0, 0, 0, 1, 0, 'espanhol'], [4.6, 0, 0, 0, 0, 2, 0, 0, 'português'], [3.8333333333333335, 0, 1, 0, 0, 1, 0, 0, 'espanhol'], [5.833333333333333, 0, 0, 0, 0, 3, 0, 2, 'português'], [3.6666666666666665, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [5.8, 0, 0, 0, 0, 2, 1, 0, 'espanhol'], [5.0, 0, 1, 0, 0, 0, 1, 0, 'espanhol'], [4.75, 0, 0, 0, 0, 1, 1, 0, 'inglês'], [3.5, 0, 0, 0, 0, 3, 0, 0, 'inglês'], [4.2, 0, 0, 0, 0, 3, 0, 0, 'inglês'], [4.333333333333333, 0, 0, 0, 0, 0, 0, 0, 'português'], [3.2857142857142856, 0, 0, 0, 0, 2, 0, 0, 'inglês'], [4.5, 0, 0, 0, 0, 2, 0, 2, 'português'], [5.75, 0, 2, 0, 0, 2, 0, 0, 'português'], [2.5714285714285716, 0, 0, 0, 1, 2, 0, 0, 'inglês'], [7.0, 0, 0, 0, 0, 3, 2, 0, 'espanhol'], [4.5, 0, 0, 0, 0, 2, 0, 0, 'português'], [4.5, 0, 0, 0, 0, 0, 2, 0, 'espanhol'], [3.6, 0, 0, 0, 0, 1, 0, 0, 'inglês'], [3.3333333333333335, 0, 0, 0, 0, 2, 1, 0, 'espanhol'], [4.4, 0, 1, 0, 0, 1, 0, 0, 'espanhol'], [5.166666666666667, 0, 1, 0, 0, 2, 0, 0, 'português'], [4.8, 0, 1, 0, 0, 0, 1, 0, 'espanhol'], [5.4, 0, 0, 0, 0, 4, 0, 0, 'inglês'], [4.125, 0, 0, 0, 0, 2, 0, 0, 'português'], [4.2, 0, 0, 0, 0, 2, 1, 0,

```
'inglês'], [5.857142857142857, 0, 0, 0, 0, 2, 0, 0, 'inglês'],
[3.5714285714285716, 0, 0, 0, 0, 4, 1, 0, 'inglês'], [5.25, 0, 0, 0, 0, 1,
0, 0, 'espanhol'], [5.333333333333333, 0, 0, 0, 0, 3, 0, 1, 'português'],
[3.7142857142857144, 0, 1, 0, 0, 3, 0, 1, 'português'], [4.333333333333333,
0, 2, 0, 0, 0, 2, 0, 'espanhol'], [5.333333333333333, 0, 3, 0, 0, 0, 1, 0,
'espanhol'], [6.25, 0, 0, 0, 0, 1, 0, 1, 'português'], [6.0, 0, 1, 0, 0, 3,
1, 0, 'espanhol'], [4.2, 0, 1, 0, 0, 1, 0, 0, 'português'],
[3.5714285714285716, 0, 0, 0, 0, 4, 0, 0, 'inglês'], [3.3333333333333335, 0,
0, 0, 0, 2, 0, 0, 'inglês'], [4.25, 0, 0, 0, 0, 3, 0, 0, 'inglês'],
[2.3333333333333335, 0, 0, 0, 0, 0, 0, 0, 'inglês'], [4.0, 0, 0, 0, 0, 3, 0,
0, 'inglês'], [4.142857142857143, 0, 1, 0, 0, 2, 0, 1, 'português']]

from sklearn.model_selection import train_test_split
import numpy as np

vet = np.array(padroes)
classes = vet[:,-1]          # classes = [p[-1] for p in padroes]

padroes_sem_classe = vet[:,0:-1]

X_train, X_test, y_train, y_test = train_test_split(padroes_sem_classe,
classes,                  test_size=0.25,                  stratify=classes)

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)

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

y_pred = modelo.predict(X_train)
cm = confusion_matrix(y_train, y_pred)
print(cm)
print(classification_report(y_train, y_pred))

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: 7
[[18  4  1]
 [ 2 19  1]
 [                3                                                    17]]

                          precision    recall    f1-         support
espanhol                0.78          0.78       0.78          23
inglês                     0.70          0.86       0             22
português              0.89          0.71       0.79          24

accuracy                                              0.78        69
macro avg            0.79         0.          0.78        69
weighted avg       0.80          0.7          0.78        69

métricas mais confiáveis
[[5 0 2]
 [0 7 1]
 [3                                  0                                  5]]

                          precision    recall  f1-score   support

espanhol               0.62          0.71       0.67          7
inglês                          1.00            0.88          0.93           8
português              0.62        0.62       0.62          8

accuracy                                           0.74        23
macro avg            0.75      0.74        0.74        23
weighted  avg           0.76          0.74              0.74             23
```

2

```
import numpy as np
import pandas as pd

https://archive.ics.uci.edu/dataset/915/differentiated+thyroid+cancer+recur
rence
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='?')

print(thyro.head())

thyro['Recurred'].value_counts(dropna=False)
```

```
    Age Gender Smoking  Hx Smoking    Hx Radiothreapy   Thyroid Function  \
0    27       F           No                    No                          No
Euthyroid
1    34       F           No                   Yes                          No
Euthyroid
2    30       F           No                    No                          No
Euthyroid
3    62       F           No                    No                          No
Euthyroid
4    62       F           No                    No                          No
Euthyroid

                     Physical                     Examination      Adenopathy
Pathology     Focality Risk  \
0       Single nodular goiter-left              No           Micropapillary
Uni-Focal           Low
1         Multinodular goiter                    No           Micropapillary
Uni-Focal           Low
2       Single nodular goiter-right             No           Micropapillary
Uni-Focal           Low
3       Single nodular goiter-right             No           Micropapillary
Uni-Focal           Low
4         Multinodular goiter                     No           Micropapillary
Multi-Focal         Low

     T      N    M   Stage       Response      Recurred
0  T1a   N0  M0     I         Indeterminate        No
1  T1a   N0  M0     I            Excellent            No
2  T1a   N0  M0     I            Excellent            No
3  T1a   N0  M0     I            Excellent            No
```

```
4  T1a  N0  M0     I            Excellent            No

No     275
Yes    108
Name:                 Recurred,                 dtype:                 int64
```

```
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']

num_cols_data = thyro[num_cols]
cat_cols_data = thyro[cat_cols]
tgt_cols_data = thyro[tgt_cols]

cat_cols_num_data = cat_cols_data.apply(LabelEncoder().fit_transform)

ori_num_data = pd.concat([num_cols_data, cat_cols_num_data, tgt_cols_data],
axis=1)
print(ori_num_data.head())
```

```
    Age  Gender  Smoking  Hx Smoking  Hx Radiothreapy  Thyroid Function  \
0    27       0             0                   0                          0
2
1    34       0             0                   1                          0
2
2    30       0             0                   0                          0
2
3    62       0             0                   0                          0
2
4    62       0             0                   0                          0
2

   Physical Examination  Adenopathy  Pathology  Focality  Risk  T  N  M  \
0                     3                      3                 2            1
2     0  0  0
1                     1                      3                 2            1
2     0  0  0
```

```
2                    4                       3                   2              1
2      0   0   0
3                    4                       3                   2              1
2      0   0   0
4                    1                       3                   2              0
2      0   0   0

    Stage  Response Recurred
0       0         2             No
1       0         1             No
2       0         1             No
3       0         1             No
4               0                       1                                  No
```

```
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())
```

```
     Age  Gender  Smoking  Hx Smoking  Hx Radiothreapy  Thyroid Function  \
0    27        0              0                   0                          0
2
1    34        0              0                   1                          0
2
2    30        0              0                   0                          0
2
3    62        0              0                   0                          0
2
4    62        0              0                   0                          0
2

   Physical Examination  Adenopathy  Pathology  Focality  Risk  T  N  M  \
0                    3                       3                   2              1
2      0   0   0
1                    1                       3                   2              1
2      0   0   0
2                    4                       3                   2              1
2      0   0   0
```

```
3                    4                    3                    2             1
2     0  0  0
4                    1                    3                    2             0
2     0  0  0

   Stage  Response
0      0         2
1      0         1
2      0         1
3      0         1
4      0         1
['No'                                                                  'Yes']

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())
X['Age']                        =                      minmax_scale(X['Age'])
X.drop(columns=['M'],                  axis=1,                  inplace=True)

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


['No'                                                                  'Yes']
[0                                                                        1]
Age   Gender   Smoking   Hx  Smoking   Hx  Radiothreapy   Thyroid  Function   \
0    27        0              0                  0                            0
2    1    34        0             0                 1                          0
2
```

```
2   30        0              0                  0                          0
2
3   62        0              0                  0                          0
2
4   62        0              0                  0                          0
2

   Physical Examination  Adenopathy  Pathology  Focality  Risk  T  N  M  \
0                     3                      3                 2            1
2     0  0  0
1                     1                      3                 2            1
2     0  0  0
2                     4                      3                 2            1
2     0  0  0
3                     4                      3                 2            1
2     0  0  0
4                     1                      3                 2            0
2     0  0  0

   Stage  Response
0      0         2
1      0         1
2      0         1
3      0         1
4                           0                                              1
```

```
from sklearn.model_selection import train_test_split
import numpy as np

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)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.25,
                      stratify=Y,random_state=10)
from sklearn import svm
from sklearn.metrics import confusion_matrix
from          sklearn.metrics          import          classification_report

treinador       =       svm.SVC()               #algoritmo       escolhido
```

```
modelo_orig = treinador.fit(X_oring_train, y_orig_train)

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))
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
```

```
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)

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))

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 7 – APRENDIZADO DE MÁQUINA

A – ENUNCIADO

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

B – RESOLUÇÃO

Seed: 202485

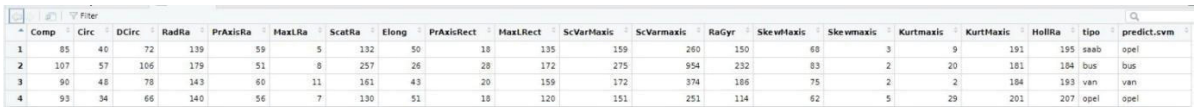
CLASSIFICAÇÃO

Veículo

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	ScVarMaxic	ScVarmaxis	RaGyr	SkewMaxis	Skewmaxis	Kurtmaxis	KurtMaxis	HollRa	tipo	predict.svm
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

Comandos emitidos no RStudio:

```
library("caret")
```

```
setwd("C:/Users/pamar/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/06 - Veículos")

dados <- read.csv("6 - Veiculos - Dados.csv")
dados_novos <- read.csv("6 - Veiculos - Dados - Novos.csv")

dados$a <- NULL
dados_novos$a <- NULL

View(dados)
View(dados_novos)
```

KNN

```
set.seed(202485)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

tuneGrid <- expand.grid(k = c(1,3,5,7,9))

set.seed(202485)
knn <- train(tipo~., data = treino, method = "knn", tuneGrid=tuneGrid)
knn

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

RNA

```
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
rna <- train(tipo~., data=treino, method="nnet", trace=FALSE)
rna

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

```
ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna <- train(tipo~., data=treino, method="nnet", trace=FALSE, trControl=ctrl)
rna

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

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

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

SVM

```
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
svm <- train(tipo~., data=treino, method="svmRadial")
svm

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

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
svm <- train(tipo~., data=treino, method="svmRadial", trControl=ctrl)
svm

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

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

Random Forest

```
set.seed(202485)
indices <- createDataPartition(dados$tipo, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
rf <- train(tipo~., data=treino, method="rf")
rf

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

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rf <- train(tipo~., data=treino, method="rf", trControl=ctrl)
rf

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

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

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

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

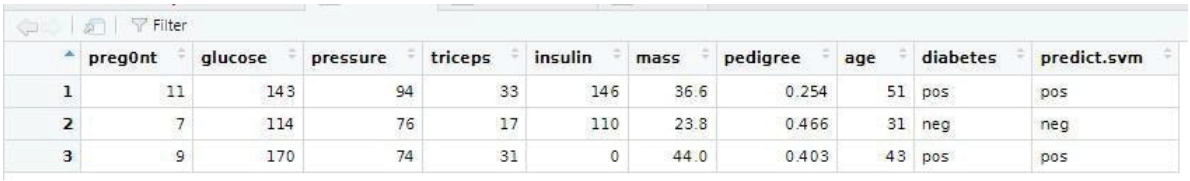
```
resultado <- cbind(dados_novos, predict.svm)
View(resultado)
```

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: 0,7934162

Predição de novos casos:



	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

Comandos emitidos no RStudio:

```
library("caret")
setwd("C:/Users/pamar/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/10 - Diabetes")

dados <- read.csv("10 - Diabetes - Dados.csv")
dados_novos <- read.csv("10 - Diabetes - Dados - Novos.csv")

dados$num <- NULL
dados_novos$num <- NULL
```

```
View(dados)
View(dados_novos)
```

KNN

```
set.seed(202485)
ran <- sample(1:nrow(dados), 0.8 * nrow(dados))
treino <- dados[ran,]
teste <- dados[-ran,]

tuneGrid <- expand.grid(k = c(1,3,5,7,9))

set.seed(202485)
knn <- train(diabetes~., data = treino, method = "knn", tuneGrid=tuneGrid)
knn

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

RNA

```
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
rna <- train(diabetes~., data=treino, method="nnet", trace=FALSE)
rna

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

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rna <- train(diabetes~., data=treino, method="nnet", trace=FALSE,
trControl=ctrl)
rna

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

```
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

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

SVM

```
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
svm <- train(diabetes~., data=treino, method="svmRadial")
svm

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

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
svm <- train(diabetes~., data=treino, method="svmRadial", trControl=ctrl)
svm

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

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

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

Random Forest

```
set.seed(202485)
indices <- createDataPartition(dados$diabetes, p=0.80, list=FALSE)
treino <- dados[indices,]
teste <- dados[-indices,]

set.seed(202485)
rf <- train(diabetes~., data=treino, method="rf")
rf

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

ctrl <- trainControl(method = "cv", number = 10)
set.seed(202485)
rf <- train(diabetes~., data=treino, method="rf", trControl=ctrl)
rf

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

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

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

predict.svm <- predict(svm, dados_novos)
resultado <- cbind(dados_novos, predict.svm)
View(resultado)
```

REGRESSÃO

Admissão

Técnica	Parâmetro	R²	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

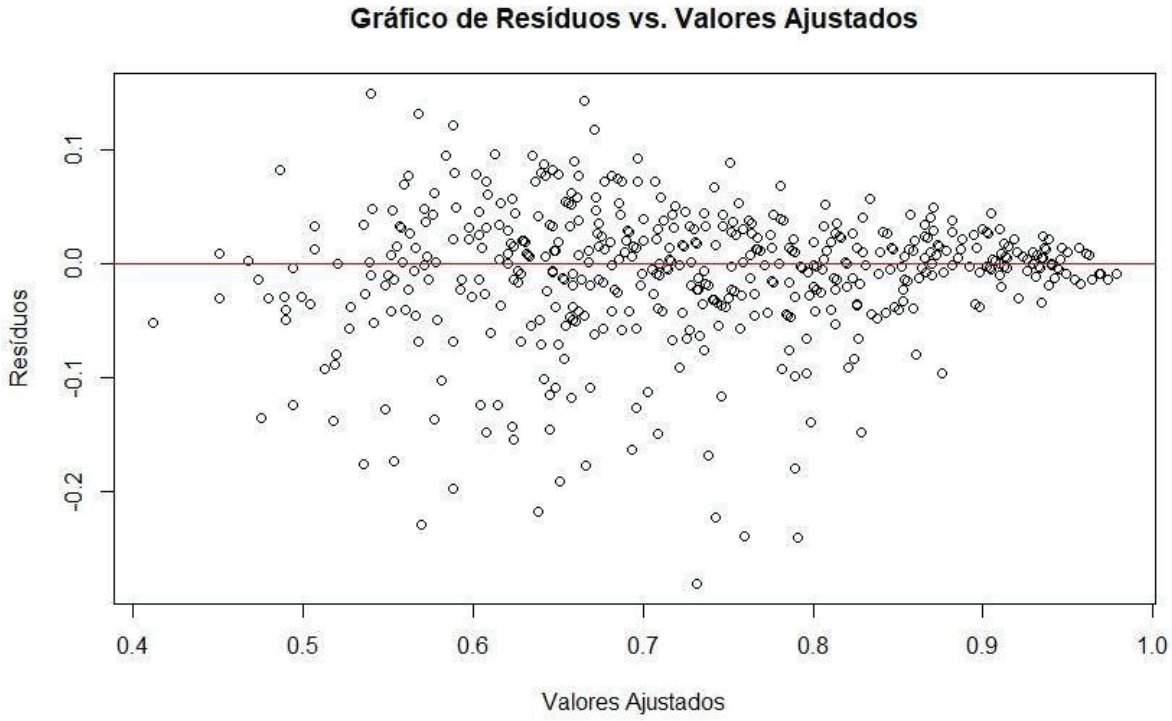
Técnica com melhor desempenho: SVM – CV com grid, R²: 0,789376

Predição de novos 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:



Comandos emitidos no RStudio:

```
r2 <- function(observados, estimados) {
  ret <- (1 - (sum((observados-estimados)^2)
               /sum((observados-mean(observados))^2)))
  return(ret)
}

trabRegressao <- function(seed, filepath, filename, filenewcases,
fileheader=T,
                          colX, metodo, separacao="ho", usaGrid=F){

  if(metodo == "knn"){
    library(caret)
    library(Metrics)
    library(Fgmutils)
  } else if(metodo == "nnet") {
    library(mlbench)
    library(caret)
    library(mice)
    library(Metrics)
    library(Fgmutils)
  } else if( (metodo == "svmRadial") || (metodo == "rf") ) {
```

```
  library(e1071)
  library(kernlab)
  library(caret)
  library(Metrics)
  library(Fgmutils)
}

setwd(filepath)
dados <- read.csv(filename, header=fileheader)
dados$num <- NULL

set.seed(seed)
ind <- createDataPartition(dados[,ncol(dados)], p=0.80, list=F)
treino_df <- dados[ind,]
teste_df <- dados[-ind,]

if(metodo == "knn") {
  tng <- expand.grid(k = c(1,3,5,7,9))
  set.seed(seed)
  modelo <- train(colX, data = treino_df, method = metodo, tuneGrid = tng)
} else {
  if(separacao == "cv"){
    ctrl <- trainControl(method = "cv", number = 10)
    if(usaGrid){
      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 {
    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 {
  if(usaGrid){
    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 {
    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)
    }
  }
```

```
    }
  }

  pred <- predict(modelo, teste_df)

  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)

  novos_casos <- read.csv(filenewcases, header=fileheader)
  novos_casos$num <- NULL

  pred_novos <- predict(modelo, novos_casos)
  novos_casos[,ncol(dados)] <- NULL
  result_pred <- cbind(novos_casos, pred_novos)

  return(list(model=modelo, R2=r2, Syx=syx, Pearson=pearson, RMSE=rmse,
MAE=mae,
              dfPredicao=result_pred))
}

filepath <- "C:/Users/pamar/Documents/Cursos/IAA-2024/IAA008 - APM/Bases/09
- Admissão"
filename <- "9 - Admissao - Dados.csv"
filenewcases <- "9 - Admissao - NovosCasos.csv"
fileheader <- TRUE
colX <- eval(ChanceOfAdmit~.)
seed <- 202485

source("Funcoes_Trabalho.R")
setwd(filepath)
```

KNN

```
metodo <- "knn"

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("### KNN ###")

print("..HOLDOUT - SEM GRID")

print(vRet$model)

sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)
```

RNA

```
metodo <- "nnet"

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")

print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)

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)

View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$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")
```

```
print(vRet$model)

sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)

sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)

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)
```

```
View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)
```

RANDOM FOREST

```
metodo <- "rf"

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")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)

sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)
```

Biomassa

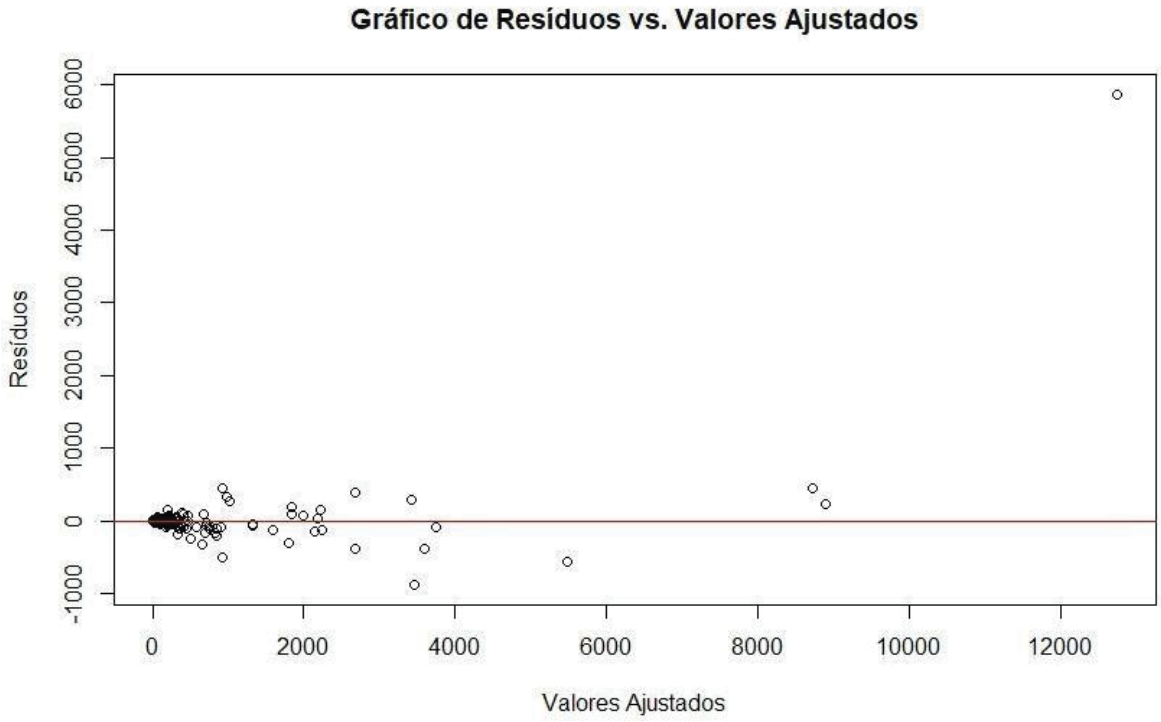
Técnica	Parâmetro	R²	Syx	Pearson	Rmse	MAE
RF – Hold out Sem grid	mtry=2	0.984068	157.786610	0.992201	155.134545	63.856373
RF – CV Com grid	mtry=2	0.976586	181.697563	0.990147	178.643604	72.972522
RF – CV Sem grid	mtry=2	0.973855	190.189821	0.989462	186.993126	74.669903
RF – Hold out Com grid	mtry=2	0.965333	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	-714789053 1188040.0	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:



Comandos emitidos no RStudio:

```
r2 <- function(observados, estimados) {
  ret <- (1 - (sum((observados-estimados)^2)
               /sum((observados-mean(observados))^2)))
  return(ret)
}

trabRegressao <- function(seed, filepath, filename, filenewcases,
fileheader=T,
                          colX, metodo, separacao="ho", usaGrid=F){

  if(metodo == "knn"){
    library(caret)
    library(Metrics)
    library(Fgmutils)
```

```
} else if(metodo == "nnet") {
  library(mlbench)
  library(caret)
  library(mice)
  library(Metrics)
  library(Fgmutils)
} else if( (metodo == "svmRadial") || (metodo == "rf") ) {
  library(e1071)
  library(kernlab)
  library(caret)
  library(Metrics)
  library(Fgmutils)
}

setwd(filepath)
dados <- read.csv(filename, header=fileheader)
dados$num <- NULL

set.seed(seed)
ind <- createDataPartition(dados[,ncol(dados)], p=0.80, list=F)
treino_df <- dados[ind,]
teste_df <- dados[-ind,]

if(metodo == "knn") {
  tng <- expand.grid(k = c(1,3,5,7,9))
  set.seed(seed)
  modelo <- train(colX, data = treino_df, method = metodo, tuneGrid = tng)
} else {
  if(separacao == "cv"){
    # controlador para Cross Validation usando 10 divisões
    ctrl <- trainControl(method = "cv", number = 10)
    if(usaGrid){
      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 {
    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 {
  if(usaGrid){
    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 {
    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)
        }
      }
    }
  }
  pred <- predict(modelo, teste_df)


  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)

  novos_casos <- read.csv(filenewcases, header=fileheader)
  novos_casos$num <- NULL

  pred_novos <- predict(modelo, novos_casos)
  novos_casos[,ncol(dados)] <- NULL
  result_pred <- cbind(novos_casos, pred_novos)

  return(list(model=modelo, R2=r2, Syx=syx, Pearson=pearson, RMSE=rmse,
MAE=mae,
              dfPredicao=result_pred))
}

filepath <- "C:/Users/pamar/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

source("Funcoes_Trabalho.R")
setwd(filepath)
```

KNN

```
metodo <- "knn"

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("### KNN ###")
print("..HOLDOUT - SEM GRID")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)

sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)
```

RNA

```
metodo <- "nnet"
```

```
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")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)
sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)
sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
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")

print(vRet$model)
```

```
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)
```

RANDOM FOREST

```
metodo <- "rf"
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")
print(vRet$model)
sprintf("R2     : %.6f", vRet$R2)
sprintf("Syx    : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE   : %.6f", vRet$RMSE)
sprintf("MAE    : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")

print(vRet$model)
sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)
separacao <- "cv"
usaGrid <- FALSE
vRet <- trabRegressao(seed=seed, filepath=filepath, filename=filename,
                      filenewcases=filenewcases, fileheader=fileheader,
                      colX=colX, metodo=metodo, separacao=separacao,
                      usaGrid=usaGrid)
print(" ")
print("..CROSS VALIDATION - SEM GRID")
print(vRet$model)
sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
View(vRet$dfPredicao)

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")
print(vRet$model)
sprintf("R2      : %.6f", vRet$R2)
sprintf("Syx     : %.6f", vRet$Syx)
sprintf("Pearson: %.6f", vRet$Pearson)
sprintf("RMSE    : %.6f", vRet$RMSE)
sprintf("MAE     : %.6f", vRet$MAE)
```

```
View(vRet$dfPredicao)
```

AGRUPAMENTO

Veículo

```
K-modes clustering with 10 clusters of sizes 47, 62, 110, 95, 92, 102, 76, 82, 96, 84

Cluster modes:
   Comp Circ DCirc RadRa PrAxisRa MaxLRa ScatRa Elong PrAxisRect MaxLRect ScVarMaxis ScVarmaxis RaGyr
1   107   51   103   160       66      6    213    31         24       162         228         396   148
2    88   52    96   197       62      9    185    35         22       161         202         284   121
3    86   39    66   127       59      7    128    52         18       131         164         246   139
4   104   53   108   197       64     11    215    31         24       168         226         669   218
5   100   48   104   201       62     10    201    32         23       158         223         635   186
6    91   43    73   141       56      7    150    44         19       145         170         341   173
7   100   55   101   177       58     11    222    30         25       174         225         706   213
8    85   45    70   154       56      6    151    45         19       147         170         333   186
9    89   47    85   149       56     11    157    43         20       149         173         354   186
10   85   42    70   120       54      6    149    46         19       145         169         317   172
   SkewMaxis Skewmaxis Kurtmaxis KurtMaxis HollRa tipo
1         66         0        11       191    198  bus
2         67         3         8       193    201 opel
3         70         1         2       183    185 saab
4         72        11        11       188    199 saab
5         68         5        16       191    197 saab
6         69         2        12       189    196  bus
7         73         0         4       187    196 opel
8         81         0         1       180    183  bus
9         75         1         7       183    193  van
10        85         6        14       179    182  bus
```

Comandos emitidos no RStudio:

```
library(klaR)
setwd("C:/base/ ")
dados <- read.csv("veiculos.csv")
View(dados)
## execução do cluster
K = 10
cluster.res <- kmodes(dados, K, iter.max = 10, weighted = FALSE )
cluster.res

dados$a <- NULL

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

REGRAS DE ASSOCIAÇÃO

Musculação

```
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].
> summary(rules)
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
```

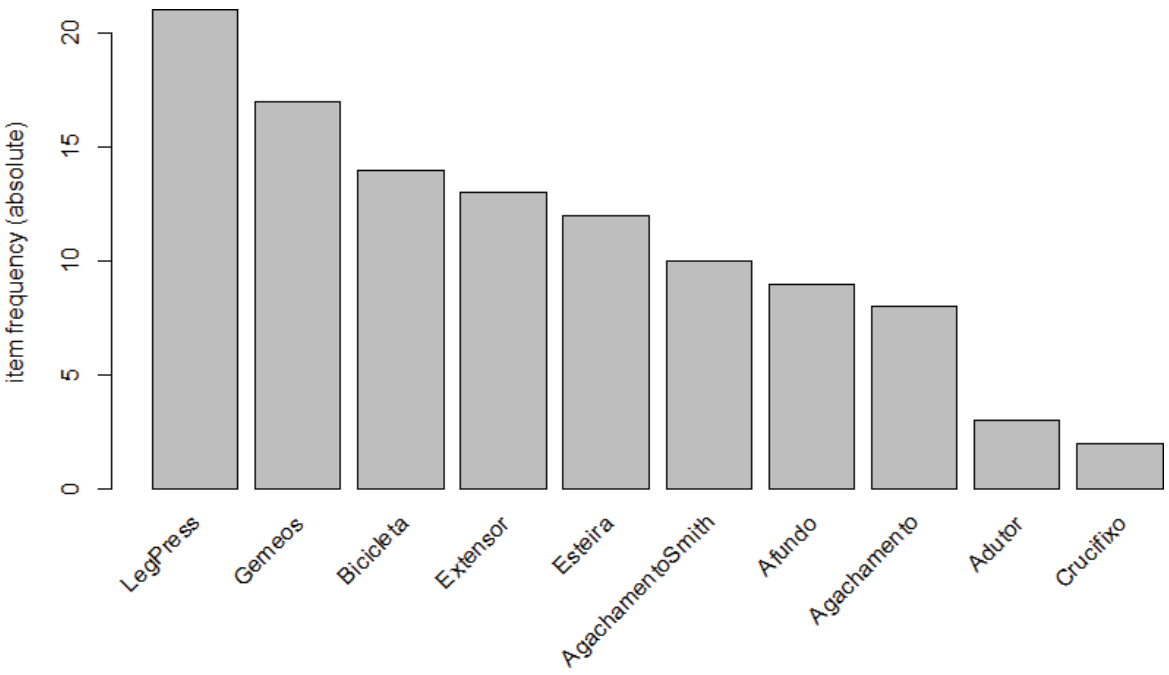
Regras geradas com uma configuração de Suporte e Confiança.

Comandos emitidos no RStudio:

```
setwd("C:/Users/pamar/Downloads")
dados <- read.transactions(file="2 - Musculacao - Dados.csv",format="basket",sep=";")
inspect(dados[1:4])
```

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

```
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 8 – DEEP LEARNING

A – ENUNCIADO

1 Classificação de Imagens (CNN)

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

2 Detector de SPAM (RNN)

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

3 Gerador de Dígitos Fake (GAN)

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

4 Tradutor de Textos (Transformer)

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

B – RESOLUÇÃO

1

```
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

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

x_train, x_test = x_train / 255.0, x_test / 255.0
```

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

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)

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

x = Flatten()(x)

x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.2)(x)
x = Dense(K, activation="softmax")(x)

model                    =                    Model(i,                    x)

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)
```

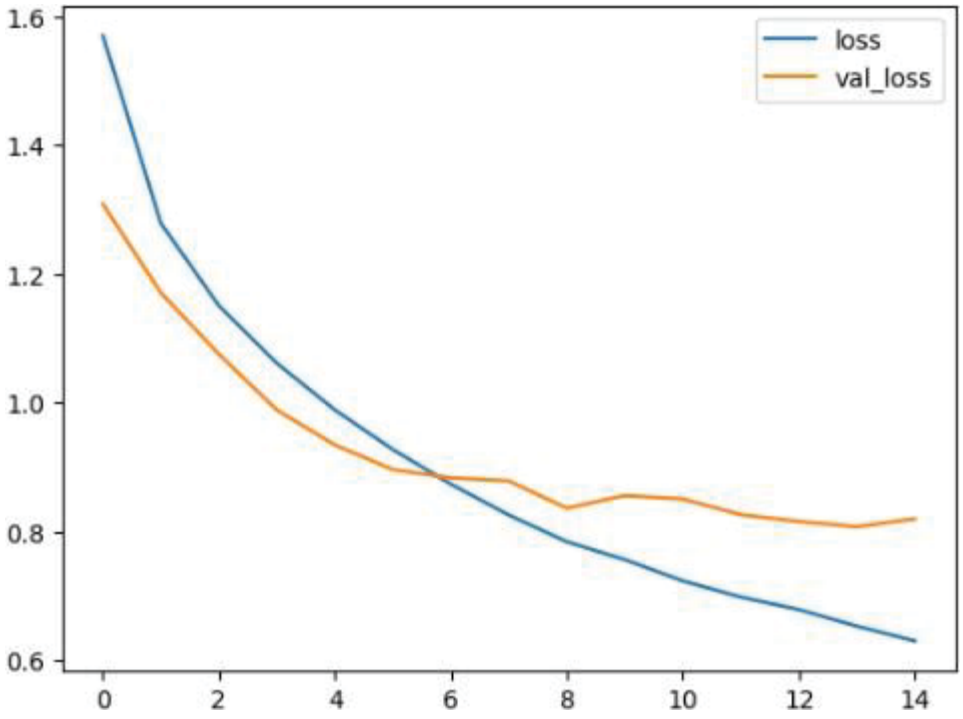
```
model.compile(optimizer="adam", loss="sparse_categorical_crossentropy",
metrics=["accuracy"])
```

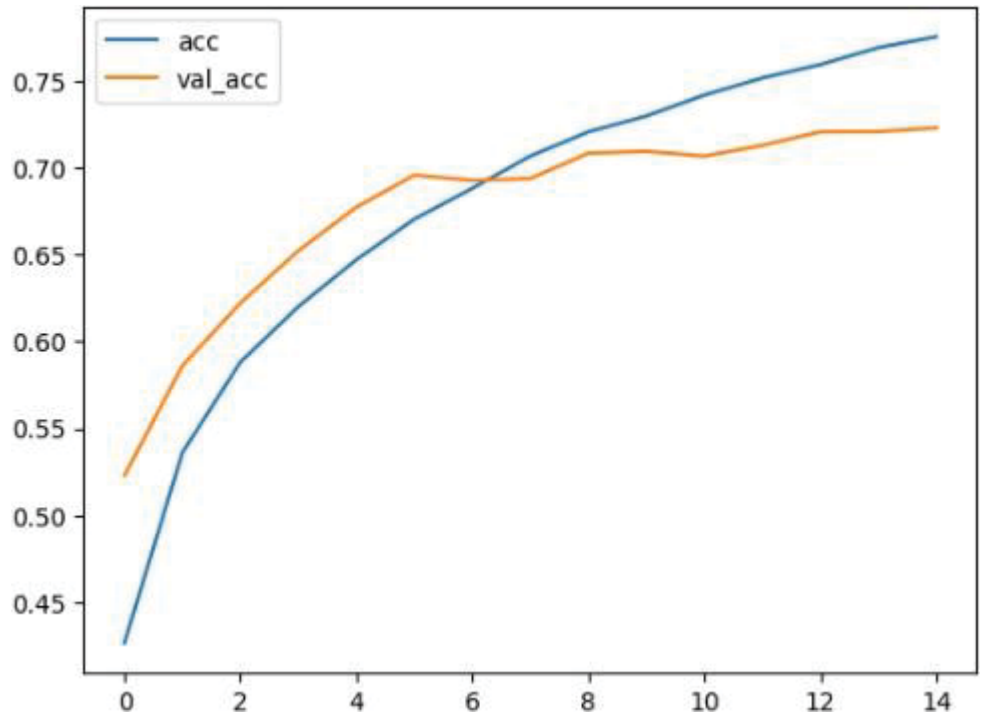
```
r = model.fit(x_train, y_train, validation_data=(x_test, y_test), epochs=15)
```

```
Epoch 1/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17s 7ms/step - accuracy: 0.3518
- loss: 1.7613 - val_accuracy: 0.5229 - val_loss: 1.3085
Epoch 2/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.5243
- loss: 1.3063 - val_accuracy: 0.5861 - val_loss: 1.1708
Epoch 3/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10s 3ms/step - accuracy: 0.5832
- loss: 1.1618 - val_accuracy: 0.6222 - val_loss: 1.0755
Epoch 4/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6192
- loss: 1.0688 - val_accuracy: 0.6523 - val_loss: 0.9890
Epoch 5/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.6510
- loss: 0.9825 - val_accuracy: 0.6773 - val_loss: 0.9348
Epoch 6/15
```

```
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.6717
- loss: 0.9248 - val_accuracy: 0.6958 - val_loss: 0.8965
Epoch 7/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.6939
- loss: 0.8651 - val_accuracy: 0.6927 - val_loss: 0.8837
Epoch 8/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7133
- loss: 0.8044 - val_accuracy: 0.6938 - val_loss: 0.8788
Epoch 9/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7229
- loss: 0.7807 - val_accuracy: 0.7084 - val_loss: 0.8367
Epoch 10/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.7324
- loss: 0.7475 - val_accuracy: 0.7095 - val_loss: 0.8563
Epoch 11/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.7427
- loss: 0.7178 - val_accuracy: 0.7067 - val_loss: 0.8509
Epoch 12/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7580
- loss: 0.6788 - val_accuracy: 0.7129 - val_loss: 0.8265
Epoch 13/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 3ms/step - accuracy: 0.7654
- loss: 0.6619 - val_accuracy: 0.7207 - val_loss: 0.8161
Epoch 14/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11s 4ms/step - accuracy: 0.7753
- loss: 0.6369 - val_accuracy: 0.7208 - val_loss: 0.8079
Epoch 15/15
1563/1563 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - accuracy: 0.7823
- loss: 0.6083 - val_accuracy: 0.7229 - val_loss: 0.8202

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
plt.show()

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

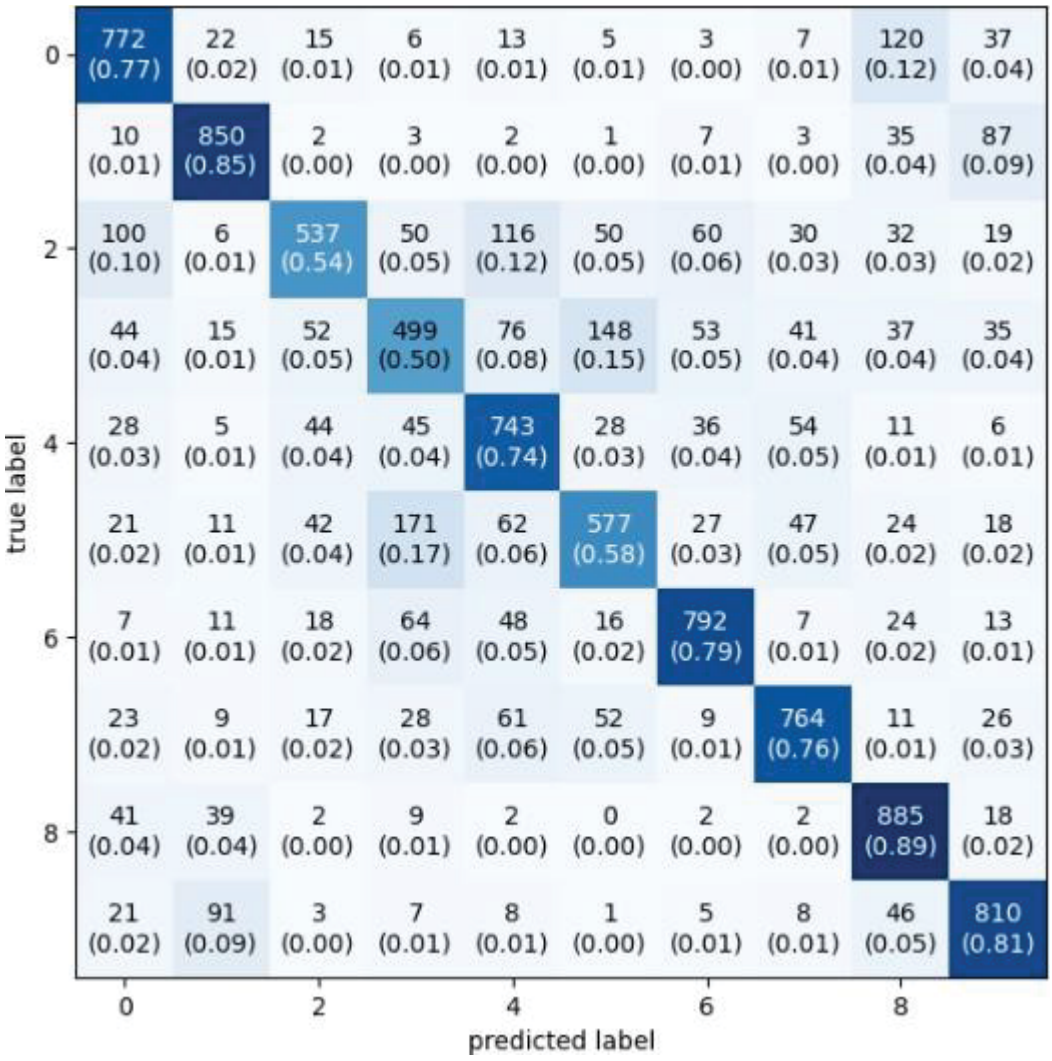
```
plt.show()
```






```
y_pred = model.predict(x_test).argmax(axis=1)
```

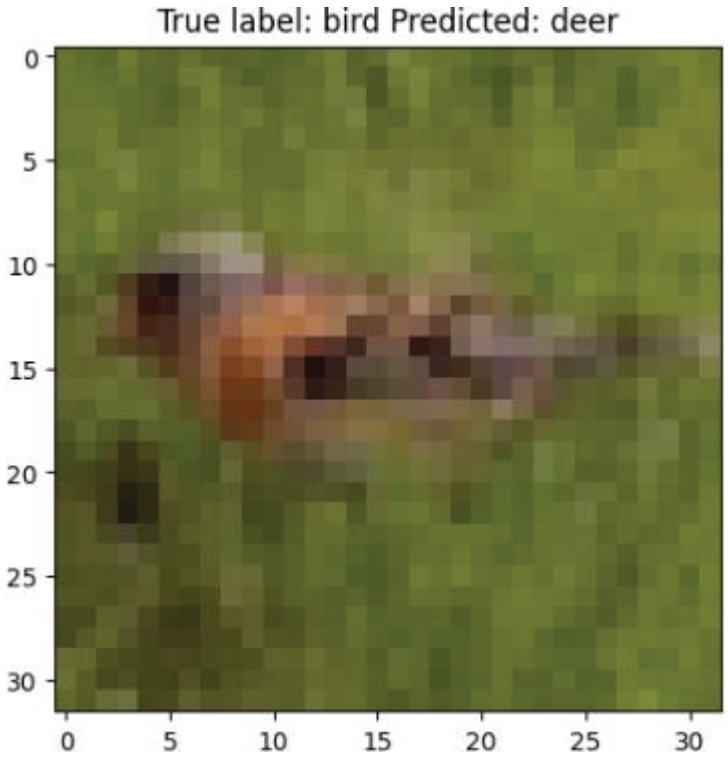
```
# Mostrar a matriz de confusão
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm,    figsize=(7,    7),    show_normed=True)
```




```
labels= ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog",
"horse", "ship", "truck"]
    misclassified = np.where(y_pred != y_test)[0]

i = np.random.choice(misclassified)

plt.imshow(x_test[i], cmap="gray")
```

```
plt.title("True label: %s Predicted: %s" % (labels[y_test[i]], labels[y_pred[i]]))
```

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



2

```
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

--2025-08-10 20:08:33-- http://www.razer.net.br/datasets/spam.csv
Resolving www.razer.net.br (www.razer.net.br)... 178.128.150.229
Connecting to www.razer.net.br (www.razer.net.br)|178.128.150.229|:80...
connected.
HTTP request sent, awaiting response... 200 OK
Length: 503663 (492K) [application/octet-stream]
Saving to: ‘spam.csv’

spam.csv 100%[===================>] 491.86K --.-KB/s in 0.1s
2025-08-10 20:08:34 (3.56 MB/s) - ‘spam.csv’ saved [503663/503663]

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)
    7197 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()
```

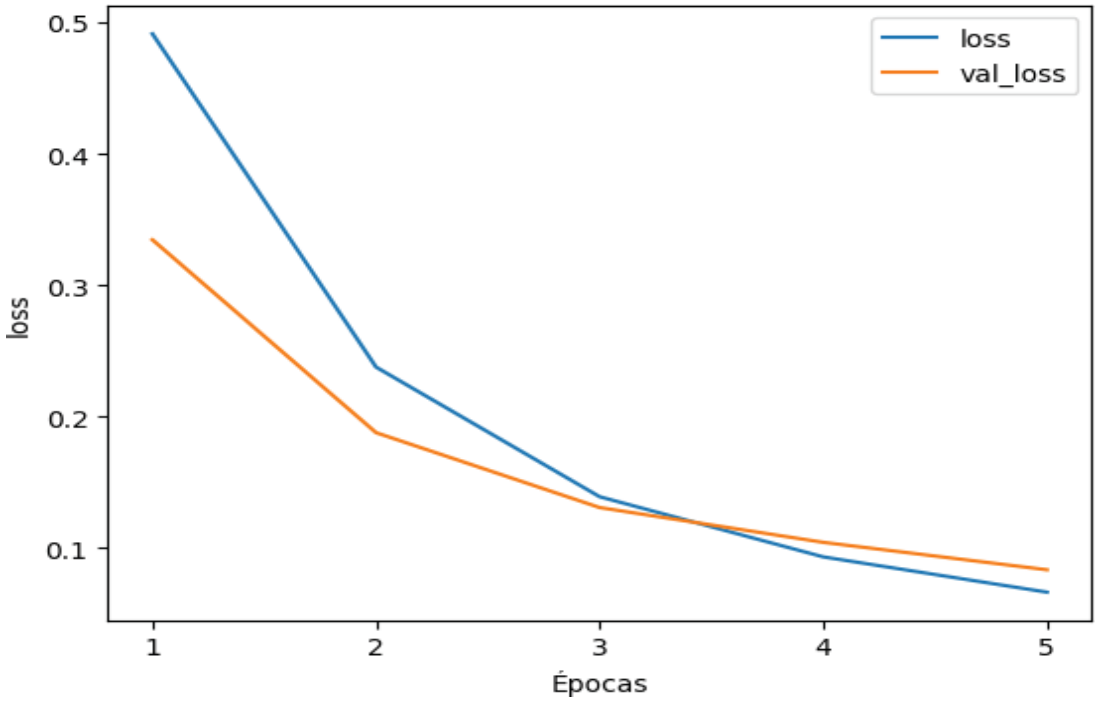
Model: "functional"

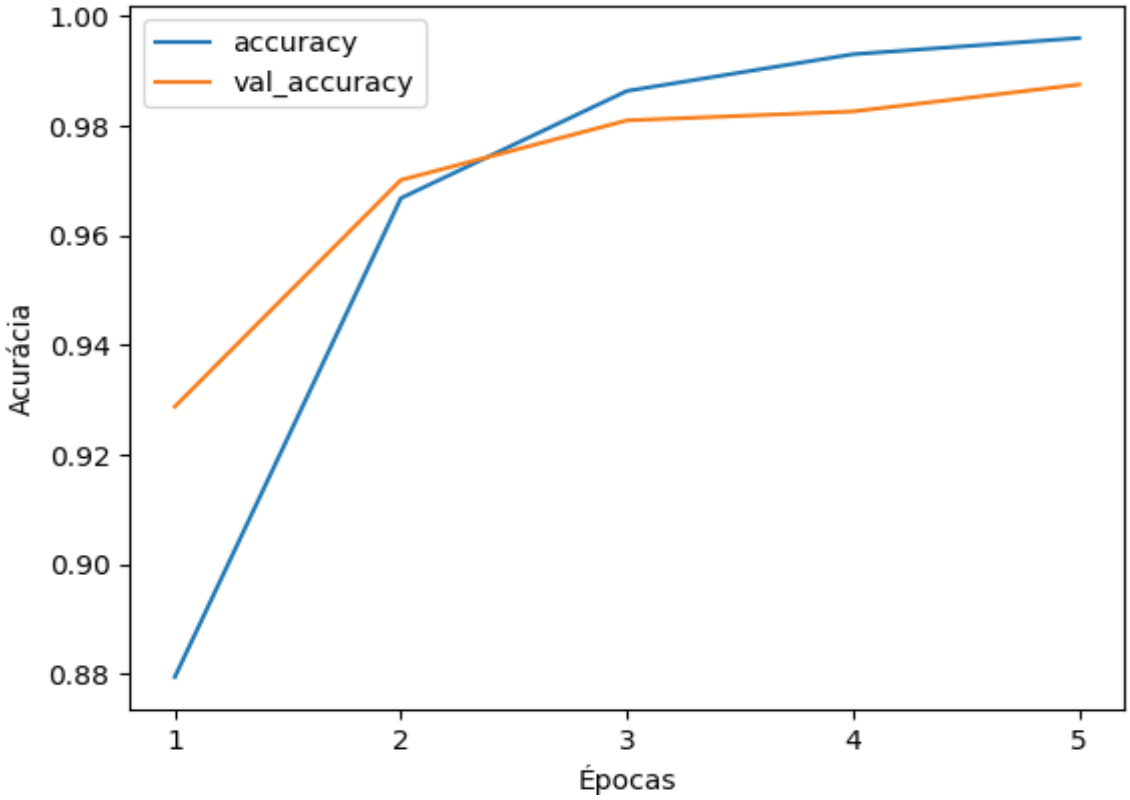
Layer (type)	Output Shape	Param #
input_layer (InputLayer)	(None, 189)	0
embedding (Embedding)	(None, 189, 20)	143,960
lstm (LSTM)	(None, 5)	520
dense (Dense)	(None, 1)	6

```
Total params: 144,486 (564.40 KB)
Trainable params: 144,486 (564.40 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 19ms/step - accuracy:
0.8707 - loss: 0.5696 - val_accuracy: 0.9288 - val_loss: 0.3345
Epoch 2/5
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy:
0.9606 - loss: 0.2757 - val_accuracy: 0.9701 - val_loss: 0.1877
Epoch 3/5
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy:
0.9814 - loss: 0.1530 - val_accuracy: 0.9810 - val_loss: 0.1307
Epoch 4/5
```

```
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9899
- loss: 0.1086 - val_accuracy: 0.9826 - val_loss: 0.1042
Epoch 5/5
117/117 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9943
- loss: 0.0784 - val_accuracy: 0.9875 - val_loss: 0.0833
```

```
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()
```





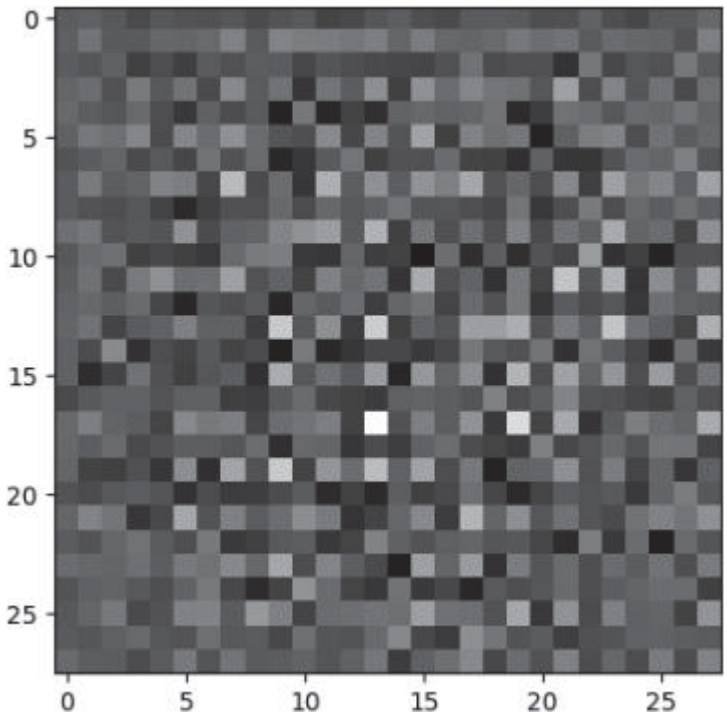

```
texto = "Is your car dirty? Discover our new product. Free for all.
Click the link."
seq_texto = tokenizer.texts_to_sequences([texto])
data_texto = pad_sequences(seq_texto, maxlen=T)
pred = model.predict(data_texto)
print(pred)
print ("SPAM" if pred >= 0.5 else "OK")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step
[[0.7058363]]
SPAM
```

```
3
```

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

(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).b
atch(BATCH_SIZE)

generator = make_generator_model()

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

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




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

  model.add(layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same'))
  model.add(layers.LeakyReLU())
  model.add(layers.Dropout(0.3))
  model.add(layers.Flatten())
  model.add(layers.Dense(1)) #camada densa com somente 1 neurônio
```

```
  return model

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

tf.Tensor([[-0.00400878]], shape=(1, 1), dtype=float32)

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

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

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

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

checkpoint_dir = './training_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)

EPOCHS = 100
noise_dim = 100
num_examples_to_generate = 16

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

@tf.function
def train_step(images):
  noise = tf.random.normal([BATCH_SIZE, noise_dim])

  with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
    generated_images = generator(noise, training=True)
    real_output = discriminator(images, training=True)
    fake_output = discriminator(generated_images, training=True)

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

  gradients_of_generator = gen_tape.gradient(gen_loss,
generator.trainable_variables)
```

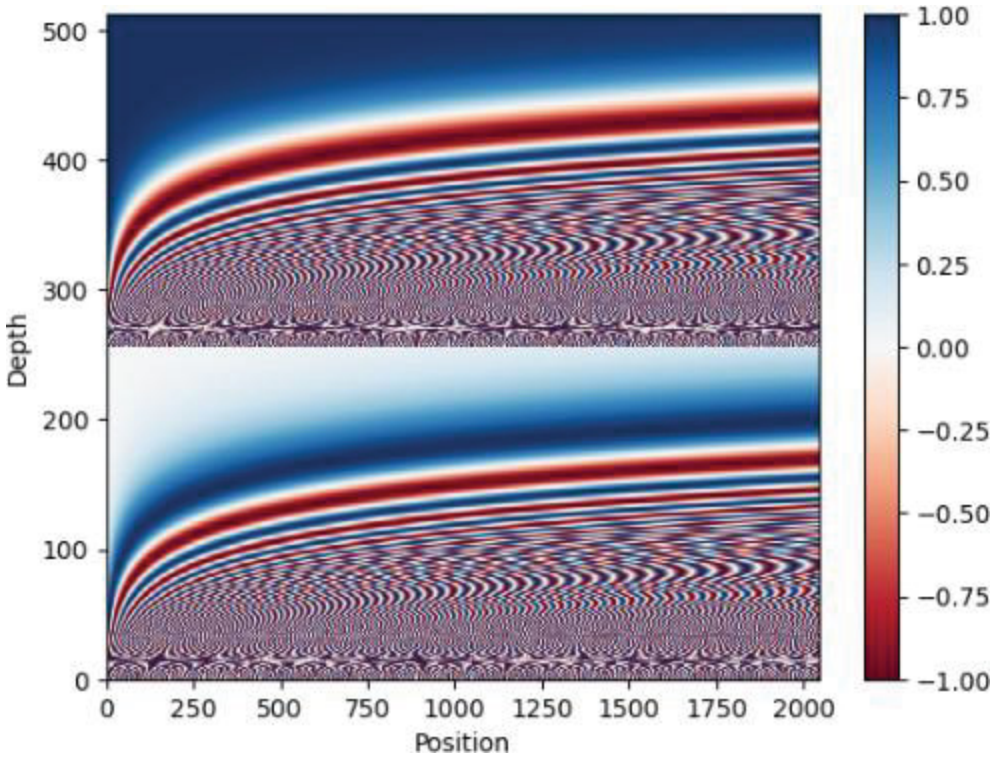
```
  gradients_of_discriminator = disc_tape.gradient(disc_loss,
discriminator.trainable_variables)

  generator_optimizer.apply_gradients(zip(gradients_of_generator,
generator.trainable_variables))


discriminator_optimizer.apply_gradients(zip(gradients_of_discriminator,
discriminator.trainable_variables))


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)
    generate_and_save_images(generator, epoch + 1, seed)

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

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

def 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

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()

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 telatolciss and in the concover .'
```

APÊNDICE 9 – BIG DATA

A – ENUNCIADO

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

B – RESOLUÇÃO

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 10 – VISÃO COMPUTACIONAL

A – ENUNCIADO

1) Extração de Características

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

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

Tarefas:

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

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%

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%

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

```
BATCH_SIZE = 32
WORK_FOLDER = '.'
BASES_FOLDER = WORK_FOLDER + '/Bases'
FEATURES_FOLDER = WORK_FOLDER + '/Features'
TRAINING_TXTFILE_PATH = WORK_FOLDER + '/Train.txt'
VALIDATION_TXTFILE_PATH = WORK_FOLDER + '/Valid.txt'
TEST_TXTFILE_PATH = WORK_FOLDER + '/Test.txt'
TRAINING_BASE_PATH = BASES_FOLDER + '/Train'
VALIDATION_BASE_PATH = BASES_FOLDER + '/Valid'
TEST_BASE_PATH = BASES_FOLDER + '/Test'
LBP_RADIUS = 1
```

```
LBP_NUM_POINTS = 8
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

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

Num GPUs Available: 1

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

x Bases/
x Bases/Test/
x Bases/Test/0/
x Bases/Test/0/66_HER2_10094.png
.
.
.
x Bases/Valid/3/22_HER2_18855.png
x Bases/Valid/3/22_HER2_19575.png
x Bases/Valid/3/22_HER2_9790.png
x Valid.txt
x Test.txt
x Train.txt
```

```
os.makedirs(FEATURES_FOLDER,                                    exist_ok=True)
def lbp_riu(img, n_points:int, radius:int):
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

def calc_features(arq):
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'
img = cv2.imread(line.strip(),0)
if img is None:
print(f"Erro ao carregar a imagem: {line.strip()}")
continue
if len(img.shape) > 2:
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
features, lbp = lbp_riu(img, LBP_NUM_POINTS, LBP_RADIUS)
features_str = '%s' % (label)
for P in features:
features_str += ",%d" % (P)
features_str += '\n'
arquivo = open(nome_arq, 'a')
arquivo.write(features_str)
arquivo.close()
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!')

Gerando Features da base de Treino...
Features de Treino geradas!
Gerando Features da base de Validação...
Features de Validação geradas!
Gerando Features da base de Teste...
Features de Teste geradas!

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()

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
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
=================================================================

train_generator                                                                  =
ImageDataGenerator(preprocessing_function=preprocess_input)
valid_generator                                                                  =
ImageDataGenerator(preprocessing_function=preprocess_input)
test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)
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.

print('Gerando Features da base de Treino...')
features_vgg_train = vgg.predict(traingen)
print('Features de Treino Geradas!')

print('---------------------------------------------------- ')
print('Gerando Features da base de Validação. . ')
features_vgg_valid = vgg.predict(validgen)
print('Features de Validação Geradas!')

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!

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)):
features_str = '%d' % (generator.labels[i])
for ft in features[i]:
if ft == 0:
features_str += ",%d" % (ft)
else:
features_str += ",%f" % (ft)
features_str += '\n'
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")

Salvando features da base de Treino...
Salvando features da base de Validação...
Salvando features da base de Teste...
fim

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")
```

```
Carregando features LBP...
Features LBP Carregadas
Carregando features VGG16...
Features VGG16 Carregadas
```

```
model_svm_lbp = svm.SVC(kernel='linear')
print ("Treinando Modelo...")
```

```
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...")
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
```

```
model_svm_vgg = svm.SVC(kernel='rbf')
print ("Treinando Modelo.  ")
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.  ")
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
```

```
model_rf_lbp = RandomForestClassifier()
print ("Treinando Modelo.  ")
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.  ")
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
```

```
model_rf_vgg = RandomForestClassifier()
print ("Treinando Modelo...")
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...")
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
```

```
model_mlp_lbp = MLPClassifier(random_state=1, max_iter=300,
learning_rate='adaptive')
print ("Treinando Modelo...")
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...")
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=30
0, 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
```

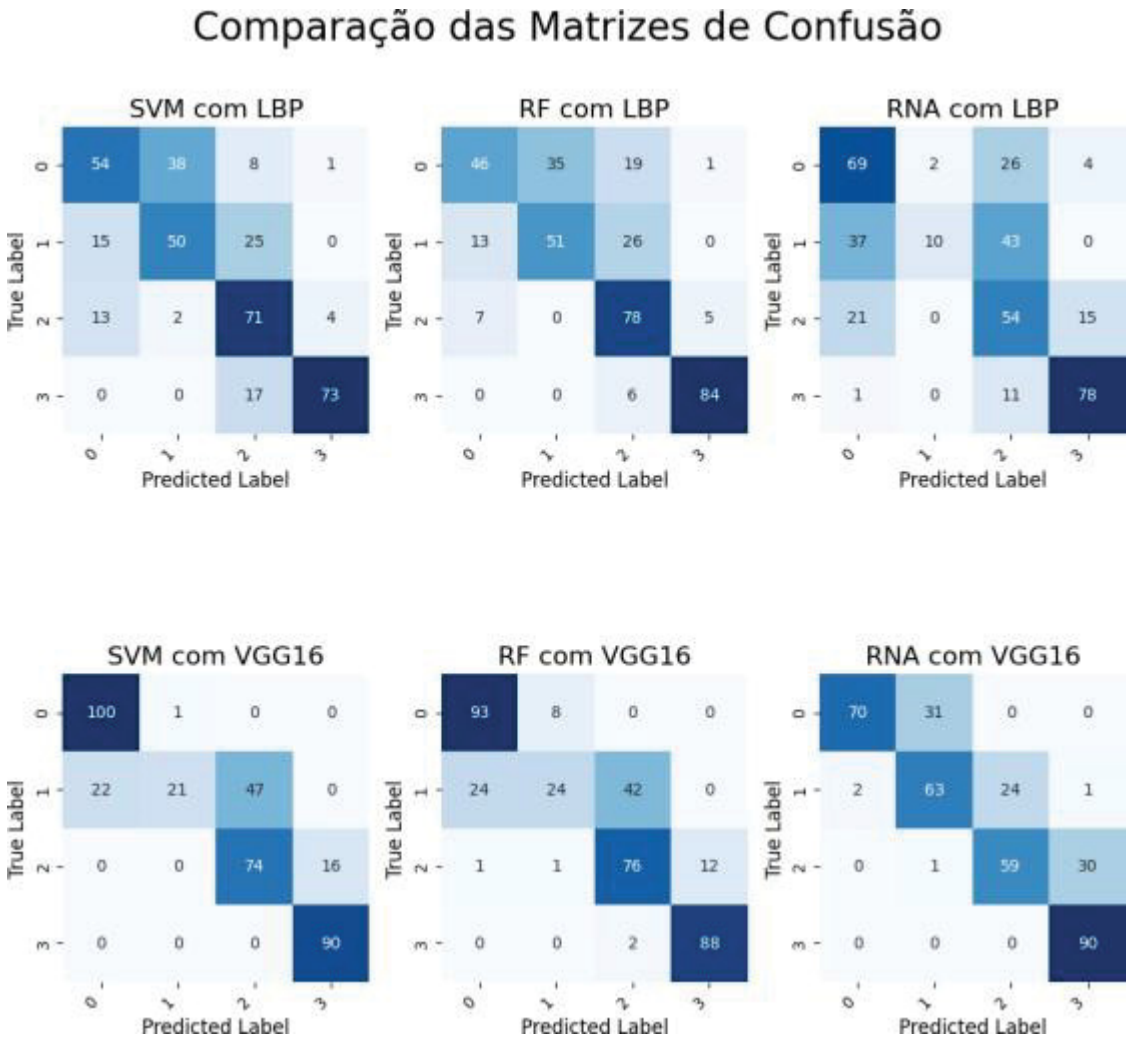
```
model_mlp_vgg = MLPClassifier(random_state=1, max_iter=300,
learning_rate='adaptive')
print ("Treinando Modelo...")
model_mlp_vgg.fit(X_train_vgg, y_train_vgg)
print (f"Treinamento finalizado.")
print ("Efetuando Predições com a base de validação para validar o modelo...")
predicted_mlp = model_mlp_vgg.predict(X_valid_vgg)
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=30
0, 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

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")

Carregando features LBP...
Features LBP Carregadas
Carregando features VGG16...
Features                           VGG16                           Carregadas

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!")

Efetuando Predições...
Predições                                                  Efetuadas!

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 sklear
cm = confusion_matrix(true_classes, pred_classes)
specificities = []
for i in range(len(cm)):
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)
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}%")

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%

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%

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%

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).

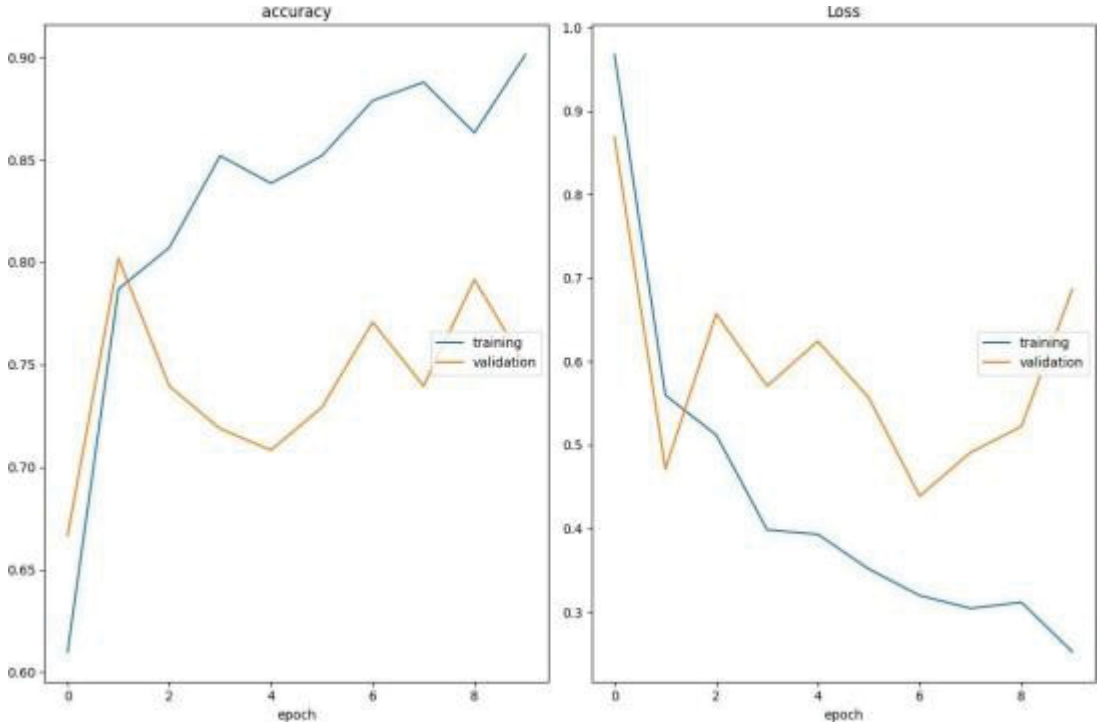
Código

```
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)

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'],
    subset='training',
    shuffle=True,
    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'],
    subset='validation',
    shuffle=True,
    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.

def monta_camada_top(model_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)
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

resnet = ResNet50(
input_shape=(224,224,3)
, weights='imagenet',
include_top=False
)
resnet.trainable = False

model_resnet_da = monta_camada_top(resnet)

model_resnet_da.summary()

Model: "model"
Layer (type) Output Shape Param # Connected to
```

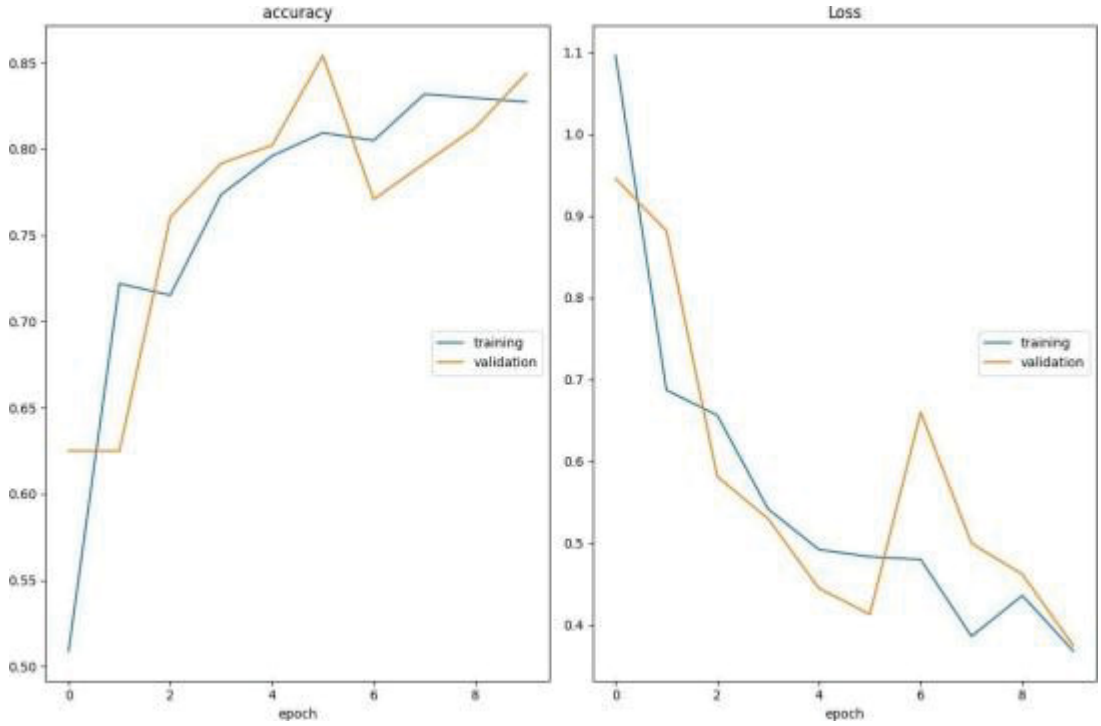
```
==================================================================================
===================================
input_1 (InputLayer) [(None, 224, 224, 3)] 0 []
conv1_pad (ZeroPadding2D) (None, 230, 230, 3) 0 ['input_1[0][0]']
conv1_conv   (Conv2D)   (None,   112,   112,   64)   9472   ['conv1_pad[0][0]']
.
.
.
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

steps_per_epoch = resnet_traingen.samples // BATCH_SIZE
val_steps = resnet_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)

model_resnet_da.compile(loss='categorical_crossentropy',
                        optimizer=optimizer, metrics=['accuracy'])

checkpoint = ModelCheckpoint(filepath=BEST_WEIGHTS_RESNET_DA_PATH,
                             verbose=1,
                             save_best_only=True)

history_resnet_da = model_resnet_da.fit(resnet_traingen,
                                        steps_per_epoch=steps_per_epoch,
                                        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
CPU times: total: 4min 12s
Wall                    time:                    3min                    4s

del history_resnet_da
del model_resnet_da
gc.collect()

64773

vgg = VGG16(
    input_shape=(224,224,3),
    weights='imagenet',
```

```
    include_top=False
)
vgg.trainable                                =                                False

model_vgg_da                        =                    monta_camada_top(vgg)
model_vgg_da.summary()

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
.
.
.
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

steps_per_epoch = vgg16_traingen.samples // BATCH_SIZE
val_steps = vgg16_valldgen.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'])

checkpoint = ModelCheckpoint(filepath=BEST_WEIGHTS_VGG16_DA_PATH,
                             verbose=1,
                             save_best_only=True)

history_vgg_da = model_vgg_da.fit(vgg16_traingen,
                                  epochs=30,
```

```
                                    steps_per_epoch=steps_per_epoch,
                                    validation_data=vgg16_valldgen,
                                    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
CPU times: total: 4min 55s
Wall time: 3min 31s

del history_vgg_da
del model_vgg_da
```

```
gc.collect()
```

```
84448
```

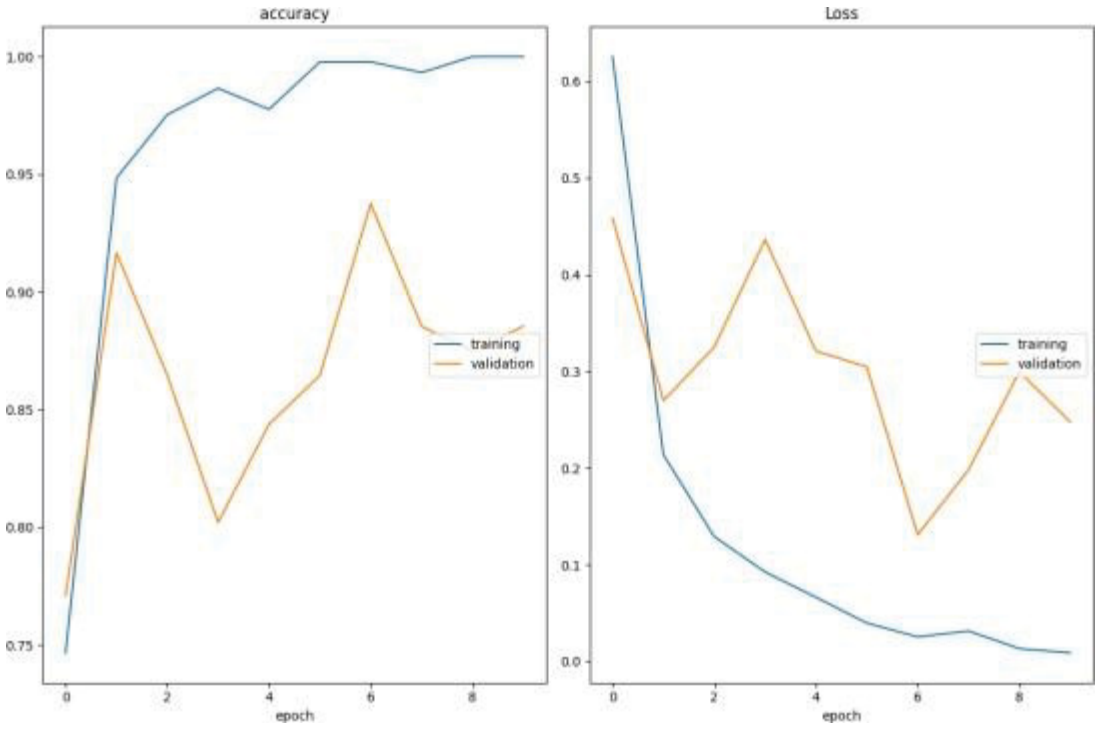
```
resnet_train_generator = ImageDataGenerator(
    validation_split=0,
    preprocessing_function=resnet_preprocess_input
)

vgg16_train_generator = ImageDataGenerator(
    validation_split=0,
    preprocessing_function=vgg16_preprocess_input
)

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.
```

```
model_resnet = monta_camada_top(resnet)
```

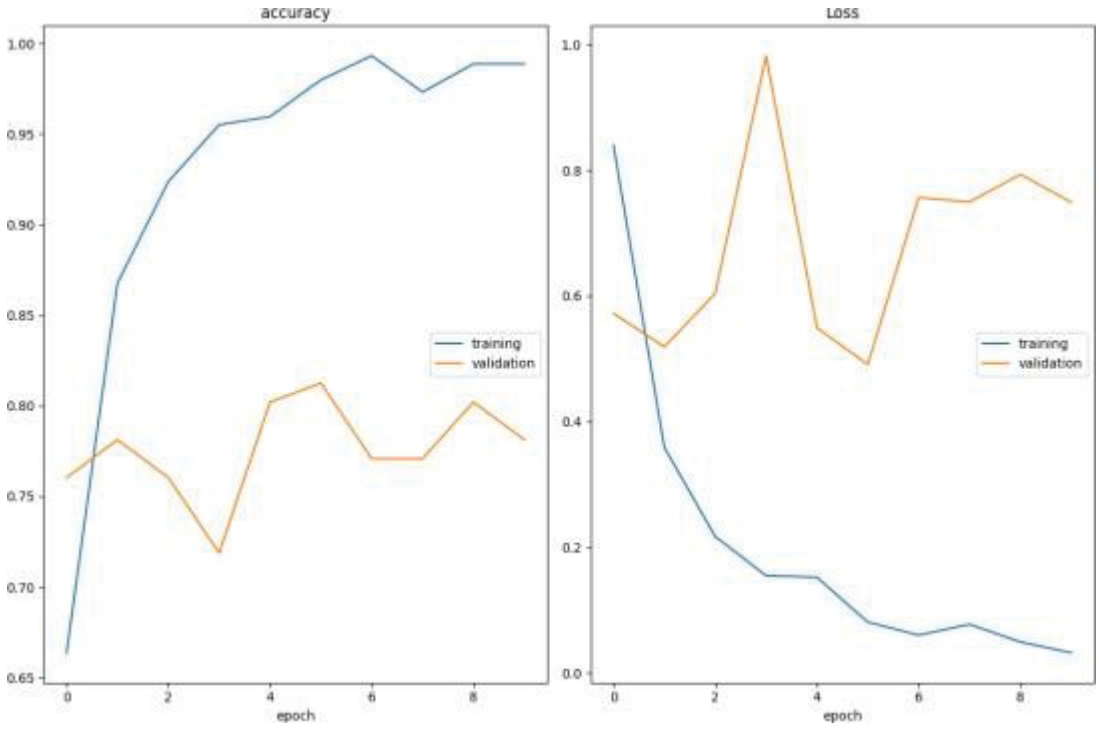
```
steps_per_epoch = resnet_traingen.samples //
```

```
BATCH_SIZE val_steps = resnet_validgen.samples //
BATCH_SIZE
optimizer                    =                    RMSprop(learning_rate=0.0001)

model_resnet.compile(loss='categorical_crossentropy',    optimizer=optimizer,
metrics=['accuracy'])

checkpointer = ModelCheckpoint(filepath=BEST_WEIGHTS_RESNET_PATH,

verbose=1,
save_best_only=True)

history_resnet = model_resnet.fit(resnet_traingen,
epochs=NUM_EPOCHS,
steps_per_epoch=steps_per_epoc
h,
validation_data=resnet_validge
n, 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

del history_resnet
del model_resnet
gc.collect()

32050

model_vgg                       =                    monta_camada_top(vgg)

steps_per_epoch = VGG16_traingen.samples // BATCH_SIZE
val_steps = vgg16_validgen.samples // BATCH_SIZE

optimizer = RMSprop(learning_rate=0.0001)

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

checkpoint = ModelCheckpoint(filepath=BEST_WEIGHTS_VGG16_PATH,
                             verbose=1,
                             save_best_only=True)

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

del history_vgg_da
del model_vgg_da
gc.collect()

82010

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.
```

```
model_resnet_pred = monta_camada_top(resnet)
model_vgg_pred = monta_camada_top(vgg)
```

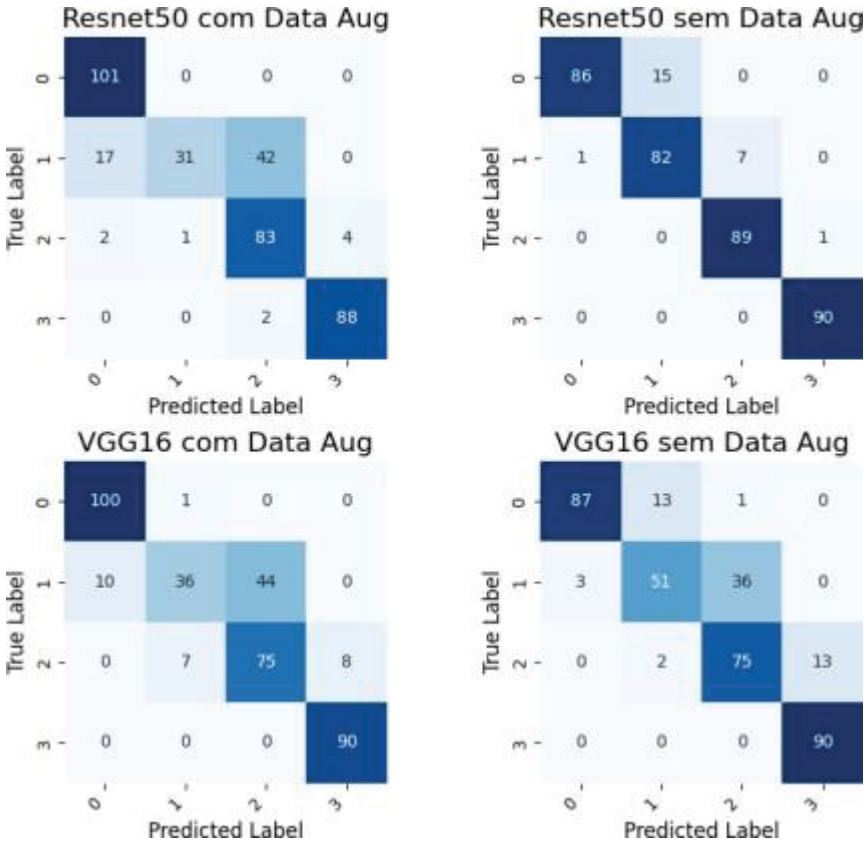
```
model_resnet_pred.load_weights(BEST_WEIGHTS_RESNET_DA_PATH)
model_vgg_pred.load_weights(BEST_WEIGHTS_VGG16_DA_PATH)
predicted_resnet_da = model_resnet_pred.predict(resnet_testgen)
predicted_classes_resnet_da = np.argmax(predicted_resnet_da, axis=1)
predicted_vgg_da = model_vgg_pred.predict(vgg16_testgen)
predicted_classes_vgg_da = np.argmax(predicted_vgg_da, axis=1)
```

```
12/12 [==============================] - 12s 980ms/step
12/12 [==============================] - 19s 2s/step
```

```
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

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 skle
cm = confusion_matrix(true_classes,
pred_classes) specificities = []
for i in range(len(cm)):

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)

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 11 – ASPECTOS FILOSÓFICOS E ÉTICOS DA IA

A – ENUNCIADO

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

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

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

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

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

9. Formatação Adequada: O trabalho deverá ser apresentado nas normas da ABNT de acordo com as diretrizes fornecidas, incluindo margens, espaçamento, tamanho da fonte e estilo de citação. Deve-se seguir o seguinte template de arquivo: 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.

APÊNDICE 12 – GESTÃO DE PROJETOS DE IA

A – ENUNCIADO

1 Objetivo

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

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

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

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

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

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

2 Orientações adicionais

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

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

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

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

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

Não altere as questões do template.

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

B – RESOLUÇÃO

Aluna/o:	DARIO CALDAS MARTINS
Nome do artigo escolhido:	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

Qual o **objetivo** do estudo descrito pelo artigo?	Qual o **problema/oportunidade/situação** que levou à necessidade de realização desse estudo?	Qual a **metodologia** que os autores usaram para obter e analisar as informações do estudo?	Quais os **principais resultados** obtidos pelo estudo?
O objetivo do estudo descrito no artigo é identificar as lacunas existentes nas práticas atuais de Engenharia de Requisitos para Inteligência Artificial (RE4AI), com foco em abordagens centradas no ser humano. O estudo busca responder a duas perguntas de pesquisa principais: mapear a pesquisa existente e as diretrizes da indústria para o desenvolvimento de IA centrada no ser humano, destacando os aspectos ausentes, e identificar as lacunas entre a pesquisa e as práticas industriais.	A necessidade de realizar este estudo surge do reconhecimento de que, embora a IA tenha se tornado um foco principal nas tecnologias atuais, muitos sistemas ainda falham em abordar aspectos centrados no ser humano. Isso inclui problemas como preconceitos e falhas em sistemas de IA, que não refletem necessariamente as necessidades dos usuários. Além disso, as práticas de Engenharia de Requisitos para IA não têm abordado adequadamente esses aspectos, resultando em sistemas que podem ser tecnicamente sólidos, mas que não atendem às necessidades humanas.	Os autores realizaram uma revisão sistemática da literatura (SLR) para mapear diretrizes industriais e estudos acadêmicos sobre RE4AI. Em seguida, conduziram uma pesquisa com profissionais da indústria para determinar quais diretrizes centradas no ser humano devem ser incluídas na Engenharia de Requisitos. A pesquisa foi projetada para identificar as ferramentas, linguagens de modelagem e práticas atualmente utilizadas, bem como os desafios enfrentados durante a fase de RE ao construir software de IA.	Os principais resultados do estudo incluem: - Identificação de que todas as abordagens centradas no ser humano mapeadas devem ser abordadas durante a fase de Engenharia de Requisitos ao construir sistemas de IA. - Descoberta de que a maioria das ferramentas e métodos usados atualmente precisa ser revisada para gerenciar RE4AI de forma eficaz. - Identificação de 15 ferramentas adicionais e diferentes notações e plataformas usadas na prática. - Comparativo entre RE4AI na pesquisa e na prática, destacando as lacunas em ambas as áreas. - Formulação de cinco recomendações de pesquisa futuras para melhorar as práticas de RE4AI. Essa análise detalhada fornece uma visão abrangente do estudo, abordando o objetivo, a situação que motivou a pesquisa, a metodologia utilizada e os resultados obtidos.

APÊNDICE 13 – FRAMEWORKS DE INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1 Classificação (RNA)

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

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

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

```
data = tf.keras.datasets.fashion_mnist

(x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()
```

2 Regressão (RNA)

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

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

Informações:

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

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv"

data = pd.read_csv(url, delimiter=';')
```

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

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

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

3 Sistemas de Recomendação

Implementar o exemplo de Sistemas de Recomendação usando a base de dados Base_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

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.

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

Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/t10k-images-idx3-ubyte.gz
4422102/4422102      ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━       0s      0us/step

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,)

display(x_train)

array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
```

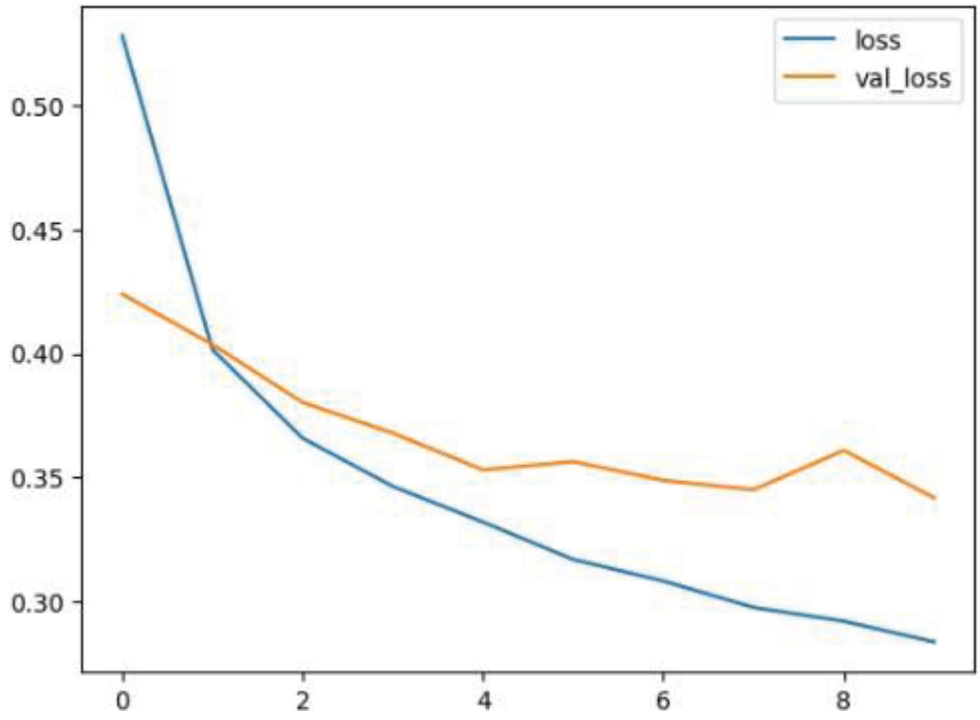
```
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       ...,

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

```
display(y_train)
```

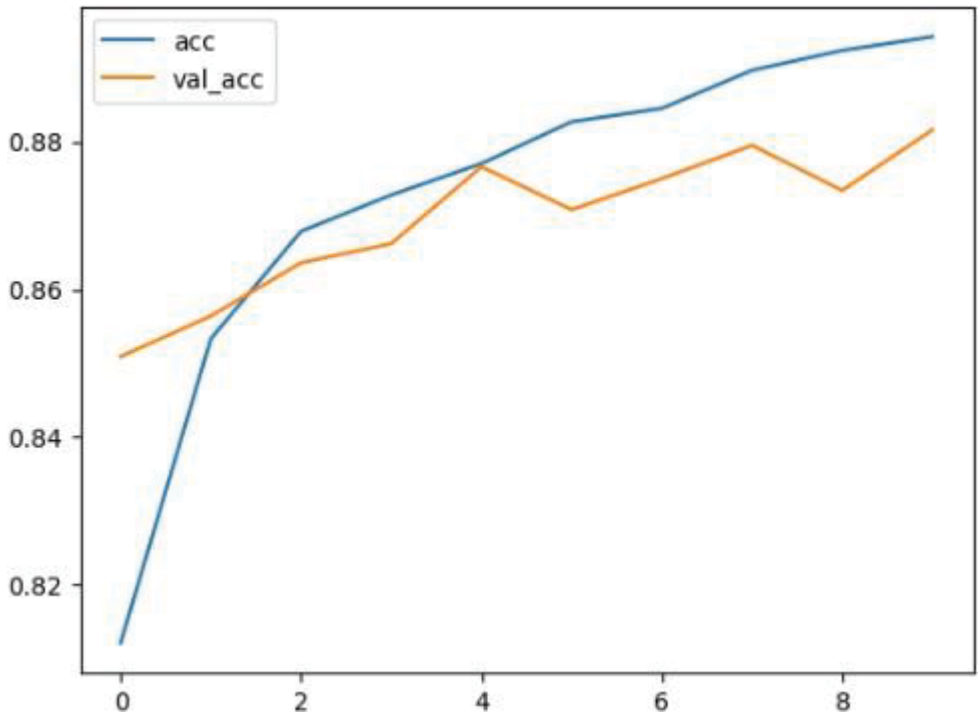
```
array([9,      0,      0,      ...,      3,      0,      5],      dtype=uint8)

x_train,         x_test         =         x_train/255.0,         x_test/255.0

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)

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 2/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step  - accuracy: 0.8473
- loss: 0.4126 - val_accuracy: 0.8564 - val_loss: 0.4036
Epoch 3/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step  - accuracy: 0.8666
- loss: 0.3683 - val_accuracy: 0.8636 - val_loss: 0.3802
Epoch 4/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step  - accuracy: 0.8709
- loss: 0.3492 - val_accuracy: 0.8662 - val_loss: 0.3679
Epoch 5/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step  - accuracy: 0.8765
- loss: 0.3294 - val_accuracy: 0.8766 - val_loss: 0.3531
Epoch 6/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step  - accuracy: 0.8829
- loss: 0.3152 - val_accuracy: 0.8708 - val_loss: 0.3564
```

```
Epoch 7/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8872
- loss: 0.3029 - val_accuracy: 0.8750 - val_loss: 0.3488
Epoch 8/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.8914
- loss: 0.2930 - val_accuracy: 0.8795 - val_loss: 0.3450
Epoch 9/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8931
- loss: 0.2901 - val_accuracy: 0.8734 - val_loss: 0.3609
Epoch 10/10
1875/1875 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8940
- loss: 0.2835 - val_accuracy: 0.8816 - val_loss: 0.3418
```

```
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()
```



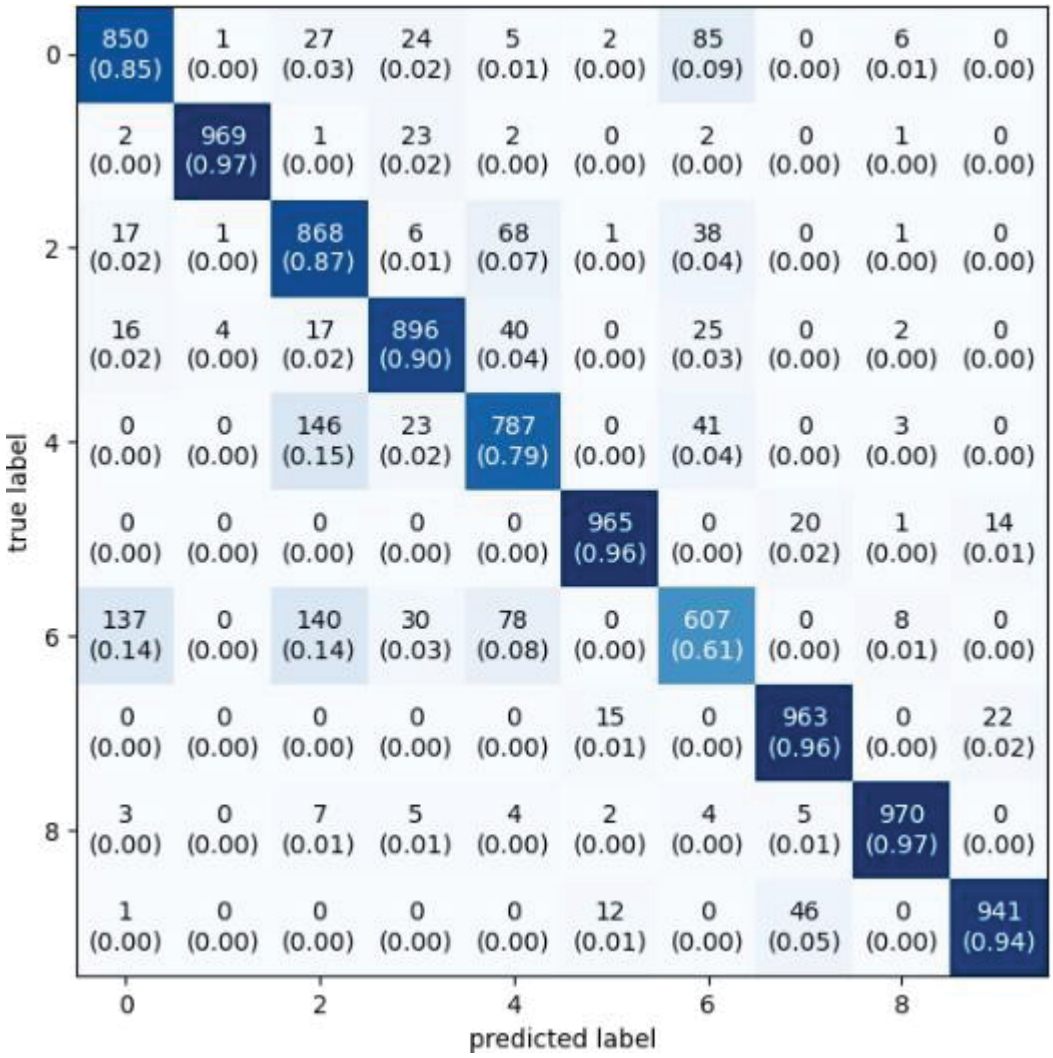

```
print(              model.evaluate(x_test,              y_test)              )
```

```
313/313 ———————————————————————————— 0s 1ms/step - accuracy: 0.8839 -
loss: 0.3343
[0.3418148458003998,                                          0.881600022315979]
```

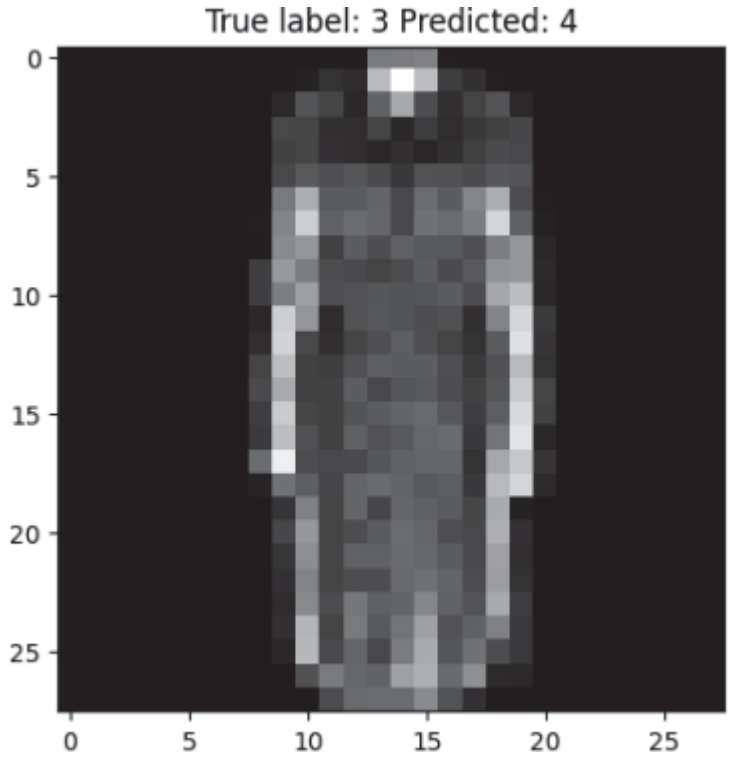
```
y_pred = model.predict(x_test).argmax(axis=1)
print(y_pred)
```

```
313/313 ———————————————————————————— 1s 2ms/step
[9           2           1           ...           8           1           5]
```

```
cm = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat=cm, figsize=(7, 7),
                      show_normed=True)
```




```
misclassified = np.where(y_pred != y_test)[0]

i                    =                    np.random.choice(misclassified)

Text(0.5,       1.0,       'True       label:       3       Predicted:       4')
```



2

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, R2):

Apesar dos valores apresentados nas métricas de erro serem pequenos, analisando se o coeficiente de determinação R2 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 da 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.

```
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
]

data.head()
```

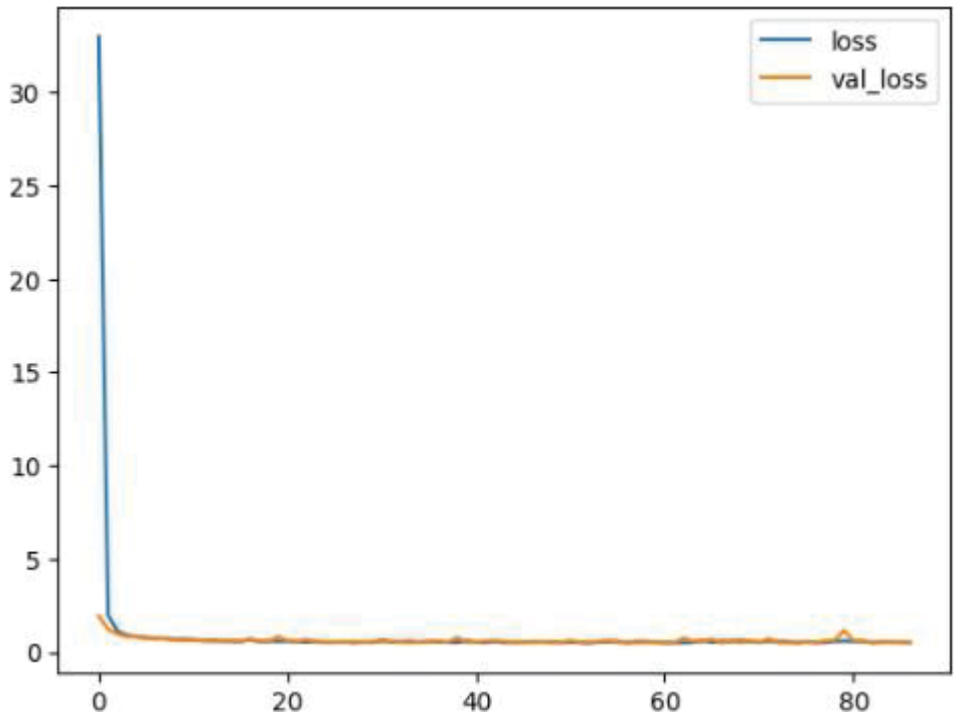
	acidez_fixa	acidez_volatil	acido_citrico	acucar_residual	cloretos	dioxido_de_enxofre_livre	dioxido_de_enxofre_total	densidade	pH	sulfatos	alcool	score_qualidade_vinho
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5
2	7.8	0.76	0.04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

```
data.shape
(1599, 12)

X = data.iloc[:,0:10].astype(float)
y = data.iloc[:,11].astype(float)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.25)

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)

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)

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

model.compile(optimizer=optimizer,
              loss=tf.keras.losses.mse,
              metrics=[rmse,                                                  r2])

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 2/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 2.2471 - r2: -
2.8260 - rmse: 1.1428 - val_loss: 1.2225 - val_r2: -1.0808 - val_rmse: 0.8756
Epoch 3/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 1.1147 - r2: -
0.8799 - rmse: 0.8419 - val_loss: 0.9557 - val_r2: -0.6054 - val_rmse: 0.7652
Epoch 4/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.9284 - r2: -
0.6026 - rmse: 0.7516 - val_loss: 0.8254 - val_r2: -0.3956 - val_rmse: 0.7223
Epoch 5/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.7919 - r2: -
0.2572 - rmse: 0.6976 - val_loss: 0.8008 - val_r2: -0.3269 - val_rmse: 0.7006
.
```

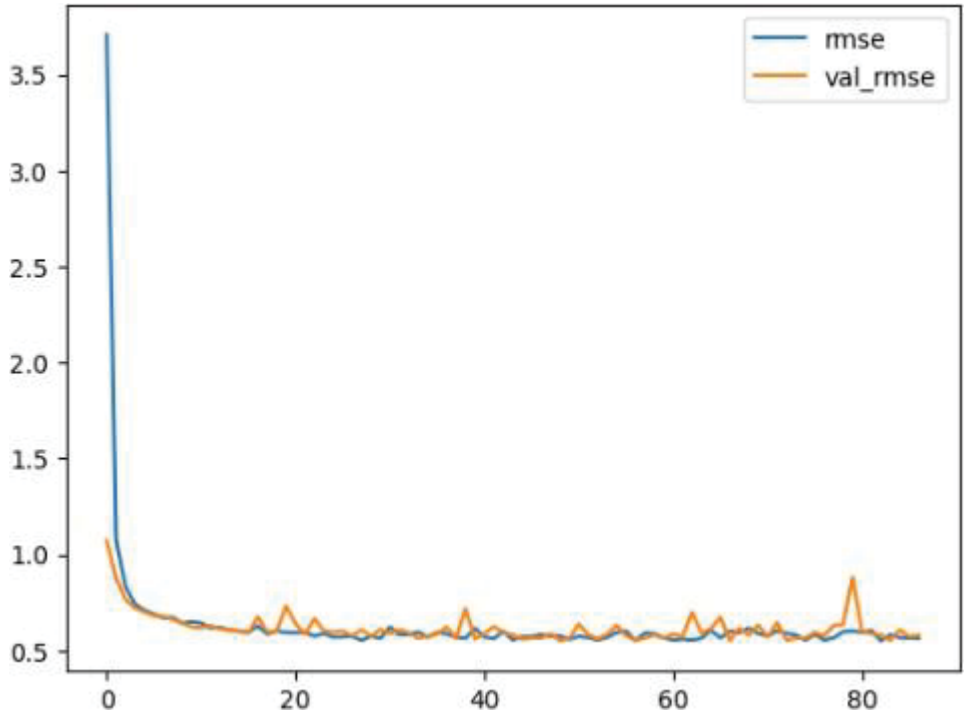
```
.
.
Epoch 83/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.5307 - r2:
0.2058 - rmse: 0.5748 - val_loss: 0.5257 - val_r2: 0.1395 - val_rmse: 0.5822
Epoch 84/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.5557 - r2:
0.1184 - rmse: 0.5868 - val_loss: 0.5025 - val_r2: 0.1633 - val_rmse: 0.5538
Epoch 85/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.5572 - r2:
0.1371 - rmse: 0.5764 - val_loss: 0.5563 - val_r2: 0.0689 - val_rmse: 0.6082
Epoch 86/1500
38/38 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - loss: 0.4785 - r2:
0.1995 - rmse: 0.5467 - val_loss: 0.5073 - val_r2: 0.1434 - val_rmse: 0.5728
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
```

```
plt.plot( r.history["loss"], label="loss" )
plt.plot( r.history["val_loss"], label="val_loss" )
plt.legend()
```

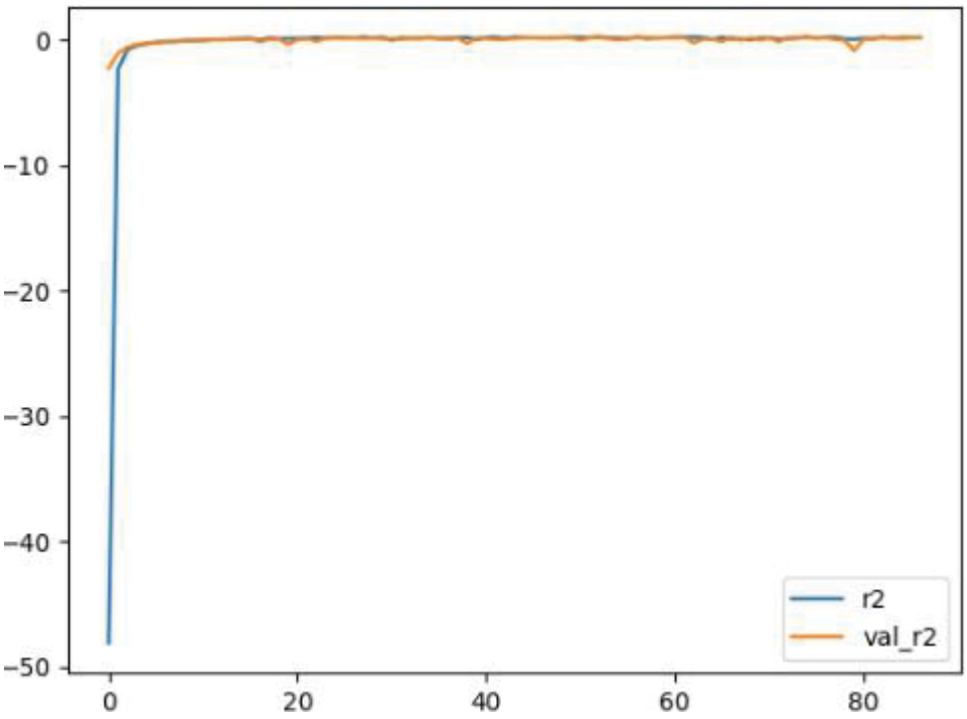



```
plt.plot( r.history["rmse"], label="rmse" )
```

```
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()
```



```
y_pred = model.predict(X_test).flatten()
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 13ms/step

mse  = mean_squared_error(y_test, y_pred)
rmse = sqrt(mse)
r2 = r2_score(y_test, y_pred)
print("mse      = ", mse)
print("rmse     = ", rmse)
print("r2 = ", r2)

mse      =  0.4866209823398623
rmse     =  0.6975822405565255
r2 = 0.21903228640689731
```

3

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.

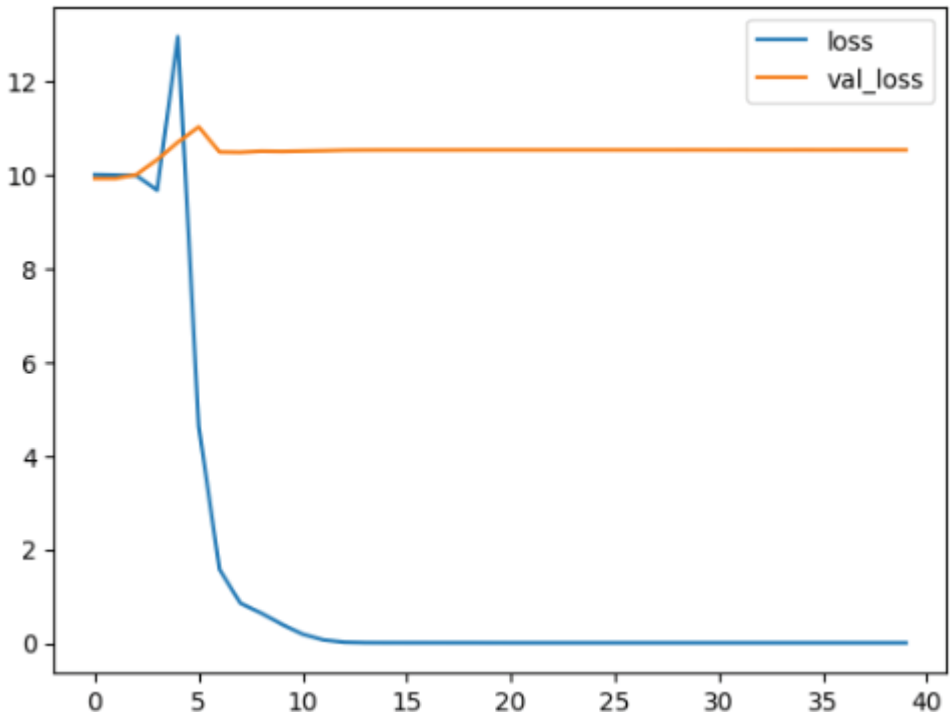
```
df = pd.read_csv('Base_livros.csv')
df.head()
```

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

```
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

N = len(set(df.new_ID_usuario))
M = len(set(df.new_ISBN))

K = 10

u = Input(shape=(1,))
u_emb = Embedding(N, K)(u) # saída : num_samples, 1, K
u_emb = Flatten()(u_emb)   # saída : num_samples, K

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)

model.compile(
    loss="mse",
    optimizer=SGD(learning_rate=0.08, momentum=0.9)
)

user_ids, isbn_ids, ratings = shuffle(df.new_ID_usuario, df.new_ISBN,
df.Notas)

Ntrain = int(0.8 * len(ratings))

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:]

avg_rating = train_ratings.mean()
train_ratings = train_ratings - avg_rating
test_ratings = test_ratings - avg_rating

epochs = 40
r = model.fit(
    x=[train_user, train_isbn],
    y=train_ratings,
    epochs=epochs,
    batch_size=1024,
    verbose=2,
    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 3/40
101/101 - 0s - 3ms/step - loss: 9.9887 - val_loss: 10.0067
Epoch 4/40
101/101 - 0s - 4ms/step - loss: 9.6719 - val_loss: 10.3250
.
.
.
Epoch 37/40
101/101 - 0s - 3ms/step - loss: 0.0035 - val_loss: 10.5325
Epoch 38/40
101/101 - 0s - 3ms/step - loss: 0.0035 - val_loss: 10.5323
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

plt.plot(r.history["loss"], label="loss")
plt.plot(r.history["val_loss"], label="val_loss")
plt.legend()
```

```
plt.show()
```




```
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

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.

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

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)

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

def show(img):
  display.display(PIL.Image.fromarray(np.array(img)))

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>'))
```

```
base_model = tf.keras.applications.InceptionV3(include_top=False,
weights='imagenet')
Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_n
otop.h5
87910968/87910968 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step

names = ['mixed3', 'mixed5']
layers = [base_model.get_layer(name).output for name in names]

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

def calc_loss(img, model):
  img_batch = tf.expand_dims(img, axis=0)
  layer_activations = model(img_batch)
  if len(layer_activations) == 1:
    layer_activations = [layer_activations]

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

  return tf.reduce_sum(losses)

class DeepDream(tf.Module):
```

```
  def __init__(self, model):
    self.model = model

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

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

        gradients = tape.gradient(loss, img)

        gradients /= tf.math.reduce_std(gradients) + 1e-8
        img = img + gradients*step_size
        img = tf.clip_by_value(img, -1, 1)

      return                              loss,                              img

deepdream                         =                      DeepDream(dream_model)

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)
```

```
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
```

```
15.100679159164429
```

APÊNDICE 14 – VISUALIZAÇÃO DE DADOS E STORYTELLING

A – ENUNCIADO

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

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

Entregue em um PDF:

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

- Explicação do **contexto e o 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

O conjunto de dados utilizado para a visualização é uma tabela que mostra a porcentagem de consumo de maconha, cocaína e crack por faixa etária, com foco nos últimos 12 meses. Esses dados representam um panorama do uso de substâncias ilícitas entre jovens e adultos, fornecendo uma análise detalhada da prevalência de consumo de cada substância em diferentes idades.

age	marijuana	crack	cocaine
12	1.10%	0.10%	0.00%
13	3.40%	0.10%	0.00%
14	8.70%	0.10%	0.00%
15	14.50%	0.50%	0.10%
16	22.50%	1.00%	0.00%
17	28.00%	2.00%	0.10%
18	33.70%	3.20%	0.40%
19	33.40%	4.10%	0.50%
20	34.00%	4.90%	0.60%
21	33.00%	4.80%	0.50%
22-23	28.40%	4.50%	0.50%
24-25	24.90%	4.00%	0.50%
26-29	20.80%	3.20%	0.40%
30-34	16.40%	2.10%	0.50%
35-49	10.40%	1.50%	0.50%
50-64	7.30%	0.90%	0.40%
65+	1.20%	0.00%	0.00%

Esses dados foram coletados de uma pesquisa nacional sobre o uso de drogas entre jovens e adultos. O foco é observar a evolução do consumo de maconha, cocaína e crack conforme a idade dos participantes.

Contexto e Público-Alvo

O objetivo dessa visualização de dados é entender como o consumo de substâncias ilícitas varia de acordo com a idade, especialmente em adolescentes e jovens adultos, que são as faixas etárias mais vulneráveis ao uso dessas substâncias. Ao analisar essas mudanças, podemos identificar os períodos críticos de maior consumo e, assim, direcionar melhor as políticas públicas e ações preventivas.

O público-alvo para essa visualização são os profissionais de saúde e educadores. Eles podem usar a visualização para adaptar programas de prevenção e reabilitação, focando principalmente nas idades mais vulneráveis.

A Visualização de Dados

A visualização foi criada utilizando o Power BI, uma plataforma que permite a criação de gráficos dinâmicos. A ferramenta foi escolhida devido à sua capacidade de gerar gráficos de fácil leitura, personalizáveis e interativos, o que permite que o público explore os dados de forma detalhada.

Para mostrar a comparação entre as substâncias e as faixas etárias, foi utilizado um gráfico de barras horizontais. Esse tipo de gráfico foi escolhido porque ele facilita a leitura das porcentagens de cada substância, permitindo uma comparação clara entre as substâncias em cada faixa etária. O

gráfico de barras horizontais é ideal para este tipo de dado, pois oferece uma visualização clara das diferenças percentuais entre as substâncias e facilita a comparação visual de diferentes faixas etárias, permitindo uma leitura mais intuitiva.

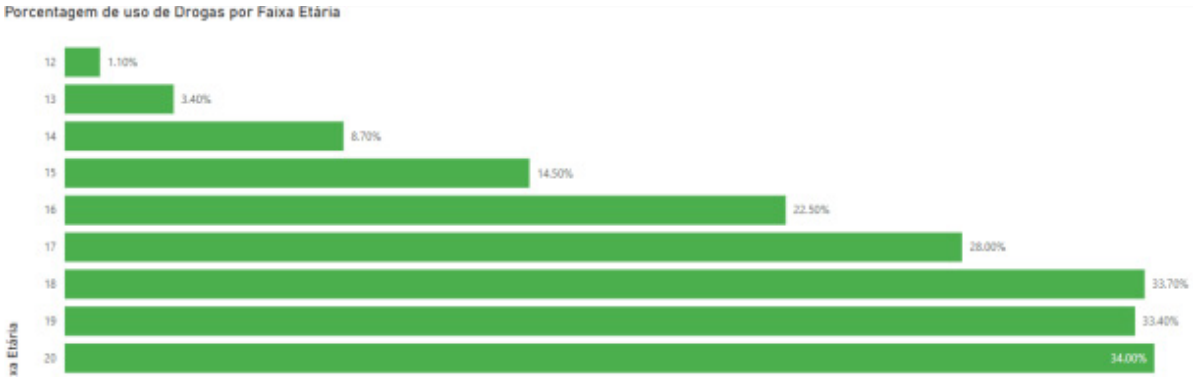
As cores escolhidas para cada substância são:

- Maconha: Verde (associada à cor da droga).
- Cocaína: Azul escuro (cor que transmite a ideia de impacto significativo).
- Crack: Vermelho escuro (associado ao perigo e à natureza altamente viciante e destrutiva do crack).

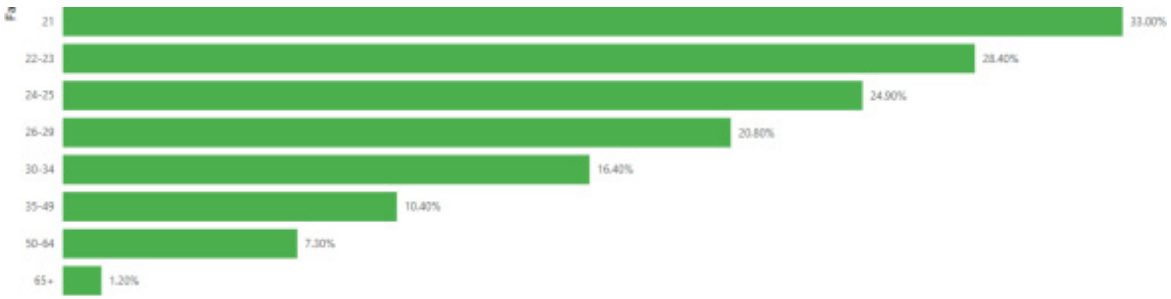
Essas cores foram escolhidas com base em psicologia das cores para transmitir as características e os impactos de cada substância de maneira intuitiva.

Descrição da Narrativa/Storytelling

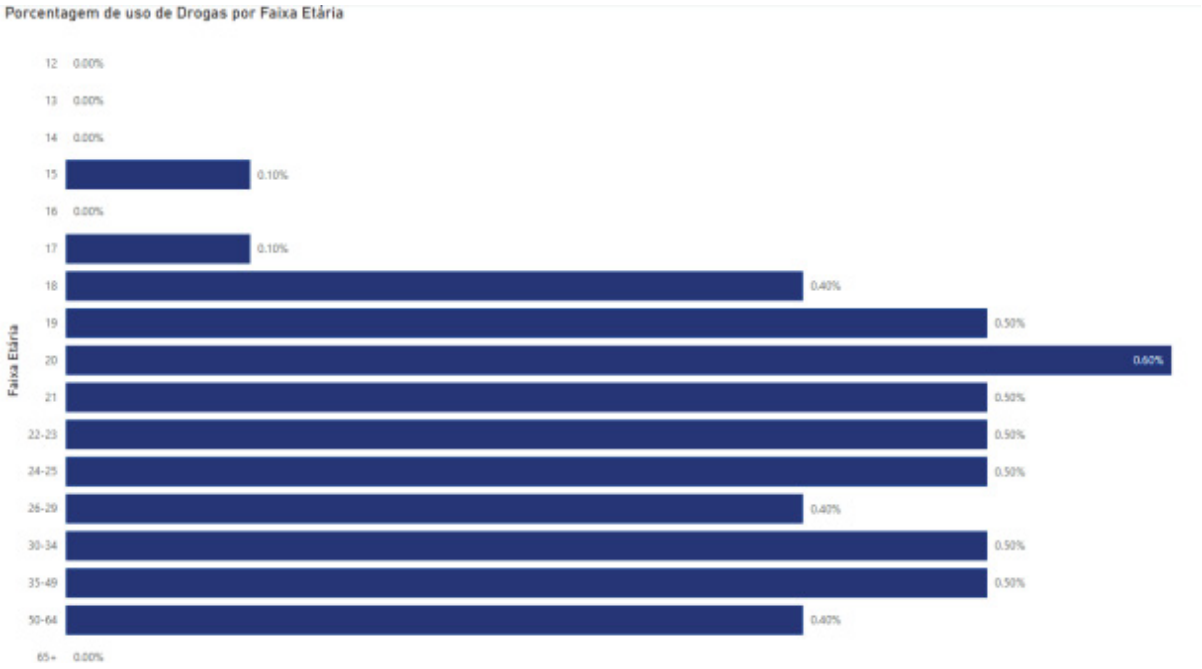
A narrativa criada a partir dessa visualização busca contextualizar os dados e explicar os padrões observados no gráfico. A história começa com a análise do consumo de maconha, que começa de forma modesta aos 12 anos (1,1%) e aumenta de forma constante até atingir seu pico aos 20 anos (34%). Esse pico inicial sugere que a maconha é uma droga de iniciação, consumida principalmente durante a adolescência e o início da idade adulta.



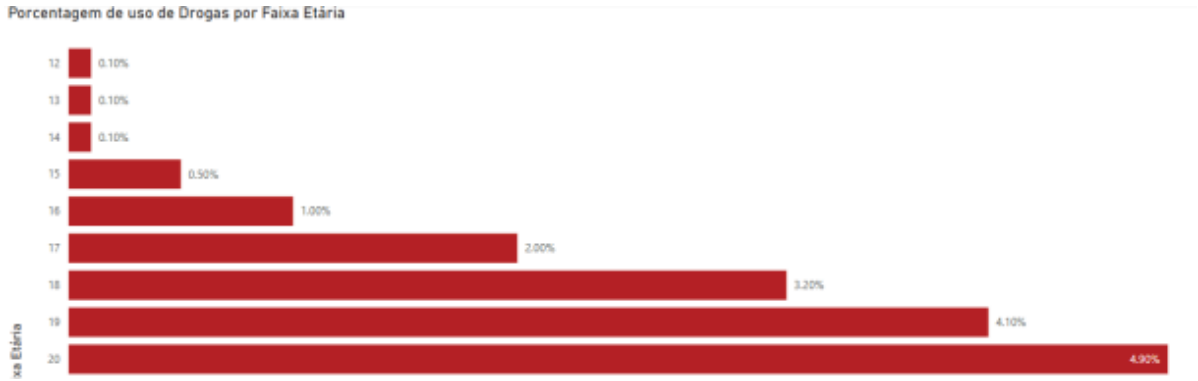
Após os 20 anos, observa-se uma queda gradual no consumo, indicando que muitas pessoas deixam de consumir maconha após a juventude.



A cocaína mostra um padrão diferente. Embora o consumo também tenha um pequeno aumento até os 20 anos, ele permanece em níveis baixos e estáveis após essa faixa etária, com um pico de 0,6% aos 20 anos. Isso sugere que, embora a cocaína seja consumida por uma parcela da população jovem, seu uso não se espalha de forma tão prevalente quanto o da maconha.

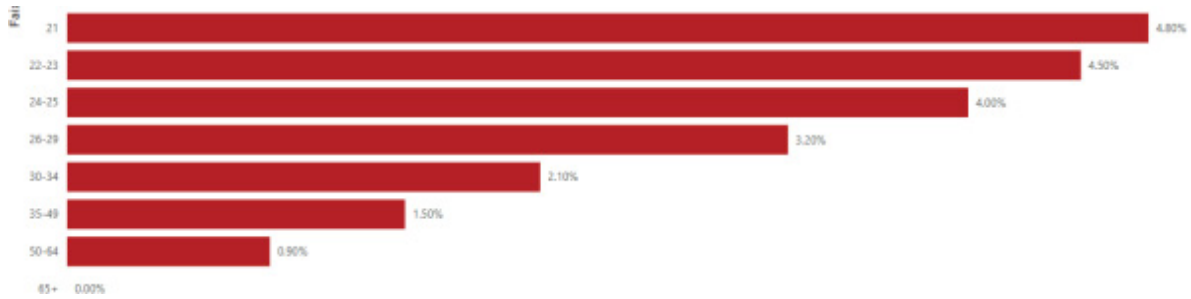


O crack, por sua vez, apresenta um padrão semelhante ao da maconha, com uma rápida ascensão no consumo até os 20 anos (4,9%), o que indica o impacto devastador dessa substância nas faixas etárias mais jovens.

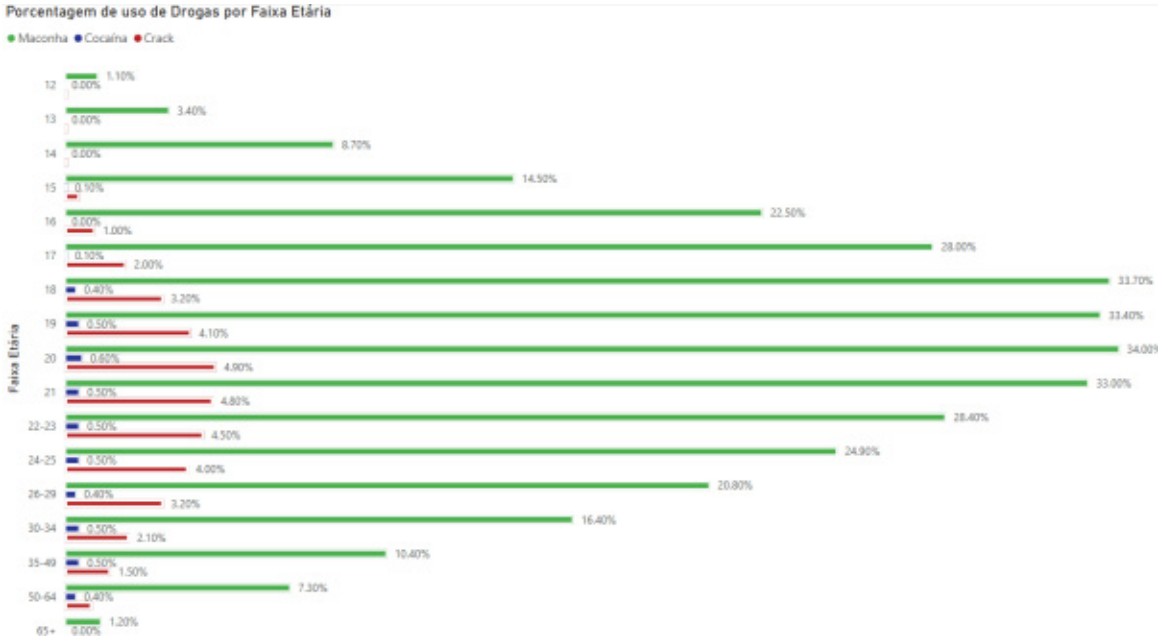


O consumo de crack, assim como o de maconha, diminui após os 20 anos, mas permanece presente em níveis baixos em faixas etárias mais velhas, refletindo a natureza altamente viciante e

prejudicial dessa droga.



Esses dados trazem importantes insights sobre o comportamento de consumo ao longo da vida. A visualização sugere que as políticas de prevenção devem se concentrar principalmente nas faixas etárias de 12 a 20 anos, quando o consumo de drogas, especialmente maconha e crack, está em seu auge. Além disso, a constância do consumo de cocaína e crack em faixas etárias mais velhas indica a necessidade de programas contínuos de reabilitação e tratamento.



A visualização também pode ser usada para aumentar a conscientização sobre os riscos do consumo precoce de substâncias, servindo como um alerta para adolescentes e jovens adultos sobre os perigos das drogas.

APÊNDICE 15 – TÓPICOS EM INTELIGÊNCIA ARTIFICIAL

A – ENUNCIADO

1) Algoritmo Genético

Problema do Caixeiro Viajante

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

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

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

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

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

Sugestão:

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

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

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
```

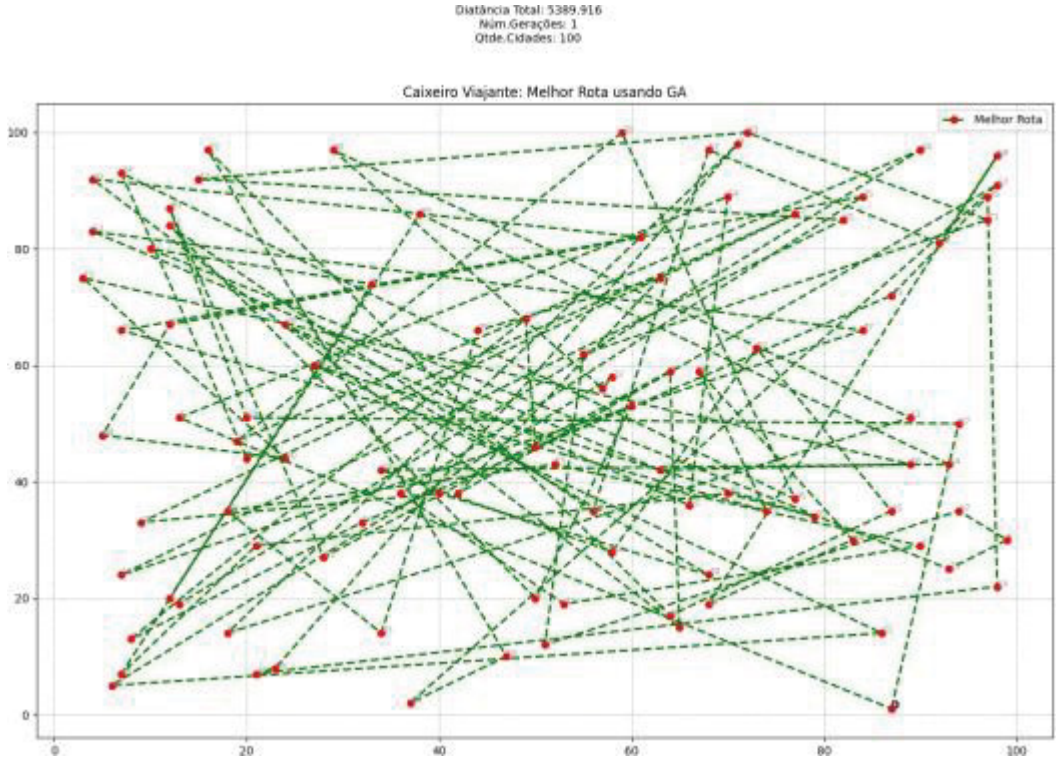
```
i==0 else 'normal', fontsize=10 if i==0 else 8, color='#000000' if i==0 else
'#999999')

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:
ret = []
while len(ret) < qtdeCidades:
cidade = (rd.randint(1, xMax), rd.randint(1, yMax))
while cidade in ret:
cidade = (rd.randint(0, xMax), rd.randint(0, yMax))
ret.append(cidade)
ret.append(ret[0])
return ret

def distanciaEuclidiana(pontoA : tuple, pontoB : tuple) -> float:
return ( np.sqrt(np.sum((np.array(pontoA) - np.array(pontoB))**2)) )

def distanciaTotal(individuo : list, coordenadas: list) -> float:
coordInd = [ coordenadas[c] for c in individuo ]

return sum([ distanciaEuclidiana(c1, c2) for c1, c2 in list(zip(coordInd[:-
1],coordInd[1:])) ])

def criaPopulacaoInicial(qtdeIndividuos : int, qtdeCidades : int) -> list:
from math import factorial

ret = []
individuo = list(range(1, qtdeCidades))
ret.append([0]+individuo+[qtdeCidades])
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')
```

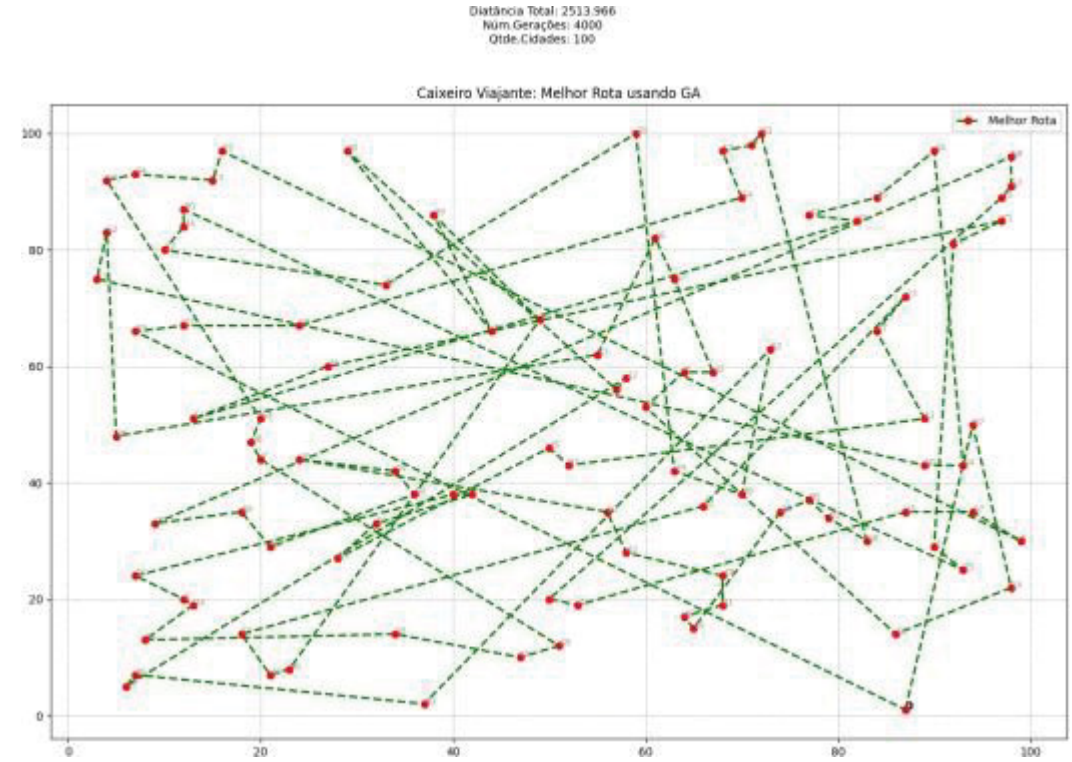
```
while len(ret) < qtdeIndividuos:

rd.shuffle(individuo)
while (individuo in ret): # previne permutações repetidas
rd.shuffle(individuo)
ret.append([0]+individuo+[qtdeCidades])

return ret
def avaliaSolucao(populacao : list, coordenadas : list, xMax : int, yMax :
int) -> list:
ret=[]

maiorDistancia = distanciaEuclidiana((1, 1), (xMax, yMax))
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

def ranqueiaPopulacao(populacao : list, coordenadas : list, xMax : int, yMax
: int) -> list:
return np.argsort(avaliaSolucao(populacao, coordenadas, xMax , yMax))[::-1]

def selecionaMelhores(nMelhores : int, populacao : list, coordenadas : list,
xMax : int, yMax : int) ->
list:
ret = [ populacao[i] for i in np.argsort(avaliaSolucao(populacao,
coordenadas, xMax , yMax))[::-1]
][:nMelhores]
return ret

def cruzamento(populacaoOrigem : list, populacaoDestino : list, qtde : int =
1) -> list:
ret = []
while len(ret) < qtde:

posicaoA = rd.randrange(0, len(populacaoOrigem))
posicaoB = posicaoA
while posicaoA == posicaoB:
posicaoB = rd.randrange(0, len(populacaoOrigem))
```

```
individuoA = populacaoOrigem[posicaoA]
individuoB = populacaoOrigem[posicaoB]
indA     =     populacaoOrigem[posicaoA][1:len(populacaoOrigem[posicaoA])-1]

indB = populacaoOrigem[posicaoB][1:len(populacaoOrigem[posicaoB])-1]
corte = rd.randrange( round(len(indA)/2), len(indA) )
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)
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)
if (len(ret) < qtde) and (novoIndividuoB not in populacaoDestino):
ret.append(novoIndividuoB)

return                                                                ret

def mutacao(populacaoOrigem : list, populacaoDestino : list, qtde : int = 1)
-> list:
ret = []
while len(ret) < qtde:
posicao = rd.randrange(0, len(populacaoOrigem))
individuo = populacaoOrigem[posicao]
gene1 = rd.randrange( 1, round(len(individuo)/2) )
gene2 = rd.randrange( round(len(individuo)/2)+1, len(individuo)-1 )
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]]
if individuo not in populacaoDestino: # valida se o novo individuo já existe
ret.append(individuo)

return                                                                    ret

qtdeIndividuos = 300
qtdeCidades = 100
xTamMax = 100
yTamMax = 100
percMuta = 0.02
percCruza = 0.9
totalGeracoes = 4000

populacaoInicial = criaPopulacaoInicial(qtdeIndividuos, qtdeCidades)
coordenadas = criaCoordenadas(qtdeCidades, xTamMax, yTamMax)
print("Coordenadas das Cidades: ",end='')
print(coordenadas)

qtdeCruza = round(percCruza*qtdeIndividuos)
qtdeMuta = round(percMuta*qtdeIndividuos)
qtdeMelhores = qtdeIndividuos - qtdeCruza - qtdeMuta

numGeracao                                =                                1

melhorIndividuo = selecionaMelhores(len(populacaoInicial), populacaoInicial,
coordenadas,
xTamMax, yTamMax)[0]
menorDistancia = distanciaTotal(melhorIndividuo, coordenadas)

print(    f'Geracao    {numGeracao}:    {menorDistancia}\nMelhor    solucao:
{melhorIndividuo}')
plotaGrafico(melhorIndividuo,    coordenadas,    numGeracao,    qtdeCidades,
menorDistancia)

geracaoAtual = populacaoInicial
while numGeracao <= totalGeracoes-1:
```

```
proximaGeracao = []

proximaGeracao.extend(     selecionaMelhores(qtdeMelhores,     geracaoAtual,
coordenadas,
xTamMax,                                yTamMax)                                )

proximaGeracao.extend( cruzamento(geracaoAtual, proximaGeracao, qtdeCruza) )

proximaGeracao.extend( mutacao(geracaoAtual, proximaGeracao, qtdeMuta) )

geracaoAtual    =    selecionaMelhores(len(proximaGeracao),    proximaGeracao,
coordenadas,
xTamMax, yTamMax)
numGeracao += 1
print('.',end='')

if numGeracao%50 == 0:
melhorIndividuo = geracaoAtual[0]
menorDistancia = distanciaTotal(melhorIndividuo, coordenadas)
print( f'\nGeracao {numGeracao}: {menorDistancia}')

print(          f'Geracao          {numGeracao}:          {menorDistancia}')

print(f'Melhor solucao: {melhorIndividuo} com {menorDistancia} unidades')
plotaGrafico(melhorIndividuo,    coordenadas,    numGeracao,    qtdeCidades,
menorDistancia)
```





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 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 para Execução

```
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."
]

vectorizer = CountVectorizer()
X = vectorizer.fit_transform(sentences).toarray()

pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

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()
```

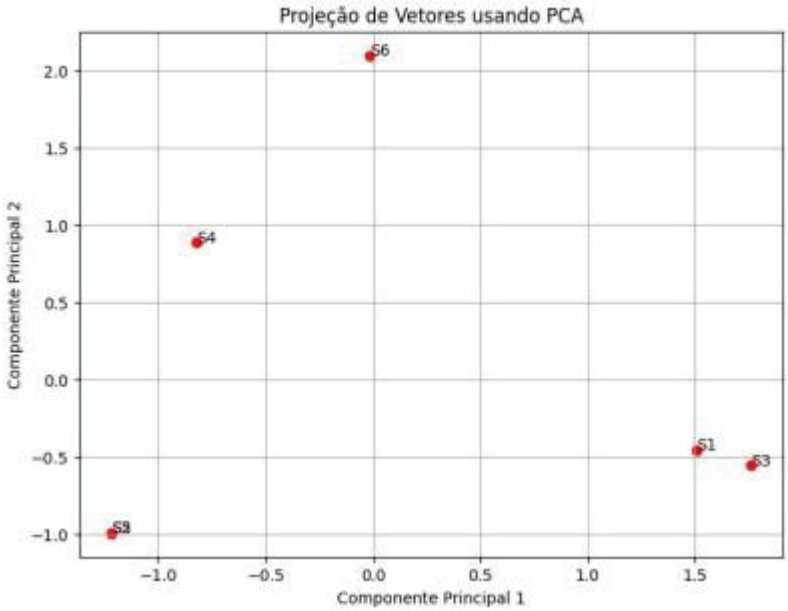
Resumo

Sentenças Similares: As sentenças 1 e 3, assim como, 2 e 5, são projetadas próximas umas das outras devido às palavras comuns.

PCA: Facilita a visualização da relação semântica entre as frases.

Visualização: O gráfico exibirá a dispersão das sentenças em um espaço 2D, destacando similaridades.

Esse processo fornece uma visão clara das semelhanças e diferenças semânticas entre as frases através das projeções de PCA.


Projeção de Vetores usando PCA
S6
S4
S1
S3
S5
Componente Principal 2
Componente Principal 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5